JOURNAL OF COMPUTATIONAL PHYSICS **125,** 313–324 (1996)
ARTICLE NO. 0096

Sensitivity Derivatives for Advanced CFD Algorithm and Viscous Modeling Parameters via Automatic Differentiation

LAWRENCE L. GREEN, PERRY A. NEWMAN, AND KARA J. HAIGLER

NASA Langley Research Center, Hampton, Virginia 23681-0001

Received February 15, 1994; revised September 27, 1995

The computational technique of automatic differentiation (AD) is applied to a complicated computer program to illustrate the simplicity, efficiency, and versatility of AD with complex algorithms for use within a sensitivity analysis. Many algorithmic and physics modeling coefficients appear in computer programs that are routinely set in an ad hoc manner; AD can be used to enhance computer programs with derivative information suitable for guiding formal sensitivity analyses, which allows these coefficient values to be chosen in a rigorous manner to achieve particular program properties such as an improved convergence rate or improved accuracy. In this paper, AD is applied to a three-dimensional thin-layer Navier–Stokes multigrid flow solver to assess the feasibility and computational impact of obtaining exact sensitivity derivatives with respect to algorithmic and physics modeling parameters typical of those needed for sensitivity analyses. Calculations are performed for an ONERA M6 wing in transonic flow with both the Baldwin–Lomax and Johnson–King turbulence models. The wing lift, drag, and pitching moment coefficients are differentiated with respect to two different groups of input parameters. The first group consists of the second- and fourth-order damping coefficients of the computational algorithm, whereas the second group consists of two parameters in the viscous turbulent flow physics modeling. Results obtained via AD are compared for both accuracy and computational efficiency with the results obtained with divided differences (DD). The AD results are accurate, extremely simple to obtain, and show significant computational advantage over those obtained by DD for some cases.

1. INTRODUCTION

Sensitivity analysis (SA) provides a natural systematic means for both analyzing and predicting the behavior of diverse physical approximations and numerical algorithms or for identifying significant input parameters in a computational system. Generally SA is used in the latter context to identify significant input parameters in a system where the outputs are assumed to be functionally dependent only upon the inputs. That is, the output changes in response to specified changes in the input; however, everything else within the system (the numerical algorithm, physical approximations, etc.) is considered fixed. Changes in the system outputs are related to the changes in the system inputs through a sensitivity derivative (SD) matrix, or a Jacobian system. This SD matrix may be used to control processes or designs that depend upon the system output, frequently to achieve some optimal or constrained condition. Numerous examples with SA used in such a context can be found [1–9], including several examples of multidisciplinary design optimization (MDO). The current study is relevant to SA used in the former context as an aid in improving the accuracy of physical approximations and numerical algorithms.

The use of SA depends upon the SD matrix that has been computed in the past by divided differences (DD), direct differentiation (handcoding), or symbolic manipulation; the method used depends upon the complexity of the system to be modeled. However, when the system includes an advanced three-dimensional (3D) computational fluid dynamics (CFD) model, all of these differentiation methods have serious drawbacks, and few SA's for advanced 3D CFD models have been attempted until recently. Results from the first application of automatic differentiation (AD) to advanced CFD codes to obtain SD's have been reported recently [10]. That work demonstrated the feasibility of obtaining the exact nongeometric SD's of the wing C_L, C_D, and C_M with respect to M, α, and Re from a complex state-of-the-art 3D thin-layer Navier–Stokes flow code (TLNS3D, [11]). The emphasis and direction of that initial work continues in order to demonstrate that SD's (both geometric and nongeometric) can be obtained for ultimate use in an MDO of flight vehicle concepts. Results from another successful AD application in structures have also been reported recently [12].

In the present application, we propose that an AD-based SA can also be useful for investigating the adequacy of approximation models within a system. That is, an SA can be used to adjust the approximation models within the system in order to improve its physical simulation (i.e., match with experimental data) or to obtain desired numerical solution properties such as stability or convergence acceleration. In this regard, this work demonstrates that exact nongeometric SD's of the wing lift, drag, and pitching moment coefficients with respect to the CFD algorithm

0021-9991/96 $18.00

and turbulence modeling parameters can be obtained from TLNS3D via AD. Several sample results are shown; preliminary results for this feasibility study were generated in O(man-week), not O(man-year), as is typically required to generate derivative code by other means.

Advanced CFD codes that calculate SD, automatic differentiation methods and capabilities, and, in particular, the ADIFOR (AD of Fortran) source translator and its application to the iteratively solved nonlinear implicit functions of advanced CFD codes are briefly discussed in this paper. More detailed information about these topics can be found in Ref. [10] and the references cited therein.

2. METHODOLOGY

2.1. TLNS3D Flow Solver

The TLNS3D code [11] solves the time-dependent 3D thin-layer Navier–Stokes equations with a finite-volume formulation. The code employs grid sequencing, multigrid, and local time stepping to accelerate convergence and efficiently obtain steady-state high Reynolds number turbulent flow solutions. When temporally converged to a steady-state solution, the method is globally second-order accurate. The TLNS3D code is a central-difference code that employs second-order central differences for all spatial derivatives. The code used in this study employs a blending of scalar second- and fourth-difference artificial dissipation to maintain numerical stability. The solution is advanced explicitly in time with a five-stage Runge–Kutta time-marching scheme. The code includes both Baldwin–Lomax (B-L) [13] and Johnson–King (J-K) [14, 15] turbulence models and 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.

The forthcoming multiblock version [16] of the TLNS3D code promises the flexibility needed for modeling complex geometric configurations. Initial work has been reported [17] on implementing the code on parallel processors. All of these facets enhance its usefulness in applications to real engineering solutions and, eventually, to MDO problems. Obtaining consistent SD information is, therefore, of genuine interest, and the straightforward application of ADIFOR appears to be the most direct route for achieving this goal.

2.2. Automatic Differentiation

Automatic differentiation [18] is a chain-rule-based technique for evaluating the derivatives of functions defined by computer programs with respect to their input variables. Automatic differentiation has two basic modes, which are usually referred to as the forward and reverse modes, respectively. The reverse mode is closely related to adjoint methods and has a lower operations count for gradient computations but potentially larger memory requirements [19] than the forward mode.

In contrast to the approximation of derivatives by DD, AD does not incur any added truncation error so that at least for noniterative 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 requirements can be bounded a priori in terms of the complexity of the original functions for all modes of AD. Calculation of SD's by hand differentiation of functions defined by computer codes can be very tedious, time-consuming, and error prone. 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 [20] on the theory, implementation, and some earlier applications is suggested for more information. A review [21] of the earlier AD tools is also recommended to the reader.

Advanced CFD codes typically employ iterative solution algorithms to solve the implicit nonlinear partial differential equations that express the fluid conservation laws. Application of AD to such solution algorithms [10] will not be repeated here because the iterative paradigm used in this present feasibility study for the SD computation is the same as that previously reported.

2.3. ADIFOR Preprocessing Tool

The ADIFOR [22–25] (automatic differentiation of Fortran) source translator is an AD tool jointly developed by Argonne National Laboratory (Mathematics and Computer Science Division) and Rice University (Center for Research on Parallel Computation) that differentiates programs written in Fortran 77. That is, given a Fortran subroutine (or collection of subroutines) that describe a function and an indication of which variables in parameter lists or common blocks correspond to independent and dependent variables with respect to differentiation [26], ADIFOR produces Fortran 77 code that allows 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 [20]. 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; the code is then employed in the forward mode to propagate overall derivatives. The resulting decrease in complexity in comparison with an entirely forward mode implementation is usually substantial.

The ADIFOR tool uses the facilities of the ParaScope Fortran environment [27, 28] to parse the code and to extract control flow and dependence information to determine the other variables that require derivative informa-

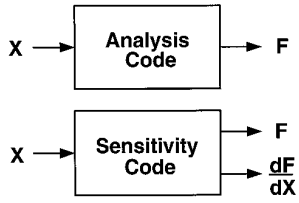

FIG. 1. I/O data for analysis and sensitivity codes.

tion propagation. This approach allows for an intuitive interface and greatly reduces the storage requirements of the derivative code. The ADIFOR tool produces portable Fortran 77 code and accepts much of Fortran 77—in particular, arbitrary calling sequences, nested subroutines, common blocks, and equivalences.

2.4. Application of ADIFOR to TLNS3D

This work was performed to appraise the feasibility and computational impact of applying AD to advanced CFD codes to generate the SD's typical of those needed for use in an SA. Figure 1 depicts the transformation of an original analysis system into an SA system. The system input is X, and the output is F. Both F and X may be scalar, vector, or array quantities, and each may involve one or many variables. The input X for the system may consist of several types of variables such as those that specify initial and boundary conditions, material properties, constraints, physical approximations, and numerical solution parameters. Similarly, the output F for the system may consist of local and global solution properties, accuracy measures, and performance indicators. Depending on the nature of the SA to be performed, presumably any combination of derivatives of output functions with respect to input variables could be obtained. In the present AD paradigm, the analysis code is operated on by the AD source translator, which produces a new sensitivity code with the same inputs and outputs, as well as the SD matrix represented by dF/dX.

Application of ADIFOR to Fortran code requires the specification of the independent and dependent variables to be used in forming the SD = ∂(dependent)/∂(independent). The partial derivative notation ($\partial(\ )/\partial(\ )$) will be used hereafter for SD, although these are total derivatives for the single-discipline analysis code; typically, they would be used in an MDO and viewed there as partial derivatives. In this paper, the wing C_L, C_D, and C_M coefficients have been used as dependent variables. Two ADIFOR applications with two different sets of independent variables were considered. The first was the second- and fourth-order damping coefficients (VIS2 and VIS4) of the CFD solution algorithm, and the second was two turbulent flow physics modeling parameters (the Clauser constant K and the Van Driest constant A^+). Note that other dependent variables (such as the shock location at one or more spanwise wing sections for the turbulent flow physics parameters or the spectral radius of convergence for the algorithmic parameters) might be more appropriate for an SA with the present independent variables; but, the dependent variables used (the wing C_L, C_D, and C_M coefficients) were chosen simply to demonstrate the technique.

Application of ADIFOR to TLNS3D was performed in a simple and straightforward manner. Minor changes to the TLNS3D code required for ADIFOR processing were made, and the TLNS3D code was passed to ADIFOR as input. The ADIFOR tool differentiated through the entire multigrid solution algorithm, the specified dependencies were traced, and new SD coding was generated as required. The resulting SD modules were then assembled into a working code, and the initial results were generated quickly. The SD code was later modified to improve its performance on a Cray Y-MP computer. Several different cases have been examined with the two SD codes, and comparisons with DD have been made.

3. COMPUTATIONAL RESULTS

The flow conditions used herein are taken from previous research of Ref. [29]. In that study the B-L and J-K solutions are quite similar at $\alpha = 3.06°$; however, at $\alpha = 5.06°$ the two differ significantly with respect to local flow details and boundary-layer transition locations. It was expected that SD's computed in the current study with respect to K and A^+ might also reveal differences for these two flow conditions; however, the choice of integrated force and moment coefficients as dependent variables may tend to mask significant local differences. No direct comparisons are made with the research of Ref. [29] due to differences in the dissipation models used and in the convergence-level requirements. The goal of that previous study was to compare different code and turbulence model results at *nominal* convergence levels. The goal of the present study is to compare the SD's computed by AD with those computed by DD's, the latter of which requires *extremely good* convergence levels for the small perturbations needed to adequately approximate the derivatives. All results in this paper were obtained on the NASA Langley Research Center Cray Y-MP.

3.1. SD on 97 × 25 × 17 Grid

Initial results were calculated for transonic turbulent flow about an ONERA M-6 wing at M = 0.84, $\alpha = 3.06°$, and Re = 11.7×10^6 on a 97 × 25 × 17 C-O mesh with three-level multigrid (MG) for the B-L turbulence model. A view of a sample 25 × 13 × 9 mesh is shown in Fig. 2. For SD calculations by DD, perturbations (Δ) of 10^{-6} times the independent variables were used.

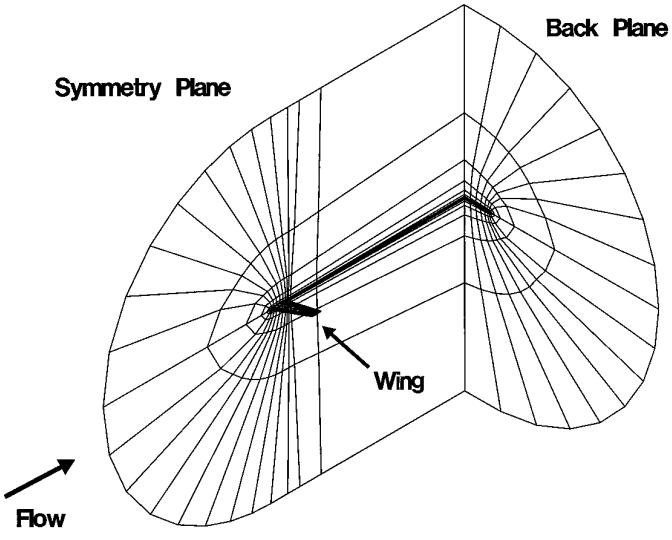

FIG. 2. Schematic of $25 \times 13 \times 9$ grid.

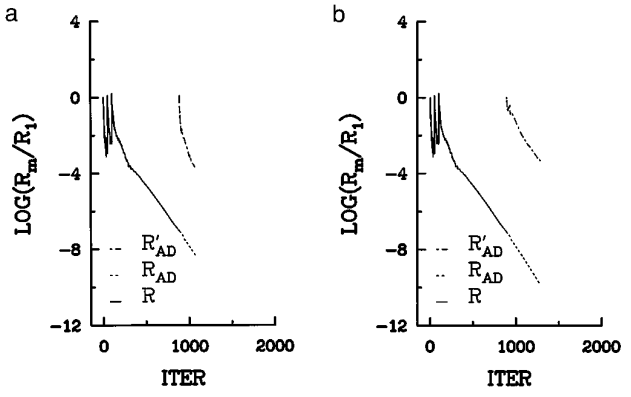

FIG. 3. Iteration convergence histories, $97 \times 25 \times 17$ grid, $\alpha = 3.06°$, B-L model, TLNS3D and AD codes: (a) Algorithm parameters; (b) Turbulence modeling parameters.

3.1.1. *CFD Algorithm Parameters*

The CFD algorithm parameters chosen here as independent variables to demonstrate that ADIFOR can obtain consistent discrete SD were essentially the coefficients of the second- and fourth-order artificial dissipation terms. These variables [11] are $\kappa^{(2)}$ and $\kappa^{(4)}$; in the TLNS3D code input, these are proportional, respectively, to VIS2 and VIS4 as

$$\kappa^{(2)} \propto \text{VIS2}$$
$$\kappa^{(4)} \propto \text{VIS4}. \tag{1}$$

The iterative paradigm for using the AD-generated code to calculate the SD is as follows. First, run the original code to a reasonably good convergence, and then start the SD calculations from a restart file of that solution. In this case, the original code was run 890 MG iterations on the $97 \times 25 \times 17$ grid. Calculations of SD by AD and by DD were then both initiated from the restart file. In the AD code, the matrix of derivatives of the independent variables with respect to each other (the "seed" matrix [10]) was an identity matrix for both scratch starts and restarts, whereas the matrix of derivatives of the dependent variables with respect to the independent variables was iterated from a zero initialization. The AD code was run for an additional 180 iterations, whereas the DD baseline continuation and perturbations were run for an additional 550 iterations. A convergence measure for the analysis solution is the logarithm of the normalized rms (root mean square) residual

$$\varepsilon = \log_{10}\left(\frac{R_m}{R_1}\right), \tag{2}$$

where m is the total number of iterations, including the original run plus restarts. For the DD solutions, ε was about -12; for the AD solution, ε was about -9.5. A similar convergence measure,

$$\varepsilon' = \log_{10}\left(\frac{R'_m}{R'_1}\right), \tag{3}$$

was used for the normal of the residual gradient in the AD code only. In this case, ε' was about -3.5 for the AD solution. The iterative convergence histories for the residual R for both TLNS3D and the AD code, as well as that for the derivative R' from the AD code, are shown in Fig. 3a. The three distinct peaks at the start are caused by the grid sequencing that is used to initiate the finest grid solution. The AD code is started at 890 MG iterations, and, as will be shown in the next section, the work/iteration is not the same for the original and AD codes. Thus, the abscissa is not indicative of the computational work in this figure.

The relative accuracy of SD via AD and DD is shown in Table I as the ratios $D_{\text{AD}}/D_{\text{DD}}$. Obviously, the closer the SD ratio is to unity, the better the agreement between AD and DD for the particular SD. Approximately three-

TABLE I

SD Ratios for Algorithm Parameters, $97 \times 25 \times 17$ Grid, $\alpha = 3.06°$, B-L Model

$\frac{D_{\text{AD}}}{D_{\text{DD}}}$	C_{L}	C_{D}	C_{M}
VIS2	0.9972	1.0008	0.9954
VIS4	0.9943	1.0015	1.0016

TABLE II

SD Ratios for Turbulence Modeling Parameters, $97 \times 25 \times 17$ Grid, $\alpha = 3.06°$, B-L Model

$\frac{D_{AD}}{D_{DD}}$	C_L	C_D	C_M
K	0.9880	0.9882	0.9879
A^+	0.9997	0.9950	0.9997

significant-digit agreement exists at the convergence levels obtained. All of these SD are $O(10^{-3}$ or $10^{-4})$. Since the DD step size Δ is 10^{-6} times the baseline values of the independent variables, the third digit in the Table I entries corresponds to the 11th digit of the dependent variables that are subtracted to form the numerator of the D_{DD}.

3.1.2. *Turbulence Model Parameters*

For this demonstration of ADIFOR-generated SD with respect to turbulence model parameters, the Baldwin–Lomax algebraic model is used. The two parameters chosen are the Clauser constant K (equal to 0.0168 in Ref. [30]) of the outer-region eddy viscosity coefficient and the constant A^+ (equal to A in Ref. [30]) of the Van Driest correction to the Prandtl mixing length of the inner-region viscosity coefficient.

The same TLNS3D restart solution and derivative initialization that was used for the previous example was used here; DD's were run an additional 550 MG iterations as before, but AD was run 400 iterations. Again, ε was about -12 for the DD solutions, but ε was about -11 and ε' was -3.3 for the AD solution, even though more AD iterations were run than before. Figure 3b shows the iterative convergence histories for the residual R from the original TLNS3D and the AD code, as well as the convergence history for R' from the AD code. The relative accuracy of SD by AD and DD is shown in Table II as the ratios D_{AD}/D_{DD}. Again, nearly three-significant-digit agreement exists at the convergence levels obtained.

Initial runs of the original TLNS3D code with the J-K turbulence model for both two- and three-level MG on the $97 \times 25 \times 17$ grid could not be converged to an ε of -5 before the residual became extremely noisy. These baseline runs were intended to create restart files for starting both DD and AD code runs to obtain similar SD. A finer grid was required to obtain solutions needed to verify the SD's for the J-K turbulence model.

3.1.3. *Initial Results Summary*

These initial results on the $97 \times 25 \times 17$ grid demonstrate that the ADIFOR-generated derivatives are accurate. That is, the mechanical aspect of ADIFOR differentiation, which *does not* depend upon grid size, has been verified for both the CFD algorithm and turbulence modeling parameters considered. The basic flow solution, however, does exhibit truncation error and, perhaps, resolution effects. Both of these effects, of course, can affect the SD values, whether computed by the AD or the DD method. Resolution of issues other than the accuracy of ADIFOR-generated SD prompted the use of a finer mesh. Realistic problems will need finer meshes for resolution of flow details and numerous mesh blocks for complete vehicle configurations that require far more grid points.

3.2 SD on $193 \times 49 \times 33$ Grid, B-L Model

All results on this larger grid were obtained with a two-level, rather than a three-level, MG because of concerns about the J-K turbulence model solutions. These concerns, however, were unfounded; the primary effect in the present solutions is a somewhat slower convergence than would have been attained with a three-level MG.

Results on this larger grid have been obtained only for the viscous turbulence modeling parameters, which were deemed to be of greater interest, based upon the initial studies. Results to be presented include those for (1) the nominal case on this larger grid; (2) an additional flow condition at $M = 0.8447$ and $\alpha = 5.06°$; (3) variation of the DD perturbation step size Δ; (4) variation of the convergence levels of ε and ε' within the SD calculation; (5) variation of convergence level of the baseline restart file (i.e., at initiation of SD calculations); and (6) the J-K turbulence model. Each of these variants are discussed separately in the following sections. However, some comments must be made about the ADIFOR postprocessing that was required to improve AD code vectorization in order to reasonably obtain AD solutions on this larger grid.

3.2.1. *ADIFOR Postprocessing*

The two previous examples that comprised the initial research for this study were done on a $97 \times 25 \times 17$ grid because the AD code could not be reasonably run for many MG iteration cycles on a larger grid. The original sequential TLNS3D code is an efficient, highly vectorized code. The initial ADIFOR-generated version of TLNS3D, however, was not nearly as efficient on the Cray Y-MP. Derivative code that was produced with the present version of ADIFOR had to be postprocessed manually in order to restore much of the vectorization. The Cray compiler Flowtrace and Loopmark options were used to identify subroutines, function calls, and ''do loops'' that did not vectorize as well as the corresponding ones in the original code; thus, far too much execution time was consumed in the AD code. After this evaluation, simple code modifications were made (i.e., by changing one recurrent subroutine argument to a parameter; by restoring intrinsic Cray vector

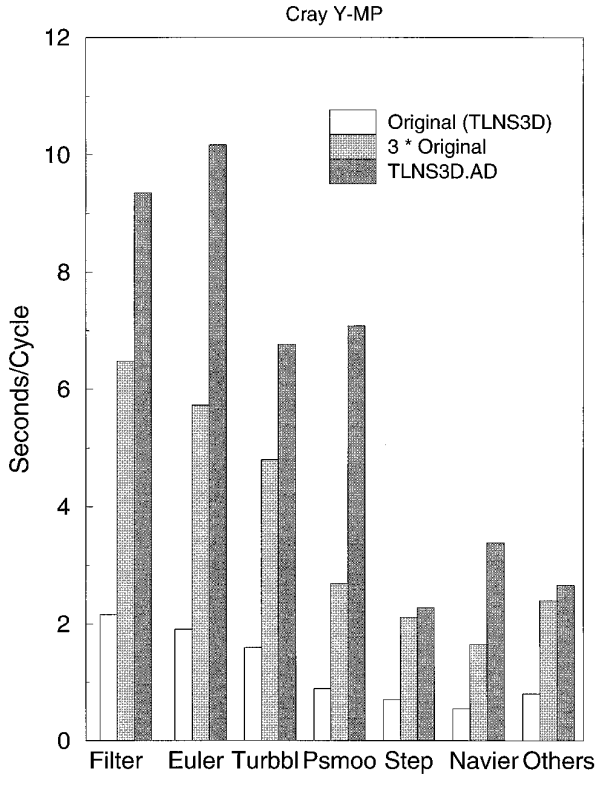

FIG. 4. Flow trace profiles comparing the six most time consuming subroutines of TLNS3D and AD codes.

functions, which ADIFOR could not process; or, in one case, by segmenting a large loop into several smaller ones). More compiler options were used (i.e., by disabling double precision, selecting the aggressive compile option, and inlining the Fortran intrinsic and error handling functions provided by ADIFOR) to improve the derivative code vectorization.

Figure 4 compares Flowtrace timing profile results for the original TLNS3D code with those for three times the original TLNS3D code (which approximate one-sided DD run times for SD with two independent variables) and those from the best (to date) AD-version TLNS3D code. Some degradation of the vector efficiency still exists. This may be due in part to automatic introduction by ADIFOR of additional intermediate variables in the original function evaluations in order to recognize the basic derivatives of all unary and binary operations. Another point to be made concerns the relatively good efficiency obtained for the STEP subroutine, as seen in Fig. 4. In the initial AD version, only about half of the loops in this routine vectorized (even after inlining and the aggressive compile options were added), whereas all the loops vectorized in the STEP routine of the original TLNS3D code. The timing result for such poorly vectorized code would be well off the plot shown in Fig. 4. With loop segmentation, however, the STEP routine timing was brought to the current level, which perhaps indicates that such segmentation might improve vector performance of other routines. The optimum vector efficiency has surely not yet been achieved; however, the AD code could be run on the $193 \times 49 \times 33$ grid to generate results for the present study.

3.2.2. *Nominal SD Results*

The problem shown previously for the turbulence modeling SD on the $97 \times 25 \times 17$ grid was run on a finer ($193 \times 49 \times 33$), more highly stretched grid. The original code was run for 1000 iterations to generate a restart file; DD's were then run for another 1000 iterations. The AD code executed 650 iterations from the restart. For the DD solutions, ε was about -11.1; for AD, ε was about -10 and ε' was about -3.5. As shown in Table III(a), the agreement among all the SD ratios is significantly improved over those in Table II. Actual values of the SD computed by AD are shown in Table III(b).

The iterative convergence histories for both the original and AD-version SD codes are given in Fig. 5. In Fig. 5a, both ε and ε' are plotted versus the iteration index previously used in Fig. 3 for the coarser grid results. The ADIFOR postprocessing for the Cray computer not only allowed for meaningful runs on this finer grid but also produced the best execution times to date. From results such as those in Fig. 4, the iteration scale can be converted to a work scale with a common unit, taken here as a single iteration sweep on the fine grid of the original TLNS3D code. Figure 5b, then, is the iteration residual histories of Fig. 5a replotted versus the work. The AD code for SD produced by differentiating through the iterative solution algorithm is still not computationally efficient. The current emphasis, however, as in Ref. [10], has been on a simple demonstration that AD can obtain SD with respect to

TABLE III

SD for Turbulence Modeling Parameters, $193 \times 49 \times 33$ Grid, $\alpha = 3.06°$, B-L Model

$\frac{D_{AD}}{D_{DD}}$	C_L	C_D	C_M
(a) *SD ratios*			
K	0.9999	1.0000	0.9998
A^+	0.9997	1.0001	0.9999

D_{AD}	C_L	C_D	C_M
(b) *SD values via AD*			
K	0.5893	0.1054	-0.3527
A^+	0.1538×10^{-3}	-0.2800×10^{-4}	-0.7607×10^{-4}

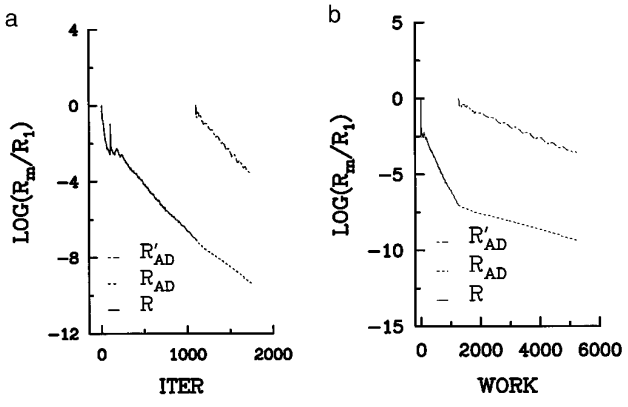

FIG. 5. Convergence histories, 193 × 49 × 33 grid, $\alpha = 3.06°$, B-L model, TLNS3D and AD codes: (a) iteration history; (b) work history.

various input and output quantities for a multigrid flow solver.

The 193 × 49 × 33 grid was also used to calculate SD's with the B-L turbulence model at M = 0.8447 and $\alpha = 5.06°$. For the DD solutions, ε was about −9.7; for the AD solution, ε was about −8.7, and ε' was about −3.5. Ratios of the calculated SD's are shown in Table IV(a) and the actual SD values are shown in Table IV(b). Agreement of the SD ratios is similar to that shown in Table III(a). Generally, the sensitivity (Table IV(b)) to these parameters is amplified at the higher alpha in comparison with that at the lower alpha conditions (Table III(b)); but the sensitivity of C_D to the A^+ has decreased.

3.2.3. *DD Step-Size Effects*

In all previous examples, $\partial C_L/\partial K$ was the largest derivative and $\partial C_D/\partial A^+$ was the smallest in magnitude. The derivatives spanned four to five orders of magnitude. For smoothly converging functions such as the B-L solution at low angles of attack, a variation of the perturbation size Δ in the DD has little effect. This effect is illustrated in Table V for both $\partial C_L/\partial K$ and $\partial C_D/\partial A^+$ derivatives, with SD ratios where the numerator (obtained from the AD solution) is fixed. Several perturbation sizes Δ for the independent variables should be used to determine if a suitable step size can be found to use in constructing the DD. Perhaps no single step size is satisfactory for approximating all the derivatives of interest.

TABLE V

Effect of DD Perturbation Size Δ on SD Ratios, 193 × 49 × 33 Grid, $\alpha = 3.06°$, B-L Model

Δ	Ratio $\partial C_L/\partial K$	Ratio $\partial C_D/\partial A^+$
1×10^{-3}	0.9991	0.9995
1×10^{-4}	0.9988	1.0000
1×10^{-5}	0.9999	1.0000
1×10^{-6}	0.9999	1.0001

3.2.4. *SD Convergence Effects*

The effect of SD convergence levels (i.e., levels of both ε and ε') on the SD ratios can be seen in Table VI, where both the AD and DD solutions are shown for iterations that range from 250 to 1000. For the original code used in the DD, 1000 iterations represent a Cray Y-MP job of about 3 h. For the AD code, 250 iterations represent approximately a 3-h job, and 650 iterations represent an 8-h job; a total of 450 iterations was chosen as a convenient intermediate point between the 3- and 8-h jobs. Note that no attempt was made to run the AD code for 1000 iterations.

In general, the AD derivatives converge at nearly the same rate as the DD. This similarity is illustrated in Table

TABLE IV

SD for Turbulence Modeling Parameters, 193 × 49 × 33 Grid, $\alpha = 5.06°$, B-L Model

$\frac{D_{AD}}{D_{DD}}$	C_L	C_D	C_M
(a) *SD ratios*			
K	0.9997	0.9998	0.9997
A^+	0.9996	1.0027	0.9996

D_{AD}	C_L	C_D	C_M
(b) *SD values via AD*			
K	1.4846	0.1949	−0.8782
A^+	0.3455×10^{-3}	-0.5453×10^{-5}	-0.1799×10^{-3}

TABLE VI

Effect of Convergence on SD Ratios, 193 × 49 × 33 Grid, $\alpha = 3.06°$, B-L Model

ITER	AD250	AD450	AD650
(a) $\partial C_L/\partial K$ *ratios*			
DD250	0.9999	1.0609	1.0640
DD450	0.9427	1.0000	1.0031
DD650	0.9397	0.9969	1.0000
DD1000	0.9396	0.9968	0.9999
(b) $\partial C_D/\partial A^+$ *ratios*			
DD250	0.9998	0.9691	0.9679
DD450	1.0313	0.9996	0.9985
DD650	1.0327	1.0010	0.9999
DD1000	1.0327	1.0010	0.9999

TABLE VII

Effect of Convergence on SD Values, $193 \times 49 \times 33$ Grid, $\alpha = 3.06°$, B-L Model

ITER	DD	AD
(a) $\partial C_L/\partial K$ *values*		
250	0.55388	0.55384
450	0.58753	0.58755
650	0.58935	0.58933
1000	0.58941	N/A
(b) $\partial C_D/\partial A^+$ *values*		
250	0.28923×10^{-4}	0.28916×10^{-4}
450	0.28039×10^{-4}	0.28028×10^{-4}
650	0.28000×10^{-4}	0.27996×10^{-4}
1000	0.28000×10^{-4}	N/A

VI(a) for $\partial C_L/\partial K$ of the B-L solution at a low angle of attack, at which the convergence was extremely good and the DD step size showed little effect. The similar convergence of the SD both by AD and by DD is observed in Table VI(a) in that the ratios are all nearly unity on the diagonal, where the number of iterations for AD equals the number of iterations for DD. Conversely, as the difference between the two iteration values increases (i.e., moves off the diagonal), the ratio departs from unity. Thus, comparisons between AD and DD are best made at the same number of iterations when the solutions are well converged (i.e., in regions where the solutions are converging at the asymptotic rate). The accuracy of both AD and DD improves as more work is done; this result is illustrated by the general improvement of the SD ratio in moving from upper left to lower right in Table VI(a). Similar effects are observed in the SD ratios for $\partial C_D/\partial A^+$ in Table VI(b). The effects of SD convergence on the accuracy of the individual SD values is illustrated in Tables VII(a) and VII(b) for the same two derivatives, respectively.

Note that in all these exercises, the lift, drag, and moment coefficients were printed in exponential format to 12 decimal places and serious attempts were made to drive the solutions to extremely good levels of convergence. These attempts led to an interesting observation about the AD code that would not be noticed in execution to normal convergence levels. The round-off error of the AD code is different than that of the original code; the evaluation of the function is globally equivalent, but locally consists of different code statements. Thus, as the convergence proceeds, small differences in the least significant digits of the function evaluation can be observed between the original and the AD code. These differences are unlikely to cause any meaningful changes in the sensitivity derivatives, yet the discrepancy can make the comparison of SD by AD with those by DD less than perfect.

3.2.5. *Restart Convergence Effects*

In this section, variations in the initial convergence level for the B-L solution at the lower angle of attack are considered, where the SD can apparently be converged to any level by either AD or DD. Thus, the impact of the starting convergence level can be judged. Table VIII shows the effect of the initial starting point on the DD calculation of $\partial C_L/\partial K$. The scratch start derivative convergence is not monotonic; the derivative convergence from a solution restart is monotonic. Both runs attain the same final derivative (0.58941); however, the scratch start does this in 900 iterations, whereas the restart requires a total of 1700 iterations. Thus, to obtain the DD approximations of all six SD's with the assumption that all converge at the same rate requires $3 \times 900 = 2700$ iterations from scratch, or $1000 + 3 \times 700 = 3100$ iterations with a restart. In this case, for two independent variables the DD can be obtained more efficiently by starting the baseline and perturbations from scratch rather than by using a restart. However, the restart is better for more than four independent variables. Here, only the largest of the six derivatives has been considered; others may converge at different rates.

These results also impact the interpretation of the AD results. Because that code only converged a total of 1650 iterations (1000 start + 650) and because the DD and AD derivatives appear to converge at the same rate, we conclude that the AD derivative is still converging and that it should ultimately approach the same value of 0.58941. The derivative convergence cannot be faster than the functional convergence [31], so the number of iterations required to achieve this result is not clear.

The initial convergence level seems to have little effect on the convergence rate of the derivatives via AD, as

TABLE VIII

Effect of Restart Convergence Level on $\partial C_L/\partial K$ via DD, $193 \times 49 \times 33$ Grid, $\alpha = 3.06°$, B-L Model

Scratch start		1000 iteration restart	
ITER	$\partial C_L/\partial K$	ITER	$\partial C_L/\partial K$
100	0.59300	1100	0.27765
200	0.48694	1200	0.52018
300	0.59124	1300	0.57300
400	0.57053	1400	0.58565
500	0.58682	1500	0.58847
600	0.58741	1600	0.58924
700	0.58947	1700	0.58941
800	0.58929	1800	0.58941
900	0.58941	1900	0.58941
1000	0.58941	2000	0.58941

TABLE IX

Effect of Restart Convergence Level on $\partial C_L/\partial K$ via AD, $193 \times 49 \times 33$ Grid, $\alpha = 3.06°$, B-L Model

AD ITER	250 start	500 start	1000 start
100	0.27432	0.27729	0.27761
150	0.43726	0.44557	0.44759
200	0.51751	0.52035	0.52020
250	0.55256	0.55393	0.55384
450	N/A	N/A	0.58755
650	N/A	N/A	0.58933

shown in Table IX. Three different restart levels (250, 500, and 1000 iterations) are compared for the convergence of $\partial C_L/\partial K$. All appear to converge the derivative at approximately the same rate, independent of the initial convergence level. Only 250 AD iterations were run for the restarts at 250 and 500 iterations. Similar convergence characteristics were observed for $\partial C_D/\partial A^+$.

3.2.6. *Time and Memory Requirements*

For smoothly converging functions, the SD obtained via AD and DD should be compared at the same number of iterations. For the two such cases examined, both AD and DD results are comparable in accuracy in approximately the same time if a proper Δ for the DD can easily be determined. Table X(a) shows a timing comparison for the restart after 250 iterations, whereas Table X(b) shows a timing comparison for the restart after 1000 iterations. Both tables show only the time required to obtain the DD by forward differencing; the ratios would decrease if centered differences were used. A slight computational advantage to DD is shown in Tables X(a) and X(b). The AD code is competitive, however, and may become more attractive if more derivatives or SD's with more independent variables are computed. Moreover, if a proper Δ cannot be easily determined, then a single AD run will surely be more efficient than perhaps many DD runs.

TABLE X

Timing Comparison, $193 \times 49 \times 33$ Grid, $\alpha = 3.06°$, B-L Model

ITER	AD time (seconds)	DD time (seconds)	Ratio
(a) *Restart at 250 iterations*			
250 (start)	2230	2230	1.00
500	12630	8670	1.46
(b) *Restart at 1000 iterations*			
1000 (start)	8330	8330	1.00
1250	18750	14590	1.29
1450	27090	19600	1.38
1650	35430	24610	1.44
2000	N/A	33380	N/A

TABLE XI

Memory Requirements for Original and AD Codes

MW memory required	Original code	AD code
$97 \times 25 \times 17$ grid	2.1	5.1
$193 \times 49 \times 33$ grid	16.4	40.5

Because of two subtle differences, the timing information presented in Table X is not directly comparable to that in Ref. [10], in which a value of 2.59 was quoted for the timing ratio. First, comparisons there were made with the AD and the DD at different numbers of MG iterations, whereas current timing numbers are presented with both AD and DD at the same number of iterations. A current timing ratio calculated in the manner of Ref. [10] would be a value close to unity. Also, the current timings employ the aggressive compile option for both the original and AD codes, whereas those of Ref. [10] did not.

Memory requirements for the original and AD codes for two grid sizes are shown in Table XI. Naively, one would expect that the memory would increase by a factor of about 3 over the original code in order to accommodate calculation and storage of the function plus the two derivatives by an iterative scheme. For both grids, the actual increase in memory is about 2.5 due to the ADIFOR dependency analysis, which augments only parts of the code.

3.3. SD on $193 \times 49 \times 33$ Grid, J-K Model

In the following examples, the J-K turbulence model is used to obtain SD at both the low and high angle-of-attack conditions. The TLNS3D code with the J-K turbulence model frequently appears to have significantly different asymptotic convergence characteristics than that of the B-L model, as illustrated by the comparison of the normalized residuals in Fig. 6. At the low angle-of-attack condition, the J-K residual converges several orders of magnitude and then hangs up in a low-amplitude noise (which may be oscillatory); the B-L residual can be driven down to machine zero, as seen in Fig. 6a. At the high angle-of-attack condition, the amplitude of the noise in the J-K residual is increased as seen in Fig. 6b. Noise also exists in each J-K solution for the dependent variables for which SD computations were required. This finding has dire implications for obtaining SD via DD, as illustrated subsequently.

Because the restart point at about 1260 work units was already in the noisy phase, the prospect of computing

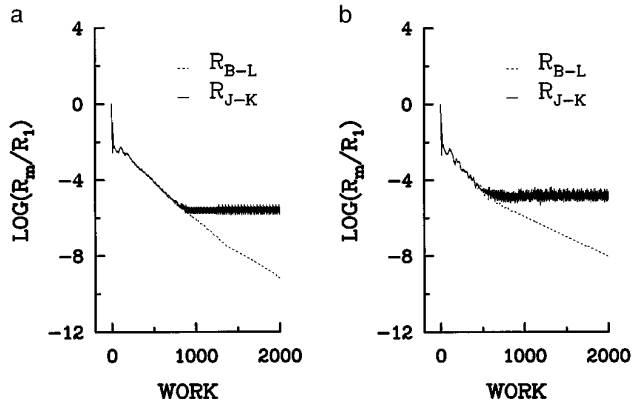

FIG. 6. Work convergence histories, 193 × 49 × 33 grid, B-L and J-K models, TLNS3D and AD code: (a) $\alpha = 3.06°$; (b) $\alpha = 5.06°$.

meaningful derivatives by either AD or DD appeared dubious. However, the SD that were calculated via AD appeared to converge to about three significant digits. After experimenting with several DD perturbation step sizes Δ, the best comparison of SD ratios was obtained with $\Delta = 10^{-3}$ times the independent variable. These results are shown in Table XII(a) for the J-K solution at M = 0.84, $\alpha = 3.06°$ on the 193 × 49 × 33 grid. Two ratios are near unity, but others depart considerably from that value. In terms of the noise, these ratios are the DD based upon instantaneous values and cannot be expected to give meaningful SD because they represent any of the infinite number of possible derivatives within a band of points. However, the AD's are expected to give meaningful results because the SD calculations via AD are driven by the MOST significant digits of the function evaluation, whereas the DD are constructed by subtraction, where the MOST significant digits of the function evaluation are lost.

TABLE XII

SD Ratios for Turbulence Modeling Parameters, 193 × 49 × 33 Grid, $\alpha = 3.06°$, J-K Model

D_{AD}/D_{DD}	C_L	C_D	C_M
(a) *Restart, instantaneous DD*			
K	0.7874	0.7750	0.8037
A^+	1.0083	1.0403	0.7359
(b) *Scratch start, instantaneous DD*			
K	0.9825	0.9764	0.9779
A^+	1.2348	0.2062	1.0469
$D_{AD}/D_{\overline{DD}}$	C_L	C_D	C_M
(c) *Restart, time-averaged DD*			
K	0.9940	0.9826	0.9935
A^+	0.9694	0.9544	1.0191

TABLE XIII

SD Ratios for Turbulence Modeling Parameters, 193 × 49 × 33 Grid, $\alpha = 5.06°$, J-K Model, Time-Averaged DD

$D_{AD}/D_{\overline{DD}}$	C_L	C_D	C_M
K	1.0513	0.9512	1.0355
A^+	0.7680	−0.0807	0.7363

A second attempt to assess the SD for the J-K solution at the low angle-of-attack condition was made by starting the SD calculations from scratch rather than from a restart file that was already noisy. The AD code was again run 650 iterations, which converged it to approximately the point at which the noise began, near the knee of the J-K residual curve of Fig. 6a. The idea was to see if meaningful SD ratios could be obtained on the convergent leg of the solution. Again, several DD perturbation sizes Δ were investigated and the best appeared to be $\Delta = 10^{-3}$ times the input variables. These SD ratios are presented in Table XII(b). The SD with respect to K show improvement over those in Table XII(a).

A final attempt to obtain a meaningful correlation between the SD obtained by AD and DD for this case was made by averaging the restart run results over a number of iterations. The lift, drag, and pitching moment coefficients were averaged over the last 350 iterations for which the functional evaluation was considered to be noisy. The DD were then constructed with these averaged output coefficients for various perturbations in the independent variables. These averaged results, denoted as $D_{\overline{DD}}$, are shown in the SD ratios of Table XII(c). The ratios are all significantly improved over theose presented in either Table XII(a) or Table XII(b).

The SD values for the J-K case, similar to those in Table III(b), are not shown since the SD ratios of Tables XII(a–c) indicate that the AD computations cannot be accurately verified by DD. The SD values in Table III(b) of D_{AD} with respect to K are $O(10^{-1})$. As shown in Table XII(c), the corresponding SD ratios $D_{AD}/D_{\overline{DD}}$ indicate SD agreement to about 1%. On the other hand, the D_{AD} with respect to A^+ are $O(10^{-3}$ and $10^{-4})$, and Table XII(c) shows the corresponding SD ratios to indicate SD agreement to only 2% to 5% accuracy.

An attempt was also made to calculate SD for the J-K turbulence model at M = 0.8447 and $\alpha = 5.06°$ on the 193 × 49 × 33 grid, despite the poor convergence characteristics of the solution. Again, the SD via AD converged to two or three significant digits. The averaging technique described above was also applied to the DD for this case. Again, after some experimentation, the best SD ratios are shown in Table XIII and were constructed with a DD

perturbation step size of $\Delta = 10^{-3}$ times the independent variables. As in the previous example, some SD ratios are near unity, and some are not; one has the wrong sign on it.

The SD values of D_{AD} with respect to K (also not shown for this J-K case, but similar to those shown for the B-L case in Table IV(b)) are $O(10^0$ and $10^{-1})$. In Table XIII these SD ratios $D_{AD}/D_{\overline{DD}}$ indicate SD agreement to about 5% accuracy. However, the D_{AD} with respect to A^+ are now $O(10^{-3}$ and $10^{-5})$; the corresponding SD ratios in Table XIII indicate SD agreement to only 25% and not at all, respectively. That is, the small SD's obtained by the AD calculations have been completely swamped by the noise for the DD calculations.

4. CONCLUDING REMARKS

These results demonstrate the efficiency and adequacy of obtaining sensitivity derivatives (SD) with respect to computational fluid dynamics (CFD) algorithms and viscous modeling parameters for advanced CFD codes via the ADIFOR automatic differentiation (AD) tool. Preliminary SD results were obtained in a very short lead time (O(man-week)) and would enable sensitivity analysis (SA) for a number of parameters to be done simultaneously. In the past, such sensitivity studies have generally been done in a rather unsystematic and brute-force manner. This present technology enables an SA, which determines the adequacy of numerical and physical modeling parameters to be automated, for example, into a formal constrained optimization procedure.

The SD's for the two-algorithm parameters chosen were all small, yet these were adequately calculated by both AD and divided differences (DD) at the low angle-of-attack flow for the B-L turbulence model on the $97 \times 25 \times 17$ grid. Agreement between the SD via AD and DD was good despite the fact that the DD made use of data in the 9th–12th-decimal place of the output functions.

The SD for the two turbulence modeling parameters varied by several orders of magnitude, with both positive and negative derivatives present. The calculated SD agreed well between AD and DD for the Baldwin–Lomax model cases, where the output function converged in a smooth, continuous, nearly monotonic fashion. Perturbation step size had little effect on the DD when the solutions were converging asymptotically; convergence of the SD via AD and DD were shown to be similar for well-converged cases. The restart convergence level also had little effect on the convergence of the SD either by AD or by DD.

The AD code was shown to be competitive with DD (by forward differencing) for smoothly converging solutions. The AD code would show even better timing comparisons with DD if centered differences were used. The memory required for the AD code was about 2.5 times that required for the original code for propagation of SD with respect to two independent variables.

In cases for which the convergence was poor due to a noisy output, the DD, based upon instantaneous output values, failed to adequately approximate the SD. Meaningful correlation of some SD between AD and DD could be made by time (iteration) averaging of the output functional values before constructing the DD. In these instances, the AD code captured several meaningful digits of each SD without any time averaging. Because the best DD perturbation step size cannot generally be predicted in advance for the construction of all of the SD to reasonable levels of accuracy, the AD solution is much faster than DD when the time required to assess several perturbation step sizes is considered.

In these studies, much more time has been spent trying to verify the AD-generated derivatives using DD than in generating the AD code or in obtaining the SD's via AD. The second most time-consuming task has been identifying and implementing the postprocessing steps required in the AD code to acquire respectable vectorization efficiency on the Cray Y-MP. All things considered, however, the AD-generated derivative code appears to be the best means for obtaining the SD's. The AD method has several distinct advantages in comparison to DD: it requires a short, initial, code-development lead time; verification can be accomplished on coarse grids; small derivative values can be calculated easily; derivatives of noisy output function evaluations can be obtained; and convergence of the SD via AD can be monitored during code execution, whereas SD via DD are usually constructed after the runs are complete, which precludes convergence monitoring during execution. The experience gained in this present feasibility study will be incorporated into future versions of ADIFOR, to increase the vector performance of ADIFOR-generated code and simultaneously decrease the manual postprocessing time.

Despite the advantages previously noted for AD as a means to obtain derivatives, the ADIFOR-generated code will continue to require some manual postprocessing to improve its performance. As indicated in [10], other techniques may be useful in reducing the time required to obtain the SD in future ADIFOR applications to CFD codes. These techniques include application of ADIFOR through the incremental iterative method, which (in 2D) has now been shown [32] to reduce the execution time and memory requirements of the derivative code. Because the multigrid algorithm operator for the flow solution variables is in incremental iterative form, significant improvements may be possible. The multigrid convergence of the SD via AD is monotonic and appears to be rather predictable, independent of the starting convergence level; this characteristic may make it possible to use a 200- to 300-iteration restart file from which to execute the AD code for an

additional 200–300 iterations, while extrapolating the SD from known information.

ACKNOWLEDGMENTS

We thank Chris Bischof, George Corliss, and Andreas Griewank of the Mathematics and Computer Science Division of Argonne National Laboratory; Alan Carle of the Center for Research on Parallel Computation of Rice University, and Paul Hovland of Michigan State University for their hospitality, patience, and stimulating discussions during the critical learning/believing stages of this work. The authors are also deeply indebted to Veer Vatsa for numerous useful and informative discussions concerning the TLNS3D code and to Jim Thomas and Tom Zang (all of NASA Langley Research Center) for giving us the opportunity to pursue this long shot.

REFERENCES

1. H. M. Adelman and R. T. Haftka, (Eds.), *Proceedings, 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, Leon, Norway, May 1989*, AGARD CP-463.
3. *Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, A Collection of Technical Papers, Cleveland, Ohio, Sept. 1992*, AIAA CP-9213.
4. *Thirty-Third AIAA/ASME/ASCE/AHS/ASC Structure, Structural Dynamics and Materials Conference, A Collection of Technical Papers. Dallas, TX, Apr. 1992*, AIAA CP-922.
5. G. S. Dulikravich (Ed.), *Proceedings, Third International Conference on Inverse Design Concepts and Optimization in Engineering Sciences, ICIDES-III, Washington, DC. Oct. 1991*.
6. O. Baysal, (Ed), *Symposium on Multidisciplinary Applications of Computational Fluid Dynamics, ASME Winter Annual Meeting, Dec. 1991*, FED-Vol. 129.
7. P. G. Coen, J. S-. Sobieski, and S. Dollyhigh, AIAA 92-1004, Feb. 1992 (unpublished).
8. J. S-. Sobieski, NASA TM 101566, NASA, Mar. 1989 (unpublished).
9. A. C. Taylor III, G. J.-W. Hou, and V. M. Korivi, *AIAA J.* **30,** No. 10, 2411 (1992).
10. C. Bischof, G. Corliss, L. Green, A. Griewank, K. Haigler, and P. Newman, *Comput. Systems Eng.* **3**(6) (1993); presented at the *Symposium on High-Performance Computing for Flight Vehicles, Dec. 7–9, 1992, Arlington, VA*.
11. V. N. Vatsa and B. W. Wedan, *Comput. & Fluids* **18**(4), 391 (1990).
12. 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*, p. 424, AIAA 92-4743-CP, Sept. 1992.
13. B. Baldwin and H. Lomax, AIAA 78-257, 1978 (unpublished).
14. D. Johnson and L. King, *AIAA J.* **23,** No. 11, 1684 (1985).
15. R. Abid, V. Vatsa, D. Johnson, and B. Wedan, *AIAA J.* **28,** No. 8, 1426 (1990).
16. V. N. Vatsa, M. D. Sanetrik, and E. B. Parlette, AIAA 93-0677, Jan. 1993 (unpublished).
17. 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, Williamsburg, VA, Apr. 1992*, SHPCC-92, p. 276.
18. L. B. Rall, ''Automatic Differentiation: Techniques and Applications,'' Lecture Notes in Computer Science, Vol. 120 (Springer Verlag, Berlin, 1981).
19. A. Griewank, ''On Automatic Differentiation,'' in *Mathematical Programming: Recent Developments and Applications*, edited by M. Iri and K. Tanabe (Kluwer Academic, Boston, 1989), p. 83.
20. A. Griewank and G. F. Corliss (Eds.), *Automatic Differentiation of Algorithms: Theory, Implementation, and Application* (SIAM, Philadelphia, 1991).
21. D. Juedes, ''A Taxonomy of Automatic Differentiation Tools,'' in *Proceedings, Workshop on Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, edited by A. Griewank and G. F. Corliss (SIAM, Philadelphia, 1991), p. 315.
22. C. H. Bischof, A. Carle, G. F. Corliss, A. Griewank, and P. Hovland, *Sci. Programming* **1**(1), 1 (1992).
23. C. H. Bischof and P. Hovland, ANL-MCS-TM-158, Mathematics and Computer Science Division, Argonne National Laboratory, 1991 (unpublished).
24. C. H. Bischof, G. F. Corliss, and A. Griewank, ANL-MCS-TM-159, Mathematics and Computer Science Division, Argonne National Laboratory, 1991 (unpublished).
25. C. H. Bischof, A. Carle, G. F. Corliss, and A. Griewank, ''ADIFOR: Automatic Differentiation in a Source Translator Environment,'' in *International Symposium on Symbolic and Algebraic Computing 92*, edited by P. Wang (ACM, Washington, DC, 1992), p. 294.
26. C. H. Bischof, A. Carle, G. F. Corliss, A. Griewank, and P. Hovland, ANL-MCS-TM-164, Mathematics and Computer Science Division, Argonne National Laboratory, 1992 (unpublished).
27. D. Callahan, K. Cooper, R. T. Hood, K. Kennedy, and L. M. Torczon, *Int. J. Supercomput. Appl.* **2**(4) (1988).
28. A. Carle, K. D. Cooper, R. T. Hood, K. Kennedy, L. Torczon, and S. K. Warren, *IEEE Comput.* **20**(11), 75–89 (1987).
29. C. L. Rumsey and V. N. Vatsa, *J. Aircraft* **32,** No. 3, 510 (1995).
30. C. Hirsch, ''Numerical Computation of Internal and External Flows,'' in *Computational Methods for Inviscid and Viscous Flows, Vol. 2* (Wiley, New York, 1990), Section 22.3.1.
31. A. Griewank, C. Bischof, G. Corliss, A. Carle, and K. Williamson, *Optimization Math and Software* **2,** 321 (1993).
32. L. L. Sherman, A. C. Taylor III, L. L. Green, P. A. Newman, G. J.-W. Hou, and V. M. Korivi, AIAA 99-4262, 1993; presented at the *5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, FL, Sept. 7–9, 1994.*