AIAA 2003–0071

A Newton-Krylov Algorithm for Turbulent Aerodynamic Flows

Todd T. Chisholm, and David W. Zingg
University of Toronto Institute for Aerospace Studies
4925 Dufferin Street, Toronto, ON, M3H 5T6, Canada

39th Aerospace Sciences Meeting & Exhibit
January 6–9, 2003
Reno, NV

A Newton-Krylov Algorithm for Turbulent Aerodynamic Flows

Todd T. Chisholm,* and David W. Zingg†
University of Toronto Institute for Aerospace Studies
4925 Dufferin Street, Toronto, ON, M3H 5T6, Canada

A fast Newton-Krylov algorithm is presented for solving the compressible Navier-Stokes equations on structured multi-block grids with application to turbulent aerodynamic flows. The one-equation Spalart-Allmaras model is used to provide the turbulent viscosity. The optimization of the algorithm is discussed. ILU(4) is suggested for a preconditioner, operating on a modified Jacobian matrix. An efficient startup method to bring the system into the region of convergence of Newton's method is given. Three test cases are used to demonstrate convergence rates. Single-element cases are solved in less than 100 seconds on a desktop computer, while the solution of a multi-element case can be found in about 10 minutes.

Introduction

Recently, Newton-Krylov methods have been shown to be very effective in reducing the time required to compute numerical solutions to the Navier-Stokes equations. Blanco and Zingg[1] studied the solution of the Euler equations on unstructured grids with a matrix-free Newton-Krylov method. Geuzaine[2] used a similar method with the compressible Navier-Stokes equations, modeling turbulence with the Spalart-Allmaras model. Barth and Linton[3] studied a parallel implemention of a Newton-Krylov solver on unstructured grids for two- and three-dimensional flows. Pueyo and Zingg[4] solved the turbulent, compressible Navier-Stokes equations on structured grids.

Pueyo and Zingg have demonstrated that this approach is competitive with state of the art multigrid methods. However, their work was limited to representing turbulence with the algebraic Baldwin-Lomax model on single block grids. Here we discuss the solution of the Navier-Stokes equations on single- and multi-element airfoils, using the one-equation Spalart-Allmaras turbulence model.[5]

Algorithm Description

Governing Equations

We study the solution of the steady compressible thin-layer Navier-Stokes equations on structured grids. A generalized curvilinear coordinate transformation is used to map the physical space to a rectangular computational domain. The use of multiple blocks allows for complex geometries such as multi-element airfoils. A circulation correction is used to reduce the effect of the farfield boundary. The Spalart-Allmaras turbulence model, including trip terms, is implemented as described by Godin,[6] with a small change in the calculation of the modified voriticity factor, first used by Ashford[7]

$$\tilde{S} = S f_{v3} + \frac{\tilde{\nu}_T}{\kappa^2 d^2} f_{v2} \tag{1}$$

$$f_{v2} = \left(1 + \frac{\chi}{c_{v2}}\right)^{-3} \tag{2}$$

$$f_{v3} = \frac{(1+\chi f_{v1})(1-f_{v2})}{\chi} \tag{3}$$

with $c_{v2} = 5.0$. The original form, which allowed $\tilde{S}$ to become negative, introduced a local minimum quite close to the solution root at some nodes at the edge of recirculation bubbles. This can cause the residual to hang, despite the majority of the flow being converged. The new form seems to avoid this problem.

Spatial Discretization

The spatial discretization follows that used by Nelson, et al.[8] Second-order centred differences are used to approximate derivatives. Both Jameson's[9] scalar and Swanson and Turkel's[10] matrix second- and fourth-difference dissipation models can be used to stabilize the centred difference scheme. A pressure switch is used to control the activation of second-difference dissipation. The matrix dissipation model uses two switches V_l and V_n to avoid the effect of overly small eigenvalues in the flux Jacobian matrix. We use $V_l = V_n = 0.1$ for subsonic cases, and $V_l = 0.025$ and $V_n = 0.25$ for transonic cases. The turbulent viscosity convection and diffusion terms are discretized using first-order upwinding and second-order centred differencing, respectively, as suggested by Spalart and Allmaras,[5] with two differences. The turbulence equation is scaled by J^{-1}, the grid-metric Jacobian, and the state variable $\tilde{\nu}$ is replaced with $J^{-1}\tilde{\nu}$. These changes

*Ph.D Candidate, todd@oddjob.utias.utoronto.ca
†Professor, Senior Member AIAA, http://goldfinger.utias.utoronto.ca/~dwz/

help keep the entries in each block of the Jacobian of similar magnitude. This improves the conditioning of the system, making the Krylov solver more efficient.

Since a Newton solver is used to solve the resulting nonlinear system, it is important that all of the boundaries be handled fully implicitly. This includes the interfaces between blocks. The Navier-Stokes equations are solved on these interfaces in the same manner as the interior nodes.

Newton-Krylov Algorithm

The Nonlinear System. After spatial discretization, we have a system of the form

$$R(\hat{Q}^*) = 0 \tag{4}$$

where each block of $\hat{Q}$, the conservative state variables with the turbulence variable, is

$$\hat{Q}_i = J_i^{-1} Q_i = J_i^{-1}[\rho_i, \rho u_i, \rho v_i, e_i, \tilde{\nu}_i]^T$$

To find $\hat{Q}^*$ which satisfies Eq. 4, we apply the implicit Euler method repeatedly until some convergence criterion, typically $\|R\|_2 < 10^{-12}$, is reached:

$$\left[\frac{I}{\Delta t_n} - \frac{\partial R_n}{\partial \hat{Q}_n}\right] \Delta \hat{Q}_n = R_n$$

$$\hat{Q}_{n+1} = \hat{Q}_n + \Delta \hat{Q}_n$$

We call these the *outer* iterations. When the time step is increased towards infinity, Newton's method is approached. If Δt is increased appropriately as $\|R\|$ decreases, the quadratic convergence characteristic of Newton's method can be achieved, while dramatically increasing the region of convergence. Note that, in order for Newton's method to converge quadratically, $\frac{\partial R}{\partial Q}$ must be accurate. This requires that the equations be fully coupled.

The Linear System. In order for $\Delta \hat{Q}_n$ to be found, a linear system needs to be solved. This system tends to be very large, so that direct solution is prohibitive in both memory and time. Fortunately, finding the exact $\Delta \hat{Q}$ is not necessary, and we may settle for finding an approximation. This is an inexact-Newton method. There are a number of popular methods of finding the approximate solution of the linear system. The proper selection and use of this method is crucial to the success of the overall solver.[11] The most successful class are the Krylov iterative methods. Specifically, the preconditioned Generalized Minimum Residual (GMRES)[12] has proven to be effective for aerodynamic systems. We call these linear iterations the *inner* iterations.

Over-solving the linear system needs to be avoided for efficiency. A stopping criterion is needed for the inner iterations. There are two considerations. First, we use a target reduction in the inner residual. Pueyo[13] found a one order of magnitude reduction ideal to balance outer and inner iteration efficiency. This is appropriate in the turbulent case during the final Newton stage, but not in the startup. This will be discussed in the next section. The second consideration is setting the maximum number of iterations of GMRES. The amount of memory and CPU time increases with each GMRES iteration, so a limit is required. GMRES may be restarted, which keeps the memory requirements lower, while allowing further solution of the linear system. However, this can significantly slow the linear system convergence, due to the very poor conditioning seen in these systems. Typically, we do not use restarting for this reason.

The convergence rate of GMRES is very sensitive to the condition number of the matrix. Since the Jacobian of the equations being solved is typically extremely ill-conditioned, a good preconditioner is required to limit the number of inner iterations. Pueyo and Zingg[4] have shown that an incomplete LU preconditioner (ILU)[14] with two levels of fill minimizes solution time. They also found that a preconditioner based on a first-order Jacobian is more efficient than the exact Jacobian, both in saving memory and CPU time. The first-order Jacobian is formed by using only second-difference dissipation. This reduces the number of entries per equation to five instead of nine. It tends to give a better conditioned matrix, which leads to a more stable LU factorization. The coefficient of the second-difference dissipation used in the approximate Jacobian matrix, ϵ_2^l, is found by

$$\epsilon_2^l = \epsilon_2^r + \sigma \epsilon_4^r$$

where σ is found empirically and ϵ_2^r and ϵ_4^r are the second- and fourth-difference dissipation coefficients used in the evaluation of the residual and the exact Jacobian.

Pueyo and Zingg used scalar artificial dissipation. The use of matrix dissipation significantly worsens the conditioning of the preconditioner, due to a reduction in diagonal dominance. This requires further modifications. Two methods have been investigated. The most obvious is to include a time step. In order to achieve preconditioner stability, a sufficiently small time step needs to be taken. This is due to the existence of negative entries on the diagonal. Large time steps (with correspondingly small additions to the diagonal) have the possibility of worsening the condition of the matrix by reducing the diagonal. Small time steps are obviously not desirable, as they dramatically slow outer residual convergence. For this reason, we add the time step only to the matrix used for the preconditioner.

Another method of increasing diagonal dominance is to increase the value of the matrix dissipation switches (V_n and V_l) used in the first-order Jacobian matrix the factorization is based on. These switches set the minimum level of dissipation, making them a natural choice for controlling small diagonals. As they are in-

creased toward unity, scalar dissipation is approached. Note that the switches in the residual evaluation remain the same, so that the final solution is unaffected. The switches in the Jacobian used for the outer iterations are also unchanged, as inaccuracies there can adversely affect convergence. There are two conflicting effects that have to be balanced to optimize the preconditioner settings. The preconditioner must be conditioned well enough to be stable, but remain close enough to the true Jacobian to provide adequate clustering of the eigenvalues. A useful tool in evaluating a preconditioner is its condition number estimate, as discussed by Chow and Saad.[14] This is simply the L_2 norm of the solution to $LU \cdot \vec{c} = \vec{1}$. If a preconditioner is performing poorly, and has a high condition number estimate of the order of 10^7 or greater, more diagonal dominance is needed. It is usually better to err on the side of being too well conditioned. While this will slow convergence somewhat, the linear solves are much more robust.

To help reduce the effect of entries dropped from the preconditioner, reverse Cuthill-McKee reordering is used.[15] Good reordering is especially important in the multiblock case, due to the increased number of far off-diagonal entries resulting from the block boundaries.

The GMRES algorithm only requires matrix-vector multiplies, and does not explicitly require the matrix, except in forming the preconditioner. A Jacobian-free implementation of GMRES may be used, which has been found by Pueyo[13] to be faster, as well as resulting in significant memory savings.

Startup. Grid sequencing is used to help rapidly eliminate initial transients. This involves partially solving on one or more coarse grids, each formed by removing every other grid line in each direction from the next finer grid. The solution is interpolated from the coarse grid to the fine grid. This helps to bring the solution on the fine grid closer to the region of convergence of Newton's method, allowing higher initial time steps. It is important to ensure that the flows on the coarsened grids are tripped before passing to the fine grid. If a region is not properly tripped when the Newton's stage is started, divergence is likely. Raising the trip coefficient c_{t1} in the turbulence model helps encourage transition on the coarse grids. Initializing the turbulence variable with a small value (≈ 10) helps to speed the first few iterations and encourages tripping in troublesome regions. The turbulent variable is set to freestream (0.01) upstream of trip points to prevent tripping in what should be laminar regions. To further ease the startup a few iterations of the mean flow solver, without the turbulence solver, are performed first on the coarse grid. This establishes flow near the body and helps to prevent large sources in the turbulence model.

Examination of the production and destruction terms of the turbulence model reveals that these terms are unstable with negative $\tilde{\nu}$. Therefore, it is crucial to take steps to ensure that $\tilde{\nu}$ remain positive. This is a particular problem during the early iterations, when the solution is rapidly changing. There are a number of strategies which can be used to avoid negative values. Spalart and Allmaras[5] recommend that a modified linearization of the equations be used during startup. This modifies the turbulence model Jacobian so that it becomes an M matrix. The flow portion of the matrix is unchanged. While this prevents quadratic convergence, it was found that this modification is only necessary during the implicit Euler stages. This modification will be effective only if the linear system is solved sufficiently well. If the same tolerance appropriate for the laminar equations is used, large negative turbulence viscosities show up very quickly, despite using a small time step. Inner tolerances of approximately 10^{-4} are necessary to see the advantage of the modified turbulence model Jacobian. This is not nearly as detrimental as it would seem at first glance. The first few orders of magnitude reduction of the linear residual happen much faster than in the laminar case, often in only two or three iterations. This phenomenon only occurs when the modified Jacobian is used. Note that during startup, Jacobian-free GMRES cannot be used, since non-negative turbulent viscosities cannot be guaranteed.

Nemec and Zingg[16] used approximate factorization with the modified Jacobian during startup with good results. This is an inexpensive approach to approximately solving the linear system. The turbulence quantities are decoupled from the mean flow equations, and the linear system for each block is solved separately.

The approach followed by Geuzaine[2] was to use a variable time step based on the switched evolution relaxation method of Mulder and van Leer.[17] The time step in the early iterations is limited, so that the updates to the turbulence quantity are well bounded.

We have found that a spatially varying time step for the turbulence model can effectively control negative values of viscosity and be more efficient than the modified Jacobian method of Spalart and Allmaras. The lack of modification allows a matrix-free method to be used during startup, saving considerable memory. The local time step is based on the desire to keep the turbulence positive. We consider only local effects when calculating the time step at each node, meaning the equation is considered completely uncoupled. We justify this by noting that the production and destruction terms, which are highly nonlinear, are the source of our startup difficulties. Both of these are local, with the exception of the vorticity in production.

If Newton's method is used on this uncoupled equa-

tion, we get the following update:

$$\Delta\tilde{\nu} = \frac{R}{-J_D} \tag{5}$$

where R and J_D are the residual and diagonal element of the equation, respectively. We want to limit this update so that

$$|\Delta\tilde{\nu}| < r \cdot \tilde{\nu} \tag{6}$$

where r is a specified ratio. Choosing $r = 1$ would keep the updated $\tilde{\nu}$ positive in the uncoupled case, but in practice a smaller value is ideal. To actually limit $\Delta\tilde{\nu}$, we use a local time step

$$\Delta\tilde{\nu} = \frac{R}{\Delta t_{\tilde{\nu}}^{-1} - J_D} \tag{7}$$

then find the local time step $\Delta t_{\tilde{\nu}}$ to get the target update.

$$\Delta t_{\tilde{\nu}} = \left[\frac{R}{r \cdot \tilde{\nu}} - J_D\right]^{-1} \tag{8}$$

Note that $\Delta\tilde{\nu}$ and therefore r should have the same sign as R.

In the case where the Newton update is sufficiently small, the time step is set to

$$\Delta\tilde{\nu} = \tau \cdot \Delta t_{ref} \tag{9}$$

τ allows us to set an appropriate scaling for the turbulence model time step. Δt_{ref} is a reference time step, used by both the mean flow equations and the turbulence model.

$$\Delta t_{ref} = max\left(\Delta t_{min}, \alpha\left[\frac{1}{||R||_2}\right]^{\beta}\right) \tag{10}$$

α and *beta* allow us to control the time step relative to the convergence. Δt_{min} keeps the time step from being too small in the first few iterations.

The geometric time step following Pulliam[18] is used for the flow equations:

$$\Delta t = \frac{\Delta t_{ref}}{1 + \sqrt{J}} \tag{11}$$

where J is the Jacobian of the metric of the curvilinear coordinate transformation.

Even with these changes, negative values of $\tilde{\nu}$ are likely to be found, especially after interpolation between grids. These values are clipped to zero after each update. Experiments have been carried out with other techniques including modifying the production and destruction terms for negative values, with varying degrees of success. Simple clipping seems to be the most robust, and allows more aggressive time steps.

Matrix dissipation tends to be somewhat more unstable than scalar, especially during startup. Scalar dissipation is used on the coarse grids for this reason. The interpolation error also seems to eliminate any gains which could be had by using matrix dissipation on the coarse grids.

Case	Mach	Alpha	Re·10^6	Airfoil
1	0.3	6.0°	9.0	NACA0012
2	0.729	2.31°	6.5	RAE2822
3	0.185	6.0°	2.51	NLR

Table 1 Flow conditions

Case	Dimensions	Nodes	Offwall Spacing
1	305× 57	17385	10^{-6}
2	257× 57	14619	$2 \cdot 10^{-6}$
3	–	44059	10^{-6}

Table 2 Grids

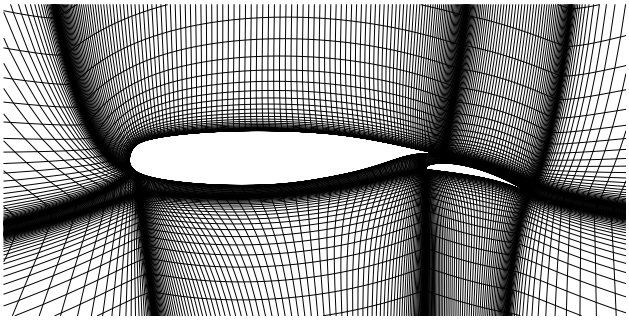

Fig. 1 Case 3 grid

Test Cases

Three test cases are presented. Two are single-element, one subsonic, the other transonic. The third case is an airfoil with a detached flap, at low Mach number and moderate angle of attack. The flow conditions are shown in Table 1, grid details in Table 2. Off-wall spacing is given relative to chord length. Figure 1 shows the grid around the multi-element airfoil.

The GMRES iterations for the single-element cases are limited to thirty search directions, with one restart. This is not sufficient for the larger, multiblock case. Sixty search directions with no restarts allow the linear solver to converge sufficiently for this case.

Algorithm Optimization

There are a number of parameters which need to be optimized to get a robust and efficient solver. These fall into two catagories: startup and preconditioning. We will optimize without the use of trip terms except where explicitly required to properly set a parameter, since these tend to make the convgence less consistent.

Startup

Choosing a proper startup sequence is crucial to obtaining maximum efficiency. The Newton steps will zero the residual much more quickly than the implicit Euler steps, so we try to raise the time step as rapidly as is feasible. We now compare two methods of startup: using a modified Jacobian following Spalart and Allmaras, and using the local time step for the

turbulence model.

When using the modified Jacobian, the time step used for the turbulence model is simply the reference time step. Δt_{ref} is set to 50 on the coarsest grid, until the turbulence model fully trips. This is indicated by a peak in the turbulent residual. While the maximum value varies strongly by case, a minimum number of iterations of 15 should ensure tripping. At this point, the turbulence has stabilized enough to use a time step of 500. When $||R||_2 < 0.01$, the solution is interpolated to the next grid. One iteration at a time step of 50 is used to smooth the interpolation error. After this $\Delta t_{ref} = 500$ is used until $||R||_2 < 0.01$, and the solution is interpolated to the final grid. At this point, the variable time step is used along with the modified Jacobian, until $||R||_2 < 0.01$. The unmodified Jacobian and variable time step are used until convergence. Note that the residual vector is comprised of both the four mean flow equations and the turbulence equation. The norm, and therefore the transition between stages, is typically dominated by the turbulence residual. This is appropriate, considering that the turbulence model is less stable than the flow equations. This method is quite robust when trip terms are not used. However, it can fail to converge in more complex multielemet cases with trip terms.

When a spatially varying time step is used for the turbulence model, the following parameters must be set: α, β, Δt_{min}, τ, and r. Fortunately, the method is not terribly sensitive to these parameters. This is a result of the local time step, which stays small and is independent of Δt_{ref} in trouble areas. $\beta = 1$, and $r = 0.3$ are good choices. A value of 100 can be use for Δt_{min}, α, and τ. When a shock is present, more conservative values need to be used on the coarse grid to establish the shock. A value of 10 is appropriate for Δt_{min}, α, and τ.

When the trip terms are being used, the turbulence model convergence will often hang around 10^{-4}. This is a result of the edge of the turbulent region moving slightly. The problem can be solved by lowering the time step which 'damps' this movement, and allows the edge of the turbulent region to settle into position. The best way to do this is to reduce α, so that the time step doesn't start to rise until after the turbulent region has settled. Setting α to one accomplishes this. This of course results in a penalty in convergence rate.

Figure 2 compares the two startup methods for case 3. The modified Jacobian takes longer to reach the Newton stage. It is also quite ill-conditioned at the beginning of the Newton iterations, making it significantly more sensitive to preconditioner settings. The second method does not require an explicit Jacobian matrix. We recommend the local time step method for these reasons.

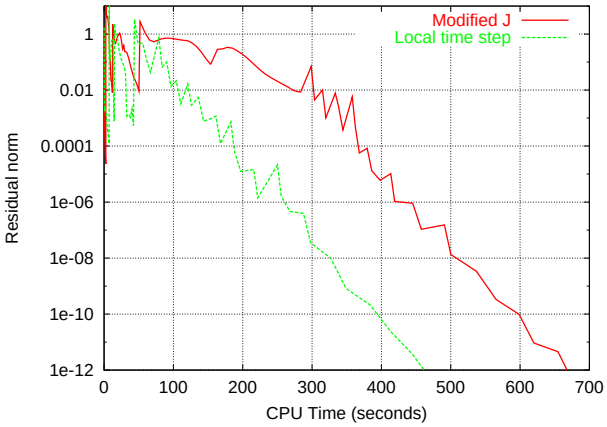

Fig. 2 Case 3 startup method comparison

	Inner iterations	
σ	Case 1	Case 2
4	430	div
5	323	263
6	334	291
7	379	308
8	401	321
9	483	345

Table 3 Inner iterations vs. σ

	CPU time (sec)		
V_l and V_n	Case 1	Case 2	Case 3
.3	dnc	100	div
.4	92	81	dnc
.6	90	90	375
.8	94	112	455
1.0	110	121	400

Table 4 CPU time vs. Preconditioner switch

Preconditioner

Choosing proper parameters for the preconditioner is crucial for stability. If the preconditioner is too poorly conditioned, the linear solves will fail. Moving the parameters past the value required for stability generally slightly slows convergence.

Table 3 shows the effect of varying σ on the total number of inner iterations needed to reach $\|R\|_2 < 10^{-12}$ on the fine grid for cases 1 and 2. *div* indicates that the case diverged. Based on these results, $\sigma = 5$ was chosen.

Table 4 shows the effect of preconditioner switch values on convergence time. *dnc* indicate the case failed to converge. The transonic case is the best conditioned of the tests. This is likely due to the higher Mach number, increased dissipation, and relatively low number of grid nodes. Choosing $V_l = V_n = 0.4$ in the preconditioner is most efficient. The other cases require higher switch values. Using $V_l = V_n = 0.6$ is a safe choice.

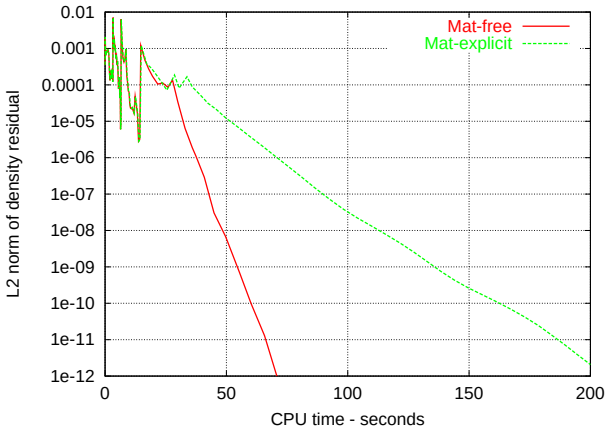

Fig. 3 Case 2 matrix-free vs. matrix-explicit GMRES

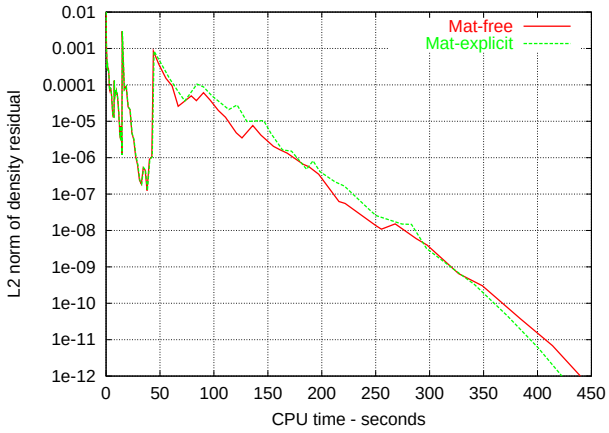

Fig. 4 Case 3 matrix-free vs. matrix-explicit GMRES

Note that the switch values in the residual and exact Jacobian remain unchanged.

Using a minimum value of time step in the preconditioner could improve the conditioning of the preconditioner, but it seemed to be too imprecise a tool compared with the switches. The time step is added to each equation, while the switches only affect those with very low dissipation. For this reason, the use of modified switches was much more successful in stabilizing the preconditioner than the use of a time step.

As mentioned previously, GMRES does not require the matrix to be explicitly formed. There is a trade-off in speed between matrix-free and matrix-explicit GMRES, which depends on the number of linear iterations. The former requires one residual evaluation per iteration. The latter requires a matrix construction when beginning the linear solve, plus a matrix-vector multiply per iteration. Since the matrix-vector multiply is cheaper than a residual evaluation, matrix-free GMRES becomes less efficient with more difficult systems. Figures 3 and 4 compares these two methods for cases 2 and 3.

The matrix-free solver requires roughly the same time as the matrix-explicit solver, despite the number of inner iterations being over forty. This makes the matrix-free GMRES more attractive, since it requires less memory. The situation for case 2, the transonic case shows a larger difference. The matrix here is better conditioned, and needs about 15 inner iterations per Newton step. Due to the presence of the shock, the pressure switch has activated second-difference dissipation. Adding the linearization of the switch and the dissipation coefficient requires significant amounts of time when calculating the Jacobian. Without these terms in the Jacobian matrix, the convergence of the outer iterations is adversely affected. Matrix-free is clearly the best choice for transonic cases.

The choice of level of fill in the preconditioner is important in balancing the memory use and CPU time.

ILU fill	Case 1	Case 3
2	91	590
3	108	533
4	101	486
5	102	460

Table 5 Convergence time (seconds) vs. ILU fill

There is an optimum level of fill to minimize CPU time. Higher fills produce more powerful preconditioners, which reduce the number of inner iterations, but require significantly more time to factorize and apply. Table 5 shows residual convergence times for cases 1 and 3. Both cases show that ILU(4) is a good choice for minimizing CPU time. This adjustable parameter has the advantage of allowing a trade-off between memory and speed. The matrix formed when a finite time step is used is quite well conditioned, so that a preconditioner with a fill of two is sufficient during the startup phase. ILU(1) can be used, but the time step has to be dropped significantly to maintain preconditioner stability.

Reordering the preconditioner matrix can increase the effectiveness of the preconditioner significantly. The reverse Cuthill-McKee algorithm is not completely defined, as the root node is only specified as a node of minimum degree. For this system, that is a boundary node. A study of different roots shows that this is an important choice and should not be left to chance. Figure 5 shows the convergence of case 3 with three different roots. This is a typical result. Choosing an upwind node gives a preconditioner with extremely poor conditioning. Downwind nodes near the centre of the outflow boundary are good candidates. The reason for this is not clear.

Results

Figures 6, 7, and 8 compare the convergence histories of the Newton-Krylov solver (NK) with an

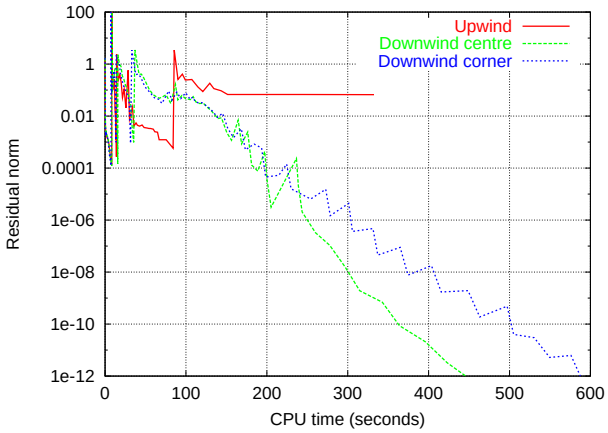

Fig. 5 Case 3 comparison of reordering root

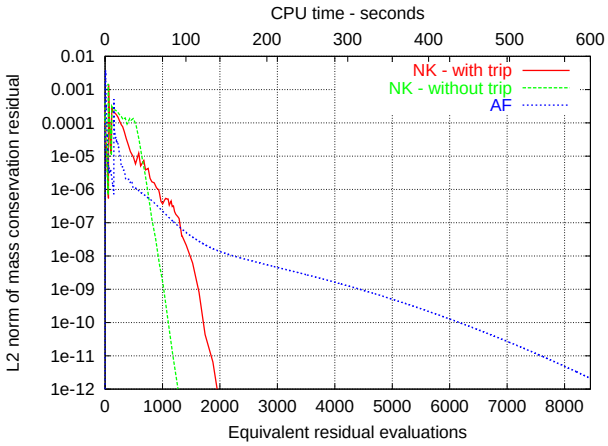

Fig. 6 Case 1 residual history

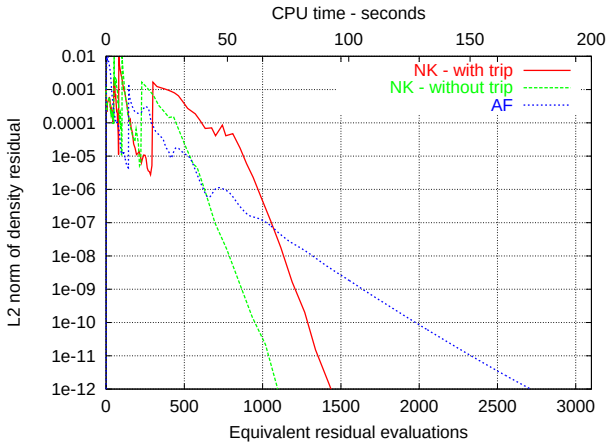

Fig. 7 Case 2 residual history

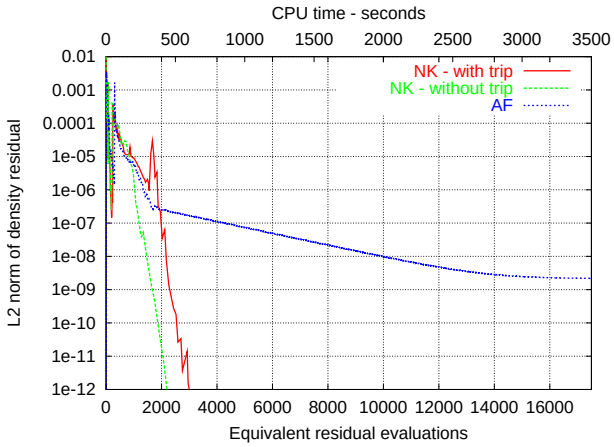

Fig. 8 Case 3 residual history

approximately-factored solver (AF) in diag form using grid sequencing. They show the conservation equation residual versus both CPU time and number of equivalent residual evaluations. We include both the evaluation of the mean flow and turbulence model equations in the residual calculation time. The time to evaluate the residual is virtually identical in both the AF and NK solvers.

The Newton-Krylov solver was run with and without trip terms. The AF solver used trip terms, although there is little difference in convergence time without them. All cases used an ILU(4) preconditioner, and were run on an AMD 1800-XP desktop computer. Cases 1 and 3 are considerably faster than the AF solver, while case 2 shows a smaller improvement. This is due to the AF solver already solving the case in 2500 residual evaluations. Case 3 is not fully converged by the AF solver. There is a small recirculation bubble at the trailing edge of the flap. Difficulties arise at the edges of these bubbles with the highly nonlinear destruction term of the turbulence model. Outside of these few nodes, the model is fully converged.

Conclusions

An efficient Newton-Krylov solver has been presented for the steady compressible Navier-Stokes equations governing turbulent flows over multi-element airfoils. Proper optimization is essential. This includes using grid sequencing and the implicit Euler method for startup. During this phase, care must be take to ensure that the turbulent viscosity remains positive. We have shown that a spatially varying time step for the turbulence model is a good approach. Incomplete LU preconditioning is used. A level of fill of four was found to be optimal with respect to CPU time and memory. The ILU factorization is applied to the first-order Jacobian matrix with modified second-difference dissipation. Time step selection and modified dissipation switches are important to stabilize the preconditioner. The single-element test cases can be solved in less than 100 seconds, while the complex flow on the multi-element case can be found in about ten minutes. The subsonic cases converge three to five times faster than an approximately factored algorithm.

Acknowledgments

This research was supported by Bombardier Aerospace, and an OGSST grant from the Government of Ontario.

References

[1]Blanco, M. and Zingg, D. W., "Fast Newton-Krylov Method for Unstructured Grids," *AIAA Journal*, Vol. 36, No. 4, April 1998, pp. 607–612.

[2]Geuzaine, P., "Newton-Krylov Strategy for Compressible Turbulent Flows on Unstructured Meshes," *AIAA Journal*, Vol. 39, No. 3, 1997, pp. 528–531.

[3]Barth, T. J. and Linton, S. W., "An Unstructured Mesh Newton Solver for Compressible Fluid Flow and Its Parallel Implementation," 1995, AIAA-95-0221.

[4]Pueyo, A. and Zingg, D. W., "Efficient Newton-Krylov Solver For Aerodynamic Calculations," *AIAA Journal*, Vol. 36, No. 11, 1998, pp. 1991–1997.

[5]Spalart, P. and Allmaras, S., "A One-Equation Turbulence Model for Aerodynamic Flows," 1992, AIAA paper 92-0439.

[6]Godin, P., Zingg, D. W., and Nelson, T., "High-lift Aerodynamics Calculations WIth One- and Two-equation Turbulence Models," *AIAA Journal*, Vol. 35, No. 11, Feb. 1997.

[7]Ashford, G. A., *An Unstructured Grid Generation and Adaptive Solution Technique for High Reynolds Number Compressible Flows*, Ph.D. thesis, University of Michigan, 1996.

[8]Nelson, T. E., Zingg, D. W., and Johnston, G. W., "Compressible Navier-Stokes Computations of Multielement Airfoil Flows Using Multiblock Grids," *AIAA Journal*, Vol. 32, No. 3, 1997, pp. 506–511.

[9]Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solutions Of The Euler Equations By Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," June 1981, AIAA paper 81-1259.

[10]Swanson, R. C. and Turkel, E., "On Central-Difference and Upwind Schemes," *J. Comp. Phys.*, Vol. 101, 1992, pp. 292–306.

[11]Barrett, R., Berry, M., Chan, T., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and van der Vorst, H., "Templates For The Solution Of Linear Systems: Building Blocks For Iterative Methods," Tech. rep., SIAM, 1994.

[12]Saad, Y. and Schultz, M. H., "GMRES: A Generalized Minimum Residual Algorithm For Solving Nonsymmetric Linear Systems," *SIAM J. Sci. Stat. Computing*, Vol. 7, 1986, pp. 856–869.

[13]Pueyo, A. and Zingg, D. W., "Airfoil Computations Using A Newton-GMRES Method." *Proceedings of the Fourth Annual Conference of the CFD Society of Canada*, CFD96, June 1997.

[14]Chow, E. and Saad, Y., "Experimental Study Of ILU Preconditioners For Indefinite Matrices," *J. of Comp. and Appl. Mathematics*, Vol. 87, 1997, pp. 387–414.

[15]Cuthill, E. H. and McKee, J. M., "Reducing the bandwidth of sparse symmetric matrices," *Proceedings of the 24th National Conference of the Association for Computing Machinery*, Brondon Press, 1969, pp. 157–172.

[16]Nemec, M. and Zingg, D. W., "Newton-Krylov Algorithm for Aerodynamic Design Using the Navier-Stokes Equations," *AIAA Journal*, Vol. 40, No. 6, 2002, pp. 1146–1154.

[17]Mulder, W. A. and van Leer, B., "Experiments With Implicit Upwind Methods For The Euler Equations," *J. Comp. Phys.*, Vol. 59, 1985, pp. 232–246.

[18]Pulliam, T. H., "Efficient Solution Methods for the Navier-Stokes Equations," Lecture Notes for The Von Karman Institute.