PROGRESS IN NEWTON-KRYLOV METHODS FOR AERODYNAMIC CALCULATIONS

Alberto Pueyo* David W. Zingg†

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

Abstract

An efficient Newton-GMRES algorithm is presented for computing steady compressible aerodynamic flows on structured grids. 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. Studies are presented which show the optimum choices of various parameters and strategies for the class of flows of interest. Finally, the algorithm developed is applied to a range of test cases and compared with a Newton-GMRES algorithm using an approximate Jacobian matrix and a well-known approximately-factored implicit algorithm. The new algorithm is shown to offer substantial reductions in the CPU time required to achieve a steady state.

Introduction

Computational methods for simulating aerodynamic flows have an important role in aircraft design. A great effort has been made to develop codes that can predict a wide range of complex flows. Many algorithms have been proposed for efficient computation of steady flows. Approximate factorization methods and methods exploiting multigrid in some manner are among the most popular and efficient.

In recent years, some authors have considered using Newton's method as a possible alternative for steady flows due to its property of quadratic convergence. Venkatakrishnan [1] and Bailey and Beam [2] successfully implemented an exact Newton's method. This approach was found to be robust, but memory and the CPU time required to reach steady state are not competitive with the methods mentioned earlier. On the other hand, quasi-Newton methods have shown promise. 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, using an iterative solver. In an approximate-Newton method, the functional Jacobian is simplified, thus producing an approximate linearization. The linear system is again solved iteratively. Since the Jacobian matrix can be very ill-conditioned and presents off-diagonal dominance, the use of a good preconditioner is critical to the success of any iterative solver. The range of options available in simplifying the system Jacobian matrix, preconditioning the system, and iteratively solving the system gives rise to a great variety of quasi-Newton methods. A popular class of methods, known as Newton-Krylov methods, uses an iterative solver of the Krylov family, often GMRES [3], usually with a preconditioner built from an incomplete LU factorization.

Venkatakrishnan and Mavriplis [4, 5] and Rogers [6] use approximate-Newton methods: they apply GMRES to the Jacobian matrix arising from a first-order approximation rather than a complete linearization of the spatial operator. They use an ILU(0) preconditioner. This approach requires less storage and fewer GMRES iterations per Newton iteration than inexact-Newton methods, but a larger number of Newton iterations.

Forsyth and Jiang [7] compared different quasi-Newton methods for inviscid transonic flows over airfoils. They showed that an inexact-Newton method, which uses the second-order Jacobian, is more effective than approximate-Newton methods, but a more expensive preconditioner, ILU(2), is needed to ensure robustness. This approach is not very attractive from a storage point of view because of the need to store the second-order matrix and its ILU(2) factorization, which produces a huge number of non-zero entries.

*Graduate Student.

†Associate Professor, Member AIAA.

The development of matrix-free GMRES [8] permits the use of the true linearization at reduced memory cost, making inexact-Newton methods competitive with approximate-Newton methods. However, if an ILU preconditioner is used, a Jacobian is needed for the factorization. The first-order Jacobian can be used; this is the approach followed by Nielsen et al. [9] and by Barth [10].

A number of practical aspects of Newton-Krylov solvers have been addressed by the authors already mentioned, as well as in other recent studies such as Refs. [11] to [15], but several issues require further study. Optimal choices of preconditioning strategy and ordering of the unknowns need to be found for specific problem classes. In addition, it is not yet clear whether approximate-Newton methods using the first-order Jacobian, which require a larger number of outer iterations but at a cheaper cost per iteration, are more or less effective overall than a matrix-free inexact-Newton method. Memory requirements need to be included in the assessment. Finally, Newton-Krylov methods should be compared with other efficient techniques such as multigrid or approximate factorization methods. These issues are addressed here in the context of inviscid and viscous flows over airfoils using a centered finite-difference operator with non-linear artificial dissipation.

The next section of the paper describes the main features of the algorithm. The test cases studied are then given. In the following section, we present a parametric study that addresses the optimum level of accuracy of the approximate-Newton iterates, preconditioning strategies, and reordering of the unknowns. The next two sections consist of a comparison with an approximate-Newton-GMRES method which makes use of the first-order Jacobian and with ARC2D, an efficient Euler and Navier-Stokes solver for structured grids developed by Steger [16] and Pulliam [17].

All the methods and strategies have been compiled and run using the same optimization flags and computer, a Pentium Pro 180. Thus, the CPU time constitutes an objective unit for efficiency comparison. For this reason, convergence histories are plotted using CPU time rather than iteration count.

Algorithm description

The spatial discretization is identical to that in ARC2D: a second-order centered-difference operator with the second and fourth-difference dissipation model of Jameson et al. [18]. A far-field circulation correction is also included. The Baldwin-Lomax algebraic model [19] is used for turbulent flows. The steady-state solutions of the present algorithm are thus identical to those computed using ARC2D.

The spatial discretization leads to a nonlinear system of equations of the form

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

where Q contains the conservative flow variables. If this system of equations is linearized, it becomes

$$A\,\Delta Q = F(Q) \tag{2}$$

If A is an exact linearization of $F(Q)$ of the form

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

i.e., the second-order Jacobian, denoted A_2, then equation (2) represents Newton's method. Alternatively, if some approximation to A_2 is used, then an approximate-Newton method is obtained. A simple approximation to A_2 with far fewer nonzero entries, which we designate A_1, is obtained by using only second-difference dissipation in forming the matrix. The coefficient of the second-difference dissipation is given by

$$\epsilon_2^l = \epsilon_2^r + \sigma\epsilon_4^r \tag{4}$$

where ϵ_4^r is the coefficient of the fourth-difference operator of the artificial dissipation on the right-hand side. Our experiments have shown that a good general value for the parameter σ in equation (4) is 10. The resulting matrix has five blocks per node, instead of nine.

Krylov solver GMRES, developed by Saad and Schultz [3], was found to be the most efficient Krylov iterative solver for the problems addressed in this paper. GMRES storage increases linearly with the number of search directions of the Krylov subspace, and CPU time 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 already shown in [20] that such a matrix-free approach can be advantageous, from the point of view of both performance and storage. Barth [10] 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

$$Av \simeq \frac{F(Q+\varepsilon v) - F(Q)}{\varepsilon} \tag{5}$$

$F(Q)$ in equation (5) is the residual vector and ε is a small scalar used to perturb the state quantities Q in the direction of v. The performance of this technique is very sensitive to ε, especially when using forward differencing [21]. Following Nielsen et al. [9], ε is determined such that

$$\varepsilon \parallel v \parallel_2 \simeq \sqrt{\varepsilon_m} \tag{6}$$

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

ILU preconditioners Incomplete LU factorizations are often regarded as efficient preconditioners for Krylov solvers. There have been two distinct approaches to forming such incomplete factorizations. The first approach, named ILU(p), uses only the graph of the matrix to determine which entries to drop. A level of fill is attributed to each fill-in element. The element is dropped if its level exceeds a certain threshold p. A drawback of this strategy is that it is difficult to predict the amount of fill-in that will be generated. In the second approach, the drop-off rule is based on the numerical value of the elements introduced in the factorization rather than on their fill levels. Unfortunately, the amount of fill-in is also hard to predict for this approach, and this preconditioner is far more expensive to form than the first approach.

ILUT(p,τ), developed by Saad [22], is a class of LU factorization that lies in between the two previous approaches. Two rules are used to determine which elements should be dropped at a given row. The first rule consists in dropping any element smaller than a relative tolerance determined by τ and a norm of the original matrix. The second rule is controlled by the parameter p: if p_l and p_u are the number of non-zeros on the lower and upper part of a given row of the original matrix A, at most the largest $p + p_l$ and $p + p_u$ are kept in the lower and upper part of the preconditioning matrix. This rule allows us to control the maximum number of elements per row and thus the memory usage.

The ILU(p) and ILUT(p,τ) factorizations are compared below, along with other aspects of the preconditioning.

Ordering of unknowns The ordering of the unknowns plays an important role in the convergence of the iterative solver [4, 23]. It can greatly affect the quality of the incomplete factorization. As mentioned in [20], for some cases, numbering the nodes across the wakecut (designated or-3) produces a more efficient algorithm than using the natural ordering (designated or-1). The downside is that the bandwidth is twice as large, which can greatly affect the performance in other cases. In order to obtain more consistent performance, we test two of the reordering algorithms typically used for unstructured grids. The Reverse Cuthill-McKee strategy [24] is a well-known bandwidth reduction algorithm. The minimum neighbouring algorithm [25] is a modification of the minimum degree reordering of George and Liu [26]. It was designed to minimize the lost information in a matrix incomplete factorization.

Start up Different relaxation techniques have been suggested in order to get past the rather violent nonlinear startup, particularly in transonic flows. One way to relax the solution is to damp the Newton updates to prevent the calculation of non-physical variable values [27]. An alternative to this technique is to rewrite equation (1) as

$$\frac{dQ}{dt} = F(Q) \tag{7}$$

and to apply implicit Euler time-differencing. A finite time step can be used initially and, as $\Delta t \to \infty$, Newton's method is obtained. This strategy is used by Mulder and Van Leer [28]. However, when a finite time step must be used, a cheaper relaxation algorithm can be employed [9] [29], significantly reducing computing time. This is particularly true for transonic flows, where too many outer iterations at low Δt are needed before fast convergence can be achieved, which is computationally expensive even when using a first-order Jacobian. In the present study, the approximately-factored algorithm of ARC2D with two levels of mesh sequencing is used for the first two orders of magnitude reduction of the residual. We also limit to 40 the maximum number of inner iterations for the first 4 outer iterations.

Test cases

Six test cases are studied using the NACA 0012 airfoil: two inviscid flows, a laminar flow and three turbulent flows. The parameters defining the test cases are given in Table 1. The transonic turbulent flow conditions listed as case 6 are used for the comparisons unless stated otherwise. The initial condition is 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 turbulent cases. These grids provide reasonable numerical accuracy for the flows considered. Figure 1 shows the computed surface pressure coefficient for the six cases.

Algorithm optimization

Approximate matrix inversion Several levels of reduction of the residual of the linear problem arising at each quasi-Newton step have been tested. The convergence

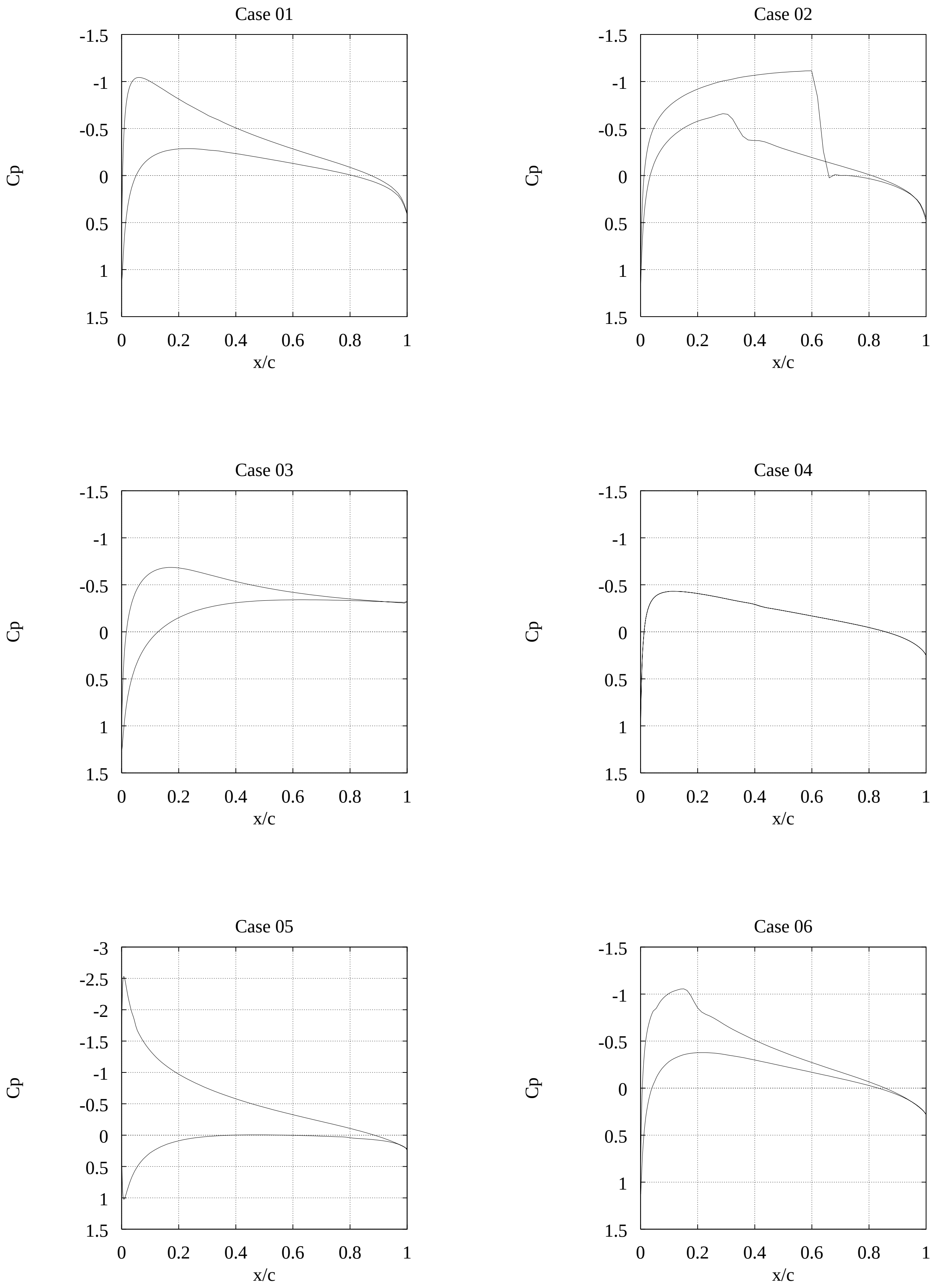

Figure 1: Pressure coefficient for the cases described in Table 1.

case	Ma	α	Re	trans.up	trans.low
Case 1	0.63	2.00	invisc	—	—
Case 2	0.80	1.25	invisc	—	—
Case 3	0.80	5.00	5.00e2	—	—
Case 4	0.30	0.00	2.88e6	0.43c	0.43c
Case 5	0.30	6.00	2.88e6	0.05c	0.80c
Case 6	0.70	1.49	9.00e6	0.05c	0.05c

Table 1: Parameters for the six flows studied.

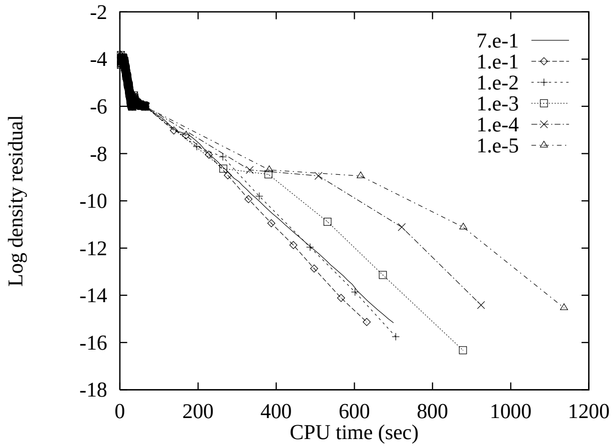

Figure 2: Convergence history for different levels of reduction of the inner residual using the matrix-free Newton-GMRES.

histories for 6 values are shown in Figure 2. Strict inner tolerances reduce the number of outer iterations, but there is an increased number of inner iterations. The results indicate that one does not benefit from the use of strict inner tolerances. The optimum tolerance for the matrix-free Newton-GMRES was found to be around 1×10^{-1}. These results are consistent with those those given in Refs. [20] and [30].

Preconditioning strategies ILU preconditioners can be built using the second-order Jacobian or any "reasonable" approximation. For instance, we could use the first-order Jacobian A_1 to form a preconditioner, M_1, which requires less storage. On the other hand, the preconditioner M_2, formed from A_2, is potentially more effective in reducing the number of iterations required by GMRES. In all cases we allow fill-in within the 4×4 blocks of the matrix used to build the preconditioner. After testing different levels of fill-in for ILU(p), we found that the optimum value is $p = 2$. Larger values require a substantial increase in memory with little or negative impact on performance.

Figure 3 shows the convergence histories obtained with three different preconditioners for case 1 on a coarse grid

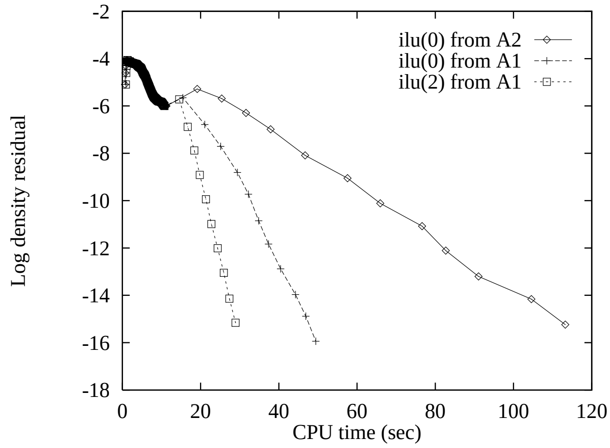

Figure 3: Convergence histories for the subsonic flow using three preconditioners: ILU(0) formed from the first-order Jacobian A_1, ILU(0) using the second-order Jacobian A_2 and ILU(2) from the first-order Jacobian.

(143×20): ILU(0) formed from the first and second-order Jacobians and ILU(2) formed from the first-order Jacobian. It should be noted that the number of non-zeros given by ILU(2) using A_1 is about the same as the number of non-zeros given by ILU(0) using A_2. ILU(2) applied to A_2 requires excessive storage and is not considered here.

The results in Figure 3 show that the preconditioners built from A_1 are more efficient than the one built from the second-order Jacobian. This is examined in more detail with ILU(0). Consider an incomplete factorization of the form:

$$A = M - E \tag{8}$$

where E is the error matrix and $M = LU$. The preconditioned matrices are given by:

$$AM^{-1} = I - EM^{-1} \tag{9}$$

Since we are solving the preconditioned system, the matrix E is not as important as the preconditioned error matrix EM^{-1}. As Saad [31] points out, when A is not diagonally dominant, L^{-1} and U^{-1} may have very large norms, causing $EM^{-1} = EU^{-1}L^{-1}$ to be very large and thus adding large perturbations to the identity matrix. In that case, the eigenvalues of AM^{-1} will not be nicely clustered around unity. This situation can occur for indefinite problems or problems with large nonsymmetric parts. The latter is the case here.

In Table 2, the Frobenius norms of the error matrices are presented. The results confirm that M_1 produces an error matrix E that has a bigger norm than the one from M_2, but the norm of its preconditioned error matrix is much smaller than the one from M_2. The matrix A_1 is more diagonally dominant than A_2 because we are adding a large amount of second-difference dissipation.

preconditioner	$\frac{\|E\|_F}{\|A_2\|_F}$	$\frac{\|EM_i^{-1}\|_F}{\|I\|_F}$
M_1	0.4870	4.6003
M_2	0.2576	6676.4449

Table 2: Frobenius norm of the error matrix and of the preconditioned error matrix for the first and second-order preconditioners.

	ILU(2)		ILUT(15,0.1)	
	RCM	MN	RCM	MN
Case 1	1	1	1.21	1.09
Case 2	1	1.07	1.32	1.52
Case 3	1	1.72	DNC	1.54
Case 4	1	1.20	DNC	1.25
Case 5	1.07	1	1.64	1.35
Case 6	1.07	1	DNC	1.16

Table 3: Relative time to converge for two ILU preconditioners, using Reverse Cuthill-McKee (RCM) and Minimum Neighbouring (MN) reorderings. DNC indicates that the case did not converge with the given parameters.

After testing many combinations, the values of the parameters p and τ of ILUT were fixed at 15 and 0.1, respectively. This produces an efficient preconditioner while the number of non-zeros is similar to ILU(2). Table 3 shows the performance of ILU(2) and ILUT(15,0.1) for the 6 test cases. Two different reorderings are shown, Reverse Cuthill-McKee and Minimum Neighbouring. These are further discussed below. The table shows that for this application, the traditional ILU(2) is superior to ILUT for both orderings.

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 it even decreases them. In fact, the most efficient strategy for our code and applications is to compute the preconditioner only once, when we switch from approximate factorization to Newton-Krylov.

Ordering of unknowns The four different orderings described above, or-1, or-3, RCM and MN, have been tested for the six cases. In Figure 4 we show that for case 3, or-3 is more efficient than or-1, although this is not always the case. The benefits of applying reordering techniques such as RCM and MN are clearly shown. The performance of MN is virtually independent of the initial ordering, or-1 or or-3. In contrast, the performance of RCM depends greatly on the initial ordering, with or-3 preferred. Table 3 shows that with ILU(2), faster convergence is achieved with RCM for cases 1 to 4, while for cases 5 and 6, MN is faster. Although further study with a wider range of cases is required, we concentrate on RCM for the remainder of the present paper. With ILUT(15, 0.1), MN is clearly superior to RCM; however, we do not consider this approach further, as it is slower than the combination of RCM and ILU(2) for all six cases.

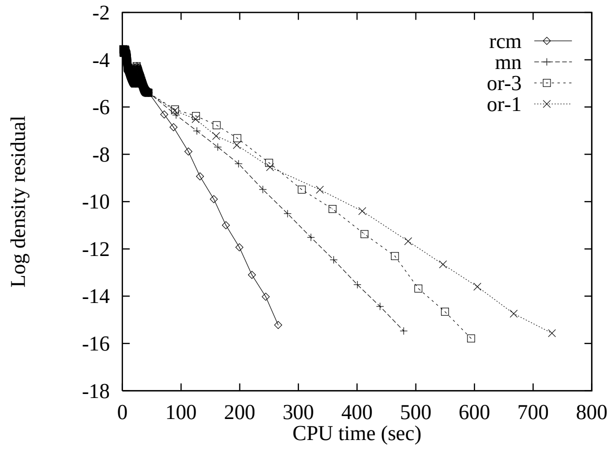

Figure 4: Convergence history for orderings or-1, or-3, Reverse Cuthill-McKee (RCM) and Minimum Neighbouring (MN) for case 3, using ILU(2) as preconditioner.

Approximate versus inexact-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. This is not an issue with the matrix-free implementation of GMRES. 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. Numerical experiments are therefore required to compare the memory use and efficiency of these two strategies.

Memory comparison Needs for memory come mainly from the matrix, the preconditioner, and the search directions forming the Krylov subspace. The first-order Jacobian with ILU(0) as preconditioner requires about 119 words per node to store the matrix (including integer arrays) and another 119 words for the preconditioner. Considering a Krylov subspace with five search directions, since the convergence criterion is generally reached within five iterations when the first-order Jacobian is used, adds another 20 words per node. Thus the total is 258 words per node. Our matrix-free solver requires 208 words per

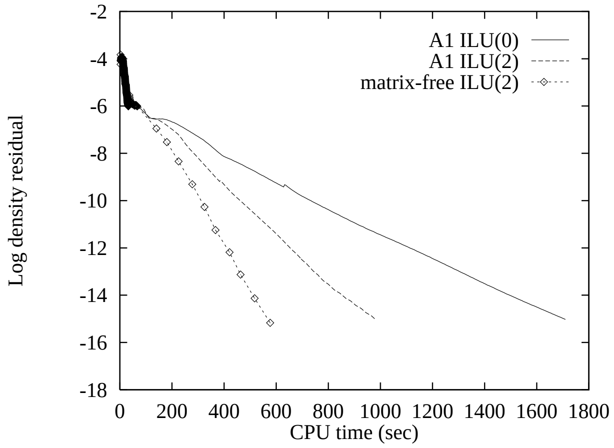

Figure 5: Convergence history for the first-order Jacobian A_1 using ILU(0) and ILU(2) as preconditioners and for the inexact-Newton matrix-free method with ILU(2).

node for the preconditioner and 80 for the Krylov subspace, which totals 288 words per node, just a 10% increase over the approximate-Newton strategy.

Performance comparison The convergence histories for the approximate-Newton strategies with two preconditioners, ILU(0) and ILU(2), and for the inexact-Newton strategy are plotted in Figure 5. The approximate-Newton strategies have been optimized in the same fashion as the inexact-Newton strategy. For example, the inner residual is reduced by a factor of 5×10^{-1} when using ILU(0) and by 1×10^{-1} when using ILU(2). The first-order Jacobian A_1 is frozen after two outer iterations without penalty on the convergence rate.

The inexact-Newton strategy is 3 times faster than the approximate-Newton strategy which uses ILU(0) as preconditioner. The approximate-Newton strategy achieves better performance using ILU(2), but then uses 20% more memory than the matrix-free inexact-Newton approach and is still substantially slower. It is clear that the matrix-free inexact-Newton strategy, which produces a very accurate linearization and thus a modest number of outer iterations, is superior to the approximate-Newton approach.

Comparison with ARC2D

We now compare our inexact-Newton-Krylov strategy, with a very successful method which is widely used on structured grids, the approximately-factored Beam and Warming algorithm in diagonal form, as used in ARC2D [17]. Our algorithm consists of the matrix-free GMRES(20) solver, preconditioned with an ILU(2) factorization of the first-order Jacobian formed using $\Delta t = 20$ and $\sigma = 10$ and the Reverse Cuthill-McKee reordering. The preconditioner is computed at the first Newton iteration and is not updated. The inner tolerance is set to reduce the residual one order of magnitude. The approximate factorization algorithm is used to reduce the outer residual two orders of magnitude before switching to the Newton-Krylov algorithm. In ARC2D, the wakecut can be treated implicitly or explicitly. The second option is faster but less robust. Both options, ARC2D-I (for implicit wakecut) and ARC2D-E (for explicit wakecut), are included in this study.

	o-it	Σi-it	i-it/o-it	CPU (sec.)	speedup
Case 1	10	264	26.4	200.0	4.9
Case 2	11	267	24.3	207.9	3.2
Case 3	10	239	23.9	265.1	15.2
Case 4	12	592	49.3	846.4	6.3
Case 5	12	671	55.9	943.6	3.9
Case 6	10	355	35.5	576.9	3.7

Table 4: 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(sec.)**: CPU time to reach 1×10^{-15}; **speedup**: CPU of ARC2D over CPU of Newton-Krylov to reach 1×10^{-15}.

Figure 6 shows the convergence histories for the six cases that we have studied and Table 4 shows some statistics for the Newton-Krylov algorithm. The number of inner and outer iterations required to reduce the residual norm 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 time, on the other hand, includes the total run time. The last column shows the speedup factor over ARC2D-E to reach a residual of 1×10^{-15}. Substantial reductions in CPU expense (at least a factor of three) are achieved in all cases. In the laminar case (case 3), the new algorithm is over 15 times faster than ARC2D. The reductions in CPU expense relative to ARC2D-I are even larger.

Conclusions

An efficient matrix-free inexact-Newton-GMRES algorithm has been presented for the computation of steady aerodynamic flows. A number of approaches have been considered, and the final algorithm incorporates the following:

- a matrix-free implementation of GMRES
- an ILU(2) preconditioner based on an approximate Jacobian matrix

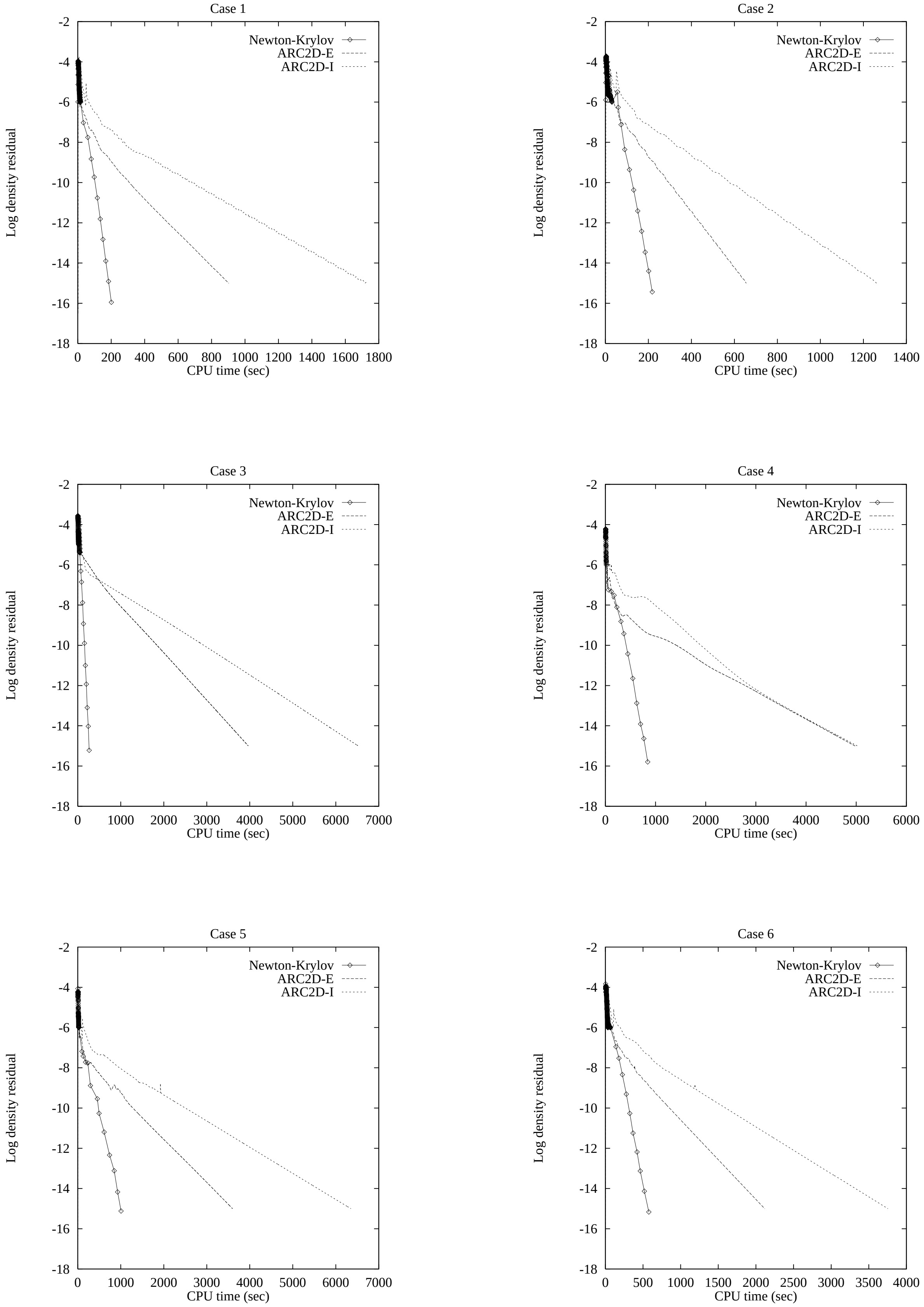

Figure 6: Convergence history for the Newton-Krylov method and for ARC2D with implicit (ARC2D-I) and explicit (ARC2D-E) treatment of the wakecut for the cases described in Table 1.

- RCM reordering
- a relatively high tolerance in the inexact computation of the Newton iterates
- use of an approximately factored algorithm to reduce the residual two orders of magnitude before switching to Newton-Krylov

The Newton-Krylov solver developed has been compared with two other flow solvers, one using an approximate-Newton-GMRES algorithm, and ARC2D, a well-known approximately-factored algorithm. Based on these comparisons, the following conclusions can be drawn:

- The matrix-free inexact-Newton algorithm is substantially faster than the approximate-Newton strategy. The increase in memory requirements is small.
- The matrix-free inexact Newton algorithm is also much faster than ARC2D for all of the cases studied. Note that there are other algorithms, generally based on multigrid, which are also faster than ARC2D, and the ARC2D algorithm can also be accelerated using multigrid [32], but ARC2D provides a useful reference because of its wide availability.

We are currently considering a number of strategies for reducing the CPU expense and memory requirements of our algorithm. Other future work involves extension to a field-equation turbulence model and multi-block grids.

Acknowledgments

The first author was supported by a FCAR grant of the Government of Québec. This work was supported by De Havilland Inc. and the Natural Sciences and Engineering Council of Canada.

References

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

2. 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.

3. 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.

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

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

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

7. 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.

8. Brown, P., and Saad, Y., "Convergence Theory of Nonlinear Newton-Krylov Algorithms," *SIAM J. Optimization*, vol. 4, pp. 297–330, 1994.

9. 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.

10. 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.

11. 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.

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

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

14. 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.

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

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

17. 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.

18. 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.

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

20. 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.

21. 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.

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

23. 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.

24. 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.

25. Martin, G., "Méthodes des préconditionnement par factorisation incomplète." Mémoire de Maîtrise, Université de Laval, Québec, Canada, 1991.

26. George, A., and Liu, J.W.H., "The evolution of the minimum degree ordering algorithm," *SIAM Rev.*, no. 31, pp. 1–19, 1989.

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

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

29. 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.

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

31. Saad, Y., "Krylov Subspace Techniques, Conjugate Gradients, Preconditioning and Sparse Matrix Solvers." 1994-05, von Karman Institute for Fluid Dynamics, March 1994.

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