AIAA JOURNAL
Vol. 33, No. 11, November 1995

Comparison of Implicit Schemes for the Incompressible Navier–Stokes Equations

Stuart E. Rogers*
NASA Ames Research Center, Moffett Field, California 94035-1000

For a computational flow simulation tool to be useful in a design environment, it must be very robust and efficient. To develop such a tool for incompressible flow applications, a number of different implicit schemes are compared for several two-dimensional flow problems in the current study. The schemes include Point–Jacobi relaxation, Gauss–Seidel line relaxation, incomplete lower-upper decomposition, and the generalized minimum residual method preconditioned with each of the three other schemes. The efficiency of the schemes is measured in terms of the computing time required to obtain a steady-state solution for the laminar flow over a backward-facing step, the flow over a NACA 4412 airfoil, and the flow over a three-element airfoil using overset grids. The flow solver used in the study is the INS2D code that solves the incompressible Navier–Stokes equations using the method of artificial compressibility and upwind differencing of the convective terms. The results show that the generalized minimum residual method preconditioned with the incomplete lower-upper factorization outperforms all other methods by at least a factor of 2.

Introduction

ALTHOUGH computational fluid dynamics (CFD) now has a lot to offer an engineer, there is still room for significant improvement. The efficiency and robustness of a flow solver are two of its most important features if it is to be successfully applied as a tool in the design process. This is especially true of Navier–Stokes methods, which require very fine resolution grids, particularly for high-Reynolds-number flows. When engineers need to study a number of design parameters, they often need to obtain hundreds of steady-state solutions to a particular problem. Use of optimization methods in design may require numerous flowfield solutions. In both these instances, a flow solver needs to produce solutions without requiring the users to tune the input numerical parameters for each computation that they run.

The goal of the current study is to determine an efficient algorithm for obtaining steady-state solutions to the incompressible Navier–Stokes solutions for two-dimensional flow problems. There is a wide range of applicability of such a flow solver in engineering flow analysis, including high-lift multi-element airfoil computations[1] and propulsion flow analysis.[2] These investigations are being performed for two-dimensional flows because these are cheaper—allowing the investigation of many parameters, schemes, and flow problems—and because a two-dimensional analysis tool can be quite valuable in its own right. This study will use the INS2D flow solver,[3,4] and later the results can be utilized to improve the INS3D flow solver.[5]

There are many different types of solution techniques for steady-state incompressible Navier–Stokes computations, too numerous to discuss in detail here. The INS2D and INS3D flow solvers by default use an implicit Gauss–Seidel line-relaxation (LR) scheme. This scheme has performed quite well for a large number of different flow problems[1–5] but has been shown to have some convergence problems for very fine meshes with multiple zones. For example, some fine-resolution, multizone grids used in recent multi-element airfoil calculations[1] have taken several thousand iterations to converge, whereas most cases that are run with this flow solver converge within 200 iterations.

Received Dec. 27, 1994; presented as Paper 95-0567 at the AIAA 33rd Aerospace Sciences Meeting, Reno, NV, Jan. 9–12, 1995; revision received April 18, 1995; accepted for publication April 21, 1995.

*Aerospace Engineer, Design Cycle Technologies Branch, Aeronautical Technological Division, Mail Stop T27B-1. Senior Member AIAA.

Recent work in iterative, Krylov-space matrix solvers[6,7] has shown that some of these methods are applicable to CFD flow solvers. In particular, the generalized minimal residual (GMRES) method[8] is well suited to the matrix problems arising in implicit CFD solvers. There are two distinct ways to implement GMRES in a flow solver. One approach is to use GMRES to solve the linearized system of equations resulting from the application of a time-marching type of scheme, such as an implicit Euler scheme. The other formulation is to use a nonlinear extension of GMRES to directly solve the discrete form of the steady-state equations. The first approach has been attempted by many authors; see Refs. 9–11 for recent examples. The second approach was introduced by Wigton et al.[12]

The most important aspect of implementing GMRES is the preconditioning of the system of equations. For implementation in a CFD code, a good preconditioner is necessary for GMRES to converge. The current work investigates the use of GMRES to solve the linearized system of equations in delta form in the INS2D code. The code for the GMRES solver was obtained from the template software available as a companion to Ref. 7. The GMRES method can be easily implemented so that any existing solution process can be utilized as a preconditioner. One commonly used preconditioner for the GMRES method is an incomplete lower-upper (ILU) factorization with zero additional fill. See Meijerink and van der Vorst[13] for a discussion of ILU methods. The ILU solution scheme was added as an option to the INS2D code, along with a Point–Jacobi relaxation (PR) scheme, to go along with the Gauss–Seidel line-relaxation scheme already in the code. Presented in this paper are comparisons between six different solution processes: the three aforementioned algorithms run independently, and the GMRES method using these three algorithms as preconditioners, denoted as GMRES+ILU, GMRES+PR, and GMRES+LR.

In the following sections, the features of the INS2D flow solver are discussed, followed by a presentation of each of the implicit methods used in this study. The next section presents the computed results of each of these methods for three different flow problems: laminar flow over a backward-facing step, turbulent flow over a NACA 4412 airfoil, and turbulent flow over a three-element airfoil.

Flow Solver

The INS2D flow solver[3,4] solves the Reynolds-averaged incompressible Navier–Stokes equations using the method of artificial compressibility,[14] which adds a pseudotime derivative of pressure p to the continuity equation, resulting in

$$\frac{\partial p}{\partial \tau} = -\beta \nabla \cdot \boldsymbol{V} \tag{1}$$

where τ is the pseudotime, V is the velocity vector, and β is the artificial compressibility constant. This relaxes the elliptic nature of the equations and results in a hyperbolic-parabolic system. The solver is capable of solving both steady-state and time-dependent flow problems, although the current study is concerned only with steady-state solutions. The INS2D code is a finite difference, structured-grid flow solver. It is capable of handling multiple-zone grids using either a patched multiblock (pointwise continuous) interface or an overlaid chimera interface between zones. The boundary conditions at the physical boundaries and at zonal boundaries are applied in an implicit fashion during the solution process. A third-order, upwind-differencing scheme based on the method of Roe[15] is used to discretize the convective terms, and the viscous terms are differenced using second-order central differences. The system of equations is integrated in pseudotime using an implicit Euler time discretization. The resulting discrete system of equations has the form

$$\frac{Q^{n+1} - Q^n}{\Delta\tau} = -R^{n+1} \tag{2}$$

where Q is the vector of dependent variables (pressure, u- and v-velocity components), $\Delta\tau$ is the time-step size in pseudotime, the superscript n is the iteration number, and R is the residual, composed of the discrete form of the convective and viscous terms. This system is linearized about pseudotime-level n, resulting in

$$\left(\frac{1}{\Delta\tau} + \frac{\partial R^n}{\partial Q}\right)\Delta Q = -R^n \tag{3}$$

where $\Delta Q = Q^{n+1} - Q^n$. This equation is iterated until a steady-state solution is obtained at which time $R^n \approx 0$. In the current implementation, the Jacobian of R on the left-hand side (LHS) of Eq. (3) is formed using a residual based on first-order differencing of the convective terms, whereas third-order differencing is used on the right-hand side (RHS). In addition, approximate Jacobians of flux differences from the upwind-differencing scheme are used, because exact Jacobians of these terms would require the formation of a tensor (see Ref. 16 for more details), and it has been assumed that the added computational costs would outweigh any benefit. The first-order LHS is used to reduce the bandwidth of the resulting LHS matrix, resulting in lower memory and computational requirements for the solution of Eq. (3). However, this use of approximate Jacobians can also slow the convergence to a steady state.

For turbulent flow calculations, the current study uses the turbulence model of Baldwin and Barth.[17] This model requires the solution of a single convective-diffusive partial-differential equation. This equation is uncoupled from the mean-flow equations during the solution process. The convective terms in the turbulence model are discretized using a first-order upwind-differencing scheme. The resulting discrete equation for the turbulence model has the same form as Eq. (3), except Q now represents a single variable at each grid point instead of three variables; the LHS of Eq. 3 is a banded matrix composed of five diagonals, each containing scalar entries. In all cases presented here, the turbulence model equation is solved using the same implicit scheme as the mean-flow equations.

Implicit Schemes

If Eq. (3) was solved exactly at each time step, and if the LHS of Eq. (3) was composed of the exact Jacobians, and an infinite time step was used, this would be a Newton iteration. In this case, quadratic convergence could be obtained if the Q^n was close to the exact solution. Since the LHS is composed of an approximate Jacobian of the RHS, and since the turbulence model is uncoupled from the mean-flow equations, the current solution procedure is not a Newton iteration. Thus the goal is to obtain an approximate solution to Eq. (3) in an efficient manner. Several methods are used to attempt to do this. This discrete form of the matrix from the LHS of Eq. (3) is a pentadiagonal banded matrix, where each entry on the diagonal consists of 3×3 blocks. Equation (3) can be written as

$$[D, 0, \ldots, 0, A, B, C, 0, \ldots, 0, E]\Delta Q = -R^n \tag{4}$$

where A, B, C, D, and E are the block diagonals. In the implementation of all of the implicit schemes, the code first computes and stores all of the terms in Eq. (4) and then proceeds with the solution procedure. This storage requires $48N$ words, where N is the total number of grid points. In the following, a subscript i refers to a single grid-point index when one can consider all of the grid points in a single vector of length N. The subscripts j and k are indices in the two computational-space directions ξ and η, respectively. When the data are stored in a single index vector, it is done so that j is the fastest changing (inner) index.

ILU Factorization

In the ILU formulation, the matrix on the LHS of Eq. (4) is replaced with the following factors:

$$[D, 0, \ldots, 0, A, B'][B']^{-1}[B', C, 0, \ldots, 0, E]$$

$$B'_i = B_i - A_i C'_{i-1} - D_i E'_{i-j\max}$$

$$C'_i = [B'_i]^{-1} C_i$$

$$E'_i = [B'_i]^{-1} E_i$$

Multiplying these factors together, one sees that a matrix of the same structure as the original LHS is obtained, except that there are additional diagonals of nonzero entries created just above the D diagonal and just below the E diagonal. These new entries are ignored in the approximation. This is known as ILU with zero additional fill, or ILU (0). See Ref. 12 for details on implementing ILU schemes with additional fill.

The ILU solver requires some significant initialization work, namely, the computation and storage of the B' diagonal. This requires $9N$ extra storage locations. When used as a preconditioner, this is done once at the beginning of the GMRES solution process and used repeatedly during the GMRES iteration cycle. This new set of factors gives an easy system to solve. The solution process can be vectorized by setting up an inner loop operating on all points on a diagonal line defined by $j + k =$ constant.

Point Relaxation

The PR algorithm iteratively solves a block diagonal system formed by multiplying all of the nonmain block diagonals by the current estimate for ΔQ and moving this to the RHS. This is done for each point, sweeping sequentially through the mesh. A forward sweep is composed of

$$[B]\Delta Q^{l+1} = -R^n - [D, 0, \ldots, 0, A]\Delta Q^{l+1}$$

$$-[C, 0, \ldots, 0, E]\Delta Q^l$$

and a backward sweep is performed by solving

$$[B]\Delta Q^{l+1} = -R^n - [D, 0, \ldots, 0, A]\Delta Q^l$$

$$-[C, 0, \ldots, 0, E]\Delta Q^{l+1}$$

In the current computations, a forward sweep plus a backward sweep counts as one sweep, denoted as PR(1). The solution process is initialized by setting ΔQ to zero. Then, a lower-upper (LU) factorization of the B block is formed. Thus the number of operations to solve this equation is minimized during the repeated sweeping process. This process can be vectorized by setting up an inner loop to compute ΔQ for all points on a diagonal line through the mesh given by $j + k =$ constant.

Line Relaxation

The LR process is similar to the PR method, except that more terms are kept on the LHS and a block tridiagonal system of equations is solved for an entire grid line at once. The algorithm is implemented so that either computational direction can be selected to be the sweep direction. For solving lines of constant k, a forward sweep is composed of

$$[A, B, C]\Delta Q^{l+1} = -R^n - [D]\Delta Q^{l+1} - [E]\Delta Q^l$$

and a backward sweep is performed by solving

$$[A, B, C]\Delta Q^{l+1} = -R^n - [D]\Delta Q^l - [E]\Delta Q^{l+1}$$

A forward plus a backward sweep counts as two sweeps and is denoted by LR(2). This process is also initialized by setting ΔQ to zero. Then, an LU factorization of the tridiagonal system is formed to minimize the number of operations during the sweeping process. The sweeping process is recursive and cannot be vectorized. It should be noted that the LR and PR schemes each require the same number of operations per point per sweep, because whereas the PR only has to solve a block diagonal system instead of a block tridiagonal system, it has additional block vector multiply operations for the additional RHS terms. The PR scheme will run faster on a vector computer because it can be vectorized. In practice, the PR routine runs about twice as fast as the LR routine on a Cray C-90 computer.

In both the PR and LR algorithms, zonal boundary conditions are enforced during the sweeping process. Typically, multiple sweeps are performed at each iteration. When computing a multiple zone grid, all zones are swept once before moving onto a second sweep. After each zone is swept, this new ΔQ^{l+1} is passed to all other zones that use this zone for their boundary conditions. In this fashion, information is propagated across zonal boundaries implicitly.

GMRES Method

The GMRES procedure of Saad and Schultz[8] is an iterative procedure for solving the linear system of equations of the form

$$Mx - b = 0$$

or, in the left preconditioned form,

$$PMx - Pb = 0$$

where P is the preconditioner that is an approximation to M^{-1}. The preconditioned matrix PM will have a smaller spectral radius than M, resulting in faster convergence for the GMRES procedure. A GMRES procedure using k search directions, known as GMRES(k), forms an approximation to the solution vector x given by

$$x_k = x_0 + y_1 v_1 + \cdots + y_k v_k$$

where x_0 is an initial guess to the solution x. The v_i are orthonormal vectors formed from the process

$$r_0 = PMx_0 - Pb, \qquad v_1 = r_0/||r_0||$$

Iterate: for $j = 1, 2, \ldots, k$:

$$h_{i,j} = (PMv_j, v_i), \qquad i = 1, 2, \ldots, j$$

$$w_{j+1} = PMv_j - h_{1,j}v_1 - h_{2,j}v_2 - \cdots - h_{j,j}v_j$$

$$h_{j+1,j} = ||w_{j+1}||$$

$$v_{j+1} = w_{j+1}/||w_{j+1}||$$

The y_i coefficients are computed so that the norm of the residual $||Mx_k - b||$ is minimized. An estimate of this norm is available during each iteration of the solution process as a function of the $h_{i,j}$ variables. The process requires approximately $(4 + k)$*3*N words of memory to apply to the mean-flow equations; therefore, it is not practical to use large values of k. Because of this memory usage, in the current work k is limited to no more than $k = 10$. A restart capability of the GMRES algorithm allows the iteration process to continue beyond $k = 10$ in this case. This is done simply by setting $x_0 = x_{10}$ and restarting from the beginning. If a total of 30 GMRES search directions is specified to the code, then two restarts are used. This is designated as GMRES(30).

The GMRES algorithm is implemented here so that the iterations can be stopped based on either of two criteria. The first is simply to specify the maximum number of search directions to be used. The second is to specify a tolerance for the error. One advantage of the GMRES process is that an accurate estimate of the norm of the residual is computed as part of the solution process. Using numerical tests, it was found that it was generally more efficient (in terms of obtaining a converged steady-state flow solution) to specify a set number of search directions for GMRES than it was to specify a tolerance for the norm of the residual.

Because the preconditioner must be utilized once for each GMRES search direction, it needs to be relatively cheap. Thus, when the PR and LR schemes are used as a preconditioner for GMRES, only two sweeps of the relaxation process are used.

Computed Results

Each of the different methods has been tested for three different geometries: laminar flow over a backward-facing step, turbulent flow over a NACA 4412 airfoil, and turbulent flow over a three-element airfoil. For each geometry and each method, calculations were run to determine an optimal time-step size, an optimal value for β, and an optimal number of relaxation sweeps or GMRES search directions. These calculations determined the best possible performance of each method for a particular case; then the most efficient run of each method was used to determined the best method for each case. Most cases ran best using an infinite time step, which is implemented by setting $\Delta\tau = 10^{12}$, so that $1/\Delta\tau$ is on the order of machine zero. In some cases, this large time step resulted in an instability, and the time step was reduced. In other cases, a large time step remained stable, but a smaller time step resulted in better efficiency. Testing different time-step sizes usually involved running with $\Delta\tau = 10^{12}$, 10.0, 1.0, and 0.1. To test different values of β, cases were run using values of 0.1, 1, 5, 10, 20, 50, 100, and 200. See Ref. 5 for an example showing the effect of β on the convergence.

In each comparison, the maximum divergence of velocity is plotted vs computing time, given in central processing unit (CPU) seconds on a Cray C-90 computer. In addition, symbols are overlaid on the plotting line at every 50 iterations, indicating the number of iterations required by each method. Most cases are run until the divergence of velocity reaches machine zero. In general, a useful steady-state solution has been obtained long before this value is reached; usually this requires only that the maximum nondimensional divergence of velocity be reduced to approximately 10^{-2}. Finally, for each computational test case, a table summarizing the convergence characteristics for each scheme is presented. The following parameters are given in these tables: the value of β and $\Delta\tau$ used in the run; the amplification factor, or average factor by which the residual is reduced at each iteration; the computing time used per point per iteration, given in microseconds; and the computing time in milliseconds used to converge the solution divided by the number of grid points. To generate these numbers, the solution was considered converged when the maximum nondimensional divergence of velocity dropped below 10^{-3}. Although the amplification factor and computing time per point per iteration are useful characteristics of an algorithm, they do not give a true measure of the efficiency; the last column in these tables is the best measure of the overall performance of the algorithm.

Backward-Facing Step

The laminar flow over a backward-facing step was computed for an expansion ratio of 1 to 1.94. The Reynolds number, based on the downstream height and the average inflow velocity, was 8×10^2. Figure 1 shows a close view of the grid near the step and the specified inflow velocity vectors. The inflow was 2 step heights upstream of the step, and the outflow boundary was placed 50 step heights downstream of the step. A single-zone H grid with dimensions of 160×61 was used. The grid points inside the step were blanked out, and the no-slip boundary conditions for the step faces were applied to internal grid points. This flow problem has been used previously as a validation case[3] for the INS2D flow solver. These previous results showed good agreement with the experimental results of Armaly et al.[18] The Reynolds number 8×10^2 case was the highest Reynolds number run in this previous study and was the slowest to converge.

For the backward-facing step flow, the only schemes that did not run best with an infinite time step were the ILU scheme, which was unstable for $\Delta\tau > 1.0$, and GMRES+LR, which was more

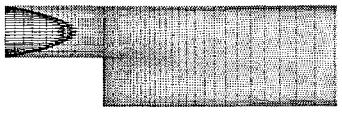
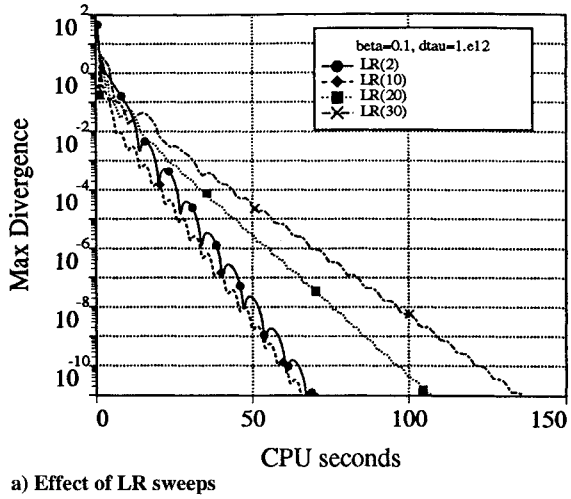
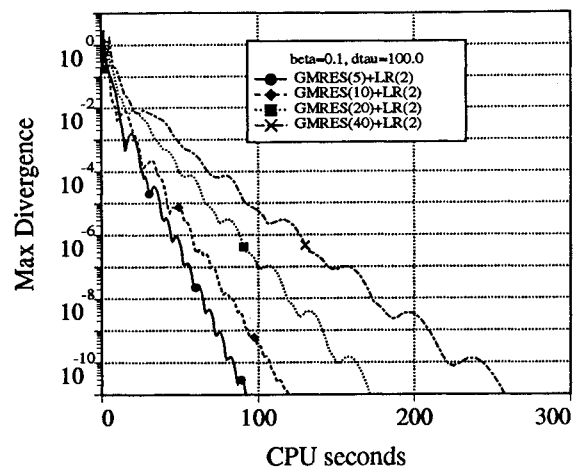


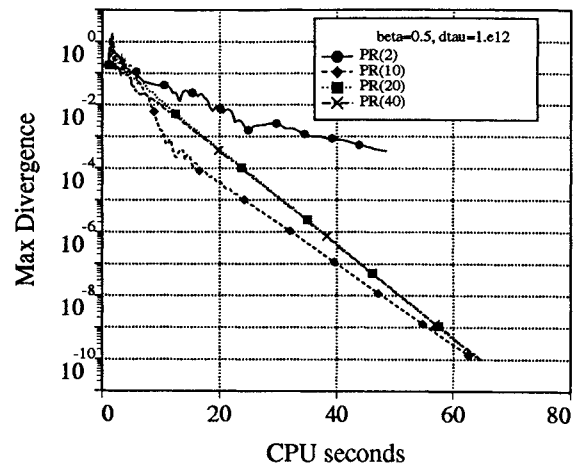
Fig. 1 Grid and inflow velocity vectors for the backward-facing step flow.



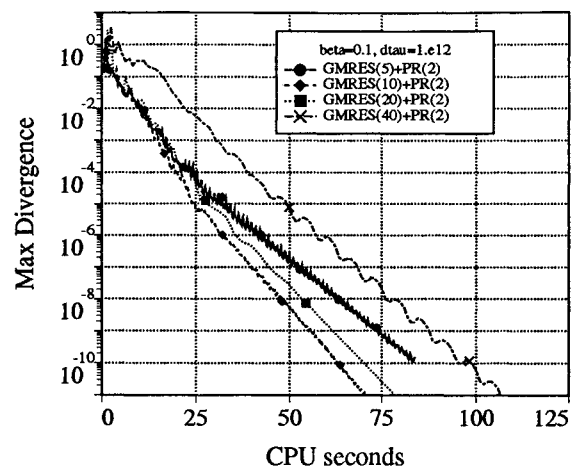
a) Effect of LR sweeps



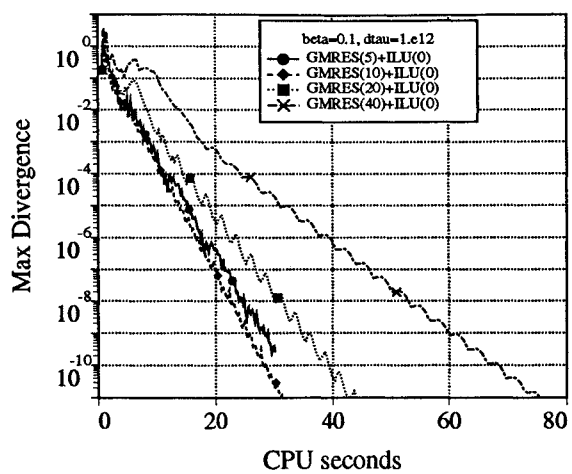
c) Effect of GMRES+LR search directions



b) Effect of PR sweeps



d) Effect of GMRES+PR search directions



e) Effect of GMRES+ILU search directions

Fig. 2 Convergence for the backward-facing step flow.

efficient with $\Delta\tau = 100$. Figure 2 shows the convergence for each scheme for a different number of sweeps/search directions. The LR scheme is best when using only 10 sweeps; the efficiency of the PR scheme is nearly constant as long as it is using at least 10 sweeps. The GMRES schemes are most efficient using either 5 or 10 search directions. For all of these schemes, it can be seen that when more sweeps/search directions are used they converge in fewer iterations, but the penalty of extra computing time per iteration causes this approach to be less efficient. Figure 3 shows the best convergence plot for each method for the backward-facing step flow; a summary of the convergence properties is shown in Table 1. Figure 3 and the last column of Table 1 show that GMRES+ILU converges faster than all other methods; the PR method is not far behind. The GMRES+LR scheme is less efficient than the LR scheme alone, even though it converges in significantly fewer iterations. The LR scheme shows the lowest amplification factor, but at nearly double the cost per iteration as the GMRES+ILU method and thus requires more time to converge.

NACA 4412 Airfoil

The flow over a NACA 4412 airfoil at a Reynolds number of 1.5×10^6 and an angle of attack of 13.87 deg was computed using

Table 1 Convergence for backward-facing step

Scheme	β	$\Delta\tau$	A.F.	Microseconds/ point/iteration	Milliseconds/ point
LR(10)	0.1	10^{12}	0.818	40.1	2.17
PR(10)	0.5	10^{12}	0.912	16.7	1.69
ILU	0.1	1	0.994	9.9	23.00
GMRES(5)+LR	0.1	100	0.788	61.4	2.70
GMRES(10)+PR	0.1	10^{12}	0.850	32.9	1.97
GMRES(10)+ILU	0.1	10^{12}	0.826	21.1	1.14

Table 2 Convergence for NACA 4412 grid 1

Scheme	β	$\Delta\tau$	A.F.	Microseconds/ point/iteration	Milliseconds/ point
LR(5)	10	10^{12}	0.862	17.8	1.98
PR(20)	10	10^{12}	0.785	34.8	2.31
ILU	20	10^{12}	0.959	9.1	3.55
GMRES(10)+LR	5	10^{12}	0.680	73.8	3.17
GMRES(10)+PR	5	10^{12}	0.682	49.3	2.12
GMRES(10)+ILU	5	10^{12}	0.676	27.2	1.14

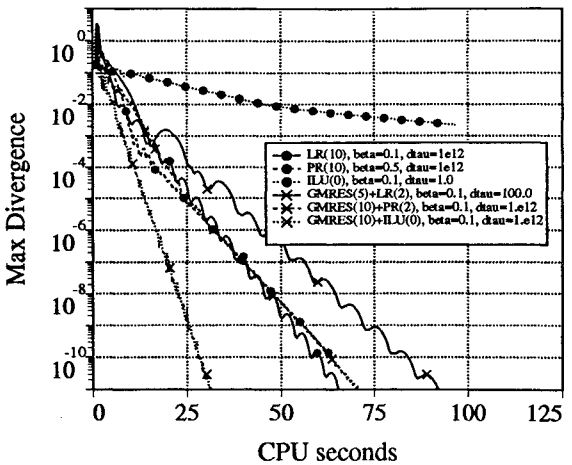


Fig. 3 Summary of convergence for the backward-facing step flow.

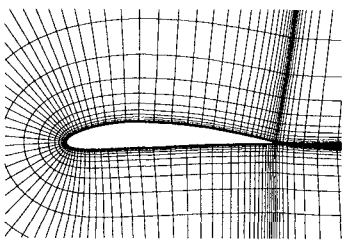


Fig. 4 119 × 31 grid around the NACA 4412 airfoil.

four grids of varying refinement. The dimensions of these grids are 119 × 31, 237 × 61, 473 × 121 and 945 × 241. These will be referred to as grids 1, 2, 3, and 4, respectively. Grids 1, 2, and 3 were generated by removing every other point in each direction from the succeeding finer grid. The normal spacing on the finest grid was set to 5×10^{-6} chords. Figure 4 shows the airfoil configuration with grid 1. This configuration has been studied experimentally by Coles and Wadcock.[19] Previous computations with the INS2D code[20] showed good agreement with the experimental data. At this angle of attack, the airfoil is near stall conditions and has a small amount of separation occurring at the trailing edge.

In the computations using grid 1, all of the methods ran with a time step of 10^{12}. It was found that the LR scheme was most efficient using five sweeps. The PR runs converged well for 10, 20, and 40 sweeps, with 20 being the best. The GMRES schemes showed a loss in efficiency when using more than 10 search directions. Figure 5 plots the best convergence of each method. This figure shows similar trends to the backward-facing step problem: GMRES+ILU outperformed all other methods. Table 2 shows that all of the GMRES methods had remarkably low amplification factors, below 0.7, for this case. All methods had higher cost per point per iteration for this case because the smaller dimensions led to shorter vector lengths. The cost per point to converge for the GMRES+ILU was the same as the laminar backward-facing step flow.

Table 3 Convergence for NACA 4412 grid 2

Scheme	β	$\Delta\tau$	A.F.	Microseconds/ point/iteration	Milliseconds/ point
LR(10)	50	10^{12}	0.888	25.2	3.68
PR(20)	50	10^{12}	0.914	26.6	5.13
ILU	10	0.1	0.990	6.7	12.00
GMRES(5)+LR	5	10^{12}	0.827	40.5	3.77
GMRES(10)+PR	50	1.0	0.865	37.4	4.48
GMRES(10)+ILU	5	10^{12}	0.780	19.6	1.45

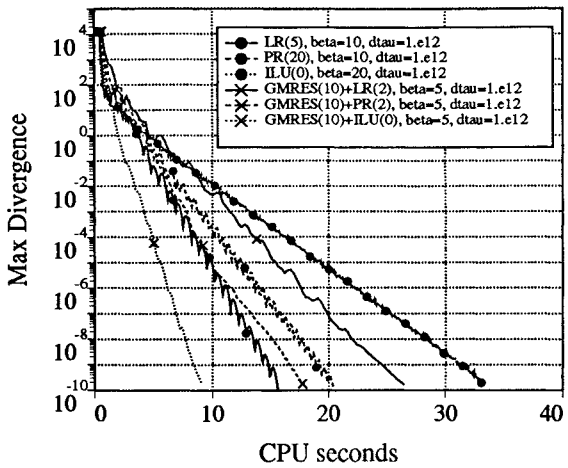


Fig. 5 Convergence for the flow over a NACA 4412 airfoil with grid 1.

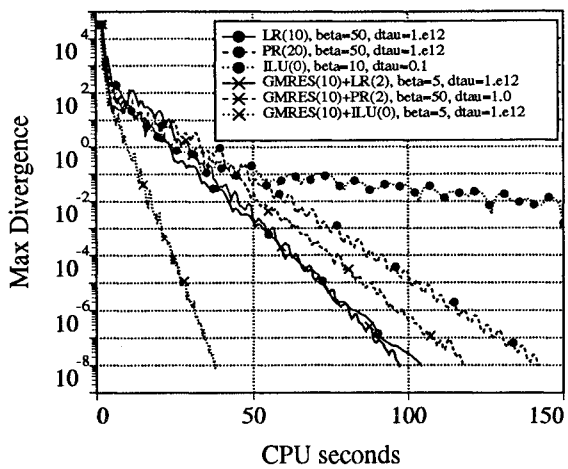


Fig. 6 Convergence for the flow over a NACA 4412 airfoil with grid 2.

The results for grid level 2 are shown in Fig. 6 and in Table 3. For this grid, the GMRES+ILU approach outperforms all of the other methods by at least a factor of 2. In all cases, the amplification factor and the cost per point have increased with the increase in grid density.

Based on the results thus far, it is apparent that LR and PR are too expensive to be used as effective preconditioners for GMRES. It was also found that the ILU scheme alone cannot handle finer grid cases, as evidenced by the decay of the ILU amplification factor to 0.99 for the previous case. Thus, for the remainder of test cases in this study, these three approaches have been eliminated. The results of the three remaining methods for the grid 3 case are shown in Fig. 7 and Table 4. All three methods ran best using a time step of 1.0. Again, the GMRES+ILU computations outperform the LR and PR runs by over a factor of 2.

Computations for grid level 4 showed that only the GMRES+ILU scheme would converge for this problem. The results of this run are

Table 4 Convergence for NACA 4412 grid 3

Scheme	β	$\Delta\tau$	A.F.	Microseconds/ point/iteration	Milliseconds/ point
LR(10)	100	1.0	0.953	23.2	8.79
PR(40)	5	1.0	0.929	51.3	12.56
GMRES(10)+ILU	5	1.0	0.917	18.7	3.91

Table 5 Convergence for NACA 4412 grid 4

Scheme	β	$\Delta\tau$	A.F.	Microseconds/ point/iteration	Milliseconds/ point
GMRES(10)+ILU	5	1.0	0.961	23.1	11.2

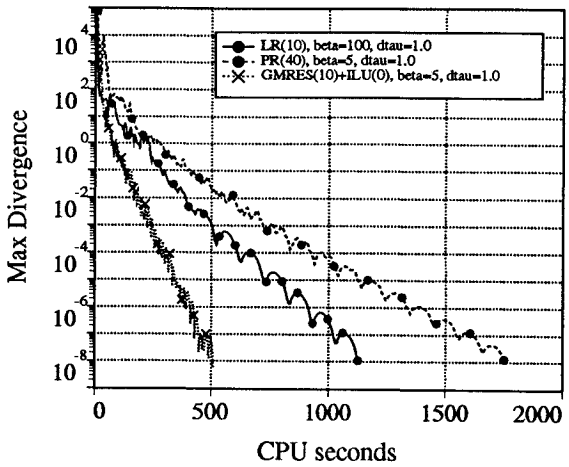


Fig. 7 Convergence for the flow over a NACA 4412 airfoil with grid 3.

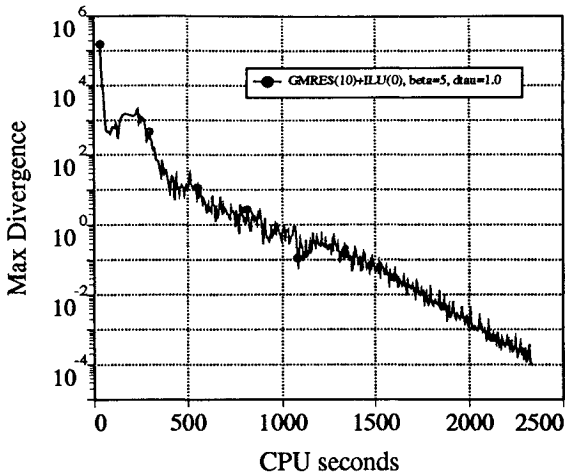


Fig. 8 Convergence for the flow over a NACA 4412 airfoil with grid 4.

shown in Fig. 8 and Table 5. Examining the trend of the convergence rate over all four grid levels shows that all methods suffered an increase in the amplification factor with increasing density. The effect of this is shown in Fig. 9, which plots the CPU time required to reach convergence per point vs the grid level for all of the airfoil computations. Even the most robust method, GMRES+ILU, saw an order-of-magnitude increase in this CPU cost from the coarsest grid to the finest grid. All of the methods required a reduction in the time-step size as the grid levels increased. Increasing the β parameter tended to be a stabilizing influence; the LR and PR methods tended to require larger β for the finer grids. One apparent contradiction to this is the use of $\beta = 5$ for the PR grid 3 case. In fact, the PR convergence for grid 3 did not vary greatly with β; values from 5 to 100 all converged slowly. The $\beta = 5$ case was slightly better than the $\beta = 100$ run. One distinct advantage of the GMRES+ILU method is that it has optimum convergence with the same value of β for all of the grid levels.

Table 6 Convergence for three-element arifoil

Scheme	β	$\Delta\tau$	A.F.	Microseconds/ point/iteration	Milliseconds/ point
LR(10)	100	1.0[a]	0.990	22.6	49.7
PR(20)	10	1.0	0.927	52.8	14.7
GMR(10)+ILU	10	10^{12}	0.892	29.2	5.37

[a] $\Delta\tau$ reduced by a factor of 0.5 every 250 iterations.

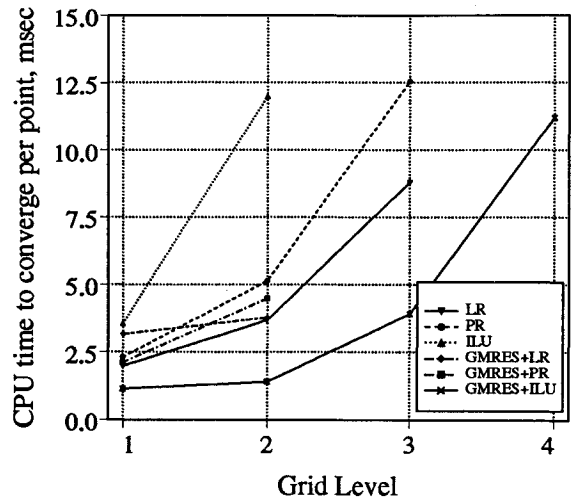


Fig. 9 CPU time to converge per point vs grid level for the flow over a NACA 4412 airfoil.

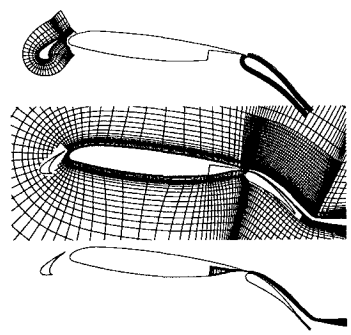


Fig. 10 Grid around the three-element airfoil.

Three-Element Airfoil

The flow over a three-element airfoil was computed at an angle of attack of 8.0 deg and a Reynolds number of 9×10^6. This geometry has recently become a standard test case for multielement airfoil flows. This geometry is a McDonnell Douglas airfoil and has been tested extensively at the NASA Langley low-turbulence pressure tunnel (LTPT).[21] The configuration consists of a leading-edge slat deflected 30 deg and a trailing-edge flap deflected 30 deg. This geometry has been used as a test case for the INS2D flow solver for a number of turbulence model and grid resolution studies.[1] The grid and flow conditions of the current problem were some of the most difficult cases to converge in the previous study.

Figure 10 shows the grids used for the three-element airfoil. For clarity, only every other grid line in each direction is shown. A total of 68,000 grid points and six zones were used: a 121×41 C grid around the slat (top of Fig. 10); a 321×101 C grid around the main element (near field shown in middle of Fig. 10); a 141×51 C grid around the flap (top of Fig. 10); a 41×31 H grid in the wake of the flap (bottom of Fig. 10); a 131×61 H grid extending from the main elements' flap cove to the downstream far field (bottom of Fig. 10); and a 141×101 embedded grid above the flap, used to help resolve the merging wake in this region (middle of Fig. 10). The normal wall spacing for all grids is 2×10^{-6} chords. The overlaid chimera scheme allows individual grids to be generated for each airfoil element. When the grid for one element intersects another airfoil element, a hole is cut to remove grid points lying inside the element. This creates a hole boundary. The fringe-point variables on the hole boundaries are updated by interpolating the value of

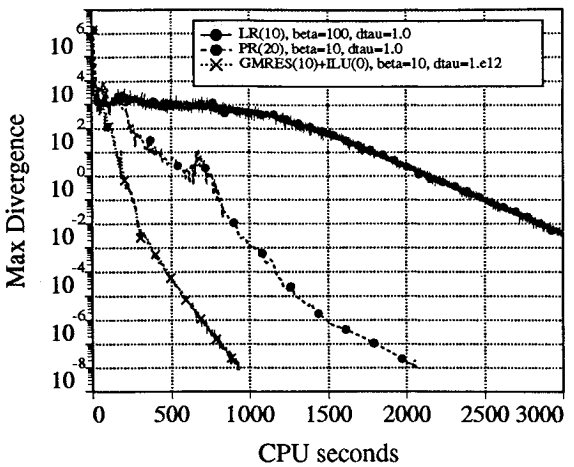


Fig. 11 Summary of convergence for the flow over a three-element airfoil.

the dependent variables from interior points of neighboring grids. Similarly, the variables on the outer boundaries of all but the main-element grid are updated using interpolation of dependent variables from neighboring grids.

The LR and PR computations for this configuration required a reduction in the time step to $\Delta\tau = 1.0$ to remain stable; GMRES+ILU ran with a time step of 10^{12}. The LR computations were extremely slow to converge and were found to benefit from a further decrease in the time step during the calculations, and so the LR time step was decreased by a factor of 2 every 250 iterations. The PR computations did not suffer from the same problems. It was found that the PR computations converged best using 20 sweeps. The GMRES+ILU computations performed best with 10 search directions. The best performance of each method is plotted in Fig. 11 and shown in Table 6. It can be seen that the GMRES+ILU method outperformed the other two; it converges about nine times faster than the LR scheme and about three times faster than the PR computations.

Conclusion

Several implicit schemes have been implemented into the INS2D code and tested for the flow over a backward-facing step, a NACA airfoil, and a three-element airfoil. The results indicate that when running on a single processor the GMRES method preconditioned with ILU factorization outperforms line relaxation and point relaxation and that the latter two methods are not efficient preconditioners for GMRES. The GMRES+ILU method has provided between a factor of 2 and 9 improvement in CPU costs over previously published results of the INS2D code. In addition, the point-relaxation scheme performed remarkably well for the three-element airfoil problem. Extensive tests have indicated that GMRES is most efficient when using 10 search directions; if more search directions are used, the computation will converge in fewer iterations, but at a greater cost. The GMRES+ILU algorithm also remained more stable than the LR or PR methods; it ran with larger time steps and did not require an increase in the value of the artificial compressibility constant β for the finger grids used in the NACA 4412 airfoil calculations.

The GMRES+ILU algorithm has been shown to work quite well for a two-dimensional flow solver. The success with this algorithm has also been observed recently by others.[6,9–11] In particular, Venkatakrishnan and Mavriplis[11] performed a similar study of implicit solvers for an unstructured flow code; they found their GMRES+ILU worked best. As it was utilized here, the GMRES algorithm would be very memory intensive for a three-dimensional flow solver. Future work will concentrate on utilizing a matrix-free method of the GMRES algorithm. The most important part of this work will be determining an effective preconditioner that will not require significant amounts of memory.

References

[1]Rogers, S. E., Menter, F. R., Mansour, N. N., and Durbin, P. A., "A Comparison of Turbulence Models in Computing Multi-Element Airfoil Flows," AIAA Paper 94-0291, Jan. 1994.

[2]Kiris, C., Kwak, D., and Rogers, S. E., "Computation of Incompressible Viscous Flows Through Turbopump Components," NASA TM-103911, March 1992.

[3]Rogers, S. E., and Kwak, D., "An Upwind Differencing Scheme for the Steady-State Incompressible Navier–Stokes Equations," *Journal of Applied Numerical Mathematics*, Vol. 8, No. 1, 1991, pp. 43–64.

[4]Rogers, S. E., and Kwak, D., "An Upwind Differencing Scheme for the Time Accurate Incompressible Navier–Stokes Equations," AIAA Paper 88-2583, June 1988; see also *AIAA Journal*, Vol. 28, No. 2, 1990, pp. 253–262.

[5]Rogers, S. E., Kwak, D., and Kiris, C., "Numerical Solution of the Incompressible Navier–Stokes Equations for Steady-State and Time-Dependent Problems," AIAA Paper 89-0463, Jan. 1989; see also *AIAA Journal*, Vol. 29, No. 4, 1991, pp. 603–610.

[6]Saad, Y., "Supercomputer Implementations of Preconditioned Krylov Subspace Methods," *Algorithmic Trends in Computational Fluid Dynamics*, edited by M. Y. Hussaini, A. Kumar, and M. D. Salas, Springer–Verlag, New York, 1993, pp. 107–136.

[7]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*, unpublished book available on NETLIB: ftp netlib2.cs.utk.edu; cd linalg; get templates.ps; 1993.

[8]Saad, Y., and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal of Scientific and Statistical Computing*, Vol. 7, No. 3, 1986, pp. 856–869.

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

[10]Ajmani, K., Liou, M., and Dyson, R. W., "Preconditioned Implicit Solvers for the Navier–Stokes Equations on Distributed-Memory Machines," AIAA Paper 94-0408, Jan. 1994.

[11]Venkatakrishnan, V., and Mavriplis, D. J., "Implicit Solvers for Unstructured Meshes," *Journal of Computational Physics*, Vol. 105, No. 1, 1993, pp. 83–91.

[12]Wigton, L. B., Yu, N. J., and Young, D. P., "GMRES Acceleration of Computational Fluid Dynamics Codes," AIAA Paper 85-1494, July 1985.

[13]Meijerink, J. A., and van der Vorst, H. A., "Guidelines for the Usage of Incomplete Decompositions in Solving Sets of Linear Equations as They Occur in Practical Problems," *Journal of Computational Physics*, Vol. 44, No. 1, 1981, pp. 134–155.

[14]Chorin, A. J., "A Numerical Method for Solving Incompressible Viscous Flow Problems," *Journal of Computational Physics*, Vol. 2, No. 1, 1967, pp. 12–26.

[15]Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372.

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

[17]Baldwin, B., and Barth, T., "A One-Equation Turbulence Transport Model for High Reynolds Number Wall-Bounded Flows," NASA TM-102847, Aug. 1990.

[18]Armaly, B. F., Durst, F., Periera, J. C. F., and Schönung, B., "Experimental and Theoretical Investigation of Backward-Facing Step Flow," *Journal of Fluid Mechanics*, Vol. 127, Feb. 1983, pp. 473–496.

[19]Coles, D., and Wadcock, A. J., "Flying-Hot-Wire Study of Flow Past a NACA 4412 Airfoil at Maximum Lift," *AIAA Journal*, Vol. 17, No. 4, 1979, pp. 321–329.

[20]Rogers, S. E., Wiltberger, N. L., and Kwak, D., "Efficient Simulation of Incompressible Viscous Flow over Single and Multi-Element Airfoils," AIAA Paper 92-0405, Jan. 1992; see also *Journal of Aircraft*, Vol. 30, No. 5, 1993, pp. 736–743.

[21]Chin, V., Peters, D., Spaid, F., and McGhee, R., "Flowfield Measurements about a Multi-Element Airfoil at High Reynolds Numbers," AIAA Paper 93-3137, July 1993.