View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by NASA Technical Reports Server

IN-28

205037

13P

NASA Technical Memorandum 106481
ICOMP–94–2; AIAA–94–0091

Discrete Sensitivity Derivatives of the Navier-Stokes Equations With a Parallel Krylov Solver

N94-24301

Unclas

G3/28 0205037

(NASA-TM-106481) DISCRETE SENSITIVITY DERIVATIVES OF THE NAVIER-STOKES EQUATIONS WITH A PARALLEL KRYLOV SOLVER (NASA) 13 p

Kumud Ajmani
Institute for Computational Mechanics in Propulsion
Lewis Research Center
Cleveland, Ohio

and

Arthur C. Taylor III
Old Dominion University
Norfolk, Virginia

Prepared for the
32nd Aerospace Sciences Meeting and Exhibit
sponsored by the American Institute of Aeronautics and Astronautics
Reno, Nevada, January 10–13, 1994

NASA

INSTITUTE FOR COMPUTATIONAL MECHANICS IN PROPULSION
NASA ICOMP
Lewis Research Center
Ohio Aerospace Institute

DISCRETE SENSITIVITY DERIVATIVES OF THE NAVIER-STOKES EQUATIONS WITH A PARALLEL KRYLOV SOLVER

Kumud Ajmani*
Institute for Computational Mechanics in Propulsion
Lewis Research Center
Cleveland, Ohio 44135

and

Arthur C. Taylor, III†
Department of Mechanical Engineering
Old Dominion University, Norfolk, Virginia 23508

Abstract

This paper solves an 'incremental' form of the sensitivity equations derived by differentiating the discretized thin-layer Navier Stokes equations with respect to certain design variables of interest. The equations are solved with a parallel, preconditioned Generalized Minimal RESidual (GMRES) solver on a distributed-memory architecture. The 'serial' sensitivity analysis code is parallelized by using the Single Program Multiple Data (SPMD) programming model, domain-decomposition techniques, and message-passing tools. Sensitivity derivatives are computed for low and high Reynolds number flows over a NACA 1406 airfoil on a 32-processor Intel Hypercube, and found to be identical to those computed on a single-processor Cray Y-MP. It is estimated that the parallel sensitivity analysis code has to be run on 40–50 processors of the Intel Hypercube in order to match the single-processor processing time of a Cray Y-MP.

Introduction

The development of parallel computers with a large number of processing nodes offers tremendous increases in computer resources for computational scientists. Parallel computers afford the realistic possibility of combining the knowledge of diverse engineering disciplines to produce an integrated, multi-disciplinary effort to solve complex problems in engineering design.

The High Speed Civil Transport (HSCT) objectives of the High Performance Computing and Communications (HPCC) program[1] depend heavily on the design of methodologies for multi-disciplinary analysis and design. Massively parallel implementations of such design methodologies would be required to ensure improved computational efficiency and tractability for large problems. The research in this paper is chiefly motivated by this need for the development of parallel, multi-disciplinary design techniques.

The field of Computational Fluid Dynamics (CFD) has matured sufficiently to the point that flow solutions provided by advanced CFD codes can be reliably used to extract information for use in aerospace vehicle design. A typical CFD code solves a system of partial differential equations (PDEs) on a discrete mesh, for a given, fixed, set of flow conditions (like Mach number, Reynolds' number etc.). However, practical design codes usually require much additional information in the form of Sensitivity Derivatives (SD), in order to produce an optimum design.

A particular CFD code may be extended to calculate aerodynamic sensitivity derivatives which are consistent with the discrete flow solution provided by the code. Each sensitivity derivative quantifies the derivative of a system response (e.g. lift on an airfoil) with respect to an independent design variable (e.g. thickness of the airfoil). A large number of sensitivity derivatives may be required to evaluate the relative influence of all the parameters which affect the vehicle design. Hence, it becomes critical that these sensitivity derivatives be computed 'cheaply', if CFD codes are to be incorporated into practical multi-disciplinary design environments.

The simplest way of calculating sensitivity derivatives is by computing the 'difference' between the two converged CFD solutions which correspond to two different values for the design variable of interest. This method is referred to as the 'finite-difference' method of calculating sensitivity derivatives. Although simple and straightforward, this method can become prohibitively expensive in terms of computational cost, particularly if the number of design variables of interest is large.

The sensitivity derivatives can also be com-

---

* Research Associate, AIAA Member

† Assistant Professor, AIAA Member

Copyright © 1993 by K. Ajmani. Published by American Institute of Aeronautics and Astronautics, Inc. with permission.

puted by differentiating the governing equations of fluid flow ; the differentiation can be undertaken prior to the numerical discretization (continuum method) or after the numerical discretization (discrete method). Both of these methods are computationally efficient as compared to the 'finite-difference' method, and have been compared by Shubin and Frank[2]. The linear system of equations obtained by this differentiation of the governing equations results in the Sensitivity Equations[3,4,5].

This research compares the 'finite-difference' method and the discrete method for calculating sensitivity derivatives on a distributed-memory machine. The sensitivity equations for the discrete approach are derived by direct differentiation of the system of discrete non-linear algebraic equations which model the thin-layer Navier-Stokes (TLNS) equations for 2-D steady flow[3]. The sensitivity equations represent a large system of coupled, linear, algebraic equations, which must be solved to yield the sensitivity derivatives of interest.

The coefficient matrix corresponding to the linear system is large and sparse, and usually reflects poor diagonal dominance. The large size of the matrix rules out solution by direct matrix inversion, as this would require prohibitively large core memory (particularly for 3-D problems). In 'standard' form, the discrete sensitivity equations must be solved 'as is' ; the method admits no approximations to the coefficient matrix. Thus, standard iterative methods may converge very slowly, or may fail due to the lack of diagonal dominance (or poor conditioning) of the coefficient matrix. The 'incremental' form of the sensitivity equations developed by Korivi et. al[6] admits 'approximations of convenience' to the coefficient matrix, and thus overcomes the problems posed in solving the 'standard' sensitivity equations.

The 'incremental' form admits approximations to improve the diagonal dominance of the coefficient matrix (e.g. a time-term may be added to the main-diagonal, certain off-diagonal terms may be neglected etc.). This implies that quasi-Newton iterative methods (which exist in most CFD codes) can be used to solve the discrete sensitivity equations. This also implies that parallel solvers developed for iterative solutions of large, sparse linear systems of equations[7,8,9], can be applied without modification to solve the system of sensitivity equations of interest. The existing literature pertaining to parallel sensitivity analysis seems to be extremely sparse. Some efforts in this direction are being made by Das et. al[10] and Olander et. al[11].

The particular quasi-Newton iterative method used in this paper is based on a preconditioned GMRES[12] solver. This solver has been successfully parallelized and validated for the original CFD code used in this research, in a message-passing environment, on a distributed-memory machine[13]. A domain-decomposition strategy has been used to partition the original problem amongst all available processors. The parallel sensitivity analysis code thus developed is validated on a 32-processor Intel Hypercube.

This paper is organized into four sections. This introduction section is followed by a discussion of (a) the analysis code — the governing equations, spatial discretizations and implicit formulation, (b) sensitivity equations in 'standard' and 'incremental' form, (c) parallel computing issues related to the parallelization of (a) and (b). This is followed by computational results and comparisons of 'finite-difference' and discrete sensitivity derivatives for laminar and turbulent flow cases. The final section is a summary of the present work, and discusses future work in this area.

Presentation of Theory

Parallel Flux-Balance Computations

The governing equations of 2-D compressible fluid flow considered in this study are the thin-layer Navier-Stokes equations written as

$$\frac{1}{J}\frac{\partial Q}{\partial t} = -\frac{\partial F}{\partial \xi} - \frac{\partial G}{\partial \eta} + \frac{\partial G_v^{tl}}{\partial \eta} = R \qquad (1)$$

The governing equations are solved computationally in their integral, conservation law form in generalized coordinates, using an implicit, upwind, cell-centered finite volume formulation. The inviscid fluxes are discretized using Van Leer's[14] flux-splitting scheme. The viscous fluxes are evaluated with second-order accurate central-differences. A nine-point stencil is used for higher-order accurate calculations of the inviscid and viscous fluxes.

In a multiple-processor system, the discretized form of equation 1 is solved by dividing the single, large uniprocessor grid into a number of smaller grids (domain-decomposition) ; each grid is then assigned to one individual processor. Note, that the memory-access for each processor in a distributed-memory environment is limited to the data residing in its local-memory only. This implies that the parallel, multiple-processor calculations will be consistent with the original uniprocessor calculations only if information is exchanged across all grid in-

terfaces which are created because of the domain-decomposition.

Since the computation of the residual vector R uses a nine-point stencil, the flux-evaluation for cell-faces which lie on (and adjacent to) domain boundaries will require information from (a maximum of) two adjacent cells which reside in a neighboring processor. This information is stored in two layers of 'ghost' cells at each domain boundary. Data from the neighboring domains is 'communicated' to these 'ghost' cells, before the flux-evaluation routines are invoked. The implementation of boundary conditions at physical boundaries may also require communication amongst processors. In particular, airfoil calculations on C and O-type meshes require communication between non-neighboring processors in order to effect C and O-type periodicity. This is achieved by communication amongst processors which lie along the 'cut' of the particular C or O-type grid. Further details regarding parallelization of the original CFD code are contained in reference 13.

A discrete numerical steady-state solution of equation 1 implies that

$$\{R(Q^*)\} = \{0\} \tag{2}$$

where $\{R(Q^*)\}$ is the residual vector, for the steady-state solution $\{Q^*\}$. The accuracy of $\{Q^*\}$ is directly affected by the accuracy of computation of the inviscid and viscous fluxes. Equation 2 represents a large system of coupled, algebraic, non-linear equations. An 'implicit' linearization of this non-linear system produces a linear system which can be solved directly by Newton's root-finding method as

$$-\left[\frac{\partial R^n(Q)}{\partial Q}\right]\{^n\Delta Q\} = \{R^n(Q)\} \tag{3}$$

$$\{Q^{n+1}\} = \{Q^n\} + \{^n\Delta Q\} \qquad n = 1, 2, 3, \ldots \tag{4}$$

In most CFD applications, $\left[\frac{\partial R^n(Q)}{\partial Q}\right]$ is a large, sparse, banded matrix, which can be very cumbersome to compute exactly (including consistent boundary-condition linearizations, true flux jacobians etc.). Even if the exact $\left[\frac{\partial R^n(Q)}{\partial Q}\right]$ is available, the core memory required to invert this matrix restricts the practical application of exact Newton's method to all but moderate sized 2-D problems.

Hence, in practice, a quasi-Newton method is used to solve equation 3. An approximate matrix $\left[\frac{\partial \widetilde{R^n(Q)}}{\partial Q}\right]$ is constructed, by introducing linearization errors, adding an artificial 'time-term' to the main diagonal, and/or splitting the original operator into simpler operators. The resulting 'approximate' system of equations is

$$-\left[\frac{\partial \widetilde{R^n(Q)}}{\partial Q}\right]\{^n\Delta Q\} = \{R^n(Q)\} \tag{5}$$

This approximate linear system is then solved iteratively for $^n\Delta Q$, followed by a solution update in equation 4. The tradeoff in using an approximate operator is the reduced error-reduction per time-step as compared to the exact Newton's method. Note, that no approximation is made in computing $\{R^n(Q)\}$ at each time-step, and that the system is solved in 'delta' or 'incremental' form. The 'delta' formulation ensures that the steady-state solutions obtained from the quasi-Newton method and the exact Newton method will be identical.

In the parallel code, the calculation of inviscid and viscous fluxes is followed by 'assembly' of the implicit coefficient matrix. The coefficient matrix is 'assembled' from linear combinations of the individual flux-jacobian matrices for each cell. Each domain computes its flux-jacobian matrices, and no extra communication is required to assemble the final coefficient matrix. This is because a five-point stencil is used to compute the implicit operator, which provides a sparse, banded, coefficient matrix with five block-diagonals.

Thus, each processor (or domain) calculates its own implicit matrix and residual vector, and the original, large, system of linear equations corresponding to the uniprocessor domain is transformed to a series of smaller linear systems, with one linear system for each processor. In this paper, a preconditioned GMRES solver is used to iteratively solve each linear system of equations for each domain. Computationally, the GMRES algorithm involves basic linear algebra kernels, namely, inner-products of vectors, *saxpy* operations and matrix-vector products. These kernels must be parallelized in order to obtain a parallel version of the GMRES solver. The parallel GMRES solver used in this paper has been validated to have the exact convergence rate of the serial GMRES solver[13].

If the implicit coefficient matrix provided to the GMRES solver lacks diagonal dominance (as is the case with the sensitivity equations), the solver converges very slowly to the solution of the corresponding linear system. The convergence rate of the solver can be improved by preconditioning the linear system. Preconditioning can greatly reduce the overall computational effort required to solve the linear system. The Lower-Upper Symmetric Gauss-Seidel

(LUSGS) scheme of Yoon and Jameson[15] is modified into a pointwise-implicit block-solver, for use as a 'local' preconditioner in this work. This preconditioner is applied individually to the linear system in each domain, and is found to be superior to the conventional preconditioners based on incomplete factorizations of the coefficient matrix[16].

Aerodynamic Sensitivity Equations

In general, the j^{th} aerodynamic system response, C_j is dependent on the vector of independent variables $\{Q^*\}$, the vector of grid coordinates $\{\bar{X}\}$, and the vector of design variables, $\{\bar{\beta}\}$. This can be written as

$$C_j = C_j(Q^*(\bar{\beta}), \bar{X}(\bar{\beta}), \bar{\beta}) \tag{6}$$

The sensitivity equations for any particular system response, C_j, can be obtained from equation 6 as

$$\left\{\frac{dC_j}{d\beta_k}\right\} = \left\{\frac{\partial C_j}{\partial Q}\right\}^T \left\{\frac{dQ^*}{d\beta_k}\right\} + \left\{\frac{\partial C_j}{\partial \bar{X}}\right\}^T \left\{\frac{d\bar{X}}{d\beta_k}\right\} + \left\{\frac{\partial C_j}{\partial \beta_k}\right\} \tag{7}$$

Equation 7 represents the total rate of change of C_j with respect to β_k.

The large system of non-linear equations which model the fluid flow (equation 2) can be generalized in the above vein and rewritten as

$$\{R(Q^*(\bar{\beta}), \bar{X}(\bar{\beta}), \bar{\beta})\} = \{0\} \tag{8}$$

The differentiation of equation 8 with respect to β_k yields

$$\left\{\frac{dR}{d\beta_k}\right\} = \left[\frac{\partial R}{\partial Q}\right]\left\{\frac{dQ^*}{d\beta_k}\right\} + \left[\frac{\partial R}{\partial \bar{X}}\right]\left\{\frac{d\bar{X}}{d\beta_k}\right\} + \left\{\frac{\partial R}{\partial \beta_k}\right\} = \{0\} \tag{9}$$

This equation represents the so-called 'standard' form of the sensitivity equations. The equation is solved for the vector of sensitivity derivatives $\left\{\frac{dQ^*}{d\beta_k}\right\}$, for each design variable of interest, β_k. This method of computing sensitivity derivatives is known as the Quasi-Analytical Method.

The matrix $\left[\frac{\partial R}{\partial Q}\right]$ is the Jacobian of the non-linear equations (evaluated at steady-state including consistently linearized boundary conditions) with respect to the field variables. The discrete sensitivity derivatives are represented by the vector $\left\{\frac{dQ^*}{d\beta_k}\right\}$, which signifies the total change in the vector of field variables for a particular design variable, β_k. The matrix $\left[\frac{\partial R}{\partial \bar{X}}\right]$ is the Jacobian of the flow equations (evaluated at steady-state) with respect to the grid coordinates ; $\left\{\frac{d\bar{X}}{d\beta_k}\right\}$ is the grid-sensitivity vector and is computed by the method of Taylor et. al[5]. The vector $\left\{\frac{\partial R}{\partial \beta_k}\right\}$ accounts for the explicit dependencies (if any) of the flow equations (including boundary conditions) on β_k.

Solutions for the 'standard' form sensitivity equations require a direct inversion of $\left[\frac{\partial R}{\partial Q}\right]$ or iterations with (a possibly poorly conditioned) $\left[\frac{\partial R}{\partial Q}\right]$ as the coefficient matrix (similar to solving equation 3). The standard form sensitivity equations are rewritten in 'incremental' form as

$$-\left[\frac{\widetilde{\partial R}}{\partial Q}\right]\left\{{}^m\Delta\frac{dQ}{d\beta_k}\right\} = \left\{\frac{dR^m}{d\beta_k}\right\} \tag{10}$$

$$\left\{\frac{dQ^{m+1}}{d\beta_k}\right\} = \left\{\frac{dQ^m}{d\beta_k}\right\} + \left\{{}^m\Delta\frac{dQ}{d\beta_k}\right\} \tag{11}$$

where

$$\left\{\frac{dR^m}{d\beta_k}\right\} = \left[\frac{\partial R}{\partial Q}\right]\left\{\frac{dQ^m}{d\beta_k}\right\} + \left[\frac{\partial R}{\partial \bar{X}}\right]\left\{\frac{d\bar{X}}{d\beta_k}\right\} + \left\{\frac{\partial R}{\partial \beta_k}\right\} \tag{12}$$

The vector $\left\{\frac{dR^m}{d\beta_k}\right\}$ represents the mth iteration on the total derivative $\left\{\frac{dR}{d\beta_k}\right\}$, and must be driven to zero to find the solution $\left\{\frac{dQ^*}{d\beta_k}\right\}$ of equation 9. Note, that no approximations are allowed in the computation of $\left\{\frac{dR^m}{d\beta_k}\right\}$, in order that the converged solution yields the correct, consistent, discrete sensitivity derivatives.

The solution of equation 10 does allow 'approximations of convenience' for the left-hand-side coefficient matrix $\left[\frac{\partial R}{\partial Q}\right]$. In practice, the approximations are introduced by using a first-order discretization for the coefficient matrix, adding a pseudo 'time-term' to the main-diagonal and neglecting consistent linearization for 'C' and 'O' type boundary-conditions. The major advantage of solving the 'incremental' form of the sensitivity equations over the standard form is that any linear-system solver that is used in the analysis code to solve equation 5 can be used without modification to solve the system of sensitivity equations in equation 10. This is because the characteristics (i.e. sparsity pattern and diagonal dominance) of the coefficient matrices in equation 5 and equation 10 are very similar to each

other. The solution of the 'incremental' form sensitivity equations, as derived by Korivi et. al[6], has been parallelized in this work, and will now be described.

Parallel Solution of Sensitivity Equations

The computational work involved in solving the 'incremental' sensitivity equations can be divided into two parts :

(i) Calculate $\left\{\frac{dR^m}{d\beta_k}\right\}$ from equation 12. Note that the matrix $\left[\frac{\partial R}{\partial Q}\right]$ is computed in parallel (from the steady-state values of the vector Q^*), and re-used at each iteration. The matrix-vector product, $\left[\frac{\partial R}{\partial Q}\right]\left\{\frac{dQ^m}{d\beta_k}\right\}$ is the only vector which needs to be recomputed at each iteration. This matrix-vector multiply needs to be parallelized across all the available processors. Note, that the exact jacobian matrix $\left[\frac{\partial R}{\partial Q}\right]$ has more non-zeroes than the approximate matrix $\left[\frac{\widetilde{\partial R}}{\partial Q}\right]$. This implies that parallel matrix-vector multiplication with the exact jacobian matrix will require more operations than with the approximate jacobian matrix. The matrix-vector product $\left[\frac{\partial R}{\partial X}\right]\left\{\frac{dX}{d\beta_k}\right\}$ and the vector $\left\{\frac{\partial R}{\partial \beta_k}\right\}$ remain constant through the iterative process. Both these vectors are computed in parallel, and stored before the iterative process begins.

Each matrix-vector multiplication is preceded by inter-processor communication. This communication is designed to provide each processor with updated values of $\left\{\frac{dQ^m}{d\beta_k}\right\}$ from the neighboring processors. This ensures that the 'parallel' matrix-vector product is identical to the 'serial' matrix-vector product.

(ii) Solve the linear system of equation 10 by an iterative method. As discussed earlier, the major motivation for developing the 'incremental' sensitivity equations is to use existing CFD solvers (e.g. spatially-split approximate factorization, Krylov solvers like GMRES etc.) to solve the linear system of equations in equation 10. This paper uses a preconditioned GMRES solver to solve this linear system. This parallel solver has been validated for solving the linear system in equation 5, and is an integral part of the existing CFD code[13]. In this work, the parallel GMRES solver incorporates a preconditioner derived from the approximate jacobian matrix, to accelerate convergence to the exact solution vector $\left\{\frac{dQ^*}{d\beta_k}\right\}$.

Recall, that the 'finite-difference' sensitivity derivatives are evaluated by computing successive 'perturbed' CFD solutions for each design variable of interest. For example, if $C_j = C_L$, the steady-state lift-coefficient, and $\beta_k = \alpha$, the angle of attack, then $\frac{dC_L}{d\alpha}$ can be approximated by

$$\frac{dC_L}{d\alpha} \approx \frac{\delta C_L}{\delta \alpha} = \frac{C_L^{\alpha+\Delta\alpha} - C_L^{\alpha-\Delta\alpha}}{2\Delta\alpha} \qquad (13)$$

$C_L^{\alpha+\Delta\alpha}$ and $C_L^{\alpha-\Delta\alpha}$ are obtained by computing new CFD solutions using the converged solution for C_L^{α} as an initial guess. In this paper, the parallel code of reference 13 is used to obtain the 'perturbed' solutions for the 'finite-difference' sensitivities, which are subsequently compared with the 'discrete' sensitivities obtained from the 'incremental' sensitivity equations.

Test Results and Discussion

A parallel, preconditioned GMRES solver has been developed to compute solutions of the sensitivity equations derived from differentiation of the discretized Navier-Stokes equations. The total computational work corresponding to the original single-processor domain is partitioned amongst the various processors of the parallel, distributed-memory machine. The SPMD (Single Program Multiple Data) model of programming is invoked as each processor runs identical copies of the computational code, on different sets of data.

The parallel code is developed on an Intel Hypercube with 32 processors. The results from the parallel sensitivity analysis code are validated against the original serial code (which is run on a single processor of a Cray YMP). The scalability of the domain decomposition algorithm and the preconditioned GMRES solver is tested by running the parallel code on a range of processors (8, 16 and 32). The two problems selected for validation are low Reynolds number subsonic flow over a NACA 1406 airfoil, and transonic turbulent flow over a NACA 1406 airfoil.

Laminar Flow — Subsonic Airfoil

The parallel sensitivity equation solver is first validated for low Reynolds number subsonic flow over a NACA 1406 airfoil. The flow conditions correspond to a freestream Mach number of $M_\infty = 0.6$, angle of attack, $\alpha = 1.0°$, and Reynolds number, $Re = 5.0*10^3$. The computational grid is a 'C' mesh of $257*65$ points, with points clustered near the airfoil surface and the far-field boundary placed five-chords from the airfoil surface. The lift-corrected

boundary conditions are implemented on the far-field boundaries[17].

The parallel, preconditioned GMRES solver is initially used to obtain a converged steady-state solution, $\{Q^*\}$, to the discrete non-linear flow equations (eqn. 2). The parallel validation for the GMRES solver is performed on 8 nodes of the Hypercube with an $8*1$ partitioning of the $257*65$ domain to yield domains of size $33 * 65$ for each processing node. The computed lift, drag, and pitching moment coefficients obtained for the steady-state solution are C_L=0.1815, C_D=0.0417, and $C_M = -0.0237$. These coefficients compare exactly with those computed with the serial version of the code on a Cray Y-MP.

The steady-state solution is used as an initial guess for the 'finite-difference' method (FDM), to compute sensitivity derivatives for C_L, C_D and C_M with respect to the angle-of-attack α, the freestream Mach Number M_∞, and the Reynolds Number Re. The forward and backward perturbations for the 'finite-difference' calculations are set to $\Delta\beta_k = \pm 5*10^{-6}*\beta_k$ (for eqn. 13). A Courant number of 25, with (a maximum of) ten GMRES sub-iterations per time-step, is used to generate a new steady-state solution for each 'perturbed' condition. The l_2 norm of the global residual vector is reduced to a value of 10^{-11} to determine the converged solution for each 'perturbed' variable. A summary of the sensitivity derivatives computed by the parallel 'finite-difference' method on 32 processors of the Intel Hypercube is presented in table 1a. The sensitivity derivatives obtained from the parallel, multiple-domain version of the FDM are identical to those obtained from the serial, single-domain version of the FDM on the Cray Y-MP.

The number of iterations to convergence (n_c) for the FDM calculations are plotted in fig. 1. The values of n_c for α and Re remain fairly constant as the number of processors increases. However, the calculations for M_∞ (Mach No.) show an increase in n_c as the number of processors increases from one (Cray Y-MP) through 8, 16 and 32 (Hypercube). This increase in n_c may be attributed to an increase in stiffness of the coefficient matrix of eqn. 10, as the single-domain problem is partitioned into multiple-domain problems on the parallel machine.

The processing times for the FDM calculations on the Cray Y-MP (1 processor) and the Intel Hypercube (8, 16, 32 processors) are summarized in fig. 2. The processing times shown do not include the time required to compute the initial (unperturbed) steady-state solution. The total time required to obtain all sensitivity derivatives for the three design variables on 32 parallel processors is 1256 seconds, as compared to 778 seconds on the Cray Y-MP. Thus, the 32 processor Intel Hypercube is 61% slower than the single-processor Cray Y-MP when calculating sensitivity derivatives by the FDM. Hence, it may be projected that 52 (or more) parallel processors would be required to match (or exceed) the single-processor performance of the Cray Y-MP for the FDM calculations.

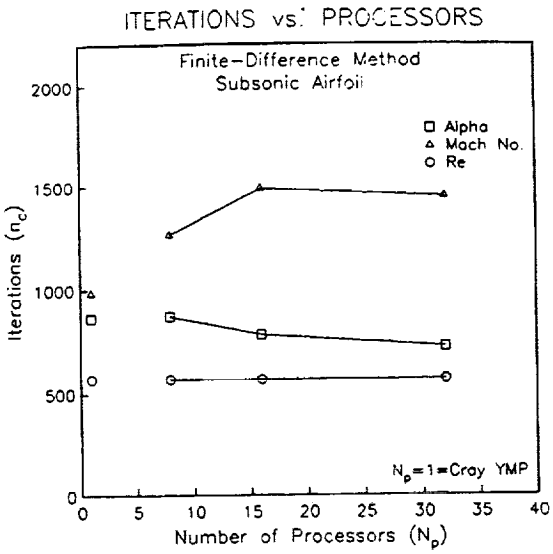

Figure 1. Iterations with FDM (Subsonic)

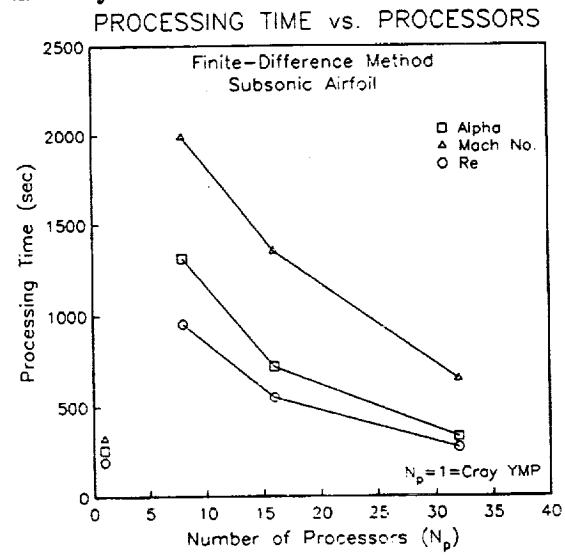

Figure 2. Processing Time with FDM (Subsonic)

The parallel, preconditioned GMRES solver developed for the original CFD code (and used to solve eqn. 5) is applied in the sensitivity analysis code to solve the 'incremental' sensitivity equations (eqn. 10). The steady-state solution, $\{Q^*\}$, is used

in equation 8 to compute the right-hand-side vector $\{\frac{dR^m}{d\beta_k}\}$ for equation 9. The sensitivity derivatives obtained from solving the 'incremental' equations by the Quasi-Analytical Method (QAM) on the parallel machine are summarized in table 1b. The values of the sensitivity derivatives in tables 1a and 1b are identical to five decimal places. This validates the accuracy of the parallel Quasi-Analytical Method for providing highly accurate sensitivity derivatives from solutions of the 'incremental' sensitivity equations (eqn. 10).

The system of sensitivity equations is declared solved when the l_2 norm of the residual vector $\{\frac{dR^m}{d\beta_k}\}$ is reduced to 10^{-6}. A Courant number of 25, with (a maximum of) ten GMRES sub-iterations are used to solve the linear system at each iteration. The variation in the number of iterations to convergence with the number of processors is plotted in fig. 3. As expected, the values of n_c for the three design variables remain fairly constant as the number of processors increases. This demonstrates the scalability of the preconditioned GMRES solver when applied as a linear-system solver for solving the sensitivity equations by the QAM.

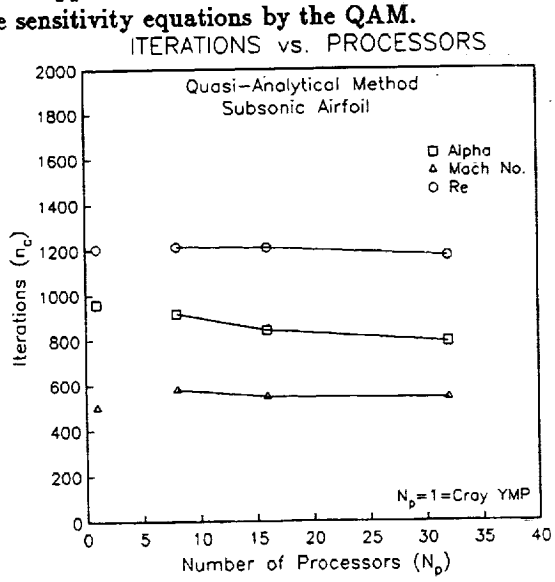

Figure 3. Iterations with QAM (Subsonic)

A comparison of the processing times over various processors for the quasi-analytical method is presented in fig. 4. The total time required on 32 processors to obtain all sensitivity derivatives for all three design variables is 944 seconds, as compared to 667 seconds on the Cray Y-MP. Thus, a complete calculation of sensitivity derivatives by the QAM on 32 processors of the Intel Hypercube requires 42% more processing time than a single processor Cray Y-MP. Assuming a linear speedup for the parallel QAM, 45 (or more) parallel processors would be required to match (or exceed) the single-processor performance of the Cray Y-MP.

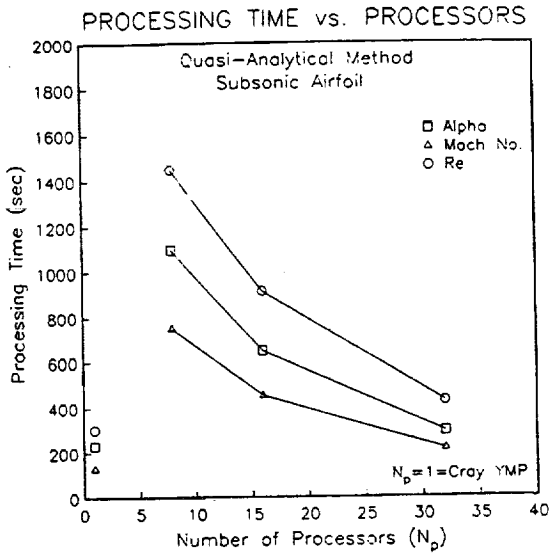

Figure 4. Processing Time with QAM (Subsonic)

A head-to-head comparisons of the parallel FDM and the parallel QAM reveals that the QAM on 32 processors (944 secs.) is 25% faster than the FDM on 32 processors (1256 secs.). As the problem size increases (for denser 2-D grids and 3-D grids) and the computational work per processor increases, the advantage of the QAM will further increase with respect to the FDM. This is because an increase in workload will cause a more significant increase in the 'computation to communication ratio' for the QAM than the FDM.

Turbulent Flow — Transonic Airfoil

This second test case demonstrates the computation of sensitivity derivatives for transonic turbulent flow over a NACA 1406 airfoil. The flow conditions correspond to $M_\infty = 0.8$, $\alpha = 1.0^\circ$, and Re=$5.0*10^6$. A 'C' mesh with 257*65 points is used, with the far-field placed five chord-lengths from the airfoil surface. The clustering near the airfoil surface is much tighter than the previous (laminar) grid, in order to account for the higher Reynolds number of the flow. The laminar viscosity is computed by Sutherland's temperature law, and the turbulent viscosity is modeled by the algebraic model of Baldwin and Lomax[18].

A steady-state solution, $\{Q^*\}$, is first obtained with the preconditioned GMRES solver on 32 processors of the Intel Hypercube. The computed lift, drag and pitching moment coefficients are

C_L=0.4166, C_D=0.7750 E−2, and C_M = −0.4563 E−1. All three coefficients compare identically with those computed by the serial code on a Cray Y-MP. This validates the accuracy of the parallel Baldwin Lomax turbulence model for this test case.

The sensitivity derivatives of C_L, C_D and C_M with respect to α, M_∞ and Re are first calculated by the 'finite-difference' method (FDM). The GMRES solver is used to obtain the 'perturbed' steady-state solutions from the unperturbed solution, $\{Q^*\}$. The time-integration parameters for this test case are identical to those used in the subsonic test case. The sensitivity derivatives obtained by the FDM on 32 processors of the parallel machine are summarized in table 2a. All the sensitivity derivatives are identical to the values obtained by serial calculations with the FDM on a single-processor Cray Y-MP.

The convergence rate of the preconditioned GMRES solver is unaffected by the number of processors used in the FDM calculations. This is clearly evident from the plots in fig. 5. The scalability of the parallel GMRES solver as used in the FDM is thus validated for 32 processors. The processing time characteristics for the three design variables are shown in fig. 6. The total time is dominated by the M_∞ calculation, which is consistent with the results for the subsonic test case. The 32 processor parallel calculations (2792 secs.) are 55% slower than the equivalent single-processor Cray Y-MP calculations (1802 secs.), which implies that 50 (or more) parallel processors would match (or exceed) the Cray Y-MP performance. These projections are very similar to those made for the subsonic FDM calculations.

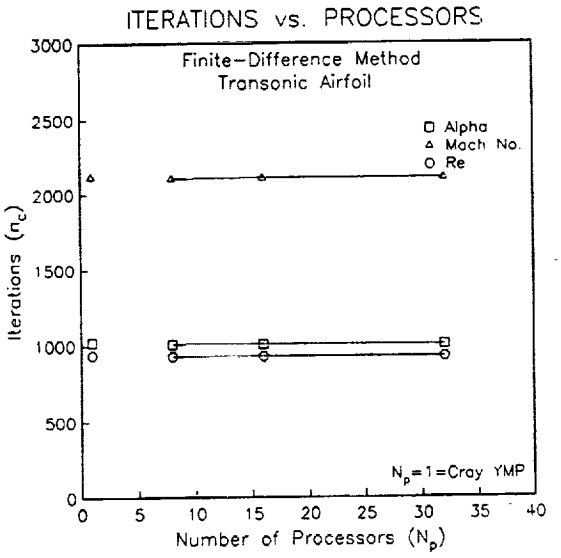

Figure 5. Iterations with FDM (Transonic)

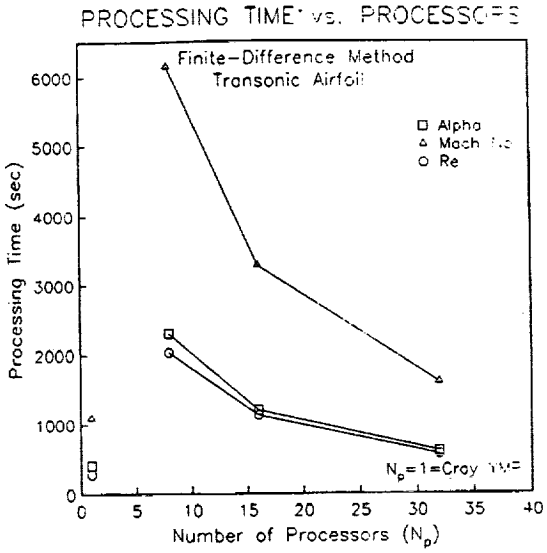

Figure 6. Proc. Time with FDM (Transonic)

The 'incremental' sensitivity equations for this test case are solved by using the parallel preconditioned GMRES solver with a Courant number of 25. The sensitivity derivatives computed with 32 processors are listed in table 2b. These sensitivity derivatives compare exactly with the sensitivity derivatives obtained from serial calculations with the QAM. However, they do exhibit some discrepancies when compared with the parallel FDM calculations. This is because the variation of laminar and turbulent viscosities with respect to the field variables, $\{Q^*\}$, and the computational grid, $\{\bar{X}\}$, is neglected in the numerical construction of the vector $\left\{\frac{dR^m}{d\beta_k}\right\}$ of equation 9. Hence, for turbulent flow cases, the 'incremental' sensitivity equations cannot provide the exact sensitivity derivatives ; the 'finite-difference'derivatives are more accurate in this case. This is true regardless of whether the sensitivity equations are solved on the serial or parallel machines.

The variation in the number of iterations with the number of processors for the QAM is plotted in fig. 7. It is clear that the number of iterations to convergence remains constant for any number of processors. This is an important result as it helps establish the scalability of the preconditioned GMRES solver for parallel sensitivity derivativecalculations with the QAM.

The processing times for the three design variables are plotted in fig. 8. The sensitivity derivativecalculations for Re require the maximum processing time, which is consistent with the results for the subsonic test case. The parallel calculations on 32 processors (2140 secs.) are 39% slower than the

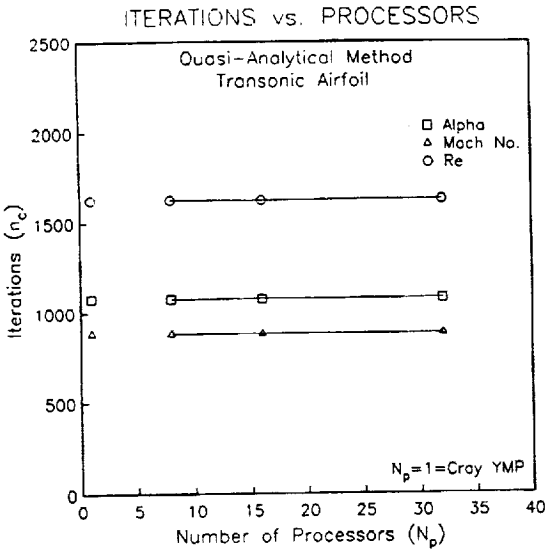

Figure 7. Iterations with QAM (Transonic)

serial Cray Y-MP calculations (1540 secs.). This result compares excellently with the processing time characteristics for the subsonic test case.

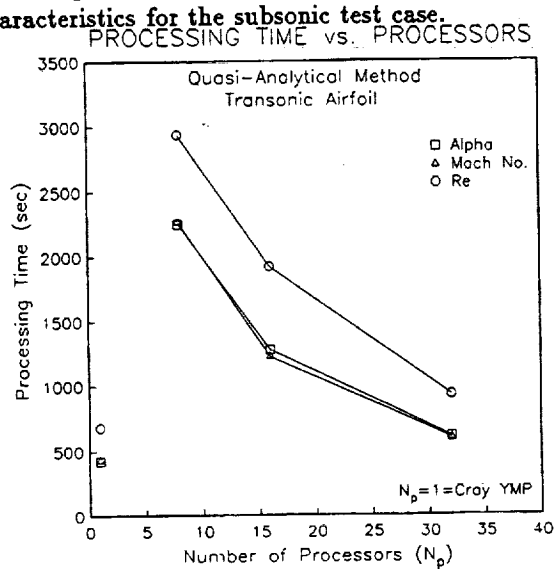

Figure 8. Proc. Time with QAM (Transonic)

A comparison of the parallel FDM and parallel QAM calculations reveals that the latter (2140 secs.) is 24% faster than the former (2792 secs.). This observation is identical to that made for the subsonic test case, and reinforces the fact that the parallel preconditioned GMRES solver performs uniformly for sensitivity derivativecalculations for subsonic and transonic flow conditions.

Conclusions and Future Work

An implicit, scalable, parallel linear-system solver, with a convergence rate independent of the number of parallel processors, is successfully designed and tested for obtaining sensitivity derivatives of the Navier-Stokes equations on a distributed memory parallel machine. The solver is based on a 'locally' preconditioned GMRES algorithm and is constructed with a general domain-decomposition strategy in an SPMD programming framework.

All tests conducted on a 32 processor Intel Hypercube indicate that the parallel GMRES solver provides consistent and accurate sensitivity derivatives for both low Reynolds number (laminar) and high Reynolds number (turbulent) flows. The accuracy of the computed sensitivity derivatives is found to be independent of the number of processors, for both flow conditions tested in this paper. The finite-difference method of calculating sensitivity derivatives is found to be more accurate than the quasi-analytical method, particularly for high Reynolds number (turbulent) flows. The quasi-analytical method of calculating sensitivity derivatives is 25% more efficient than the finite-difference method, in terms of processing time. The parallel processing times for both the low and high Reynolds number test cases indicate that 40–50 parallel processors of an Intel Hypercube would match the performance of a Cray Y-MP. The parallel, preconditioned GMRES solver exhibits similar processing time characteristics and scalability when calculating sensitivity derivatives for both the laminar and turbulent flow cases.

In future work, the procedure for obtaining sensitivity derivatives developed in this paper will be tested on larger parallel machines. This will be done to further study the scalability of the code, and the effectiveness of the parallel solver on large numbers of processors. The sensitivity analysis code will also be ported to a 'cluster' of workstations, in order to study its performance characteristics in a loosely coupled parallel environment. The feasability of obtaining sensitivity derivatives in a parallel environment by automatic differentiation of the Navier-Stokes equations with a software package like ADIFOR[19], will also be investigated.

Acknowledgements

The authors wish to thank Dr. P. A. Newman of NASA Langley Research Center for his continued encouragement and suggestions during this research. Thanks are also due to the Advanced Computational Concepts Lab (ACCL) at NASA Lewis Research Center for providing the computational resources for this work.

References

1. Holst, T. L., Salas, M. D., and Claus, R. W., "The NASA Computational Aerosciences Program — Toward Teraflops Computing," AIAA Paper 92-0558, January 1992.

2. Shubin, G. R., and Frank, P. D., "A Comparison of the Implicit Gradient Approach and the Variational Approach to Aerodynamic Design Optimization," Applied Mathematics and Statistics Technical Report, AMS TR-163, Boeing Computer Services, Seattle, WA, April 1991.

3. Taylor, A. C. III, Hou, G. J.-W., and Korivi, V. M., "Sensitivity Analysis, Approximate Analysis, and Design Optimization for Internal and External Viscous Flows," AIAA Paper 91-3083, September 1991.

4. Sobieski, J. S-., "The Case for Aerodynamic Sensitivity Analysis," in *Sensitivity Analysis in Engineering*, NASA CP-2457, 1987.

5. Hou, G. J.-W., Taylor, A. C. III, and Korivi, V. M., "Discrete Shape Sensitivity Equations for Aerodynamic Problems," AIAA Paper 91-1680, June 1991.

6. Korivi, V. M., Taylor, A. C. III, Newman, P. A., Hou, G. J.-W., and Jones, H. E., "An Incremental Strategy for Calculating Consistent Discrete CFD Sensitivity Derivatives," NASA TM-104207, February 1992.

7. Anderson, E., and Saad, Y., "Preconditioned CG Methods for General Sparse Matrices on Shared Memory Machines," in *Parallel Processing for Scientific Computing*, Proceedings of the Third SIAM Conference, Los Angeles, CA, 1987, pp. 88–92.

8. Gropp, W. D., and Keyes, D. E., "Domain Decomposition with Local Mesh Refinement," NASA-CR-187528, ICASE Report No. 91-19, 1991.

9. Shadid, J. N., and Tuminaro, R. S., "Iterative Methods for Nonsymmetric Systems on MIMD Machines," Sandia National Labs Report No. 90-2689C, Albuquerque, NM.

10. Das, R., Mavripilis, D. J., Saltz, J., Gupta, S., and Ponnusamy, R., "The Design and Implementation of a Parallel Unstructured Euler Solver Using Software Primitives," AIAA Paper 92-0562, January 1992.

11. Olander, D., and Schnabel, R. B., "Preliminary Experience in Developing a Parallel Thin-Layer Navier Stokes and Implications for Parallel Language Design," *in Proceedings, Scalable High Performance Computing Conference*, SHPCC-92, pp. 276–283, Williamsburg, VA, April 1992.

12. Saad, Y., and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal of Scientific and Statistical Computing*, Vol. 7, 1986, pp. 856–869.

13. Ajmani, K., Liou, M. S., Dyson, R. W., "Preconditioned Implicit Solvers for the Navier-Stokes Equations on Distributed-Memory Machines," AIAA Paper 94-0408, January 1994.

14. Van Leer, B., "Flux Vector Splitting for the Euler Equations," *Lecture Notes in Physics*, Vol. 170, 1982, pp. 507–512 (also ICASE Report 82-30, September 1982).

15. Yoon, S., and Jameson, A., "An LU-SSOR Scheme for the Euler and Navier-Stokes Equations," AIAA Paper 87-0600, January 1987.

16. Ajmani, K., "Preconditioned Conjugate Gradient Methods for the Navier-Stokes Equations," Ph.D Dissertation, Virginia Polytechnic Institute and State University, Blacksburg, VA, 1991.

17. Thomas, J. L., and Salas, M. D., "Far-Field Boundary Conditions for Transonic Lifting Solutions to the Euler Equations," AIAA Paper 85-0020, January 1985.

18. Baldwin, B., and Lomax, H., "Thin-Layer Approximation and Algebraic Model for Separated Turbulent Flows," AIAA Paper 78-0257, January 1978.

19. Bischof, C., Carle, A., Corliss, G., Griewank, A., and Hovland, P., "ADIFOR : Generating derivative codes from Fortran programs," *Scientific Programming*, Vol. 1, 1992, pp. 1–29.

Table 1a. Subsonic Airfoil ; Finite-Difference Method

β_k	$dC_L/d\beta_k$	$dC_M/d\beta_k$	$dC_D/d\beta_k$
α	6.1218E+0	9.1815E−2	−3.1690E−2
M_∞	5.4302E−3	1.6279E−2	−4.7328E−3
Re	5.9580E−6	−4.9120E−6	−6.5630E−7

Table 1b. Subsonic Airfoil ; Quasi-Analytical Method

β_k	$dC_L/d\beta_k$	$dC_M/d\beta_k$	$dC_D/d\beta_k$
α	6.1218E+0	9.1813E−2	−3.1675E−2
M_∞	5.4248E−3	1.6279E−2	−4.7296E−3
Re	5.9577E−6	−4.9123E−6	−6.5637E−7

Table 2a. Transonic Airfoil ; Finite-Difference Method

β_k	$dC_L/d\beta_k$	$dC_M/d\beta_k$	$dC_D/d\beta_k$
α	1.2976E+1	4.3337E−1	−6.2317E−1
M_∞	2.0293E+1	1.9710E−1	−5.9554E−1
Re	−1.1112E−9	−2.8051E−10	1.4250E−10

Table 2b. Transonic Airfoil ; Quasi-Analytical Method

β_k	$dC_L/d\beta_k$*	$dC_M/d\beta_k$*	$dC_D/d\beta_k$*
α	1.1981E+1	4.1926E−1	−4.6152E−1
M_∞	1.7419E+0	1.9215E−1	−5.3973E−1
Re	−6.4846E−9	−7.3551E−10	1.3584−9

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1994	Technical Memorandum

4. TITLE AND SUBTITLE

Discrete Sensitivity Derivatives of the Navier-Stokes Equations With a Parallel Krylov Solver

5. FUNDING NUMBERS

WU-505-90-5K

6. AUTHOR(S)

Kumud Ajmani and Arthur C. Taylor III

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E-8411

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, D.C. 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM-106481
ICOMP-94-2
AIAA 94-0091

11. SUPPLEMENTARY NOTES

Prepared for the 32nd Aerospace Sciences Meeting and Exhibit sponsored by the American Institute of Aeronautics and Astronautics, Reno, Nevada, January 10-13, 1994. Kumud Ajmani, Institute for Computational Mechanics in Propulsion, NASA Lewis Research Cetner and Arthur C. Taylor, Old Dominion University, Department of Mechancial Engineering, Norfolk, Virginia 23508, (work funded under NASA Cooperative Agreement NCC3-233). ICOMP Program Director, Louis A. Povinelli, organization code 2600, (216) 433-5818.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 28

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This paper solves an 'incremental' form of the sensitivity equations derived by differentiating the discretized thin-layer Navier Stokes equations with respect to certain design variables of interest. The equations are solved with a parallel, preconditioned Generalized Minimal RESidual (GMRES) solver on a distributed-memory architecture. The 'serial' sensitivity analysis code is parallelized by using the Single Program Multiple Data (SPMD) programming model, domain-decomposition techniques, and message-passing tools. Sensitivity derivatives are computed for low and high Reynolds number flows over a NACA 1406 airfoil on a 32-processor Intel Hypercube, and found to be identical to those computed on a single-processor Cray Y-MP. It is estimated that the parallel sensitivity analysis code has to be run on 40-50 processors of the Intel Hypercube in order to match the single-processor processing time of a Cray Y-MP.

14. SUBJECT TERMS

Sensitivity derivatives; Krylov solvers; Navier-Stokes equations

15. NUMBER OF PAGES

12

16. PRICE CODE

A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	