AN EFFICIENT NEWTON-GMRES SOLVER FOR AERODYNAMIC COMPUTATIONS

Alberto Pueyo* David W. Zingg†

University of Toronto Institute for Aerospace Studies
4925 Dufferin Street, Downsview, Ontario
M3H 5T6 Canada

Abstract

An efficient inexact-Newton-Krylov algorithm is presented for the computation of steady aerodynamic flows. The algorithm uses preconditioned, restarted GMRES in matrix-free form to solve the linear system arising at each Newton iteration. The preconditioner is formed using an ILU(2) factorization of an approximate Jacobian matrix after applying the Reverse Cuthill-McKee reordering. The algorithm has been successfully applied to a wide range of test cases which include inviscid, laminar, and turbulent aerodynamic flows. In all cases except one, convergence of the residual to 10^{-12} is achieved with a CPU cost equivalent to fewer than 1200 function evaluations. The sole exception is a low Mach number case where some form of local preconditioning is needed. Several other efficient implicit solvers have been applied to the same test cases, and the matrix-free inexact-Newton-GMRES algorithm is seen to be the fastest and most robust of the methods studied. Hence this strategy is an excellent option for flow computations in which memory use is not critical, such as two-dimensional applications.

Introduction

Early efforts in the development of efficient solvers for aerodynamic calculations concentrated on approximate factorization methods and methods exploiting multigrid in some manner. Beam and Warming [1] introduced the approximate factorization method for the Euler and Navier-Stokes equations in 1976. Steger [2] used this algorithm in the well-known flow solver ARC2D. ARC2D was further developed by Pulliam [3] with the addition of local time-stepping, diagonalization, and grid sequencing. Recently, multigrid acceleration has been added [4, 5], increasing the convergence rate by factors of three to six.

Multigrid acceleration techniques for transonic potential flows were first introduced in the late 70's [6,7]. Ni [8] and Jameson [9] extended the application of multigrid to the Euler equations. Application to the Navier-Stokes equations was done a few years later by Martinelli et al. [10]. Jameson's approach generally includes an explicit multi-stage iterative method, local time-stepping, and implicit residual smoothing. This approach has received considerable use for aerodynamic flows [11].

In the late 80's, consideration was given to Newton's method as a solver for the Navier-Stokes equations. Venkatakrishnan [12] and Bailey and Beam [13] successfully implemented exact Newton methods. This approach was found to be robust, but memory use and the CPU time required to reach steady state are not competitive with the methods mentioned earlier.

The development of reliable iterative solvers for nonsymmetric matrices, such as GMRES [14], opened the door to the development of quasi-Newton methods. Quasi-Newton methods can be classified as inexact-Newton methods or approximate-Newton methods. In the first category, the large linear system arising at each Newton step is solved approximately. In an approximate-Newton method, the functional Jacobian is simplified, thus producing an approximate linearization. The linear system is again solved iteratively. The advantage of this approach is that the approximate Jacobian has fewer entries and is thus less expensive to use and can be better conditioned. The disadvantage is that the number of Newton iterations is increased. In either case, the use of a good preconditioner is critical to the success of the iterative solver. Preconditioners based on variants of the incomplete lower-upper factorization have proven to be effective.

The range of options available in simplifying the system Jacobian matrix, preconditioning the system, and iteratively solving the system gives rise to a great va-

*Graduate Student.
†Associate Professor, Member AIAA.

riety of quasi-Newton methods. A number of practical aspects in the implementation of Newton-Krylov solvers have been addressed in Refs. 15 to 25. The development and optimization of an inexact-Newton solver using a matrix-free approach to form the matrix-vector products required by GMRES was presented by the present authors [26]. Issues such as optimal choices of preconditioning strategy, ordering of the unknowns and tolerance level at each Newton-step for specific problem classes were addressed. Furthermore, Ref. 26 demonstrated that the inexact-Newton approach can be more efficient than the approximate-Newton approach.

In this paper, we present an improved version of the inexact-Newton solver given in Ref. 26. The solver, which is known as PROBE, is evaluated for a wide range of flows over airfoils and compared to other efficient implicit solvers. The objective is to demonstrate that the present solver is both efficient and robust and thus provides a promising alternative to the solvers currently in use.

Algorithm description

Spatial discretization The spatial discretization is the same as that used in ARC2D[3]. It consists of second-order centered-difference operators with the second and fourth-difference artificial dissipation model of Jameson et al. [27] A far-field circulation correction is also included. The Baldwin-Lomax algebraic model [28] is used for turbulent flows. The steady-state solutions of PROBE are thus identical to those computed using ARC2D.

Inexact-Newton solver The spatial discretization leads to a nonlinear system of equations in the form

$$F(Q) = 0 \tag{1}$$

where Q contains the conservative flow variables. An inexact-Newton method is a generalization of Newton's method where, at each step k, we try to find ΔQ_k such that

$$\|F(Q_k) + A(Q_k)\Delta Q_k\| \leq \eta_k \|F(Q_k)\| \tag{2}$$

where A is an exact linearization of $F(Q)$ given by

$$A = -\frac{\partial F(Q)}{\partial Q} \tag{3}$$

If $\eta_k = 0$, we recover Newton's method. During the first few iterations, the solution is far from the converged solution, and thus the linear approximation of F can be very inaccurate. The use of a strict tolerance (i.e., a small value of η_k) during these early iterations is thus not beneficial to the rate of convergence and wastes CPU time. This is known as oversolving [29]. Strategies for choosing a sequence of η_k's leading to an efficient rate of convergence of the inexact-Newton method without oversolving have been developed by several authors [29]. We found that choosing

$$\eta_k = \gamma \left(\frac{\|F(Q_k)\|}{\|F(Q_{k-1})\|} \right)^{\alpha} \tag{4}$$

with $\gamma \in [0,1], \alpha \in (1,2]$, and $\eta_0 \in [0,1)$ is an effective strategy. However, for our applications, we have found the following approach to give slightly better efficiency. We use $\eta_k = 0.5$ for the first 10 iterations. This guarantees no oversolving for most problems. We then use $\eta_k = 0.1$ for the remaining iterations. This approach is similar to those given in Refs. 30 and 31. Note that "undersolving" at each Newton step is not particularly wasteful, while oversolving should be avoided.

GMRES There are several effective iterative solvers for non-symmetric linear systems available, as reviewed by Dutto [32]. It is very difficult to establish general rules about which one is *the best method*. This depends on the particular problem one is attempting to solve. We found GMRES [14] to be the most efficient Krylov iterative solver for the problems addressed in our studies. This solver has the property of minimizing, at every step, the norm of the residual vector over a Krylov subspace. The disadvantage of GMRES is that its storage requirements increase linearly with the number of search directions in the Krylov subspace, and its CPU expense increases quadratically. For this reason, we use a restarted version of the algorithm limiting the search directions to 20. In order to make a distinction between the Newton iterations on the non-linear problem and the GMRES iterations on the linear one, the terms "outer iterations" and "inner iterations", respectively, will be used.

Since GMRES requires only matrix-vector products, the algorithm can be implemented without forming the Jacobian matrix explicitly. It was shown in [30] that such a matrix-free approach can be advantageous, from the point of view of both performance and storage. Barth [24] developed a technique to compute the exact product of the second-order Jacobian and a vector, using linearizations of the lower-order Jacobian. In this paper the finite-difference technique is employed, where the product is approximated by

$$A(Q_k)v \simeq \frac{F(Q_k + \varepsilon v) - F(Q_k)}{\varepsilon} \tag{5}$$

where $F(Q_k)$ is the current residual vector and ε is a small scalar used to perturb the state quantities Q_k in the direction of v. The performance of this technique is sensitive to ε [33]. Following Nielsen et al. [23], ε is determined such that

$$\varepsilon \, \| v \|_2 \simeq \sqrt{\varepsilon_m} \tag{6}$$

where ε_m is the value of machine zero for the hardware used.

ILU preconditioner Incomplete LU factorizations are efficient preconditioners for Krylov solvers. There have been two distinct approaches to forming such incomplete factorizations. The first approach, named ILU(n), uses only the graph of the matrix to determine which entries to drop. ILUT(p,τ), developed by Saad [34], uses two rules to determine which elements should be dropped at a given row. The first rule consists of dropping any element smaller than a relative tolerance determined by τ and a norm of the original matrix. The maximum number of non-zeros in a given row is controlled by the second parameter p. When using ILU(n), we have found $n = 2$ to be optimal. For ILUT(p,τ), $p = 15$ and $\tau = 0.1$ generally lead to good performance. A comparison between ILU(2) and ILUT(15,0.1)is presented in [26]. ILU(2) proved to be both faster and more robust. In all cases we allow fill-in within the 4×4 blocks of the matrix used to build the preconditioner.

ILU preconditioners can be built using the second-order Jacobian or any reasonable approximation. There are some advantages in using an approximation. For instance, if we form the Jacobian using only second-difference dissipation, the resulting matrix has five blocks per node, instead of nine, which produces a significant reduction in memory requirements. It has been shown by Pueyo and Zingg [26] that choosing an approximate-Jacobian can be more effective than using the actual second-order Jacobian. An efficient approximation consists in modifying the second-difference dissipation coefficients on the left-hand-side by

$$\epsilon_2^l = \epsilon_2^r + \sigma \epsilon_4^r \tag{7}$$

where ϵ_2^r and ϵ_4^r are the coefficients of the second- and fourth-difference artificial dissipation on the right-hand side. Our experiments have shown that a good general value for the parameter σ in equation (7) is 9.

A second approximation is introduced to make the matrix more diagonally dominant by adding $1/\Delta t$ to the diagonal, where Δt is a local time step given by,

$$\Delta t = \frac{\Delta t_0}{1 + \sqrt{J}} \tag{8}$$

where J is defined to be the metric Jacobian of the generalized curvilinear coordinate transformations,

$$J = \frac{1}{x_\xi y_\eta - x_\eta y_\xi} \tag{9}$$

and Δt_0 is a constant. For all cases shown, $\Delta t_0 = 200$. Note that these approximations are made only in forming the preconditioner. They do not affect the convergence of the outer iterations.

It was found that freezing the preconditioner after a few iterations does not significantly increase the number of iterations required by GMRES, and in some cases even decreases them. The most efficient strategy for our applications is to compute the preconditioner only once.

Ordering of unknowns Since it can greatly affect the quality of the incomplete factorization, the ordering of the unknowns plays an important role in the convergence of the iterative solver [19, 35]. Several ordering strategies have been compared in [26]. The Reverse Cuthill-McKee strategy [36], a well-known bandwidth reduction algorithm, was shown to be the most efficient of those studied, which included two natural orderings and the minimum neighbouring ordering.

Start up For some flow cases, especially those with shocks, the early Newton iterations can diverge. Different relaxation techniques have been suggested to overcome this difficulty. One way to relax the solution is to damp the Newton updates to prevent the calculation of non-physical variable values [37]. An alternative to this technique is to rewrite equation (1) as

$$\frac{dQ}{dt} = F(Q) \tag{10}$$

and to apply implicit Euler time-differencing. A finite time step can be used initially and, as $\Delta t \to \infty$, the inexact-Newton method is obtained. This strategy is used by Mulder and Van Leer [38]. However, when a finite time step must be used, a cheaper relaxation algorithm can be employed [23] [39], significantly reducing computing time. This is particularly true for transonic flows, where many outer iterations at low Δt can be needed before fast convergence can be achieved. This can be computationally expensive even when using an approximate-Newton method. In the present study, the approximately-factored algorithm of ARC2D in diagonal form is used with two levels of mesh sequencing. We do 150 iterations or reduce the residual two orders of magnitude, whichever comes first, on the coarse grid, followed by ten iterations on the fine grid.

Summary All results presented using PROBE were obtained using the following strategies and parameters:

- inexact-Newton strategy
- matrix-free GMRES(20)
- ILU(2) preconditioner based on the first-order Jacobian formed using $\Delta t_0 = 200$ and $\sigma = 9$.
- Reverse Cuthill-McKee reordering
- preconditioner computed at first iteration and not updated

case	airfoil	Ma	α	Re	tr.up	tr.low
1	NAC	0.63	2.00	invisc	—	—
2	NAC	0.80	1.25	invisc	—	—
3	NAC	0.80	5.00	5.00e2	—	—
4	NAC	0.30	0.00	2.88e6	0.43c	0.43c
5	NAC	0.30	6.00	2.88e6	0.05c	0.80c
6	NAC	0.70	1.49	9.00e6	0.05c	0.05c
7	NAC	0.16	12.00	2.88e6	0.01c	0.95c
8	RAE	0.729	2.31	6.50e6	0.03c	0.03c

Table 1: Parameters for the six flows studied. NAC refers to the NACA 0012 airfoil and RAE to the RAE 2822 airfoil.

- inner tolerance (η_k) set to 0.5 for the first ten iterations, 0.1 for the remainder
- approximate factorization algorithm used to reduce the residual two orders of magnitude initially

Test cases

Seven test cases are studied using the NACA 0012 airfoil and one using the RAE 2822 supercritical airfoil. Two are inviscid flows, one laminar, and five are turbulent. The parameters defining the test cases are given in Table 1. The initial condition is always freestream flow.

For the inviscid cases, the grid used has 249×39 nodes with the wall spacing set to 2×10^{-3} chords. For the laminar case, the grid used has 249×49 nodes and a wall spacing of 5×10^{-4} chords. A 331×51 grid with the wall spacing set to 1×10^{-5} chords is used for the NACA 0012 turbulent cases. For the RAE airfoil case, the grid has 321×49 nodes with similar wall spacing. In some tests which include comparisons with multigrid, the number of nodes may vary slightly in order to be able to obtain coarser grids by removing every other point in the finer grid. These grids provide reasonable numerical accuracy for the flows considered.

Units for comparing efficiency

When comparing the speed of different algorithms, CPU time is the appropriate unit. However, this is dependent on the computer, compiler, and the coding details. Furthermore, in comparing iterative methods, one would like to exclude the cost of the flux evaluation from the comparison. Although it is by no means perfect, the number of function evaluations (or right-hand-side evaluations) required to reduce the residual by a given amount is a useful unit for assessing the speed of an iterative algorithm [40]. This unit allows the relative performance of different algorithms to be compared across various platforms, compilers, and flux evaluation methods. Shortcomings of this choice are that it tends to favour expensive flux evaluation methods (overhead appears smaller) and there is some arbitrariness as to what is included in a function evaluation. For example, local time stepping and the circulation correction are optional.

For most of our comparisons, we will use the number of function evaluations as our basic unit. In the function evaluation we include the flux evaluation, pressure field update, the computation of the artificial dissipation coefficients, the computation of the molecular and eddy viscosity and the evaluation of the right-hand-side at the boundaries. This permits comparison with other solvers. Since all of the methods compared in this paper use the same right-hand-side, the number of function evaluations translates directly into CPU time on a given computer.

In addition to speed, memory use is also an important consideration in algorithm development. PROBE requires the storage of approximately 280 real numbers per node. This is not competitive with many current solvers, and thus the inexact-Newton approach is presently restricted to applications where memory use is not a dominant concern, such as two-dimensional flows. Development of an effective matrix-free preconditioner would be a significant advance.

Performance of the algorithm

Ideally, a solver should be able to solve different flow conditions on varying grids with relatively consistent performance. Figure 1 shows the convergence histories for the 8 cases that we have studied. Table 2 shows some statistics for the algorithm, including the number of inner and outer iterations required to reduce the residual norm by twelve orders of magnitude (to machine zero) are shown. The outer iterations include only those done using the Newton-Krylov solver and not those of the approximately-factored algorithm. CPU/f.e. time gives the total run time normalized by the CPU time of a function evaluation. Figure 1 shows that except at low Mach numbers, convergence is achieved in 600 to 1200 function evaluations. Case 7, in which the freestream Mach number is 0.16, shows a slower convergence. The number of inner iterations go up dramatically for this case. In order to minimize oversolving in cases like this, we always limit the maximum number of inner-iterations at each quasi-Newton step to 40. We are examining the use of local preconditioning to improve the performance of the solver at lower Mach numbers [41].

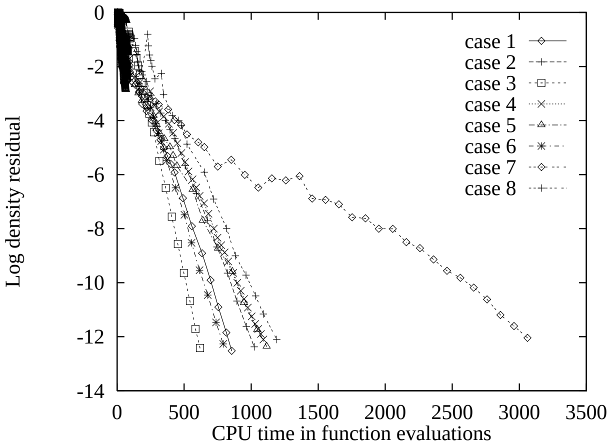

Figure 1: Residual history for PROBE for the 8 cases described in Table 1.

	o-it	Σi-it	i-it/o-it	CPU/f.e.
Case 1	18	209	11.6	823.8
Case 2	19	255	13.4	993.1
Case 3	19	172	9.1	598.6
Case 4	18	379	21.1	1082.3
Case 5	18	413	22.9	1075.2
Case 6	18	278	15.4	773.5
Case 7	35	1174	33.5	3051.1
Case 8	25	430	17.2	1180.4

Table 2: Statistics for the Newton-Krylov algorithm for the cases studied; **o-it**: outer iterations, **Σi-it**: total number of inner iterations, **i-it/o-it**: average number of inner iterations per outer iteration; **CPU/f.e.)**: CPU time in equivalent function calls to reduce the residual by twelve orders of magnitude.

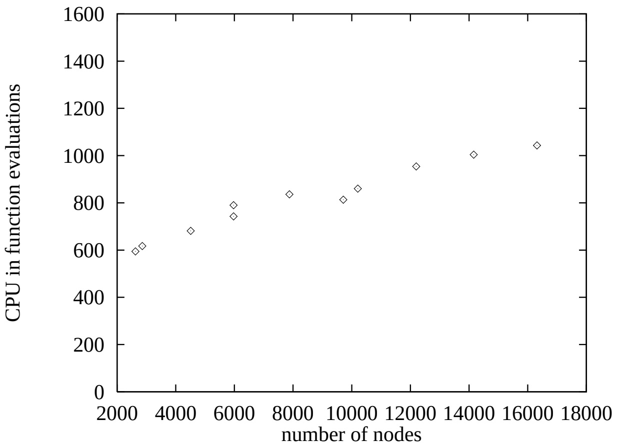

Figure 2: CPU time in function evaluations required to converge to machine zero for case 1 as a function of grid size.

Figure 2 shows the number of function evaluations needed to reach convergence for case 1 as we increase the grid density. A factor of six increase in the number of nodes leads to an increase in the number of function evaluations of less than a factor of two. Although we would like to see a constant number of function evaluations, this performance is competitive with other solvers.

Comparison with other solvers

We now compare our inexact-Newton-Krylov strategy with other implicit solvers.

Approximate factorization (ARC2D) The approximate factorization algorithm in diagonal form, as used in ARC2D, is explained in detail in Ref. 3. In ARC2D, the wakecut can be treated implicitly or explicitly. In this paper we consider only the explicit treatment of the wakecut, which is faster for our present test cases.

Approximate factorization with multigrid (ARC2D-MG) Other authors [4, 5] have already shown that multigrid can substantially increase the convergence rate of the approximate factorization algorithm. For our study, we use a three level V-cycle.

Incomplete factorization (ILU(2)) ILU(2) has proven to be an efficient preconditioner for our applications. It can also be used as a solver, with the same strategy as used for the preconditioner: we use the first-order Jacobian to form the factorization, with values of 9 and 200 for the parameters σ and Δt_0, and we compute the factorization once and reuse it at each iteration.

Approximate-Newton (approx. Newton) Some of the most popular approximate-Newton methods use a first-order Jacobian on the left-hand side. One of the original reasons is that this matrix requires less storage than the second-order Jacobian. Another reason for using a first-order Jacobian is that it is better conditioned than the second-order Jacobian; hence the inner iterations can converge faster. The penalty is an increased number of outer iterations. Approximate-Newton solvers which use a first-order Jacobian are typically preconditioned with an ILU(0) factorization. With this preconditioner, the solver requires approximately the same amount of storage as our matrix-free inexact-Newton solver. An efficient strategy which helps to reduce the overall CPU time without harming the convergence rate consists of freezing the left-hand-side after the first ten iterations. The preconditioner is updated only three times during the first ten iterations and then it is also frozen.

Performance comparison The residual convergence histories of PROBE are compared with the other four solvers in Figures 3 and 4. In several cases, ILU(2) converges just slightly more slowly than PROBE. However, for cases with stronger shocks, such as cases 2 and 8, ILU(2) diverges, and for case 4 it is significantly slower. Nevertheless, the use of ILU(2) as a solver can be much faster than the approximate factorization algorithm and merits further development, perhaps in conjunction with multigrid.

In several cases, ARC2D-MG converges almost as quickly as PROBE. However, for cases 3 and 4, PROBE is significantly faster than the multigrid algorithm. The approximate-Newton solver is robust, but it is much slower than PROBE, while requiring the same amount of storage.

Conclusions

An efficient matrix-free inexact-Newton-GMRES algorithm has been presented for the computation of steady aerodynamic flows. The algorithm has been successfully applied to a wide range of test cases which include inviscid, laminar, and turbulent flows. In all cases except one, convergence of the residual to 10^{-12} is achieved with a CPU cost equivalent to fewer than 1200 function evaluations. The sole exception is a low Mach number case where some form of local preconditioning is needed. Several other efficient implicit solvers have been applied to the same test cases, and the matrix-free inexact-Newton-GMRES algorithm is seen to be the fastest and most robust of the methods studied. Thus this strategy is an excellent option for flow computations in which memory use is not critical, such as two-dimensional applications.

Acknowledgments

The first author was supported by an OGS grant of the Government of Ontario. This work was supported by de Havilland Inc. and the Natural Sciences and Engineering Council of Canada.

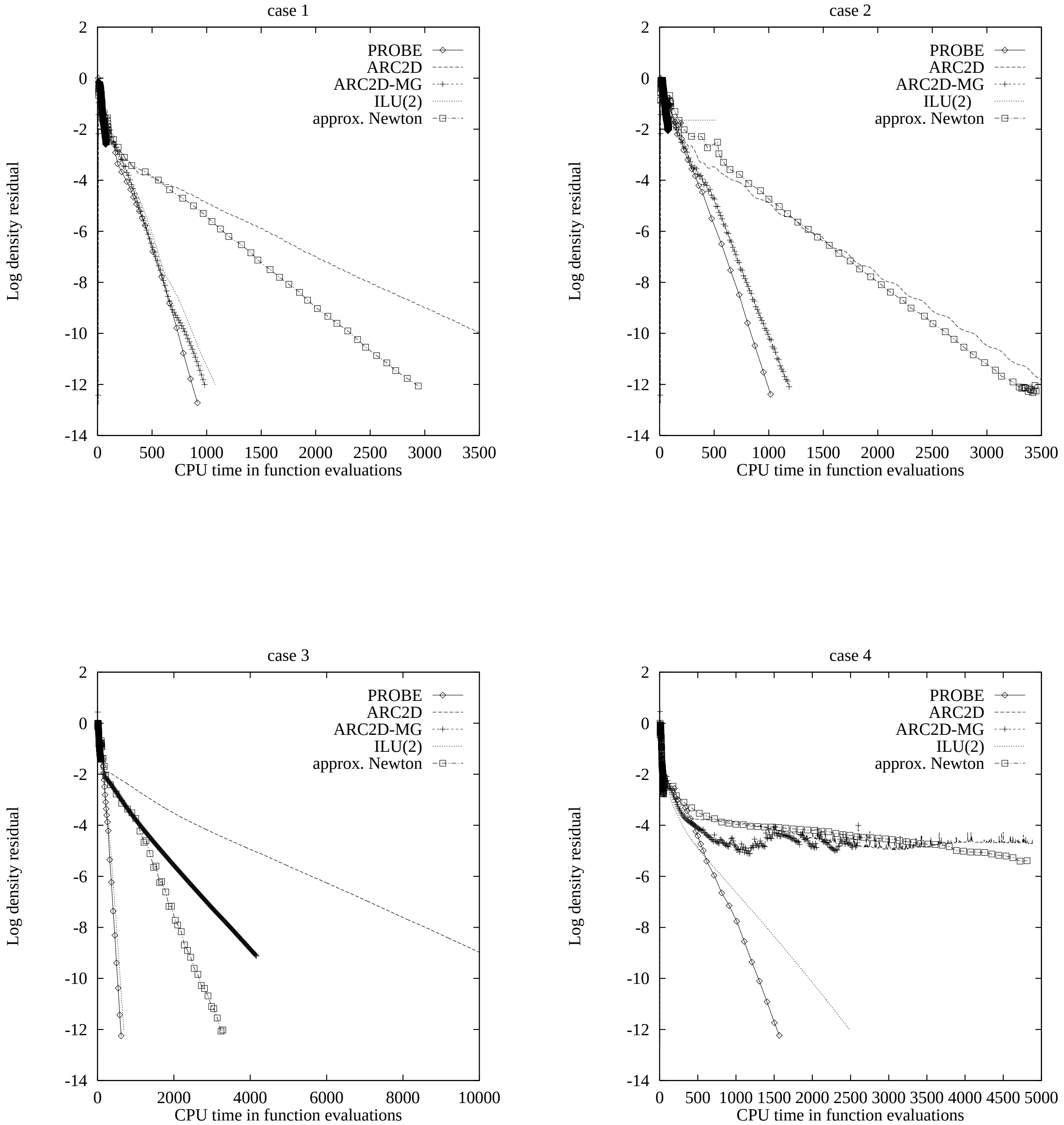

Figure 3: Cases 1 to 4 - convergence histories for the inexact-Newton-Krylov method (PROBE), the approximately-factored method (ARC2D), the approximately-factored method with three-level multigrid (ARC2D-MG), the incomplete factorization method (ILU(2)), and the approximate-Newton method (approx. Newton).

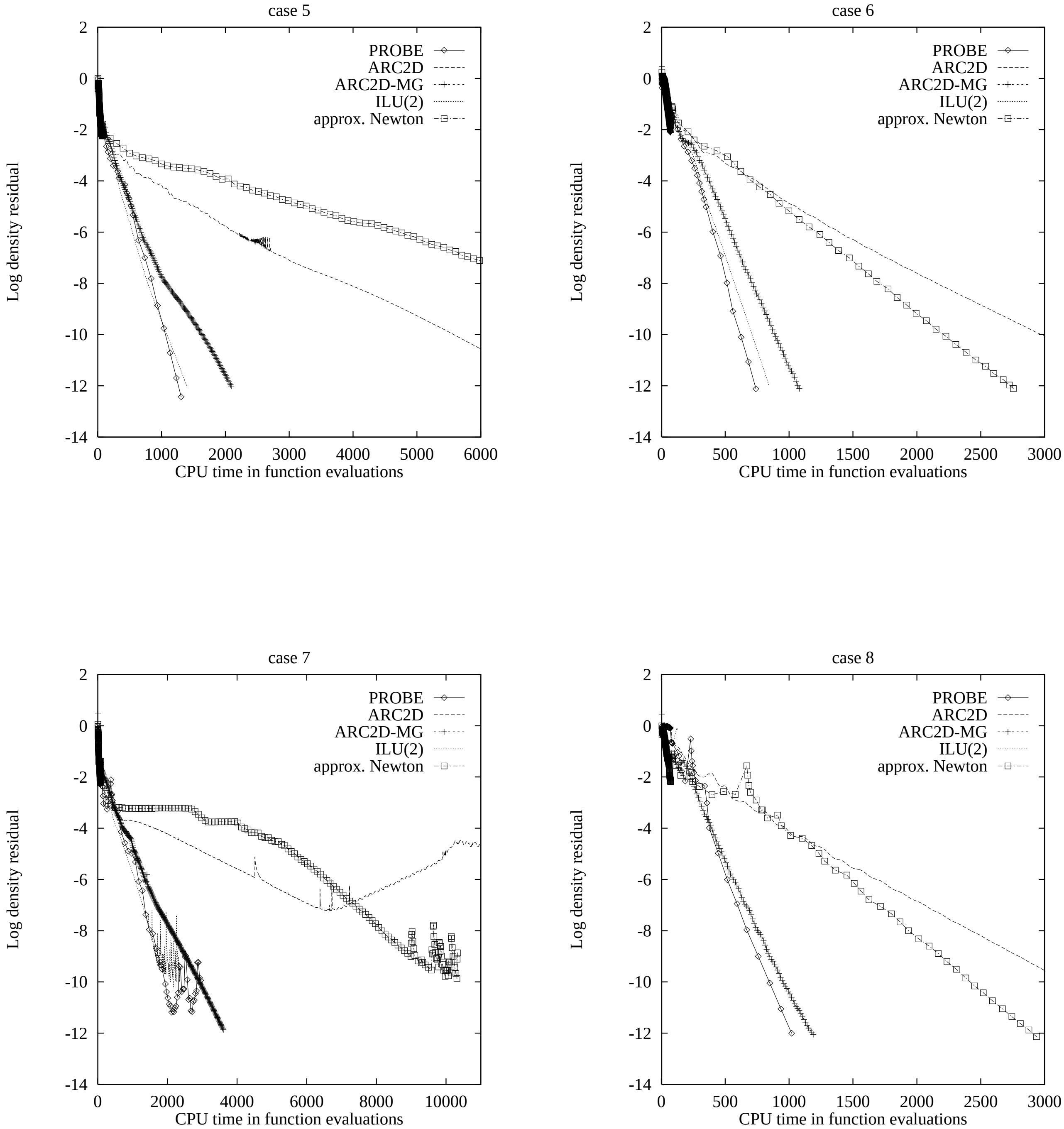

Figure 4: Cases 5 to 8 - convergence histories for the inexact-Newton-Krylov method (PROBE), the approximately-factored method (ARC2D), the approximately-factored method with three-level multigrid (ARC2D-MG), the incomplete factorization method (ILU(2)), and the approximate-Newton method (approx. Newton).

References

1. Beam, R., and Warming, R.F., "An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation Law Form," *J. Comp. Phys.*, vol. 22, pp. 87–110, 1976.

2. Steger, J.L., "Implicit Finite Difference Simulation of Flow About Arbitrary Geometries with Application to Airfoils." AIAA Paper 77-665, 1977.

3. Pulliam, T.H., "Efficient Solution Methods for the Navier-Stokes Equations." Lecture Notes For The Von Karman Institute For Fluid Dynamics Lecture Series, January 1986.

4. Jespersen D., Pulliam T., and Buning P., "Recent Enhancements to OVERFLOW." AIAA Paper 97-0644, January 1997.

5. Chisholm, T., and Zingg, D.W., "Multigrid Acceleration of an Approximately Factored Algorithm for Aerodynamic Flows." 44th Annual Conference of the Canadian Aeronautics and Space Institute, April 1997.

6. South, J.C., and Brandt, A., "Application of a Multi-Level Grid Method to Transonic Flow Calculations." Proc. of Workshop on Transonic Flow Problems in Turbomachinery, Monterrey, 1976, edited by T.C. Adamson and M.F. Platzer, Hemisphere, 1977, pp.180-206.

7. Jameson, A., "Acceleration of Transonic Potential Flow Calculations on Arbitrary Meshes by the Multigrid Method." Proc. AIAA 4th Computational Fluid Dynamics Conference, Williamsburg, 1979, pp. 122-146.

8. Ni, R.H., "A Multiple Grid Scheme for Solving the Euler Equations." Proc. AIAA 5th Computational Fluid Dynamics Conference, Palo Alto, 1981, pp. 257-264.

9. Jameson, A., "Solution of the Euler Equations for Two Dimensional Transonic Flow by a Multigrid Method," *Applied Mathematics and Computation*, vol. 13, pp. 327–356, 1983.

10. Martinelli, L., Jameson, A., and Grasso, F., "A Multigrid Method for the Navier-Stokes Equations." AIAA paper 86-0208, 1986.

11. Jameson A., "Computational Aerodynamics for Aircraft Design," *Science*, vol. 245, pp. 361–371, 1989.

12. Venkatakrishnan, V., "Newton Solution of Inviscid and Viscous Problems." AIAA-88-0413, January 1988.

13. Bailey, H.E., and Beam, R.M., "Newton's Method Applied to Finite-Difference Approximations for the Steady-State Compressible Navier-Stokes Equations," *Journal of Computational Physics*, vol. 93, pp. 108–127, 1991.

14. Saad, Y., and Schultz, M.H., "GMRES: A Generalized Minimal Residual Algorithm For Solving Nonsymmetric Linear Systems," *SIAM J. Sci. Stat. Comput.*, vol. 7, July 1986.

15. Pulliam, T.H., Rogers, S., and Barth, T.J., "Practical Aspects of Krylov-Subspace Iterative Methods in CFD." AGARD 77th Fluid Dynamics Panel Symposium on Progress and Challenges in CFD Methods and Algorithms, Seville, Spain, 2-5 October, 1995.

16. Ajmani, K., and Liou, M., "Implicit Conjugate-Gradient Solvers On Distributed-Memory Architectures." AIAA 95-1695, 1995.

17. Anderson, W.K., Rausch, R.D., and Bonhaus, D.L., "Implicit/Multigrid Algorithms for Incompressible Turbulent Flows on Unstructured Grids." AIAA 95-1740, 1995.

18. Choquet, R., "A Matrix-free Preconditioner Applied to CFD," Tech. Rep. No. 2605, INRIA, Institut National de Recherche en Informatique et en Automatique, Juin 1995.

19. Venkatakrishnan, V., and Mavriplis, D.J., "Implicit Solvers for Unstructured Meshes." AIAA-91-1537-CP, 1991.

20. Venkatakrishnan, V., "Implicit Schemes and Parallel Computing in Unstructured Grid CFD," ICASE report 95-28 CR-195071, NASA, 1995.

21. Rogers, S.E., "A Comparison of Implicit Schemes for the Incompressible Navier-Stokes Equations with Artificial Compressibility." AIAA 95-0567, January 1995.

22. Forsyth, P.A., and Jiang, H., "Iterative Methods for Full Newton Solution of the Euler Equations." Sixth International Symposium on Computational Fluid Dynamics, pp. 318–323, Lake Tahoe, Nevada, September 1995.

23. Nielsen, E.J., Anderson, W.K., Walters, R.W., and Keyes, D.E., "Application of Newton-Krylov Methodology to a Three-Dimensional Unstructured Euler Code." AIAA 95-1733-CP, June 1995.

24. Barth, T.J., and Linton, S.W., "An unstructured Mesh Newton Solver for Compressible Fluid Flow and Its Parallel Implementation." AIAA 95-0221, January 1995.

25. Cai, X., Keyes, D.E., and Venkatakrishnan, V., "Newton-Krylov-Schwarz: an Implicit Solver for CFD," Tech. Rep. No. 95-87, ICASE, December 1995.

26. Pueyo, A., and Zingg, D.W., "Progress in Newton-Krylov Methods for Aerodynamic Calculations." AIAA Paper 97-0877, January 1997.

27. Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes." AIAA Paper 81-1259, Palo Alto, Calif., 1981.

28. Baldwin, B.S., and Lomax, H., "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows." AIAA Paper 78-257, Hunstville, Alabama, 1978.

29. Eisenstat, S.C., and Walker, H.F., "Choosing the Forcing Terms in an Inexact Newton Method," *SIAM J. Sci. Comput.*, vol. 17, pp. 16–32, January 1996.

30. Pueyo, A. and Zingg, D.W., "Airfoil Computations Using a Newton-GMRES Method." CFD96, Proceedings of the Fourth Annual Conference of the CFD Society of Canada, June 1996.

31. Orkwis, P.D., "Comparison of Newton's and Quasi-Newton's Method Solvers for the Navier-Stokes Equations," *AIAA J.*, vol. 31, pp. 832–836, May 1993.

32. Dutto, L.C., "On Iterative Methods for Solving Linear Systems of Equations," *Revue européenne des éléments finis*, vol. 2, 1993.

33. Saleem, M., Pulliam, T.H., and Cheer, A.Y., "Acceleration of Convergence and Spectrum Transformation of Implicit Finite Difference Operators Associated with Navier-Stokes Equations," *Journal of Computational Physics*, no. 104, pp. 1–13, 1993.

34. Saad, Y., "ILUT: A Dual Threshold Incomplete LU Factorization," Tech. Rep. UMSI 92/38, University of Minnesota Supercomputer Institute, March 1992.

35. Dutto, L.C., "The Effect of Ordering on Preconditioned GMRES Algorithm, for Solving the Compressible Navier-Stokes Equations," *International Journal for Numerical Methods in Engineering*, vol. 36, pp. 457–497, 1993.

36. Cuthill, E.H., and McKee, J.M., "Reducing the bandwidth of sparse symmetric matrices." Proc. 24th National Conference of the Association for Computing Machinery, Brondon Press.

37. McHugh, P.R., and Knoll, D.A., "Comparison of Standard and Matrix-Free Implementations of Several Newton-Krylov Solvers," *AIAA J.*, vol. 12, pp. 2394–2400, December 1994.

38. Mulder, W.A., and Van Leer, B., "Implicit Upwind Methods for the Euler Equations." AIAA Paper 83-1930, 1983.

39. Ajmani, K., Ng, W., and Liou, M., "Preconditioned Conjugate Gradient Methods for the Navier-Stokes Equations," *Journal of Computational Physics*, vol. 110, pp. 68–81, 1994.

40. Ollivier-Gooch, C.F., "Improved Asymptotic Convergence Rates For An Unstructured Multigrid Solver." submitted to *AIAA J.*, 1996.

41. Unrau, D., and Zingg, D.W., "Viscous Airfoil Computations Using Local Preconditioning." AIAA Paper 97-2027, June 1997.