AIAA JOURNAL
Vol. 31, No. 5, May 1993

Comparison of Newton's and Quasi-Newton's Method Solvers for the Navier-Stokes Equations

Paul D. Orkwis*
University of Cincinnati, Cincinnati, Ohio 45221

The results of a computational evaluation of several Newton's and quasi-Newton's method solvers are discussed and analyzed. Computer time and memory requirements for iterating a solution to the steady state are recorded for each method. Roe's flux-difference splitting together with the Spekreijse/Van Albada continuous limiter is used for the spatial discretization. Sparse matrix inversions are performed using a modified version of the Boeing real sparse library routines and the conjugate gradient squared algorithm. The methods are applied to exact and approximate Newton's method Jacobian systems for flat plate and flat-plate/wedge-type geometries. Results indicate that the quasi-Newton's method solvers do not exhibit quadratic convergence, but can be more efficient than the exact Newton's method in select cases.

Introduction

A CONSISTENT goal of computational fluid dynamics (CFD) research has been to improve the efficiency of numerical algorithms. One measure of this efficiency is the log of the residual, which varies linearly with iteration count for typical CFD techniques. These methods can be expected to converge at the same rate no matter how close they are to the final solution. An exception to this behavior is the quadratic convergence Newton's method for which the convergence rate actually increases as the final solution is approached. When quadratic convergence has been attained, the number of correct digits in the solution doubles with each iteration. This behavior has been illustrated by several researchers who have obtained solutions of the potential equation,[1] the Euler equations,[2-5] and the Navier-Stokes equations.[2,4,6-13] Unfortunately, the number of iterations is a less-important measure of the efficiency of a code than the total computation time. Quadratic convergence rates are typically accompanied by significant increases in algorithm complexity and hence increased per-iteration computational requirements.

It was the intent of this research to explore how the convergence rate and per-iteration efficiency vary for similar Newton's and quasi-Newton's method solvers so that the most efficient computational strategy can be identified. To accomplish this goal, the quadratic convergence Newton's method solver developed by Orkwis and McRae[6-8] was modified to create several quasi-Newton's method solvers. Approximations were made to the form of the Jacobian and to the Jacobian matrix inversion process. The resulting schemes are evaluated based on their robustness and efficiency for two test problems.

Governing Equations

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

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

where

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

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

with τ_{ij} and b_i as defined below:

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

$$\tau_{xy} = \frac{\mu}{R_e}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)$$

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

$$b_x = \left(\frac{\gamma\mu}{R_eP_r}\right)\frac{\partial e_i}{\partial x} + u\tau_{xx} + v\tau_{xy}$$

$$b_y = \left(\frac{\gamma\mu}{R_eP_r}\right)\frac{\partial e_i}{\partial y} + u\tau_{xy} + v\tau_{yy}$$

$$e_i = \frac{e}{\rho} - \frac{1}{2}(u^2 + v^2)$$

$$p = (\gamma - 1)\left[e - \frac{1}{2}\rho(u^2 + v^2)\right]$$

Equation (1) was transformed into generalized coordinates and discretized, as described previously by the author,[6-8] using Roe's flux-difference splitting (FDS)[14] and the Spekreijse/Van Albada[15,16] continuous limiter. This set of equations was then used to form the Newton's or quasi-Newton's method systems.

Numerical Methods

All of the methods used in this work were derived from the exact Newton's method solver of Orkwis and McRae,[6-8] which has the form

$$\left(\frac{\partial \mathfrak{F}}{\partial \bar{U}}\right)^n \Delta^n \bar{U} = -\mathfrak{F}^n(\bar{U}) \qquad (2)$$

where $\mathfrak{F}(\bar{U})$ is the spatial discretization of the governing equations.

The exact quadratic convergence Newton's method (EN) is formed by creating the Jacobian matrix based on the actual spatial discretization. Quadratic convergence will be obtained if a "close enough" initial guess is supplied and an exact

Received April 24, 1992; presented as Paper 92-2644 at the AIAA 10th Applied Aerodynamics Conference, Palo Alto, CA, June 22-24, 1992; revision received Sept. 12, 1992; accepted for publication Sept. 14, 1992. Copyright © 1992 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Assistant Professor of Aerospace Engineering and Engineering Mechanics, M.L. 70. Member AIAA.

matrix inversion routine is employed. Unfortunately, a "close enough" initial guess is not often known a priori. To avoid this difficulty the $1/\Delta t$ diagonal term modification was used which adds a term to the diagonal of the Jacobian matrix that is modulated by the size of the residual. This modification can either be viewed as an operator that averages the full Newton iterate $\Delta^n \bar{U}$ with $\Delta^n \bar{U} = \emptyset$, or as a modification that makes the scheme a standard time-relaxation algorithm. The result is a more robust scheme with minimal requirements on the initial guess. Orkwis and McRae have shown that quadratic convergence can still be obtained and the method started from poor initial conditions when this modification is included.

The first quasi-Newton's method (QN1) used an approximate matrix inversion routine and the same Jacobian matrix as the EN method. The conjugate gradient squared (CGS) routine of Sonneveld[17] with a slightly modified incomplete lower-upper [ILU(0)] preconditioner was employed. An approximate matrix inverse is obtained by halting the iterations at a convergence level above machine zero. Details of the CGS algorithm are presented below.

The second quasi-Newton's method (QN2) is derived from EN by forming the Jacobian based on a first-order Roe's FDS. This eliminates the outer bands of the original matrix and thereby reduces decomposition fill-in by one-half. The advantage is a significantly improved per-iteration run time, although a linear convergence rate will be obtained.[2]

The third quasi-Newton's method (QN3) is a combination of QN1 and QN2. That is, it employs the approximate inverse CGS routine together with the first-order Jacobian matrix.

Matrix Inversion Techniques

Boeing Real Sparse Library Routine

The direct solver used in this work was a modified version of the Boeing Real Sparse Library (RSLIB) routines.[5] This is a direct lower-upper (LU) decomposition method that minimizes the amount of required in-core storage by writing the matrix and decomposition out-of-core. This allows the user to invert much larger matrices or to run on a smaller memory (usually quicker) queue. The disadvantage is significant input/output (I/O) times which slow down the overall operation of the machine. The modifications to the RSLIB routines consisted primarily of a rearrangement of the matrix input sequence to eliminate costly sorting and checking routines. The new methods demonstrated a 58% savings in computational effort as compared to the original scheme.[9]

Conjugate Gradient Squared Routine

The approximate inverse method used in this work was the CGS routine of Sonneveld.[17] This scheme is a member of a class of iterative procedures that are related to the method of steepest decent.[18] Several recent solvers for aerospace applications[5,10,19] have appeared that make use of the conjugate gradient type routines based on Saad and Schultz' generalized minimum residual (GMRES) algorithm.[20] The CGS routine is similar to the GMRES and bi-conjugate gradient[21] procedure but does not require a functional minimization operation. According to Sonneveld, it is more efficient per iteration and has a greater convergence rate.

The CGS algorithm with left and right preconditioning for solving $Ax - b = \emptyset$ given an initial guess x_o is

```
$r = P_L(b - Ax_o)$
$q = 0$
$p = 0$
$\rho_{-1} = 1$
$n = 0$
while $\|r\| >$ tolerance do
    $\rho = r^T r$
    $\beta = \rho/\rho_{-1}$
    $u = r + \beta q$
    $p = u + \beta(q + \beta p)$
    $v = P_L A P_R p$
    $\sigma = r^T v$
    $\alpha = \rho/\sigma$
    $q = u - \alpha v$
    $v = \alpha P_R(u + q)$
    $x = x + v$
    $r = r - P_L A v$
    $n = n + 1$
    $\rho_{-1} = \rho$
```

This method is guaranteed to converge (in exact arithmetic) in at most N iterations, where N is the number of equations. However, in most cases the method converges much more rapidly. Storage is needed for the preconditioning matrices P_L and P_R, and for five vectors of length N. Proper choice of preconditioner and initial guess x_o play a major role in determining the actual convergence rate. The method becomes approximate if *tolerance* is set above machine zero.

In this work, the original ILU(0) preconditioner and a slight modification were tested. ILU(0) is a member of a general class of incomplete LU decompositions referred to as ILU(n) approximate factorizations. The n value denotes the degree to which fill-in is allowed. ILU(0) refers to a scheme which has zero fill-in and has memory requirements on the order of the matrix storage alone. In general, the greater n becomes the more fill-in is allowed and the more closely the approximate decomposition mimics the exact decomposition.

Although Venkatakrishnan and Mavriplis[11] reported success with an ILU(0)/GMRES method, this author found that the subiterate residual of the ILU(0)/CGS scheme would not converge. It was determined that the order of magnitude of the elements in the approximate L (and U) matrix were considerably different. This problem was corrected by allowing fill-in within the 4×4 blocks of the original matrix. Note that this fill-in pattern is not the same as the complete between band fill-in of a full LU decomposition.

Results

The previously discussed methods were applied to two test cases and analyzed to determine computation time and memory requirements on a Cray Y-MP supercomputer. The test cases were chosen to illustrate the different behaviors of the solvers for supersonic viscous flows with weak shocks of various strengths. Case 1 was a $M_\infty = 2$, $Re = 1.65 \times 10^6$ flat plate, and case 2 was a flat plate/15-deg wedge with the same freestream. The 40×40 grid shown in Fig. 1 was used for case 1, and the 80×40 grid shown in Fig. 2 was used for case 2. A single grid was chosen for each case since grid-resolution studies had already been performed in previous work. Both grids have equal spacing in the x direction and are described in the y direction by the equation

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

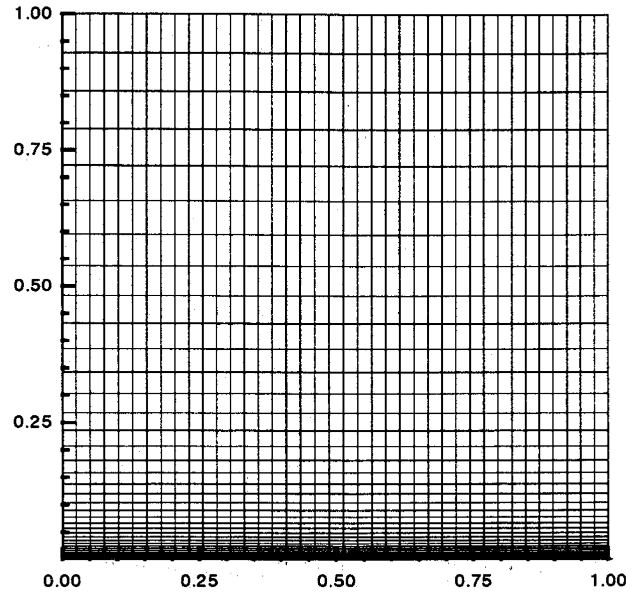

Fig. 1 40 × 40 flat-plate grid.

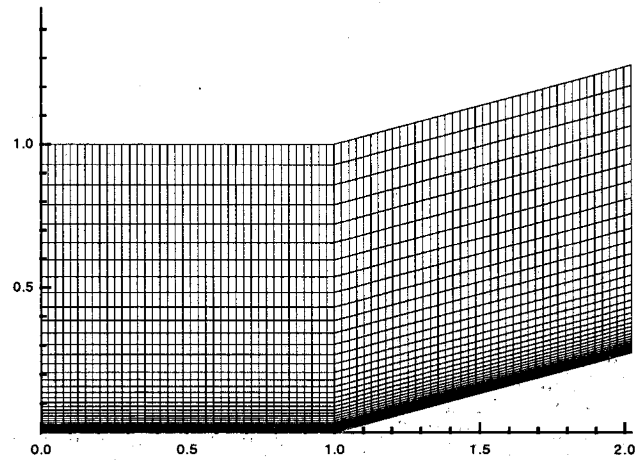

Fig. 2 80 × 40 flat-plate/15-deg wedge grid.

where $s = 1.008$, $y_{\min} = 0$, and $y_{\max} = 1.0$ for the flat-plate region in both cases. Initial Courant numbers were based on $\Delta t_o = 20{,}000$ for case 1 and $\Delta t_o = 10{,}000$ for case 2. Neither of these values should be interpreted as optimal as they were used only to illustrate the behavior of the methods. Freestream initial conditions were prescribed everywhere except at the surface where no-slip was applied.

Figures 3 and 4 are the density residuals obtained with the four methods for case 1. Figure 3 compares the exact Jacobian matrix EN and QN1 schemes and Fig. 4 the approximate Jacobian matrix QN2 and QN3 schemes. Note that the EN method produces a near quadratic superlinear convergence rate, whereas the QN1 CGS method produces residuals that are identical to the EN values until the global residual falls below the subiterate residual tolerance. In contrast, the QN2 and QN3 results indicate that the penalty for an approximate Jacobian matrix is a true linear convergence rate requiring over three times the number of iterations as the EN method.

It should be noted that quadratic convergence can be obtained with the EN method (see Refs. 6–8 and case 2) but that the convergence rate depends heavily on the initial value of Δt and its variation as the final solution is approached. The initial value of Δt and its functional variation in time were chosen only to illustrate the typical behaviors of these schemes.

The aforementioned results seem to indicate that the EN method is the most efficient method for case 1. However, measuring efficiency based on these plots can be deceptive since total computational requirements are more important than the number of iterations needed to converge the solution. Table 1 gives the computer time, memory usage, and total number of iterations for the case 1 solutions. The EN method required ten iterations to converge and 146 s of total CPU time. The QN1 CGS-based routines required between 13 and 22 iterations to converge, which produced run-time reductions of between 8 and 30%. Surprisingly, the largest total reductions came for runs which required the most iterations. This implies that the CGS algorithm becomes significantly more efficient per iteration when larger subiterate convergence tolerances are used.

A similar trend in per-iteration efficiency was experienced with the approximate Jacobian solvers. However, total iteration counts increased dramatically for both the QN2 and QN3 methods. Figure 4 shows that 34 iterations were required to converge the QN2 method and 36 for the QN3 method. Table 1 indicates that the per-iteration run times were reduced by nearly 56 and 80%, respectively, for the two schemes as com-

Table 1 Case 1 computer time and memory usage

Method	CPU time, s	Memory, MW[a]	Iterations
EN	146	1.3	10
QN1 10^{-3}	102	2.2	22
QN1 10^{-6}	114	2.2	19
QN1 10^{-9}	134	2.2	17
QN1 10^{-10}	130	2.2	13
QN2	219	1.3	34
QN3 10^{-3}	103	2.2	36

[a]Megawords.

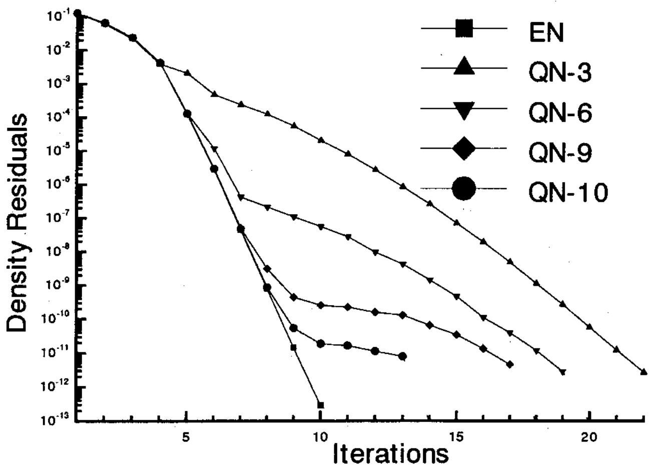

Fig. 3 EN and QN1 case 1 density residuals.

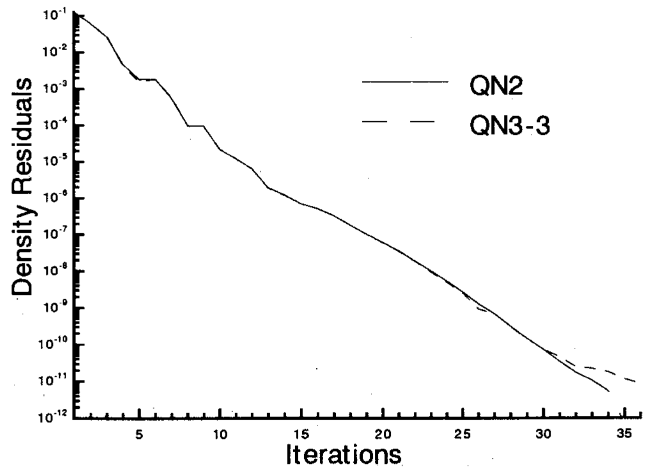

Fig. 4 QN2 and QN3 case 1 density residuals.

pared to the EN method. The combination of increased iteration counts and reduced per-iteration times resulted in a total computation time reduction of approximately 29% for the QN3 method and a computation time increase of nearly 50% for the QN2 method as compared to the EN method.

The case 1 results indicate that the Jacobian matrix and matrix solution approximations afforded by the QN1 with 10^{-3} subiterate tolerance and QN3 with 10^{-3} subiterate tolerance methods provided a similar net reduction in computational requirements. However, it is important to note that an estimation of the work associated with these methods may not be easily obtained since it is directly related to the number of CGS subiterations performed each global iteration. It was found[9] that this number varied considerably for typical runs from as large as 50 during the early iterations to as small as zero for later iterations. In addition, these numbers did not vary monotonically with the global iteration count.

It should be noted that zero subiterations implies that the preconditioner produced an initial guess which made further CGS iterations unnecessary. However, this occurred only for subiterate tolerances above 10^{-10}. No preconditioned initial guesses or CGS subiterates were obtained which were below this value, which prevented the global iteration from converging to machine zero. It is believed that this phenomena is associated with inadequacies in the preconditioner and might be eliminated by using a more accurate preconditioner.

Next, the methods were tested on the flat-plate/15-deg wedge geometry. This configuration contains a stronger wedge-induced shock wave than the bow shock wave formed in case 1, making it a much more severe test case for the approximate methods. To help minimize computational effort, the case 1 results were used as initial conditions for the flat-plate portion of the grid.

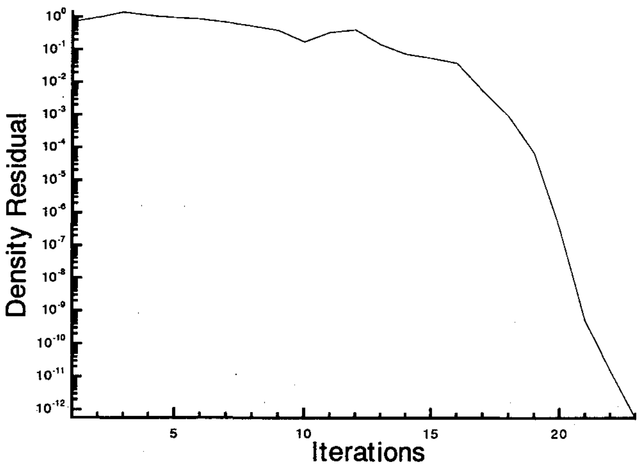

Fig. 5 EN case 2 density residual.

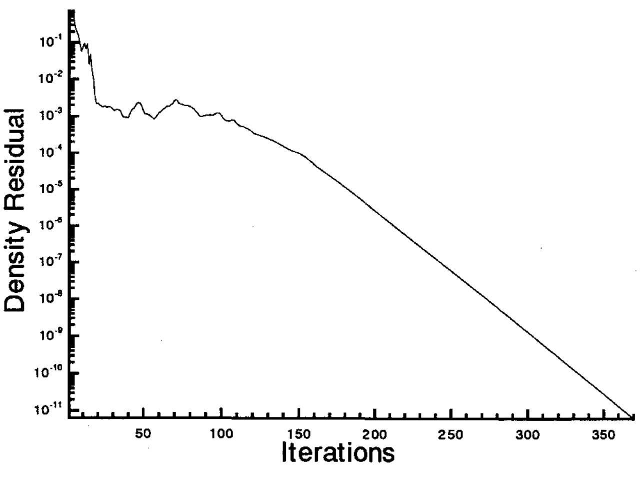

Fig. 6 QN2 case 2 density residual.

Table 2 Case 2 computer time and memory usage

Method	CPU time, s	Memory, MW[a]	Iterations
EN	1052	1.4	23
QN2	7414	1.4	372

[a]Megawords.

Figure 5 shows the density residual for the EN method applied to case 2 and indicates that quadratic convergence was obtained. In contrast, the density residual for the QN2 method, shown in Fig. 6, again illustrates a linear convergence rate for the approximate Jacobian method. Unfortunately, the CGS-based schemes did not converge for case 2 even after the guaranteed maximum number of iterations. It is believed that the increased strength of the wedge-induced shock wave further enhanced the differences in the individual terms of the approximate L and U matrices requiring a significantly more accurate preconditioner.

Table 2 gives the results for the EN and QN2 methods. It shows that 372 iterations were required to converge the approximate Jacobian method whereas 23 iterations were required for the exact Jacobian scheme. Hence, the 44% reduction in per-iteration CPU time experienced with the QN2 scheme was too small to offset the increased iteration count caused by a linear convergence rate. Total CPU time grew by nearly 700% as compared to the EN method.

Conclusions

The results indicate that methods which employ approximate Jacobians and Jacobian inverses do not exhibit quadratic convergence. They show that approximate Jacobian and Jacobian inverse CGS-based methods can provide considerable savings for problems with relatively weak shocks and that the CGS algorithm can produce results with the same accuracy as those obtained with a direct solver when the global residual is greater than the subiterate residual. However, CGS subiterate accuracy to machine zero was never obtained. In addition, approximate inverse CGS routines may not converge for problems with relatively strong shock waves. Finally, it appears that increased shock strength adversely affected the performance of all of the tested methods.

Acknowledgments

This research was supported by the U.S. Air Force Office of Scientific Research Summer Faculty Research Program. Computational resources were provided by Eglin Air Force Base and the Ohio Supercomputer Center. The author wishes to thank Dave Belk, Jack George, and Kirk Vanden for their insights into Newton's method and the CGS routine.

References

[1]Bender, E. E., and Khosla, P. K., "Application of Sparse Matrix Solvers and Newton's Method to Fluid Flow Problems," *Proceedings of the AIAA/ASME/SIAM/APS 1st National Fluid Dynamics Congress,* AIAA, Washington, DC, 1988, pp. 402-408 (AIAA Paper 88-3700-CP).

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

[3]Giles, M., Drela, M., and Thompkins, W. T., "Newton Solution of Direct and Inverse Transonic Euler Equations," *Proceedings of the AIAA 7th Computational Fluid Dynamics Conference,* AIAA, New York, 1985, pp. 394-402 (AIAA Paper 85-1530).

[4]Hafez, M., Palaniswamy, S., and Mariani, P., "Calculations of Transonic Flows with Shocks Using Newton's Method and Direct Solver, Part II," AIAA Paper 88-0226, Jan. 1988.

[5]Wigton, L. B., "Application of MACSYMA and Sparse Matrix Technology to Multielement Airfoil Calculations," *Proceedings of the AIAA 8th Computational Fluid Dynamics Conference,* AIAA, Washington, DC, 1987, pp. 444-457 (AIAA Paper 87-1142-CP).

[6]Orkwis, P. D., "A Newton's Method Solver for the Two-Dimensional and Axisymmetric Navier-Stokes Equations," Ph.D. Disserta-

tion, Dept. of Mechanical and Aerospace Engineering, North Carolina State Univ., Raleigh, NC, 1990.

[7]Orkwis, P. D., and McRae, D. S., "Newton's Method Solver for High-Speed Viscous Separated Flowfields," *AIAA Journal,* Vol. 30, No. 1, 1992, pp. 78–85; also "A Newton's Method Solver for the Navier-Stokes Equations," AIAA Paper 90-1524, June 1990.

[8]Orkwis, P. D., and McRae, D. S., "Newton's Method Solver for the Axisymmetric Navier-Stokes Equations," *AIAA Journal,* Vol. 30, No. 6, 1992, pp. 1507–1514; also *Proceedings of the AIAA 10th Computational Fluid Dynamics Conference,* AIAA, Washington, DC, 1991, pp. 297–307 (AIAA Paper 91-1554).

[9]Orkwis, P. D., "A Comparison of Newton's and Quasi-Newton's Method Solvers for the Navier-Stokes Equations," *Proceedings of the AIAA 10th Applied Aerodynamics Conference* (Palo Alto, CA), AIAA, Washington, DC, 1992, pp. 410–418 (AIAA Paper 92-2644).

[10]Venkatakrishnan, V., "Newton Solution of Inviscid and Viscous Problems," *AIAA Journal,* Vol. 27, No. 7, 1989, pp. 885–891.

[11]Venkatakrishnan, V., and Mavriplis, D. J., "Implicit Solvers for Unstructured Meshes," *Proceedings of the AIAA 10th Computational Fluid Dynamics Conference,* AIAA, Washington, DC, 1991, pp. 115–124 (AIAA Paper 91-1537-CP).

[12]Van Dam, C. P., Hafez, M., and Ahmad, J., "Calculation of Viscous Flows with Separation Using Newton's Method and a Direct Solver," *AIAA Journal,* Vol. 28, No. 5, 1990, pp. 937–939.

[13]Bender, E. E., and Khosla, P. K., "Solution of the Two-Dimensional Navier-Stokes Equations Using Sparse Matrix Solvers," AIAA Paper 87-0603, Jan. 1987.

[14]Vatsa, V. N., Thomas, J. L., and Wedan, B. W., "Navier-Stokes Computations of Prolate Spheroids at Angle-of-Attack," *Journal of Aircraft,* Vol. 26, No. 11, 1989, pp. 986–993.

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

[16]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.

[17]Sonneveld, P., "CGS, A Fast Lanczos-Type Solver for Nonlinear Systems," *SIAM Journal on Scientific and Statistical Computing,* Vol. 10, Oct. 1989, pp. 350–356.

[18]Golub, G. H., and Van Loan, C. F., *Matrix Computations,* 2nd ed., The Johns Hopkins Univ. Press, Baltimore, MD, 1989, pp. 506–509.

[19]Ajmani, K., Ng, W., and Liou, M.-S., "Generalized Conjugate Gradient Methods for the Navier-Stokes Equations," *Proceedings of the AIAA 10th Computational Fluid Dynamics Conference,* AIAA, Washington, DC, 1991, pp. 319–327 (AIAA Paper 91-1556-CP).

[20]Saad, Y., and Schultz, M. H., "GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal on Scientific and Statistical Computing,* Vol. 7, July 1986, pp. 856–869.

[21]Fletcher, R., "Conjugate Gradient Methods for Indefinite Systems," *Lecture Notes in Mathematics 506,* Springer-Verlag, New York, 1976, pp. 73–89.