AIAA JOURNAL
Vol. 34, No. 6, June 1996

Comparison of Numerical and Analytical Jacobians

Kirk J. Vanden*
U.S. Air Force Wright Laboratory, Eglin Air Force Base, Florida 32542-6810
and
Paul D. Orkwis†
University of Cincinnati, Cincinnati, Ohio 45221-0070

The results are presented of a comparison between numerical and analytical approaches for forming the Jacobian matrices that arise from implicit discretizations of the governing equations. The derivation of the exact analytical Jacobians using the MACSYMA symbolic manipulation expert system is discussed in detail. The numerical differentiation procedure used to develop the numerical Jacobians is described. The Jacobian entries obtained from the preceding approaches were tested by comparing the density residuals obtained for selected test cases. The test cases include supersonic flat plate and compression corner geometries. The results indicate that the two approaches are virtually identical, although the numerical Jacobian approach may have some implementation advantages over the exact analytical approach.

Introduction

IMPLICIT discretizations of the governing equations are popular because of their increased stability over explicit discretizations. However, implicit methods usually require the solution of a large system of equations. Whether one chooses a direct, factored, or iterative solution methodology, the individual entries, or flux Jacobians, in the resulting matrix need to be calculated. For some numerical flux calculation procedures, such as that due to Steger and Warming[1] (SW), the flux Jacobians are easily calculated by hand. Because of their simplicity, SW flux Jacobians are often used even when something other than a SW numerical flux is used to discretize the residual. The numerical flux due to Roe[2] is quite popular and has been shown to work well with SW flux Jacobians for many problems. However, to obtain the best possible convergence one should strive to maintain consistency between the numerical flux and the flux Jacobians. This is often quite difficult given the complexity of the expressions for many of the popular numerical flux calculation methods.

In a practical sense, the formation of flux Jacobians through manual differentiation is an unrealistic option. Fortunately, there exist several symbolic mathematics software packages that are capable of performing the necessary work. A number of researchers have taken this approach, including Wigton,[3] Orkwis,[4,5] Orkwis and McRae,[6,7] and Knight,[8] who employed symbolic systems to derive the Jacobian matrix. Orkwis reported that with MACSYMA the differentiation together with the output of error-free FORTRAN code could be performed in under one CPU hour using a current generation workstation. However, this approach produces a very large code that requires significant additional storage to vectorize and is potentially prone to roundoff errors if care is not taken to simplify the results sufficiently.

A number of researchers, including Whitfield and Taylor,[9] Vanden,[10] and Vanden and Whitfield,[11] surmounted this difficulty by employing numerical Frechet derivatives for the Jacobian matrix entries. As pointed out by Whitfield and Taylor the main reason for doing this is to provide for a consistent treatment of the numerical flux and the flux Jacobians. An additional advantage of this approach is the potential for a highly efficient vectorizable code with relatively low operation counts. A potential drawback is that this approach is considered to be approximate. Vanden and Whitfield showed that the actual approximation error is quite small, and hence quadratic convergence can be obtained with this scheme. This indicates that their Jacobian matrix was exact at least to machine accuracy.

Results from both the analytical and numerical Jacobian matrix approaches are compared objectively in this paper to determine the most desirable. The paper includes a brief discussion of the governing equations and Newton's method solver, an in-depth discussion of the analytical and numerical Jacobian matrix formation process, the results obtained for supersonic flat plate and compression corner test case geometries, and concluding recommendations as to the best approach.

Governing Equations

The equations that were solved discretely are the two-dimensional laminar compressible Navier–Stokes equations

$$\frac{\partial F(U)}{\partial x} + \frac{\partial G(U)}{\partial y} = 0 \tag{1}$$

where

$$U = [\rho, \rho u, \rho v, e]^T$$

$$F = \begin{bmatrix} \rho u \\ \rho u^2 + p - \tau_{xx} \\ \rho u v - \tau_{xy} \\ (e+p)u - b_x \end{bmatrix} \qquad G = \begin{bmatrix} \rho v \\ \rho u v - \tau_{xy} \\ \rho v^2 + p - \tau_{yy} \\ (e+p)v - b_y \end{bmatrix}$$

with τ_{ij} and b_i as defined next:

$$\tau_{xx} = \frac{2}{3}\frac{\mu}{R_e}\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right), \qquad \tau_{xy} = \frac{\mu}{R_e}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)$$

$$\tau_{yy} = \frac{2}{3}\frac{\mu}{R_e}\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x}\right), \qquad \tau_{\theta\theta} = \frac{2}{3}\frac{\mu}{R_e}\left(-\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right)$$

$$b_x = \frac{\gamma\mu}{R_e P_r}\frac{\partial e_i}{\partial x} + u\tau_{xx} + v\tau_{xy}, \qquad b_y = \frac{\gamma\mu}{R_e P_r}\frac{\partial e_i}{\partial y} + u\tau_{xy} + v\tau_{yy}$$

$$e_i = (e/\rho) - \tfrac{1}{2}(u^2 + v^2), \qquad p = (\gamma - 1)\left[e - \tfrac{1}{2}\rho(u^2 + v^2)\right]$$

Equation (1) is transformed into generalized coordinates and discretized, as described previously by Orkwis[4,5] and Orkwis and McRae,[6] using Roe's flux difference splitting (FDS) and the Spekreijse/Van Albada[12,13] continuous limiter. Spatially first- and second-order-accurate differences for the numerical flux are employed. The order of the Jacobians is always kept consistent with the order of the numerical flux.

Presented as Paper 94-0176 at the AIAA 32nd Aerospace Sciences Meeting, Reno, NV, Jan. 10–13, 1994; received Feb. 4, 1995; revision received July 21, 1995; accepted for publication July 24, 1995. This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

*Research Scientist, Flight Vehicles Branch. Senior Member AIAA.

†Assistant Professor, Department of Aerospace Engineering and Engineering Mechanics. Senior Member AIAA.

Numerical Method

The discrete equation (1) can be written as a system of nonlinear algebraic equations of the form

$$\mathcal{F}(\bar{U}) = 0 \tag{2}$$

where $\bar{U}$ is the composite vector of the conserved variables at each point in the discretization. Newton's method is formed by creating the system

$$\left(\frac{\partial \mathcal{F}}{\partial \bar{U}}\right)^n \Delta^n \bar{U} = -\mathcal{F}(\bar{U}^n) \tag{3}$$

and solving for $\Delta^n \bar{U}$ to update the $\bar{U}$ iterate. The preconditioned conjugate gradient squared (CGS) solver employed by Orkwis and George[14] was used for this purpose. The Jacobian matrix $\partial\mathcal{F}/\partial\bar{U}$ is updated each iteration using the latest predicted value of $\bar{U}$.

Frequently a diagonal dominance enhancement fix must be incorporated to allow the scheme to iterate until the neighborhood of the final solution is approached. The term used in the current work is similar to the quasitime step of typical relaxation type schemes. An initial value is set that is then scaled by the inverse of the two-norm of Eq. (2).

Jacobian Formation

As shown previously by Orkwis[5] and Liou and Van Leer,[15] Eq. (2) requires an exact Jacobian matrix to achieve quadratic convergence rates. The Jacobian matrix consists of the derivatives of the discretized governing equation with respect to the local conserved variables that form the discretization. The matrix entries are typically zero except for those derivatives from points within the local discretization stencil. The number of entries is equal to the number of points in the local discretization of the governing equations multiplied by the square of the number of governing equations solved at a given point.

For example, consider the two-dimensional continuity equation for incompressible flow

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

A simple central difference of this equation is

$$\frac{u_{i+1,j} - u_{i-1,j}}{\Delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{\Delta y} = \mathcal{F}(\bar{U})_{i,j} \tag{4}$$

The nonzero Jacobian matrix entries corresponding to point (i, j) are

$$\frac{\partial \mathcal{F}_{i,j}}{\partial u_{i+1,j}} = \frac{1}{\Delta x}, \qquad \frac{\partial \mathcal{F}_{i,j}}{\partial u_{i-1,j}} = -\frac{1}{\Delta x}$$

$$\frac{\partial \mathcal{F}_{i,j}}{\partial v_{i,j+1}} = \frac{1}{\Delta y}, \qquad \frac{\partial \mathcal{F}_{i,j}}{\partial v_{i,j-1}} = -\frac{1}{\Delta y}$$

The derivation of the form of these derivatives is relatively simple for functions similar to Eq. (4), but can become quite complicated for more involved discretizations such as Roe's FDS. Consideration of the expense and accuracy of such computations led to the use of the analytical and numerical calculation procedures described next.

Analytical Jacobians

Barth,[16] Wigton,[3] Orkwis,[4] and Knight[8] have all recognized that the man-hour investment needed to derive the analytical Jacobians manually would be enormous and the likelihood of error considerable. For this reason many researchers have utilized symbolic manipulation packages to assist in deriving the Jacobians.

The advantage of symbolic mathematics packages like MACSYMA is that they flawlessly perform the mundane differentiation tasks that humans are most likely to do incorrectly. The disadvantage of MACSYMA is that it does not recognize repeatable patterns in the results and is incapable, without outside intervention, of common simplifications. However, this can be overcome by interactively controlling the routines and by cleverly setting up the problem. A good example of this limitation is that MACSYMA does not employ the chain rule for differentiation in the manner most commonly used by humans.

Consider the following expression used in Roe's FDS discretization of the Navier–Stokes equations, the so-called Roe-averaged velocity:

$$\tilde{u} = \frac{u_L + u_R\sqrt{\rho_R/\rho_L}}{1 + \sqrt{\rho_R/\rho_L}} \tag{5}$$

Assume that a primitive variable higher order interpolation expression, such as

$$Q_L = Q_1 - 2Q_2 + Q_{10}$$

$$Q_R = Q_1 - 2Q_4 + Q_{12}$$

is used, where the subscripts represent locations in the local discretization. If the conserved variable vector is $U = [\rho, \rho u, \rho v, e]^T$, and we let $m = \rho u$ and $n = \rho v$, $\tilde{u}$ becomes

$$\tilde{u} = \frac{\left(\dfrac{m_1}{\rho_1} - 2\dfrac{m_2}{\rho_2} + \dfrac{m_{10}}{\rho_{10}}\right)\left(\dfrac{m_1}{\rho_1} - 2\dfrac{m_4}{\rho_4} + \dfrac{m_{12}}{\rho_{12}}\right)\sqrt{\dfrac{\rho_1 - 2\rho_4 + \rho_{12}}{\rho_1 - 2\rho_2 + \rho_{10}}}}{1 + \sqrt{\dfrac{\rho_1 - 2\rho_4 + \rho_{12}}{\rho_1 - 2\rho_2 + \rho_{10}}}}$$

Clearly the derivatives of this expression will be complicated. If the MACSYMA routine is asked to find the derivative of $\tilde{u}$ with respect to ρ_1, and no guidance is given regarding simplification, the result is as follows:

```
 UTIL = ((-2*M4/R4+M12/R12+M1/R1)*SQRT((-2*R4
1   +R12+R1)/(-2*R2+R10+R1))-2*M2/R2+M10/R10
2   +M1/R1)/(SQRT((-2*R4+R12+R1)/(-2*R2+R10+R1))+1)
 DUTIL = (-M1*SQRT((-2*R4+R12+R1)/(-2*R2+R10
1   +R1))/R1**2+(-2*M4/R4+M12/R12+M1/R1)*(1/(-2*R2
2   +R10+R1)-(-2*R4+R12+R1)/(-2*R2+R10+R1)**2)/
3   SQRT((-2*R4+R12+R1)/(-2*R2+R10+R1))/2.0-M1/
4   R1)**2)/(SQRT((-2*R4+R12+R1)/(-2*R2+R10+R1))+1)
5   -((-2*M4/R4+M12/R12+M1/R1)*SQRT((-2*R4
6   +R12+R1)/(-2*R2+R10+R1))-2*M2/R2+M10/R10+M1/R1)
7   *(2/(-2*R2+R10+R1)-(-2*R4+R12+R1)/(-2*R2+R10
8   +R1)**2)/((SQRT((-2*R4+R12+R1)/(-2*R2+R10+R1))+1)
9   **2*SQRT((-2*R4+R12+R1)/(-2*R2+R10+R1)))/2.0
```

However, MACSYMA can be taught to perform chain rule type operations by properly setting up the code. In this case the results would be simpler if $\tilde{u}$ were written as a function of u_L, u_R, ρ_L, and ρ_R. The derivative can then be written as

$$\frac{\partial \tilde{u}}{\partial \rho_1} = \frac{\partial \tilde{u}}{\partial u_L}\frac{\partial u_L}{\partial \rho_1} + \frac{\partial \tilde{u}}{\partial u_R}\frac{\partial u_R}{\partial \rho_1} + \frac{\partial \tilde{u}}{\partial \rho_L}\frac{\partial \rho_L}{\partial \rho_1} + \frac{\partial \tilde{u}}{\partial \rho_R}\frac{\partial \rho_R}{\partial \rho_1}$$

This approach can be implemented by writing macros that recognize functional relationships. The variables are then written in a cascade of dependencies in which equations are written for u_L, u_R, ρ_L, and ρ_R. The derivatives of the variables that a given term depends upon are taken before the derivative of that term. A name is given to these derivatives so that MACSYMA will use the name instead of the expanded expression when it takes the derivative of the term. When this is done the output takes the following form:

```
RL = -2*R2+R10+R1
RR = -2*R4+R12+R1
UL = -2*M2/R2+M10/R10+M1/R1
UR = -2*M4/R4+M12/R12+M1/R1
UTIL = (SQRT(RR/RL)*UR+UL)/(SQRT(RR/RL)+1)
DRL = 1
DRR = 1
DUL = -M1/R1**2
DUR = -M1/R1**2
DUTIL = ((DRR/RL-DRL*RR/RL**2)*UR/SQRT(RR/RL)
1   /2.0+DUR*SQRT(RR/RL)+DUL)/SQRT(RR/RL)+1)
2   -(DRR/RL-DRL*RR/RL**2)*(SQRT(RR/RL)*UR+UL)/
3   (SQRT(RR/RL)*(SQRT(RR/RL)+1)**2)/2.0
```

It is easy to see that the operation count drops dramatically, although the required memory increases. This illustrates a major difficulty with using the analytical Jacobian approach; that is, although an exact matrix can be formed, the code is difficult to vectorize because of the significantly greater memory requirements caused by the use of temporary variables. In addition, the complexity of the expressions invites potential roundoff errors, although this is typically not a problem with 64-bit machines. The preceding simple example provides ample reason to pursue alternate ways of obtaining Jacobians such as numerical differentiation of the numerical flux vector.

Numerical Jacobians

Whitfield and Taylor[9] point out that numerical Jacobians more easily maintain a consistent treatment with the numerical flux. Although approximate, this approach has been shown to be capable of very fast convergence. In fact, Vanden and Whitfield[11] demonstrated a numerical Frechet derivative based Newton relaxation solver that is capable of quadratic convergence. The $[p, r]$ component of a given flux Jacobian is calculated numerically by using one-sided derivatives

$$\frac{\partial f_p}{\partial u_r} \doteq \frac{f_p(U^n + \epsilon e^r) - f_p(U^n)}{\epsilon} \tag{6}$$

where f_p is the pth component of the numerical flux vector $\mathcal{F}$ and u_r is the rth component of the dependent variable vector associated with the flux Jacobian. Also, e^r is the rth canonical vector and ϵ is a small number that is chosen based on the machine accuracy of the computer being used. A discussion of numerical differentiation can be found in Dennis and Schnabel.[17] Ortega and Rheinboldt[18] show that for discretized Newton methods it is possible to obtain quadratic convergence provided the difference approximation to the Jacobian is computed in a certain way. Thus, when the analytical Jacobian is impossible or too difficult to obtain one can use a numerical approximation and still retain the quadratic convergence. Ortega and Rheinboldt point out that it is necessary that

$$\lim_{\epsilon \to 0} \frac{f_p(U^n + \epsilon e^r) - f_p(U^n)}{\epsilon} = \frac{\partial f_p}{\partial u_r} \tag{7}$$

and

$$\lim_{n \to \infty} \epsilon^n = 0 \tag{8}$$

for quadratic convergence. If a fixed ϵ is maintained for all time steps, then, in general, the convergence rate will be only linear. However, it was reported by Vanden[10] that sequencing of the perturbations was not needed to obtain quadratic convergence for some specific three-dimensional Euler calculations. For the current study ϵ was kept fixed at 10^{-9} for the entire convergence process. Although perturbation sequencing was not attempted, the calculations were repeated with constant values of ϵ of 10^{-8} and 10^{-10}, and the results remained effectively the same. An ϵ of 10^{-9} was chosen based on the method described in Dennis and Schnabel, which takes into account the accuracy of the machine.

The numerical Jacobians require only the coding of the $\mathcal{F}$ discretization in a form that will output $\mathcal{F}$ given the original or altered U vector. Being able to reuse the same code as that for the calculation of $\mathcal{F}$ is perhaps one of the biggest advantages of using numerical Jacobians over analytical Jacobians. However, it is obvious that an inefficient method of calculating $\mathcal{F}$ will naturally lead to inefficient numerical Jacobian calculations.

Results

Results have been obtained for $M = 2.0$ and 8.0 flat plates with $R_e = 1.65 \times 10^6$ and a 10-deg ramp grid with $M = 14.1$ and $R_e = 1.035 \times 10^5$. Both first- and second-order discretizations were computed. The Jacobians, both numerical and analytical, were always the same order as the right-hand side.

Grid clustering for all cases was provided by the function

$$y(j) = y_{\min} + (y_{\max} - y_{\min}) \times \left(1 - s + \frac{2s}{1 + [(s+1)/(s-1)]^{(ny-j)/(ny-1)}}\right)$$

A 40 × 40 grid was used for the $M = 2.0$ flat plate and 40 × 40, 80 × 80, and 160 × 160 grids for the $M = 8.0$ flat plate. A single 200 × 40 grid was used for the ramp case. Grid clustering factors of $s = 1.008$ and 1.004 were used for the flat plate and ramp cases, respectively.

Slug flow initial conditions were used for all cases since the authors felt that these were likely to be the most severe encountered by users.

The test cases were chosen to illustrate the comparative accuracy of the analytical and numerical Jacobians under several conditions, including the effects of increasing Mach number, grid refinement, and solution complexity (via multiple shock interactions). These three effects were tested using both first- and second-order discretizations to illustrate the effects of increased discretization complexity.

Laminar flow calculations were performed because of the inherent difficulty (or impossibility) in deriving exact analytical Jacobians from most turbulence models. Recall that the goal of this study was to compare the two different types of Jacobians; hence, although these flow conditions were almost certainly turbulent, the inclusion of a turbulence model would have obscured this comparison. This is perhaps another area in which the numerical Jacobian approach is more advantageous than the analytical. In some cases, such as with chemistry or a nondifferential turbulence model, numerical Jacobians may be the only suitable approach.

The L_2 norm of the density residual is depicted in each of the figures. Figure 1 shows the density residual for the second-order discretization of the $M = 2.0$ flat plate on a 40 × 40 grid. Figure 2 is the corresponding plot for the $M = 8.0$ case. The figures show that the cases are nearly identical, although the numerical Jacobian residual did not drop as far in the last iterations. However, the numerical Jacobians had a smaller residual until they fell below approximately

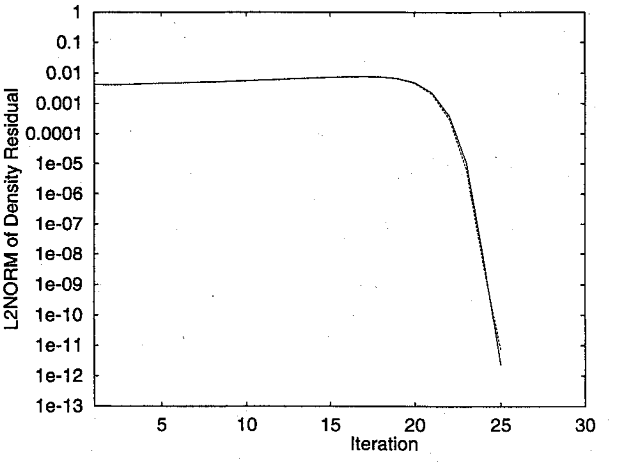

Fig. 1 40 × 40 flat plate at Mach = 2.0, second-order spatial discretization, and initial time step of 1000: ——, analytical and - - - -, numerical.

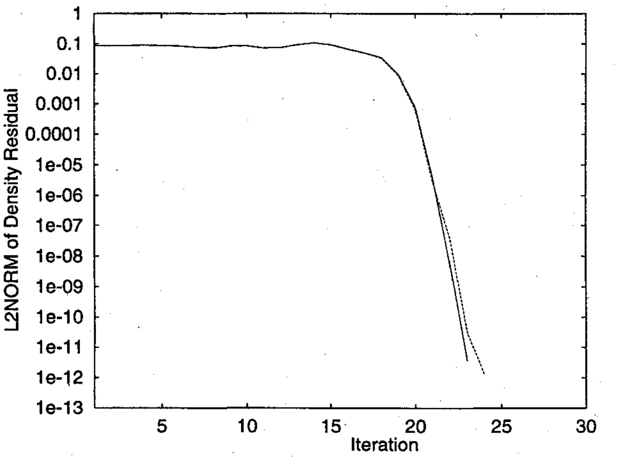

Fig. 2 40 × 40 flat plate at Mach = 8.0, second-order spatial discretization, and initial time step of 1000: ——, analytical and - - - -, numerical.

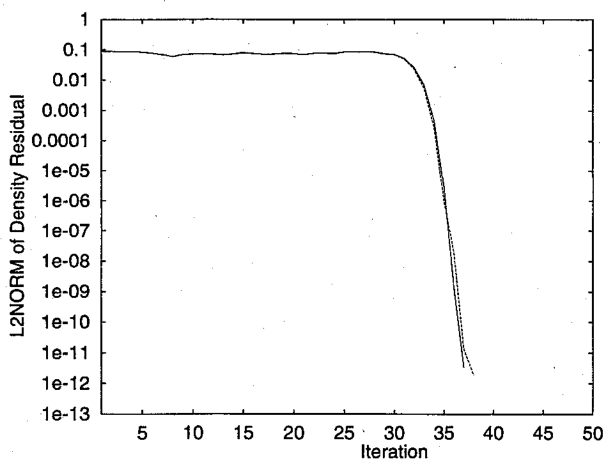

Fig. 3 80 × 80 flat plate at Mach = 8.0, second-order spatial discretization, and initial time step of 1000: ——, analytical and - - - -, numerical.

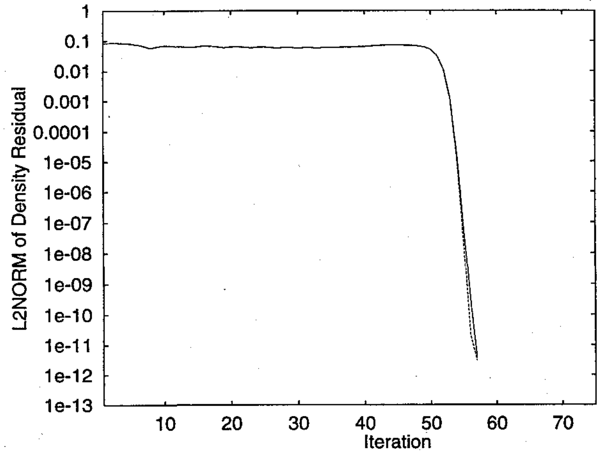

Fig. 4 160 × 160 flat plate at Mach = 8.0, second-order spatial discretization, and initial time step of 1000: ——, analytical and - - - -, numerical.

6 or 7 orders of magnitude in these cases. In addition, the difference between the two approaches expanded very slightly as the Mach number increased. This is possibly due to the steepening of the solution gradients with increased Mach number.

The preceding result should be contrasted with that obtained as the grid is refined. Density residuals for the 40 × 40, 80 × 80, and 160 × 160 grid $M = 8.0$ cases are shown in Figs. 2–4, respectively. These results indicate that more iterations are required as the mesh is refined. In all cases, the numerical Jacobian method converged in a nearly identical manner to that of the analytical Jacobian method, although slight differences did occur. The general shape of the convergence plots did not change appreciably; however, the numerical Jacobian convergence rate improved relative to the analytical as the grid was refined. Hence, although grid refinement adversely affected the total iteration count, it did not affect greatly the relative performance between the two types of Jacobians for this test case.

The 10-deg ramp density residual is shown in Figs. 5 and 6 for initial time steps of 50 and 25, respectively. Here the numerical Jacobian method outperformed the analytical method by a considerable margin near machine zero. In fact, the analytical approach did not converge to machine zero for the calculation shown in Fig. 6. However, residuals up until the analytical hang-up were again approximately identical for the $\Delta t = 50$ and 25 cases. The considerable increase in iteration count as compared with the flat plate cases further illustrates the effect of steep shock waves and multiple shock interactions. The failure of the analytical approach to fully converge can perhaps be attributed to the greater roundoff errors inherent in the calculation of the current analytical Jacobians. However, the greater apparent accuracy of the numericals would probably be academic to most users, since convergence to machine zero is typically not necessary.

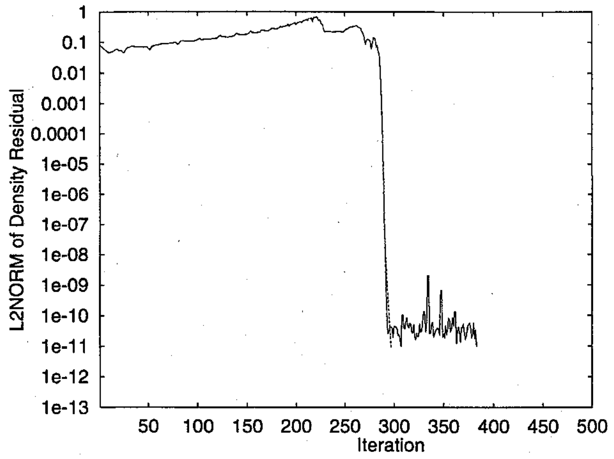

Fig. 5 200 × 40 10-deg ramp at Mach = 14.1, second-order spatial discretization, and initial time step of 50: ——, analytical and - - - -, numerical.

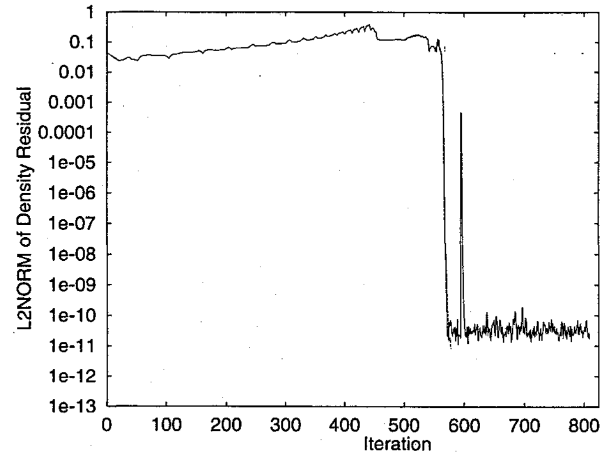

Fig. 6 200 × 40 10-deg ramp at Mach = 14.1, second-order spatial discretization, and initial time step of 25.0: ——, analytical and - - - -, numerical.

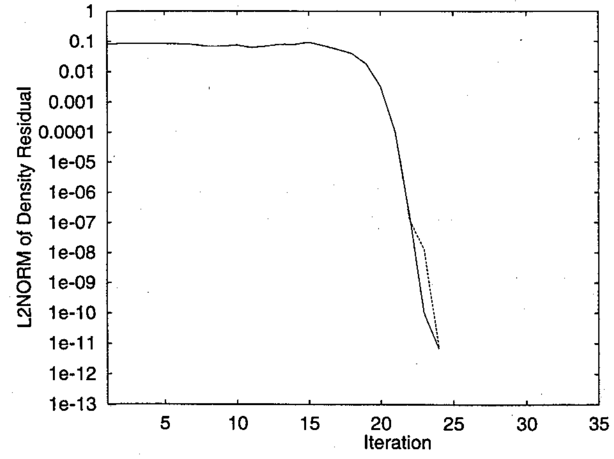

Fig. 7 40 × 40 flat plate at Mach = 8.0, first-order spatial discretization, and initial time step of 1000: ——, analytical and - - - -, numerical.

The preceding second-order results should be compared with the corresponding first-order discretization results shown in Figs. 7–9. Relatively fewer iterations were required to converge these first-order cases as compared with the second-order cases. For example, the first-order numerical Jacobian method converged in approximately 75 fewer iterations than the corresponding second-order scheme for the ramp test case. This is probably due to the smoother final solution for the more dissipative first-order discretization. Again the analytical Jacobians performed marginally better at the

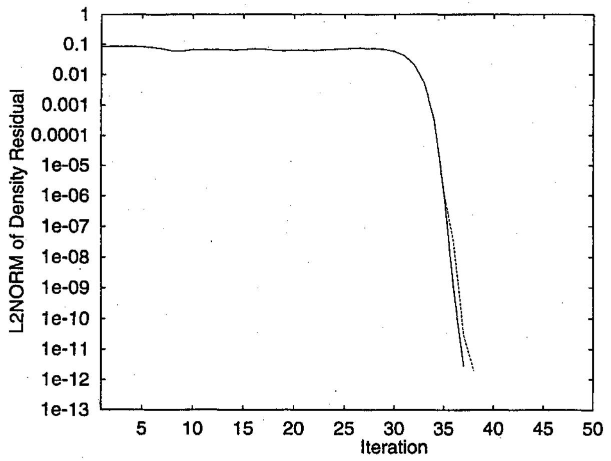

Fig. 8 80 × 80 flat plate at Mach = 8.0, first-order spatial discretization, and initial time step of 1000: ——, analytical and - - - -, numerical.

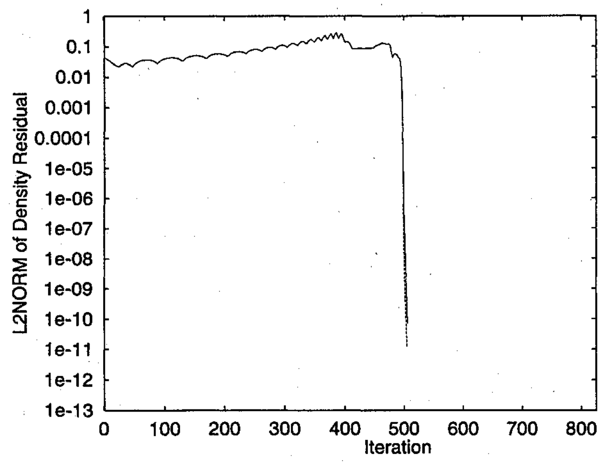

Fig. 9 200 × 40 10-deg ramp at Mach = 14.1, first-order spatial discretization, and initial time step of 25.0: ——, analytical and - - - -, numerical.

lower residual levels for the flat plate case, whereas the numerical Jacobians were slightly better for the ramp case.

It is the opinion of the authors that for all practical purposes the two approaches are identical. Therefore, the two can be used interchangeably. The choice between approaches should then be made based on which better fits the current situation. Analytical Jacobians are better suited for comparatively simple functions, whereas numerical Jacobians are more practical for complicated functions.

Since performance of the schemes cannot be used to determine the best approach, other issues must be considered. In earlier works by Orkwis[4,5] and Orkwis and McRae,[6,7] the analytical Jacobians were formed using the incorrect assumption that viscosity does not depend implicitly on the solution variables. Hence, derivatives of the viscosity with respect to the solution variables were not taken. In a later work by the authors,[19] this assumption was found to damage the convergence rate of the scheme, as the resulting method was outperformed by the numerical approach. Therein lies the beauty of the numerical approach; variable dependencies are automatically accounted for in the Frechet derivatives. This is a distinct advantage because no consideration of dependencies is needed, and hence human error is reduced. This becomes even more important when cases with turbulence models are considered, since many cannot be differentiated analytically. Therefore, in an overall sense the numerical Jacobians can be considered superior unless operation counts are significantly less with the analytical approach.

Conclusions

Tests were conducted to evaluate the relative benefits of numerical and analytical Jacobians. Supersonic flat plate and ramp test cases were computed with a Newton's method solver that employed both approaches. In all cases the two methods converged in virtually identical numbers of iterations. Since the accuracy of the two methods was shown to be effectively the same, the choice between the two methods should be made on other grounds. For relatively simple numerical flux expressions analytical Jacobians may be the most practical and most efficient. But as the complexity of the numerical flux evaluation increases, numerical Jacobians may become the preferred choice. In addition, there may be some cases, such as with chemistry or a nondifferential turbulence model, in which numerical Jacobians may be the only possible choice.

References

[1] Steger, J. L., and Warming, R. F., "Flux Vector Splitting of the Inviscid Gasdynamic Equations with Application to Finite-Difference Methods," *Journal of Computational Physics*, Vol. 40, No. 2, 1981, pp. 263–293.

[2] Roe, P. L., "Some Contributions to the Modelling of Discontinuous Flows," *Large Scale Computations in Fluid Mechanics*, edited by B. E. Enquist, S. Osher, and R. C. J. Somerville, *Lectures in Applied Mathematics*, Vol. 22, 1985, pp. 163–193.

[3] Wigton, L. B., "Application of MACSYMA and Sparse Matrix Technology to Multielement Airfoil Calculations," *Proceedings of the AIAA 8th Computational Fluid Dynamics Conference* (Honolulu, HI), AIAA, New York, 1987, pp. 444–457 (AIAA Paper 87-1142).

[4] Orkwis, P. D., "A Newton's Method Solver for the Two-Dimensional and Axisymmetric Navier–Stokes Equations," Ph.D. Dissertation, Dept. of Mechanical and Aerospace Engineering, North Carolina State Univ., Raleigh, NC, 1990.

[5] Orkwis, P. D., "A Comparison of Newton's and Quasi-Newton's Method Solvers for the Navier–Stokes Equations," *AIAA Journal*, Vol. 31, No. 5, 1993, pp. 832–836.

[6] Orkwis, P. D., and McRae, D. S., "A Newton's Method Solver for High-Speed Viscous Separated Flowfields," *AIAA Journal*, Vol. 30, No. 1, 1992, pp. 78–85.

[7] Orkwis, P. D., and McRae, D. S., "A Newton's Method Solver for the Axisymmetric Navier–Stokes Equations," *AIAA Journal*, Vol. 30, No. 6, 1992, pp. 1507–1514.

[8] Knight, D., "A Fully Implicit Navier–Stokes Algorithm Using an Unstructured Grid and Flux Difference Splitting," AIAA Paper 93-0875, Jan. 1983.

[9] Whitfield, D. L., and Taylor, L. K., "Discretized Newton-Relaxation Solution of High Resolution Flux-Difference Split Schemes," *Proceedings of the AIAA 10th Computational Fluid Dynamics Conference* (Honolulu, HI), AIAA, Washington, DC, 1991, pp. 134–145 (AIAA Paper 91-1539).

[10] Vanden, K. J., "Direct and Iterative Algorithms for the Three-Dimensional Euler Equations," Ph.D. Dissertation, Dept. of Aerospace Engineering, Mississippi State Univ., Mississippi State, MS, 1992.

[11] Vanden, K. J., and Whitfield, D. L., "Direct and Iterative Algorithms for the Three-Dimensional Euler Equations," *AIAA Journal*, Vol. 33, No. 5, 1995, pp. 851–858.

[12] Spekreijse, S. P., "Multigrid Solution of the Steady Euler Equations," Ph.D. Dissertation, Dept. of Numerical Mathematics, Centrum voor Wiskunde en Informatica, Amsterdam, 1987.

[13] Van Albada, G. D., Van Leer, B., and Roberts, W. W., Jr., "A Comparative Study of Computational Methods in Cosmic Gas Dynamics," *Astronomy and Astrophysics*, Vol. 108, No. 1, 1982, pp. 76–84.

[14] Orkwis, P. D., and George, J. H., "A Comparison of CGS Preconditioning Methods for Newton's Method Solvers," *Neural, Parallel, and Scientific Computations*, Vol. 2, No. 2, 1994, pp. 177–194.

[15] Liou, M. S., and Van Leer, B., "Choice of Implicit and Explicit Operators for the Upwind Differencing Method," AIAA Paper 88-0624, Jan. 1988.

[16] Barth, T. J., "Analysis of Implicit Local Linearization Techniques for Upwind and TVD Algorithms," AIAA Paper 87-0595, Jan. 1987.

[17] Dennis, J. E., and Schnabel, R. B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice–Hall, Englewood Cliffs, NJ, 1983, Chap. 5.

[18] Ortega, J. M., and Rheinboldt, W. C., *Iterative Solution of Nonlinear Equations in Several Variables*, Academic, New York, 1970, Chap. 7.

[19] Orkwis, P. D., and Vanden, K. J., "On the Accuracy of Numerical Versus Analytical Jacobians," AIAA Paper 94-0176, Jan. 1994.