AUTOMATIC DIFFERENTIATION OF ADVANCED CFD CODES FOR MULTIDISCIPLINARY DESIGN

C. Bischof[1] and **G. Corliss** [2]
Argonne National Laboratory, Argonne, Illinois

L. Green
NASA Langley Research Center, Hampton, Virginia

A. Griewank[1]
Argonne National Laboratory, Argonne, Illinois

K. Haigler and P. Newman
NASA Langley Research Center, Hampton, Virginia

Argonne Preprint MCS-P339-1192

Automated multidisciplinary design of aircraft and other flight vehicles requires the optimization of complex performance objectives with respect to a number of design parameters and constraints. The effect of these independent design variables on the system performance criteria can be quantified in terms of sensitivity derivatives which must be calculated and propagated by the individual discipline simulation codes. Typical advanced CFD analysis codes do not provide such derivatives as part of a flow solution; these derivatives are very expensive to obtain by divided (finite) differences from perturbed solutions. It is shown here that sensitivity derivatives can be obtained accurately and efficiently by using the ADIFOR source translator for automatic differentiation. In particular, it is demonstrated that the 3-D, thin-layer Navier–Stokes, multigrid flow solver called TLNS3D is amenable to automatic differentiation in the forward mode even with its implicit iterative solution algorithm and complex turbulence modeling. It is significant that, using computational differentiation, consistent discrete nongeometric sensitivity derivatives have been obtained from an aerodynamic 3–D CFD code in a relatively short time, e.g. $\mathcal{O}$(man-week) not $\mathcal{O}$(man-year).

[1] This work was supported by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U. S. Department of Energy, under Contract W-31-109-Eng-38.

[2] This work was supported by the National Science Foundation under Cooperative Agreement Number CCR-9120008.

1 Nomenclature

C_D	Wing drag coefficient
C_L	Wing lift coefficient
C_M	Wing pitching moment coefficient
D	Generic sensitivity derivative
I	Identity matrix
J	Jacobian matrix
M	Free stream Mach number
P	Preconditioner matrix
R	Residual vector for flow equations
Re	Reynold's number (mean chord)
S	Seed matrix
x	Design variable
y	Discrete mesh coordinates
z	Local flow (state) variable
α	Angle-of-attack
ρ	Spectral radius
Subscripts	
AD	Automatic differentiation
DD	Divided difference
m	Iteration index
x	Partial derivative w.r.t. x
y	Partial derivative w.r.t. y
z	Partial derivative w.r.t. z
$*$	Root of $R = 0$ or iteration-fixed-point
Superscripts	
$'$	(prime) Total derivative w.r.t. x
$\tilde{}$	(tilde) Approximate operator

2 Introduction

In the past, design of flight vehicles typically required the interaction of many technical disciplines over an extended period of time in a more or less sequential manner. At present, computer-automated discipline analyses and interactions offer the possibility of significantly shortening the design-cycle time, while simultaneous multidisciplinary design optimization (MDO) via formal sensitivity analysis (SA) holds the possibility of improved designs. Recent topical conferences [3] [1–8], [10, 12, 34, 35, 54, 55] for example, attest to the interest in these possibilities for improving aerospace vehicle design processes and procedures. Advances in computer hardware and software, electronic communications, and discipline solution algorithms and codes will individually contribute; however, true synergisms may be required to make it all feasible. This paper addresses one such synergism for computa-

[3] Those without published proceedings include the 1992 AIAA/AHS/ASEE Aerospace Design Conference, Feb. 1992, and the AIAA Aircraft Design Systems Meeting, Aug. 1992.

tional science and computational fluid dynamics (CFD) algorithm technologies.

Procedures for MDO of engineering systems have been addressed by Sobieski [65]. He proposes a unified system SA guided by system sensitivity derivatives (SD); the optimizer code or algorithm that uses these SD is the outermost loop of the entire design process. The objective and constraint functions are now generally composed of output functions from several disciplines. Each single-discipline analysis code is then to supply not only the output functions required for the constrained optimization process and other discipline analysis inputs, but also the derivatives of all of these output functions with respect to its input variables. These variables include not only the MDO variables, but also output functions from other disciplines that implicitly depend on the MDO variables.

Thus, a key technology required for MDO procedures is the capability to calculate the SD of outputs from the various analysis codes with respect to a set of design variables. However, a certain degree of flexibility and automation is needed, since the envisioned flight vehicle concept determines which objectives, constraint functions, MDO design variables, and discipline analysis codes are required to model the pertinent physical aspects throughout the flight regime (i.e., the particular MDO problem). Current technology cannot be counted on to deliver reliable and fast derivatives for large computer codes such as advanced 3-D CFD codes. Divided differences (DD) may not be accurate and are obtained too slowly, symbolic approaches do not appear to be feasible, and hand coding of derivatives is impractical. This situation has dire consequences, in particular, for very large scale computer models, as they are to be run on teraflops machines. Since DD errors tend to grow with problem complexity, larger models will have to deal with ever-more-inaccurate derivatives, even though a faithful modeling of their complex nonlinear behavior requires very accurate derivatives. In addition, the cost of DD will restrain the magnitude of problems that can be done in practice.

Automatic differentiation (AD) addresses this need by providing a scalable technology that computes derivatives of large codes accurately, irrespective of the complexity of the model. This paper discusses and documents the initial application of an AD system to advanced CFD codes in order to obtain SD typical of those required in an MDO. The general ideas and direction of this work, including a sample result, have been outlined in [56] and [18]. As will be seen, the initial results given here are both significant and encouraging; but challenges remain.

The organization of this paper is as follows: first, brief reviews of advanced CFD codes with SD calculations and AD of Fortran codes (ADIFOR); then, discussion of the application of ADIFOR to CFD codes; and finally, comments on the future directions of this work.

The present interest and work have been stimulated by two research programs related to incorporating advanced CFD capabilities in MDO. The NASA Langley Research Center High-Speed Airframe Integration Research (HiSAIR) project [32, 29, 28] is focused on the High-Speed Civil Transport (HSCT) design activity in order to develop a methodology and computational environment for multidisciplinary analysis and design. The emphasis is on including most of the required disciplines and interactions at a sufficiently advanced level of analysis to demonstrate improved engineering design methodology. The second stimulus is the NASA Computational Aerosciences (CAS) grand challenge of the High Performance Computing and Communications (HPCC) Program [45, 55], where one of the applications is the HSCT. The two major thrusts in this latter program are enhanced simulations via multidisciplinary formulations and improved computational efficiency via massively parallel hardware. In both programs, the primary NASA Langley approach being pursued is MDO via SA.

3 Advanced CFD with SD

The application of advanced CFD codes to provide aerodynamic analyses within an MDO via SA is severely hampered by the sheer magnitude of the computational task if these SD must be obtained by DD. Recent interest and progress have focused on quasi-analytical (QA) or "adjoint-related" techniques to get these SD. The most recent references [13, 22, 33, 37, 38, 39, 47, 50, 52, 56, 62, 63, 66, 70] from several groups engaged in this research indicate the current status and cite many references to earlier works. A number of other aerodynamic design methods have been proposed, developed, and discussed [23, 36, 48, 49, 53, 58, 60, 68, 73]. Typically these methods have been developed to solve "single-discipline" design problems, that is, problems in which the cost, or objective and constraint functions, depend only upon the aerodynamic solution output. One then has the liberty to combine the optimization or iterative design variable search with the flow analysis solution(s) at several different degrees of implicitness. Generally the computational efficiency gets better with more implicitness, whereas the flexibility to handle modified problems gets worse. It appears that use of these more efficient implicit methods in MDO would require some suboptimization at the disci-

pline level or an implicit formulation of all the relevant discipline analyses. Note that one should be able to obtain the SD of the aerodynamic design from the "adjoint-related" methods. As can be seen from the cited references, there are only a handful of applications of any method to 3–D aerodynamic configurations.

Three major issues for obtaining SD from 3–D CFD codes concern: (1) the form of the the linear sensitivity equations; (2) the means for differentiating the various terms which appear; and (3) the method for solving the resulting large systems of sensitivity equations. Direct matrix solution methods have generally been used in 2–D problems; however, their use in 3–D problems appears highly unlikely as a viable approach. In [52] and [56] an incremental iterative technique for efficiently obtaining consistent, discrete aerodynamic SD for advanced CFD codes was proposed, demonstrated, and discussed. The studies concluded that: (1) the linear sensitivity equations should be cast into an incremental (correction or delta) form; (2) one should use AD or symbolic manipulation to obtain the needed derivatives; and (3) the resulting large system of sensitivity equations should be solved iteratively, using the same operator form (i.e., code) as originally used to solve the nonlinear flow equations. The incremental form allows for approximate operators of convenience (stability, convergence acceleration, simplicity, parallel processing, etc., since only convergence is required) while maintaining consistent discrete derivative solutions. The iterative solution aspect allows for extension to large 3–D problems, which presently cannot be solved by direct means because of storage and/or run-time limitations. A very brief discussion of the fundamental equations from [52] and [56] is given in the next paragraph.

The steady-state nonlinear fluid-flow equations representing conservation of mass, momentum, and energy can be written symbolically as

$$R(z, y, x_*) \;=\; 0 \quad , \tag{1}$$

where R, the residual vector at each mesh point, is a function of the local flow (state) variables z, the mesh coordinates y and the design parameters x_*. Both y and z are implicit functions of x_*. An iterative solution for the flow variables z of (1) can be written symbolically as

$$-\left(\tilde{R}_z\right)_m * (z_{m+1} - z_m) \;=\; R(z_m, y, x_*) \quad , \tag{2}$$

where m is the iteration index, R_z denotes $\partial R/\partial z$ and the tilde means some approximation of the operator. Sensitivity derivatives of various flow solutions with respect to the design parameters x_* are calculated in the QA methods as functions of dy/dx_* and dz/dx_*, denoted respectively as y' and z'. In order to remain on the solution surface, then

$$R'(z, y, x_*) \;=\; R_z z' + R_y y' + R_x \;=\; 0 \; . \tag{3}$$

It is assumed that y', the mesh sensitivity with respect to x, is obtained from the grid generation code or grid-movement algorithm [61, 64, 69]. Thus, (3) is a linear equation in z', which has been called the standard form

$$-R_z z' \;=\; R_y y' + R_x \quad . \tag{4}$$

Equation (4) can also be solved in an iterative fashion, written symbolically as

$$-\left(\tilde{R}_z\right)_m * (z'_{m+1} - z'_m) \;=\; R'_m(z_*, y, x_*). \tag{5}$$

The significance of the incremental iterative solution forms, (2) for z and (5) for z', is that the LHS operators can be approximate; the RHS, a zero at convergence, is the condition to be satisfied. Consistent discretization for the RHS of both produces consistent z and z'. Solution of the standard form (4) for consistent z' requires exactly R_z as the LHS operator. For many CFD codes, R_z is not very well conditioned and cannot be directly inverted in practice for large (i.e., 3–D) problems. In most advanced CFD codes, grid generation is not part of the code; an array of mesh coordinates, denoted before as y, is read as input. For sensitivity analysis, the potentially much larger arrays of mesh sensitivities y' also need to be obtained and given to the flow sensitivity code.

The Jacobians R_z, R_y, and R_x are generally not computed, much less identified, in most CFD codes. To obtain them "by-hand", as has been done in the past for 2–D problems, is a very tedious, time-consuming, error-prone job, hardly practical for complicated 3–D CFD codes. A more-or-less painless, automatic, and robust means to generate them is highly desirable and appears to be realizable in AD. In fact, the straightforward application of AD to the entire iterative solution process (2) generates z', without having to identify and construct the terms in (4) or (5). The next section discusses AD in all of these aspects.

4 AD of Large Programs

Automatic differentiation [59] is a chain-rule-based technique for evaluating the derivatives of functions defined by computer programs with respect to their input variables. In contrast to the approximation of derivatives

by DD, AD does not incur any truncation error so that, at least for noniterative and branch-free codes, the resulting derivative values are usually obtained with the working accuracy of the original function evaluation. In contrast to fully symbolic differentiation, both operations count and storage requirement can be *a priori* bounded in terms of the complexity of the original function code for all modes of AD. In many cases, the calculations initiated by an AD tool for the evaluation of derivatives mirror those of a carefully handwritten derivative code. A comprehensive collection on the theory, implementation, and some earlier applications can be found in the proceedings [43].

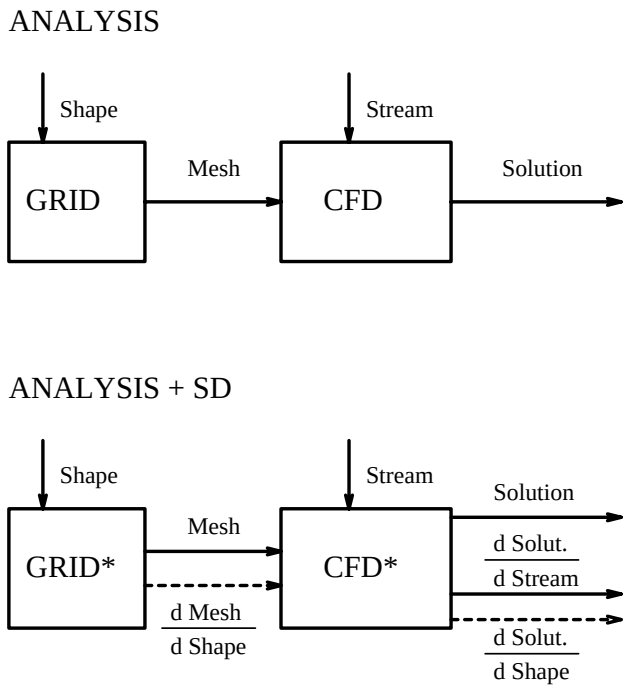
There are two basic modes of automatic differentiation, which are usually referred to as *forward* and *reverse*, respectively. The results reported in this paper were obtained with a variant of the forward mode. As discussed in [41] the reverse mode is closely related to adjoint methods and has as intriguingly low operations count for gradients. However, its potentially very large memory requirement has been a serious impediment to its application in large-scale scientific computing. When there are several independent and dependent variables the operations count for evaluating the Jacobians may be lowest for certain mixed strategies [44] rather than the forward or reverse mode. AD can also be extended for the accurate evaluation of second and higher derivatives [27, 42, 26, 17]. Second derivatives might eventually be useful for the application of higher order optimization methods in MDO. For a recent review of AD techniques and tools in the context of engineering design see [11]. An introduction to the Fortran tool ADIFOR and some preliminary numerical results on a 2–D small-disturbance model of transonic flow are given in [18].

4.1 An Advanced FORTRAN Tool

ADIFOR (Automatic Differentiation of Fortran) [15, 19, 16, 14] provides automatic differentiation for programs written in Fortran 77. Given a Fortran subroutine (or collection of subroutines) describing a "function," and an indication of which variables in parameter lists or common blocks correspond to "independent" and "dependent" variables with respect to differentiation [20], ADIFOR produces Fortran 77 code that allows the computation of the derivatives of the dependent variables with respect to the independent ones. ADIFOR employs a hybrid of the forward and reverse modes of automatic differentiation [43]. That is, for each assignment statement, code is generated for computing the partial derivatives of the result with respect to the variables on the right-hand side, and then employed in the forward mode to propagate overall derivatives. The resulting decrease in complexity compared to an entirely forward mode implementation usually is substantial.

In contrast to some earlier AD implementations [51] the source translator ADIFOR was designed from the outset with large-scale codes in mind. It uses the facilities of the ParaScope Fortran environment [24, 25] to parse the code and to extract control flow and dependence flow information. ADIFOR produces portable Fortran 77 code and accepts almost all of Fortran 77—in particular, arbitrary calling sequences, nested subroutines, common blocks, and equivalences. The ADIFOR-generated code tries to preserve vectorization and parallelism in the original code, and employs a consistent subroutine naming scheme that allows for code tuning, the use of domain-specific knowledge, and the exploitation of vendor-supplied libraries. It should be stressed that ADIFOR uses the data flow analysis information from ParaScope to determine the set of variables that require derivative information in addition to the dependent and independent ones. This approach allows for an intuitive interface, and greatly reduces the storage requirements of the derivative code.

ADIFOR-generated code can be used in various ways. Instead of simply producing code to compute the Jacobian J, ADIFOR produces code to compute $J \cdot S$, where the "seed matrix" S is initialized by the user. Therefore, if S is the identity, ADIFOR computes the full Jacobian; whereas if S is just a vector, ADIFOR computes the product of the Jacobian by a vector. "Compressed" versions of sparse Jacobians can be computed by exploiting the same graph coloring techniques [31, 30] that are used for DD approximations of sparse Jacobians. The runtime and storage requirements of the ADIFOR-generated code are roughly proportional to the number of columns of S. Hence, the computation of Jacobian-vector products and compressed Jacobians requires much less time and storage than does the generation of the full Jacobian matrix. For example, in a wing design optimization sketched below, typically only a relatively small number of geometric design variables determine the shape of the wing. On the other hand, hundreds to millions of mesh coordinates enter into the aerodynamic or structural analysis code. Provided that the grid generation process is smooth, one can determine for each mesh coordinate a comparatively short vector representing its gradient with respect to the design parameters. Declaring the mesh coordinates as the independent variables of the analysis code and initializing the rows of the seed matrix with the mesh coordinate gradients, one can run the ADIFOR-generated code to



compute the gradients of the resulting flow or displacement field with respect to the design parameters. This approach is much cheaper than first performing a full SA on the analysis code and then multiplying the resulting large Jacobian J by the matrix S representing the grid sensitivities.

Currently ADIFOR does not provide for the automatic transfer of derivative data via files. Therefore, combining the mesh-generation process and the CFD code in a multidisciplinary SA based on AD has not yet been done. When both programs are available as Fortran source, the exchange of derivative information is not very hard and will be automated in future versions of ADIFOR. However, in general the exchange of sensitivity information between single discipline codes of different origin and on various platforms will remain a difficult challenge. Another challenge stems from the tacit assumption that the outputs of all single disciplinary components depend smoothly on their input parameters. For grid-generation algorithms of eventual interest, such as adaptive unstructured grids, that assumption is probably not satisfied.

Advanced CFD codes pose several other principle challenges and uncertainties regarding the *automatic* generation of sensitivities. By far the most important difficulty is that the flow equations for all nontrivial geometries and stream conditions must be solved iteratively. The iterative flow solvers may take hundreds of steps and often involve discontinuous adjustments of solution operators, grids, shock waves, or free boundaries. The prospect of obtaining accurate solution sensitivities by simply differentiating the whole iterative process may appear dubious for multigrid methods. These are now the state of the art in 3-D CFD codes (for example [71]), despite the lack of a convergence theory under realistic assumptions. In the following section, some theoretical results from a forthcoming paper [21] are summarized; these theoretical considerations make the numerical observations of Section 5 at least plausible, even though they do not apply directly to multigrid methods in their current form.

4.2 Differentiating Implicit Functions

Large-scale codes in scientific computing frequently embody iterative solution schemes. That is, for given x_*, a nonlinear system

$$R(z, x_*) = 0 \tag{6}$$

is solved to find the value $z_* = z(x_*)$ of the function implicitly defined by R. The question is under what circumstances an AD version of the code implementing this rootfinding process computes the desired derivatives $z'_* = \frac{dz}{dx}|_{x=x_*}$. Often, iterative schemes perform discontinuous adjustments of step multipliers and preconditioners, so that the iterates themselves are very unlikely to be differentiable in the input parameters.

For the sake of discussion, assume that our iteration for solving (6) has the generic form

for $m = 1, \ldots$ **do**
  evaluate $R(z_m, x_*)$ and stop if it is small
  compute a suitable preconditioner P_m
  update $z_{m+1} = z_m - P_m\, R(z_m, x_*)$
end for

This iteration must locally converge if one can ensure that

$$||I - P_m\, R_z(z_m, x_*)|| \le \rho < 1 \tag{7}$$

The notation R_z (R_x) is shorthand for $\frac{\partial R}{\partial z}$ ($\frac{\partial R}{\partial x}$). Newton's method, for example, is a particular instance of this scheme with $P_m = (\frac{dR}{dz}|_{z=z_m})^{-1}$.

The implicit function theorem tells us that at the fixed point (z_*, x_*), one has

$$R_z z'_* + R_x = 0. \tag{8}$$

In fact, the so-called quasi-analytic (QA) approach for obtaining z' is to compute (or approximate by DD) $R_z(x_*)$

and $R_x(x_*)$ and to solve the resulting linear system (8) for z'. However, the reliability of this approach depends greatly on the conditioning of $R_z(x_*)$, as well as the accuracy of R_z and R_x. In the following discussion, a "prime" notation (such as z') always denotes total differentiation with respect to x. Applying AD to the generic iteration above, we obtain the derivative iteration

for $m = 1, \ldots$ **do**
  evaluate $R_m \equiv R(z_m, x_*)$ and
  $R'_m \equiv R_z(z_m, x_*)z'_m + R_x(z_m, x_*)$
  if (R_m and R'_m are small enough) **stop**
  compute P_m and its derivative P'_m
  update $z_{m+1} = z_m - P_m R_m$ and
  $z'_{m+1} = z'_m - P'_m R_m - P_m R'_m$
end for

Given z_m and z'_m, one can obtain the derivative residual R'_m at a cost roughly equal to that of evaluating R multiplied by the number of design parameters (i.e., components in x). In particular, this derivative evaluation does not require the calculation of the Jacobian R_z, which may contain very many elements. Note that the stopping criterion based solely on R_m has been replaced by one that also requires R'_m to be small. While it is natural to do so, an automatic tool cannot be expected to spot the stopping criterion in a potentially complicated code without some user intervention. Conceptually, one may remove the stopping criterion completely to obtain infinite sequences of iterates z_m and derivative approximations z'_m, which have been shown in [21] to converge R-linearly in that

$$\|z_m - z_*\| + \|z'_m - z'_*\| \;\sim\; \rho^m \quad \rightarrow \quad 0 \,.$$

This result was originally obtained by Gilbert [40] and Christianson [26] for the case of Newton's method and similar smooth fixed point iterations.

Recently, we have been able to extend these results (see the forthcoming paper [21]) to quasi-Newton methods, where the derivatives P'_m may grow unbounded but $P'_m R_m$ still tends to zero, because of the superlinear rate of convergence. Whenever the iterates themselves converge superlinearly there is the danger that the R-linearly convergent derivative approximations may lag behind. For such methods, it is particularly important that the stopping criterion enforce a significant reduction of $\|R'_m\|$. In large scale applications, a reasonable linear rate is often the best one can achieve, so that the asymptotic rate of convergence is likely to be the same.

For even more general preconditioners P_m, it is shown in [21] that the simple setting $P'_m = 0$ ensures convergence to the desired derivative value z'_* at the same R-linear rate, provided condition (7) is satisfied. In contrast to the previous *black-box* approach, the preconditioner P_m is treated here as a constant and hence the term $P'_m R(z_m, x_*)$ is dropped in the update of z'_{m+1}. This approach makes intuitive sense since in the end $R(z_m, x_*)$ will converge to zero anyway, thereby annihilating any contribution of P'_m. Also, P'_m is likely to involve higher derivatives that (according to the implicit function theorem) play no role in the existence of z'_*. This latter procedure is the incremental iterative form of equation (5). The implications of this observation for the speed of derivative computations are noteworthy. For example, in a Newton iteration, one saves the work of differentiating through the matrix factorization process, which is by far the dominant work of the iteration process. Exploitation of this result does require some user intervention to indicate stopping criteria and variables containing preconditioners. Depending on code modularity, this may or may not be easy to do. We are experimenting with "deactivation" concepts that would support the user in this task. Techniques such as this one, which build on AD techniques but require some understanding of the code, we call *computational differentiation* techniques.

Another point worth mentioning is that it does not make sense to start the derivative iterations until the iterations for $R(z, x_*) = 0$ have essentially converged. Obviously, the derivatives z'_* will not settle in until the "function value" z_* itself has. Again, this is not automatic and requires user intervention, but the potential savings are significant. It is important to recognize that the cost of evaluating R'_m for a given z'_m is the same whether the z_m have converged to z_* or not. Since $R_z(z, x_*)$ is not explicitly formed, one cannot exploit its constancy when $z = z_m = z_*$ for several iterations.

5 Application of AD to CFD

Numerical results reported here show that even the naive application of ADIFOR to multigrid solvers viewed as a black-box program can produce accurate sensitivity information at tolerable costs. In fact, all calculated derivatives could be reproduced with several digits agreement by carefully evaluated DD. Moreover, the implicit function theorem yields a constructive test on the accuracy of the derivative approximation, which suggests that, in most cases, at least six digits were correct. In the 2–D quasi-analytical SD code [52], the turbulence model was deemed too complicated for differentiation by hand; its treatment as constant led to sizable relative errors in some

resulting global sensitivities. Nevertheless, it was found here that for a Baldwin-Lomax model the results produced by ADIFOR agree quite well with DD and therefore convey meaningful sensitivity information even for the turbulent case.

5.1 2-D Transonic Small Disturbance

The first iterative code to which ADIFOR was applied for evaluating derivatives of an implicitly defined function was the 2–D transonic small disturbance code TAMRF of Elbana and Carlson [37]. The grid used by that code is stretched Cartesian and does not change with shape variations; such modifications are reflected in the flow boundary conditions. Therefore, in regard to the mechanics of ADIFOR, derivatives with respect to both shape (geometric) and stream (nongeometric) parameters could be treated exactly the same. This particular code was selected because it had been differentiated "by hand" to obtain SD and because it had many characteristics typical of nonlinear CFD codes. All of the flow cases reported in [37] have been recomputed to obtain SD via one-sided DD at 10^{-6} independent variable increments and via AD using the generic derivative iteration.

Table 1: Computation Time for Five 2-D SD

NACA 1406 Airfoil; TSDE Inviscid Flow				
Solution Procedure	Number Solutions	Time (sec)*		
		Subs.	Trans.	Supers.
DD	6 Nonlinear	13	36	64
QA	1 Nonlinear, 5 Linear	$40^{\dagger}$	$38^{\dagger}$	$37^{\dagger}$
AD	1 Nonlinear, 5 Linear	$12^{\#}$	$25^{\#}$	$35^{\#}$

* All calculations performed on a CRAY Y-MP.
† Average time for several matrix solution methods.
\# $\|R'_m\|$ converged below 10^{-3}.

Table 1 shows typical timing results for five SD (C_L with respect to M,α, maximum airfoil thickness, maximum camber and its location) for subsonic, transonic, and supersonic flow cases. In all these cases AD and DD agreed to more than four significant digits. For subsonic and transonic flows this agreement extended also to the results obtained by the original QA method of Elbanna and Carlson.

In Figure 1, three convergence history plots for the norms of the residuals R_m and their total derivatives R'_m with respect to the five parameters mentioned above are shown. In all three subsonic, transonic, and supersonic cases, the code was run without any derivative calculations until the norm of the residual R_m was reduced below 10^{-7} times its initial value. The resulting base-line solutions were then used as initial points for a second phase of the iterative process, which was fully differentiated in the black-box sense discussed in the previous section. For these numerical experiments, the iterations were continued until the norm of the derivative residual R'_m was reduced below 10^{-8}. In all three cases it appears that the asymptotic rate of convergence for the R'_m is very close to that for the R_m and that the latter have been driven down to the noise level. Some studies have been made on the accuracy achieved as a function of the convergence of R' (initially about 10^2). These results are shown in Table 2. It is seen that the AD time relative to the DD time is still small for rather good SD agreement.

Table 2: Computing Time and Accuracy for Five 2-D SD

NACA 1406 Airfoil; TSDE Inviscid Transonic Flow						
AD Converg.†	10^{+1}	10^{0}	10^{-1}	10^{-2}	10^{-3}	10^{-4}
AD/DD Agreement#	2	3	4	4^{+}	4^{+}	4^{+}
AD Time/ DD time	0.18	0.26	0.36	0.50	0.69	0.92
QA Time/ DD Time	1.08	1.08	1.08	1.08	1.08	1.08

† Norm of $R' = dR/dx$, # Number of significant digits.

5.2 3-D Thin Layer Navier-Stokes

The 3–D thin-layer Navier-Stokes code TLNS3D [71] employs a multigrid acceleration technique to an explicit multi-stage Runge-Kutta time-stepping scheme with central spatial differencing to efficiently obtain steady state high Reynolds number turbulent flow solutions. It has been used successfully in a number of applications across the flight speed range from low subsonic to hypersonic and for a number of flight vehicle types. Its forthcoming multiblock version [72] promises the flexibility needed for modeling complex geometric configurations. Initial work has been reported [57, 67] on implementing these codes on parallel processors. All of these facets tend to enhance its usefulness in applications to real engineering solutions and thus MDO problems. Obtaining consistent SD information is therefore of genuine interest and straightforward application of ADIFOR appeared to be the most

direct route for obtaining it. Moreover, the multigrid algorithm operator, used to obtain the flow solution variables z, is an incremental iterative form.

Application of ADIFOR to the entire iterative solution algorithm of the TNLS3D code alone yielded sensitivities with respect to the (nongeometric) stream variables M, α, and Re. The output functions chosen to be differentiated were C_L, C_D, and C_M. The primary concern about the SD was accuracy; i.e., could AD properly handle the multigrid algorithm and turbulence models? Of secondary importance, at least for this initial study, were the memory and runtime requirements of the derivative code generated by ADIFOR. The TLNS3D code is a highly vectorized code and this aspect contributes greatly to its overall computational efficiency. ADIFOR inserts loops of length equal to the number of design parameters to compute "gradient objects" for each intermediate involved in the function evaluation. Without post-ADIFOR processing, these inner-most short loops prevent the longer outer loops from being pipelined and thus lead to an inordinately long runtime. Minor code changes and a statement specifying the short vector length allowed the Cray compiler to automatically unroll these loops. However, apparently due to the volume of code added by ADIFOR, there is still some loss of vectorization efficiency, which is currently being investigated. Nevertheless, an executable Fortran derivative code was obtained that converged the SD adequately, even though it still runs much slower than anticipated.

Table 3: Accuracy for Six 3–D SD

ONERA M-6 Wing, TLNS3D Inviscid Transonic Flow, 3-Level Multigrid, $97 \times 25 \times 17$ ($M = 0.84$, $\alpha = 3.06°$)

(a) Dead Start: $\left(\frac{R_{800}}{R_1}\right)_{DD} \sim 10^{-5}$; $\left(\frac{R_{500}}{R_1}\right)_{AD} \sim 10^{-4}$ and $\left(\frac{R'_{500}}{R'_1}\right)_{AD} \sim 10^{-3}$

D_{AD}/D_{DD}	C_L	C_D	C_M
M	0.9999	0.9992	0.9993
α	0.9995	0.9993	0.9992

(b) Restart File: $\left(\frac{R_{1800}}{R_1}\right)_{DD} \sim 10^{-11}$; $\left(\frac{R_{1375}}{R_1}\right)_{AD} \sim 10^{-9}$ and $\left(\frac{R'_{575}}{R'_1}\right)_{AD} \sim 3 \times 10^{-5}$

D_{AD}/D_{DD}	C_L	C_D	C_M
M	1.0003	1.0000	1.0002
α	0.9999	0.9999	0.9999

Initial accuracy tests were run on coarse grids to ascertain that the differentiated code could be run from a dead start as well as from a restart file created by a preliminary TLNS3D run. This two stage procedure represents the delayed derivative calculation paradigm discussed in Section 4. It was also determined that the derivative code generated in a single pass through the ADIFOR successfully executed with the inviscid, laminar and turbulent options. Subsequent detailed verification of the AD results was done on larger grids ($\approx 97 \times 25 \times 17$); by current CFD standards for 3–D configurations, this is still a coarse grid.

A simple wing configuration, the ONERA M-6, was used in these initial AD verification studies and a sample C-O mesh about it ($25 \times 9 \times 9$ for clarity) is depicted in Figure 2. For the inviscid (Euler) mode, six of the nine nongeometric SD are nonzero. Comparisons of the agreement between the AD and DD (one-sided at increments of 10^{-6} times the input value) SD produced are shown in Table 3 as ratios D_{AD}/D_{DD}. This transonic flow case is for $M = 0.84$ and $\alpha = 3.06°$. The results in Table 3(a) are for a dead start of both the original TLNS3D (for DD calculations) and the differentiated version (for AD calculations). The convergence levels obtained at the indicated number of multigrid iteration cycles is shown. It can be seen that about 3 significant digits agreement is obtained. In Table 3(b) similar results are shown with both codes

Table 4: Accuracy for Nine 3-D SD

ONERA M-6 Wing, TLNS3D Laminar Subsonic Flow, 2-Level Multigrid, $97 \times 17 \times 17$ ($M = 0.2, Re = 5000$)
(a) Zero Angle-of-Attack

D_{AD}/D_{DD}	C_L	C_D	C_M
M	*	1.0004	*
α	1.0000	*	1.0000
Re	*	1.0000	*

*Numerator of D_{DD} is noise; $D_{AD} \sim O(10^{-8})$ and for DD stepsize of 10^{-6}, numerator of $D_{DD} \sim O(10^{-14})$

(b) One degree Angle-of-Attack

D_{AD}/D_{DD}	C_L	C_D	C_M
M	1.0028	0.9998	0.9985
α	1.0000	1.0001	1.0000
Re	1.0000	1.0000	1.0001

run from an original TLNS3D baseline restart file. Again the convergence levels obtained at the indicated number of multigrid operations is shown. Agreement to essentially 4 significant digits is obtained.

The relative accuracy for all nine nongeometric SD at subsonic laminar flow conditions, $M = 0.2$ and $Re = 5000$, are shown in Table 4. For $\alpha = 0°$, the resulting symmetric flow produces some very small SD which for the DD are only noise. However, the larger SD are seen from Table 4(a) to agree very well. The results for $\alpha = 1°$ are shown in Table 4(b). Here again the agreement is very good.

Table 5: Accuracy for Nine 3-D SD

ONERA M-6 Wing, TLNS3D Turbulent Transonic Flow, 3-Level Multigrid, $97 \times 25 \times 17$
($M = 0.84$,$\alpha = 3.06°$,$Re = 11.7 \times 10^6$)

(a) Mixing-length model

D_{AD}/D_{DD}	C_L	C_D	C_M
M	1.0000	1.0000	0.9999
α	1.0000	1.0000	1.0000
Re	1.0007	1.0000	1.0012

(b) Baldwin-Lomax model

D_{AD}/D_{DD}	C_L	C_D	C_M
M	1.0000	1.0000	1.0000
α	1.0000	1.0000	1.0000
Re	0.9991	1.0000	0.9961

(c) Convergence

$$\left(\frac{R_{1700}}{R_1}\right)_{DD} \sim 10^{-12}\ ;$$
$$\left(\frac{R_{1300}}{R_1}\right)_{AD} \sim 5 \times 10^{-12} \text{ and } \left(\frac{R'_{425}}{R'_1}\right) \sim 10^{-4}$$

Similar relative accuracy results at transonic turbulent flow conditions, $M = 0.84, \alpha = 3.06°$, and $Re = 11.7 \times 10^6$, are shown in Table 5. In Table 5(a) results are compared for the simple differentiable mixing-length turbulence model [46], whereas in Table 5(b), results for the Baldwin-Lomax turbulence model [46] are compared. In both cases 3 to 4 significant digits agreement between the AD and DD results is obtained. The number of multigrid iterations and convergence levels for both models is shown Table 5(c). An indication of the flow field resolution obtained on this $(97 \times 25 \times 17)$ grid is displayed in Figure 3, which shows a wing upper surface pressure contour plot. It can be seen that both the swept leading-edge shock and the almost normal wing-volume shock near the mid chord are smeared out, as one would expect from central-difference operators on such grids. It is not clear how much effect such shock smearing has on these derivative comparisons which have been presented; its effect is surely favorable though.

Table 6: Computing Time and Memory for Nine 3-D SD

ONERA M-6 Wing, TLNS3D Turbulent Transonic Flow, Baldwin-Lomax, 3-Level Multigrid, $97 \times 25 \times 17$

Solution Procedure	Number Solutions	Time (sec*)	Storage (MW)
DD one-sided	4 Nonlinear	3960	2.47
DD central	6 Nonlinear	5940	2.47
AD	1 Nonlinear, 3 Linear	10290	7.63

* All calculcations performed on CRAY Y-MP.

Convergence histories for the subsonic laminar flow (at $\alpha = 1°$) and the transonic turbulent (Baldwin-Lomax model) flow cases are presented in Figure 4. Here the relative residuals for both the original and differentiated (AD) TLNS3D codes are plotted versus work, which, for a multigrid algorithm, is taken as a unit roughly equivalent to the computational work of one iteration on the finest grid. As can be seen the subsonic laminar results are not too smooth; perhaps the flow would be seen to be unsteady on a finer grid. The delayed derivative evaluation paradigm has been used and, as can be seen, the derivative code solution (started from an original code restart file) commences just beyond 1000 work units. These are the residual histories for the accuracy results given in Tables 4(b) and 5(b). An indication of the computational time and memory requirements for the derivative code in its current form compared with those for the original code can be seen in Table 6. These are the runtime statistics for the results shown in Table 5(b) and Figure 4. The AD-version code requires about 3 times the memory and 2.5 times more runtime than that needed using the original code and DD for the problem considered. However, these are simply initial results and little has yet been done to refine the iterative derivative paradigm with regard to just what convergence levels are required. That is, what is necessary for the residual R convergence level for the original code baseline solution on the restart file and also

that for R' in the derivative code?

An indication of the derivative accuracy as a function of the derivative residual R' convergence level for the 3-D TLNS3D code is given in Table 7. As can be seen, for agreement to 2 significant digits, the AD runtime is essentially equal to the DD runtime. The accuracy results given in Table 5(b) are the last column in Table 7. It appears from comparing Tables 2 and 7 that the number of significant-digit agreement versus the convergence level of R' is essentially the same for the 2-D TSDE code and the TLNS3D code.

Table 7: Computing Time and Accuracy for Nine 3-D SD

ONERA M-6 Wing, TLNS3D Turbulent Transonic Flow, Baldwin-Lomax, 3-Level Multigrid, $97 \times 25 \times 17$

AD Convergence†	10^{+1}	10^{0}	10^{-1}	$10^{-1.5}$
AD/DD Agreement#	2	3	4	4^{+}
AD Time/DD Time	1.01	1.45	2.07	2.59

† Norm of $R' = dR/dx$, # Number of significant digits.

It appears that the newest version of ADIFOR is very easy to use. After a few examples, the NASA personnel found that ADIFOR could be applied to a code in a matter of days. Verification of the resulting derivatives by DD, however, was often much more time consuming. As the AD technology matures this extra effort will no longer be necessary.

6 Conclusion and Challenges

Computational differentiation of an advanced CFD code employing ADIFOR in order to obtain SD of output flow properties with respect to nongeometric input variables has been quantitatively demonstrated. This is a very significant and encouraging result for several reasons:

(a) The TLNS3D code is an efficient, complex, state-of-the-art 3–D CFD code.

(b) The computational efficiency of TLNS3D is based upon the rather delicate multigrid acceleration algorithm and the successful application of ADIFOR yielded similar convergence rates for the SD.

(c) ADIFOR also successfully differentiated the TLNS3D Baldwin-Lomax turbulence model.

(d) Accurate SD were obtained using AD in a relatively short lead time (O(man-week)), essentially treating ADIFOR and TLNS3D as black boxes.

While it has been shown that AD can be successfully applied to advanced CFD codes for nongeometric SD, the procedures and results need to be improved before sensitivity information on high resolution meshes can be obtained. Also, the SD reported here were restricted to a single discipline, namely the CFD calculation. However, experiments applying AD to the combination of the mesh generation process and the flow analysis are under way and preliminary results are encouraging. This interaction must be achieved in order to perform the geometric SD, which is of primary interest to MDO.

It should be stressed that, from a purely mathematical point of view, the differentiation of iterative processes does not seem to be a problem, despite the fact that the assumptions of known derivative convergence theorems have not been verified for the small disturbance code and are almost certainly not satisfied by multigrid algorithms. Since, in the latter case, not even the convergence of the iterates themselves has been proven under reasonably general assumptions, attempts to prove the convergence of their derivatives seem premature. A comparatively simple, but application and platform dependent, task is the choice of criteria in the iterative paradigm for the transition from the undifferentiated iteration to the more costly final stage, where derivative information is carried along. As our theoretical studies and numerical experiments indicate, one may assume that both solutions and derivatives converge at about the same rate once the iteration has settled down. This is of significance in design optimization calculations since the objective function and its gradient need be obtained with high accuracy only in the vicinity of the optimal design. Thus, great savings are possible through less accurate evaluations in the earlier part of the optimization.

A second goal is to avoid the unnecessary differentiation of preconditioners and other intermediates that affect only the solution process but not the solution function and its derivatives. Unless the original code is appropriately structured, "deactivating" such intermediates "by-hand" is a difficult task. However, the resulting simplified derivative calculation should be as efficient as the incremental iterative form of the QA method. Therefore, an investigation will be made to determine if and how ADIFOR can automatically perform deactivation with a minimum of directives from the user or programmer.

A third goal is improved vectorization and parallelism of the derivative code, so that their runtime is at worst equal to that of the original code multiplied by the num-

ber of design parameters. For standard test problems [9] ADIFOR achieves and undercuts this bound regularly on scalar and super-scalar chips but, as in the case of TLNS3D on a CRAY Y-MP, the situation is currently much less favorable on vector machines.

It may actually be simpler to maintain efficiency for the ADIFOR generated versions of parallel CFD codes. However, as in the interdisciplinary case it becomes necessary to pass derivative objects with intermediate data through I/O statements and interprocessor messages. While the size of the data transfers is multiplied by the number of design variables, the data flow structure between various tasks is preserved so that parallelism is essentially unchanged. The ADIFOR added code can interfere with the inner loop vectorization whereas it is not seen in the outer loop parallelization.

Acknowledgements

We would like to thank Alan Carle of Rice University and Paul Hovland of Michigan State University for many stimulating discussions and their essential roles in the ADIFOR development project. The authors are also deeply indebted to Veer Vatsa of NASA Langley for numerous useful and informative discussions concerning the TLNS3D code.

References

[1] H. M. Adelman and R. T. Haftka, editors. *Proceedings of the Symposium on Sensitivity Analysis in Engineering, NASA Langley Research Center, Hampton, VA, Sept. 1986.* NASA CP–2457, 1987.

[2] AGARD. *Computational Methods for Aerodynamic Design (Inverse) and Optimization, Loen, Norway.* AGARD–CP–463, May 1989.

[3] *Twenty-Ninth AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, A Collection of Technical Papers. Williamsburg, VA.* AIAA CP–882, Apr. 1988.

[4] *Thirtieth AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, A Collection of Technical Papers. Mobile, AL.* AIAA CP–891, Apr. 1989.

[5] *Thirty-First AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, A Collection of Technical Papers. Long Beach, CA.* AIAA CP–902, Apr. 1990.

[6] *Thirty-Second AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, A Collection of Technical Papers. Baltimore, MD.* AIAA CP–911, Apr. 1991.

[7] *Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, A Collection of Technical Papers, Cleveland, OH.* B B B k AIAA CP–9213, Sept. 1992.

[8] *Thirty-Third AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, A Collection of Technical Papers. Dallas, TX.* AIAA CP–922, Apr. 1992.

[9] B. Averick, R. G. Carter, and J. J. Moré. The MINPACK-2 test problem collection (preliminary version). Technical Report ANL/MCS–TM–150, Mathematics and Computer Science Division, Argonne National Laboratory, 1991.

[10] J. F. Barthelemy, editor. *Second NASA/Air Force Symposium on Recent Advances in Multidisciplinary Analysis and Optimization, Hampton, VA, Sept. 1989.* NASA CP–3031, 1989.

[11] J. F. Barthelemy, and L. Hall. Automatic differentiation as a tool in engineering design. In *Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, A Collection of Technical Papers, Cleveland, OH*, pages 424–432. AIAA 92–4743–CP, Sept. 1992.

[12] O. Baysal, editor. *Symposium on Multidisciplinary Applications of Computational Fluid Dynamics.* FED–Vol. 129, ASME Winter Annual Meeting, Dec. 1991.

[13] O. Baysal, M. E. Eleshaky, and G. W. Burgreen. Aerodynamic shape optimization using sensitivity analysis on third-order Euler equations. In *Proceedings of the AIAA Tenth Computational Fluid Dynamics Conference*, pages 573–583. AIAA 91-1577–CP, June 1991.

[14] C. H. Bischof, A. Carle, G. F. Corliss, and A. Griewank. ADIFOR: Automatic differentiation in a source translator environment. In Paul Wang, editor, *International Symposium on Symbolic and Algebraic Computing 92*, pages 294–302. ACM, Washington, D.C., 1992.

[15] C. H. Bischof, A. Carle, G. F. Corliss, A. Griewank, and P. Hovland. ADIFOR: Generating derivative

codes from Fortran programs. *Scientific Programming*, 1(1):1–29, 1992.

[16] C. H. Bischof, G. F. Corliss, and A. Griewank. ADIFOR exception handling. ADIFOR Working Note #3, MCS–TM–159, Mathematics and Computer Science Division, Argonne National Laboratory, 1991.

[17] C. H. Bischof, G. F. Corliss, and A. Griewank. Computing second- and higher-order derivatives through univariate Taylor series. ADIFOR Working Note #6, MCS–P296–0392, Mathematics and Computer Science Division, Argonne National Laboratory, 1992.

[18] C. H. Bischof and A. Griewank. ADIFOR: A Fortran system for portable automatic differentiation. In *Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, A Collection of Technical Papers, Cleveland, OH*, pages 433–441. AIAA 92–4744–CP, Sept. 1992.

[19] C. H. Bischof and P. Hovland. Using ADIFOR to compute dense and sparse Jacobians. ADIFOR Working Note #2, MCS–TM–158, Mathematics and Computer Science Division, Argonne National Laboratory, 1991.

[20] C. H. Bischof, A. Carle, G. Corliss, A. Griewank, and P. Hovland. Getting started with ADIFOR. ADIFOR Working Note #9, ANL–MCS–TM–164, Mathematics and Computer Science Division, Argonne National Laboratory, 1992.

[21] C. H. Bischof, A. Carle, G. F. Corliss, J. Dennis, A. Griewank, and K. Williamson. Derivative convergence for iterative equation solvers. ANL–MCS–P333–1192, Mathematics and Computer Science Division, Argonne National Laboratory, 1992.

[22] G. Burgreen, O. Baysal, and M. Eleshaky. Improving the efficiency of aerodynamic shape optimization procedures. In *Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, A Collection of Technical Papers, Cleveland, OH*, pages 87–97. AIAA 92–4697–CP, Sept. 1992.

[23] H. Cabuk, C.-H. Sung, and V. Modi. Adjoint operator approach to shape design for internal incompressible flows. In *Proceedings of Third International Conference on Inverse Design Concepts and Optimizations in Engineering Sciences ICIDES–III, Washington, DC*, pages 391–404. Oct. 1991.

[24] D. Callahan, K. Cooper, R. T. Hood, K. Kennedy, and L. M. Torczon. ParaScope: A parallel programming environment. *International Journal of Supercomputer Applications*, 2(4), Dec. 1988.

[25] A. Carle, K. D. Cooper, R. T. Hood, K. Kennedy, L. Torczon, and S. K. Warren. A practical environment for scientific programming. *IEEE Computer*, 20(11):75–89, Nov. 1987.

[26] B. D. Christianson. Reverse accumulation and accurate rounding error estimates for Taylor series coefficients. *Optimization Methods and Software*, 1(1):81–94, 1992.

[27] B. D. Christianson. Automatic Hessians by reverse accumulation. Technical Report NOC TR228, The Numerical Optimisation Center, Hatfield Polytechnic, Hatfield, U.K., Apr. 1990.

[28] P. G. Coen. Recent results from the high-speed airframe integration research project. AIAA Paper 92–4717, Sept. 1992.

[29] P. G. Coen, J. S-. Sobieski, and S. Dollyhigh. Preliminary results from the high-speed airframe integrated research project. AIAA Paper 92–1004, Feb. 1992.

[30] T. F. Coleman, B. S. Garbow, and J. J. Moré. Software for estimating sparse Jacobian matrices. *ACM Trans. Math. Software*, 10:329 – 345, 1984.

[31] T. F. Coleman and J. J. Moré. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM Journal on Numerical Analysis*, 20:187 – 209, 1984.

[32] S. Dollyhigh and J. S-. Sobieski. Recent experience with multidisciplinary analysis and optimization in advanced aircraft design. In *Third Air Force/NASA Symposium on Recent Advances in Multidisciplinary Analysis and Optimization. A Collection of Technical Papers, San Francisco, CA*, pages 404–411. Sept. 1990.

[33] M. Drela. Viscous and inviscid schemes using Newton's method. In *Special Course on Inverse Methods for Airfoil Design for Aeronautical and Turbomachinery Applications*, pages 9-1 to 9-16. AGARD Report No. 780, May1990.

[34] G. S. Dulikravich, editor. *Proceedings Second International Conference on Inverse Design Concepts and Optimization in Engineering Sciences ICIDES–II, University Park, PA*. Oct. 1987.

[35] G. S. Dulikravich, editor. *Proceedings, Third International Conference on Inverse Design Concepts and Optimization in Engineering Sciences ICIDES–III, Washington, DC.* Oct. 1991.

[36] G. S. Dulikravich. Aerodynamic shape design and optimization. AIAA Paper No. 91–0476, Jan. 1991.

[37] H.M. Elbanna and L.A. Carlson. Determination of aerodynamic sensitivity coefficients based on the small perturbation formulation. *J. Aircraft*, 27(6): 507–515, 1990. (Also appeared as AIAA Paper 89–0532.)

[38] H. M. Elbanna and L. A. Carlson. Determination of aerodynamic sensitivity coefficient based on the three-dimensional full potential equation. In *Proceeding of the AIAA 10th Applied Aerodynamics Conference*, pages 539–548. AIAA 92–2670–CP, June 1992.

[39] M. Eleshaky and O. Baysal. Aerodynamic shape optimization
via sensitivity analysis on decomposed computational domains. In *Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, A Collection of Technical Papers, Cleveland, OH*, pages 98–109. AIAA 92–4698–CP, Sept. 1992.

[40] Jean-Charles Gilbert. Automatic differentiation and iterative processes. *Optimization Methods and Software*, 1(1):13–22, 1992.

[41] A. Griewank. On automatic differentiation. In M. Iri and K. Tanabe, editors, *Mathematical Programming: Recent Developments and Applications*, pages 83–108. Kluwer Academic Publishers, 1989.

[42] A. Griewank. Automatic evaluation of first- and higher-derivative vectors. In R. Seydel, F. W. Schneider, T. Küpper, and H. Troger, editors, *Proceedings of the Conference at Würzburg, Aug. 1990, Bifurcation and Chaos: Analysis, Algorithms, Applications*, volume 97, pages 135–148. Birkhäuser Verlag, Basel, Switzerland, 1991.

[43] A. Griewank and G. F. Corliss, editors. *Automatic Differentiation of Algorithms: Theory, Implementation, and Application.* SIAM, Philadelphia, PA, 1991.

[44] A. Griewank and S. Reese. On the calculation of Jacobian matrices by the Markowitz rule. In A. Griewank and G. F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 126–135. SIAM, Philadelphia, PA, 1991.

[45] T. L. Holst, M. D. Salas, and R. W. Claus. The NASA computational aerosciences program – toward teraflop computing. AIAA Paper 92–0558, Jan. 1992.

[46] C. Hirsch. *Numerical Computation of Internal and External Flows, Computational Methods for Inviscid and Viscous Flows*, Vol. 2, Section 22.3.1, John Wiley & Sons, 1990.

[47] G. J-.W. Hou, A. C. Taylor III, and V. M. Korivi. Discrete shape sensitivity equations for aerodynamic problems. AIAA Paper No. 91–2259, June 1991.

[48] A. Jameson. Aerodynamic design via control theory, *Journal of Scientific Computing*, 3:233–260, 1988. (Also NASA CR–181749 or ICASE Report No. 88–64, Nov. 1988).

[49] A. Jameson. Automatic design of transonic airfoils to reduce the shock induced pressure drag. Princeton University MAE Report 1881, 1990. (Also appeared in *31st Israel Annual Conference in Aviation and Aeronautics*, Feb. 1990).

[50] H. Jones, G. Hou, M. Korivi, A. Taylor III, and P. Newman. Multidisciplinary analysis and sensitivity derivatives for isolated helicopter rotors in hover. In *Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, A Collection of Technical Papers, Cleveland, OH*, pages 63–86. AIAA 92–4696–CP, Sept. 1992.

[51] D. Juedes. A taxonomy of automatic differentiation tools. In A. Griewank and G. F. Corliss, editors, *Proceedings of the Workshop on Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 315–329. SIAM, Philadelphia, PA, 1991.

[52] V. M. Korivi, A. C. Taylor III, P. A. Newman, G. J.-W. Hou, and H. E. Jones. An approximately factored incremental strategy for calculating consistent discrete CFD sensitivity derivatives. In *Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, A Collection of Technical Papers, Cleveland, OH*, pages 465–478. AIAA 92–4746–CP, Sept. 1992.

[53] J. B. Malone and R. C. Swanson. Inverse airfoil design procedure using a multigrid Navier-Stokes

method. In *Proceedings, Third International Conference on Inverse Design Concepts and Optimization in Engineering Sciences ICIDES–III, Washington, DC*, pages 55–66. Oct. 1991.

[54] NASA. *Third Air Force/NASA Symposium on Recent Advances in Multidisciplinary Analysis and Optimization. A Collection of Technical Papers, San Francisco, CA.* Sept. 1990.

[55] NASA. *Computational Aerosciences Conference, Compendium of Abstracts.* NASA Ames Research Center, Moffett Field, CA, Aug. 1992.

[56] P. A. Newman, G. J.-W. Hou, H. E. Jones, A. C. Taylor, and V. M. Korivi. Observations on computational methodologies for use in large-scale, gradient-based, multidisciplinary design incorporating advanced CFD codes. In *Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, A Collection of Technical Papers, Cleveland, OH*, pages 531–542. AIAA 92–4753–CP, Sept. 1992.

[57] D. Olander and R. B. Schnabel. Preliminary experience in developing a parallel thin-layer Navier Stokes code and implications for parallel language design. In *Proceedings, Scalable High Performance Computing Conference, SHPCC-92*, pp. 276-283, Williamsburg, VA, Apr. 1992.

[58] C. Orozco and O. Ghattas. Optimal design of systems governed by nonlinear partial differential equations, In *Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, A Collection of Technical Papers, Cleveland, OH*, pages 1126–1140. AIAA 92–4836–CP, Sept. 1992.

[59] L. B. Rall. *Automatic Differentiation: Techniques and Applications*, volume 120 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, Germany, 1981.

[60] M. H. Rizk. Optimization by updating design parameters as CFD iterative flow solutions evolve. In O. Baysal, editor, *Symposium on Multidisciplinary Applications of Computational Fluid Dynamics*, pages 51–62. FED–Vol. 129, ASME Winter Annual Meeting, Dec. 1991.

[61] I. Sadrehaghighi, R. E. Smith Jr., and S. N. Tiwari. An analytical approach to grid sensitivity analysis. AIAA Paper 92–0660, Jan. 1992.

[62] G. R. Shubin. Obtaining "cheap" optimization gradients from computational aerodynamics codes. Applied Mathematics and Statistics Technical Report AMS–TR–164, Boeing Computer Services, June 1991.

[63] G. R. Shubin and P. D. Frank. A comparison of two closely-related approaches to aerodynamic design optimization, In *Proceedings of the Third International Conference on Inverse Design Concepts and Optimizations in Engineering Sciences ICIDES-III, Washington. DC*, pages 67–78. Oct. 1991. (Also appeared as Boeing Computer Services Technical Report AMS–TR–163, April, 1991).

[64] R. E. Smith Jr., and I. Sadrehaghighi. Grid sensitivity in airplane design, In *Proceedings of the 4th International Symposium of Computational Fluid Dynamics, University of California-Davis*, pages 1071–1077. Sept. 1991.

[65] J. S-. Sobieski. Multidisciplinary optimization for engineering systems: Achievements and potential. NASA TM 101566, NASA, Mar. 1989.

[66] T. Sorenson. Airfoil optimization with efficient gradient calculations. In *Proceedings of the Third International Conference on Inverse Design Concepts and Optimizations in Engineering Sciences ICIDES-III, Washington. DC*, pages 433–444. Oct. 1991.

[67] A. Sussman, J. Saltz, R. Das, S. Gupta, D. Mavriplis, R. Ponnusamy, and K. Crowley. PARTI primitives for unstructured and block structured problems, NASA CR 189662, June 1992.

[68] S. Táasan, G. Kuruvila, and M. D. Salas. Aerodynamic design and optimization in one shot. AIAA Paper No. 92–0025, Jan. 1992.

[69] A. C. Taylor III, G. J-.W. Hou, and V. M. Korivi. A methodology for determining aerodynamic sensitivity derivatives with respect to variation of geometric shape. AIAA Paper No. 91–1101, Apr. 1991.

[70] A. C. Taylor III, G.J-.W. Hou, and V. M. Korivi. Sensitivity analysis, approximate analysis, and design optimization for internal and external flows. AIAA Paper No. 91–3083, Sept. 1991.

[71] V. N. Vatsa and B. W. Wedan. Development of a multigrid code for 3–D Navier-Stokes equations and its application to a grid-refinement study, *Computers & Fluids*, 18(4):391-403, 1990.

[72] V. N. Vatsa, M. D. Sanetrik, and E. B. Parlette. Development of a flexible and efficient multigrid-based multiblock flow solver, To be presented at AIAA 31st Aerospace Sciences Meeting, Reno NV, AIAA 93-0677, Jan. 1993.

[73] N. J. Yu and R. C. Campbell. Transonic airfoil and wing design using Navier-Stokes codes, In *Proceedings AIAA 10th Applied Aerodynamics Conference*, pages 477-485, AIAA 92–2651–CP, June 1992.

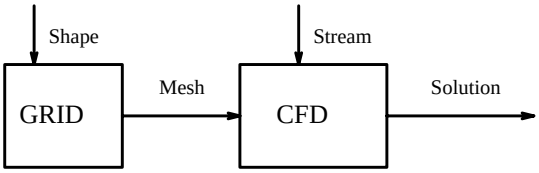
Figure 1: Iteration convergence histories for subsonic, transonic, and supersonic inviscid flow solutions from the 2–D transonic small disturbance equation code TAMRF for an NACA 1406 airfoil at $\alpha = 1°$.

Figure 2: Symmetry and back plane traces of the 3-D C-O mesh ($25 \times 9 \times 9$) produced by the grid generation code WTCO about an ONERA M-6 wing.

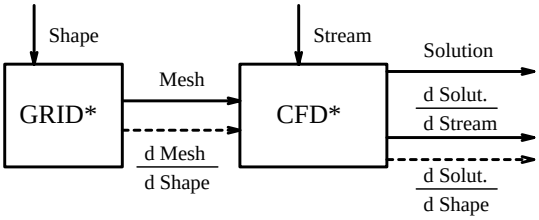
Figure 3: Upper surface pressure contour plot on an ONERA M-6 wing for TLNS3D transonic turbulent flow solution at $M = 0.84, \alpha = 3.06°, Re = 11.7 \times 10^6$ with Baldwin-Lomax model on $97 \times 25 \times 17$ computational mesh.

Figure 4: Iteration convergence histories for 3-D subsonic laminar and transonic turbulent flow solutions from the TLNS3D code for an ONERA M-6 wing.

ANALYSIS



ANALYSIS + SD



* = original code + AD added code

Table 1: Computation Time for Five 2-D SD

NACA 1406 Airfoil; TSDE Inviscid Flow				
Solution Procedure	Number Solutions	Time (sec)*		
		Subs.	Trans.	Supers.
DD	6 Nonlinear	13	36	64
QA	1 Nonlinear, 5 Linear	40†	38†	37†
AD	1 Nonlinear, 5 Linear	12#	25#	35#

* All calculations performed on a CRAY Y-MP.
† Average time for several matrix solution methods.
# $\|R'_m\|$ converged below 10^{-3}.

Table 2: Computing Time and Accuracy for Five 2-D SD

NACA 1406 Airfoil; TSDE Inviscid Transonic Flow						
AD Converg.†	10^{+1}	10^{0}	10^{-1}	10^{-2}	10^{-3}	10^{-4}
AD/DD Agreement#	2	3	4	4^{+}	4^{+}	4^{+}
AD Time/ DD time	0.18	0.26	0.36	0.50	0.69	0.92
QA Time/ DD Time	1.08	1.08	1.08	1.08	1.08	1.08

† Norm of $R' = dR/dx$, # Number of significant digits.

Table 3: Accuracy for Six 3–D SD

ONERA M-6 Wing, TLNS3D Inviscid Transonic Flow,
3-Level Multigrid, $97 \times 25 \times 17$ ($M = 0.84$, $\alpha = 3.06°$)

(a) Dead Start: $\left(\frac{R_{800}}{R_1}\right)_{DD} \sim 10^{-5}$;
$\left(\frac{R_{500}}{R_1}\right)_{AD} \sim 10^{-4}$ and $\left(\frac{R'_{500}}{R'_1}\right)_{AD} \sim 10^{-3}$

D_{AD}/D_{DD}	C_L	C_D	C_M
M	0.9999	0.9992	0.9993
α	0.9995	0.9993	0.9992

(b) Restart File: $\left(\frac{R_{1800}}{R_1}\right)_{DD} \sim 10^{-11}$;
$\left(\frac{R_{1375}}{R_1}\right)_{AD} \sim 10^{-9}$ and $\left(\frac{R'_{575}}{R'_1}\right)_{AD} \sim 3 \times 10^{-5}$

D_{AD}/D_{DD}	C_L	C_D	C_M
M	1.0003	1.0000	1.0002
α	0.9999	0.9999	0.9999

Table 4: Accuracy for Nine 3-D SD

ONERA M-6 Wing, TLNS3D Laminar Subsonic Flow,
2-Level Multigrid, $97 \times 17 \times 17$ ($M = 0.2, Re = 5000$)
(a) Zero Angle-of-Attack

D_{AD}/D_{DD}	C_L	C_D	C_M
M	*	1.0004	*
α	1.0000	*	1.0000
Re	*	1.0000	*

*Numerator of D_{DD} is noise; $D_{AD} \sim O(10^{-8})$ and for DD stepsize of 10^{-6}, numerator of $D_{DD} \sim O(10^{-14})$

(b) One degree Angle-of-Attack

D_{AD}/D_{DD}	C_L	C_D	C_M
M	1.0028	0.9998	0.9985
α	1.0000	1.0001	1.0000
Re	1.0000	1.0000	1.0001

Table 5: Accuracy for Nine 3-D SD

ONERA M-6 Wing, TLNS3D Turbulent Transonic Flow,
3-Level Multigrid, $97 \times 25 \times 17$
($M = 0.84$,$\alpha = 3.06°$,$Re = 11.7 \times 10^6$)

(a) Mixing-length model

D_{AD}/D_{DD}	C_L	C_D	C_M
M	1.0000	1.0000	0.9999
α	1.0000	1.0000	1.0000
Re	1.0007	1.0000	1.0012

(b) Baldwin-Lomax model

D_{AD}/D_{DD}	C_L	C_D	C_M
M	1.0000	1.0000	1.0000
α	1.0000	1.0000	1.0000
Re	0.9991	1.0000	0.9961

(c) Convergence

$$\left(\frac{R_{1700}}{R_1}\right)_{DD} \sim 10^{-12} \ ;$$
$$\left(\frac{R_{1300}}{R_1}\right)_{AD} \sim 5 \times 10^{-12} \text{ and } \left(\frac{R'_{425}}{R'_1}\right) \sim 10^{-4}$$

Table 6: Computing Time and Memory for Nine 3-D SD

ONERA M-6 Wing, TLNS3D Turbulent Transonic Flow,
Baldwin-Lomax, 3-Level Multigrid, $97 \times 25 \times 17$

Solution Procedure	Number Solutions	Time (sec*)	Storage (MW)
DD one-sided	4 Nonlinear	3960	2.47
DD central	6 Nonlinear	5940	2.47
AD	1 Nonlinear, 3 Linear	10290	7.63

* All calculcations performed on CRAY Y-MP.

Table 7: Computing Time and Accuracy for Nine 3-D SD

ONERA M-6 Wing, TLNS3D Turbulent Transonic Flow,
Baldwin-Lomax, 3-Level Multigrid, $97 \times 25 \times 17$

AD Convergence†	10^{+1}	10^{0}	10^{-1}	$10^{-1.5}$
AD/DD Agreement#	2	3	4	4^{+}
AD Time/DD Time	1.01	1.45	2.07	2.59

† Norm of $R' = dR/dx$, # Number of significant digits.