FED-Vol. 215, Numerical Developments in CFD
ASME 1995

INVESTIGATION OF NEWTON-KRYLOV ALGORITHMS FOR LOW MACH NUMBER COMPRESSIBLE FLOW

P. R. McHugh, D. A. Knoll, V. A. Mousseau, and G. A. Hansen
Idaho National Engineering Laboratory
P. O. Box 1625
Idaho Falls, ID. 83415

ABSTRACT

Fully coupled Newton-Krylov algorithms are used to solve steady low speed compressible flow past a backward facing step for different flow Mach and Reynolds numbers. Various preconditioned Krylov iterative methods are used to solve the linear systems that arise on each Newton step, specifically Lanczos-based and Arnoldi-based algorithms. Several preconditioning strategies are considered to improve the performance of these iterative techniques, including incomplete lower-upper factorization with various levels of fill-in [ILU(k)] and domain based additive and multiplicative Schwarz type preconditioning both with and without overlapping domains. The ILU(k) preconditioners were generally less reliable for lower values of the flow Mach number, and exhibited strong sensitivity to cell ordering. In addition, the parallel nature of the domain based preconditioners is exploited on both a shared memory computer and a distributed system of workstations. Important aspects of the numerical solutions are discussed.

NOMENCLATURE

$\tilde{c}_p$	Specific heat capacity at constant pressure
$\tilde{c}_v$	Specific heat capacity at constant volume
d	Newton damping coefficient
f_i	i-th component of F.
F	Discrete governing equations vector
J	Jacobian matrix
k	Level of ILU fill-in
m	Krylov subspace dimension
$\bar{m}$	Average Krylov iterations
Ma	Inlet Mach number
n	Newton iteration number
N	System dimension
p	Dimensionless pressure, $p = \tilde{p}/(\tilde{\rho}\tilde{u}_o^2)$
Pe	Peclet number = $Re\,Pr$
Pr	Prandtl number = $\tilde{\upsilon}/\tilde{\alpha}$
Re	Reynolds number = $\tilde{u}_o\tilde{s}/\tilde{\nu}$
R	Convergence parameter
$\tilde{s}$	Step height
T	Dimensionless temperature = $\tilde{T}/\tilde{T}_o$
u	Dimensionless principal velocity = $\tilde{u}/\tilde{u}_o$
v	Dimensionless transverse velocity = $\tilde{v}/\tilde{u}_o$
x	State variable vector
δx	Newton iteration update vector
Δx_j	Perturbation in the jth component of x
x	Principal coordinate variable
Δx	Grid spacing in x-direction
y	Transverse coordinate variable
Δy	Grid spacing in y-direction

Greek Symbols:

$\tilde{\alpha}$	Thermal diffusivity
γ	Ratio of specific heat capacities = $\tilde{c}_p/\tilde{c}_v$
ε	Tolerance parameter for Krylov iteration
$\tilde{\nu}$	Kinematic viscosity
ρ	Dimensionless density = $=\tilde{\rho}/\tilde{\rho}_o$

Subscripts:

n	Newton iteration number
o	Reference value

Superscripts:

i	Refers to inner iteration
n	Newton iteration number
o	Refers to outer iteration

Operators:

$[\,]^T$	Transpose of [ ]
$\tilde{[\,]}$	Indicates that [ ] is dimensional
$\|\cdot\|$	Euclidean norm
$\|\cdot\|_\infty$	L_∞ norm

INTRODUCTION

The objective of this work is to demonstrate the effectiveness of Newton-Krylov algorithms for steady state calculations of low Mach number compressible flow. Solution of the compressible flow equations at low Mach numbers is important in instances where: a low Mach number region is imbedded within a high speed flow, some thermally driven flows, and other flow situations where density variations are important, e.g., chemically reacting flow.

Finite volume discretization and a staggered mesh are used to discretize the nonlinear governing equations. The subsequent nonlinear algebraic equations are linearized using Newton's method. In this study, Krylov subspace based iterative algorithms are considered for solving these linear systems, specifically Lanczos-based (Lanczos, 1952) and Arnoldi-based (Arnoldi, 1951) algorithms. The Lanczos-based algorithms include the conjugate gradients squared (CGS)

DISCLAIMER

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

algorithm of Sonneveld (1989), the Bi-CGSTAB algorithm of van der Vorst (1992), and the transpose-free quasi-minimal residual (TFQMR) algorithm of Freund (1993). The restarted generalized minimal residual algorithm [GMRES(m)] of Saad and Schultz (1986) is chosen from the Arnoldi-based methods, where m is the selected dimension of the Krylov subspace.

The primary advantage in using an iterative solver is reduced memory requirements. Newton's method can be very memory intensive due to Jacobians with large matrix bandwidths. Several recent articles have used conjugate gradient-like algorithms to reduce the large memory requirements associated with problems of this type (e.g., Ajmani, 1993 and 1994; Cai, et al., 1993; Dutto, 1993 and Dutto et al., 1994a and 1994b; Habashi, 1994; Orkwis, 1993; and Venkatakrishnan 1991 and 1993). In particular, Ajmani (1994) and Dutto et al. (1994a and 1994b) have investigated efficient parallel implementations of these numerical techniques.

The objective of this paper necessitates a comparison of various Krylov solvers, listed above, which are applicable to both non symmetric and non positive definite linear systems. Specifically, the advantages and disadvantages of Lanczos-based and Arnoldi-based algorithms are compared and contrasted. Additionally, the efficiency of a Krylov algorithm is strongly tied to preconditioner effectiveness. Consequently, the performance of incomplete lower-upper (ILU) preconditioners (see Meijerink and van der Vorst, 1977) plus additive and multiplicative Schwarz preconditioners (see Cai and Saad, 1993) are investigated for different values of the flow Mach and Reynolds numbers. Preconditioner effectiveness is measured not only by lower inner iteration counts, but also by CPU efficiency and memory cost. The latter two also being dependent upon either parallel or serial implementation.

This paper is organized as follows. First, the governing equations and assumptions are described along with the model problem definition. Next, the general numerical solution algorithm is outlined. In the penultimate section, computational results are presented that illustrate the important numerical features of the different Krylov algorithms and preconditioners (including parallel implementations) with respect to solutions of the low Mach number compressible flow model problem. Important aspects of the numerical solutions are summarized in the final section.

MODEL PROBLEM

The physical situation of interest in this study is low Mach number (≤0.25) compressible flow past a backward facing step. The governing equations of interest (in dimensionless form) are given by,

$$\frac{\partial \rho u}{\partial x}+\frac{\partial \rho v}{\partial y}=0, \tag{1}$$

$$\frac{\partial \rho u^2}{\partial x}+\frac{\partial \rho uv}{\partial y}=-\frac{\partial p}{\partial x}+\frac{1}{Re}\frac{\partial}{\partial x}\left[2\frac{\partial u}{\partial x}-\frac{2}{3}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\right]+\frac{1}{Re}\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right), \tag{2}$$

$$\frac{\partial \rho uv}{\partial x}+\frac{\partial \rho v^2}{\partial y}=-\frac{\partial p}{\partial y}+\frac{1}{Re}\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)+\frac{1}{Re}\frac{\partial}{\partial y}\left[2\frac{\partial v}{\partial y}-\frac{2}{3}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\right], \tag{3}$$

$$\frac{\partial \rho uT}{\partial x}+\frac{\partial \rho vT}{\partial y}=\frac{\gamma}{Pe}\left[\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right]-\gamma(\gamma-1)Ma^2 p\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right), \tag{4}$$

and

$$p=\rho T/\left(\gamma Ma^2\right), \tag{5}$$

where constant transport properties have been assumed along with Stokes condition (White, 1974). Additionally, heat generation, viscous dissipation, and the effects of buoyancy forces have been neglected.

The backward facing step model problem geometry and boundary conditions are presented in Figure 1. Also shown in Figure 1 are the Mach number contours corresponding to a solution with an inlet Mach number of 0.25 and a Reynolds number of 100. These contours illustrate the region of separated recirculating flow that exists downstream of the step. This solution was obtained using a uniform grid with 16 cells in the x-direction and 80 cells in the y-direction (i.e., 16x80). This grid had equal mesh spacings in both directions.

NUMERICAL SOLUTION ALGORITHM

Finite volume discretization, using simple upwinding for convection terms, is used to derive a set of discrete nonlinear algebraic equations from the continuous partial differential equations described previously. This discretization assumes thermodynamic variables are located at volume centers, while velocities are located at volume faces, i.e. a staggered grid. This system of nonlinear algebraic equations is linearized using Newton's method. The resulting set of linear algebraic equations are then solved using preconditioned Krylov subspace algorithms. In addition, several performance enhancement techniques are used to improve overall algorithm robustness and to simplify implementation. These include an efficiently evaluated numerical Jacobian, the use of mesh sequencing, and an adaptively damped Newton iteration (Knoll and McHugh, 1992).

Newton's Method

Newton's method solves nonlinear systems of the form, $F(\mathrm{x})=\left[f_1(\mathrm{x}),f_2(\mathrm{x}),\ldots,f_N(\mathrm{x})\right]^T=0$, for the state vector, $\mathrm{x}=\left[x_1,x_2,\ldots,x_N\right]^T$. Application of Newton's method requires the solution of the linear system,

$$\mathrm{J}^n\delta\mathrm{x}^n=-\mathrm{F}(\mathrm{x}^n), \tag{6}$$

where the new solution approximation is obtained from,

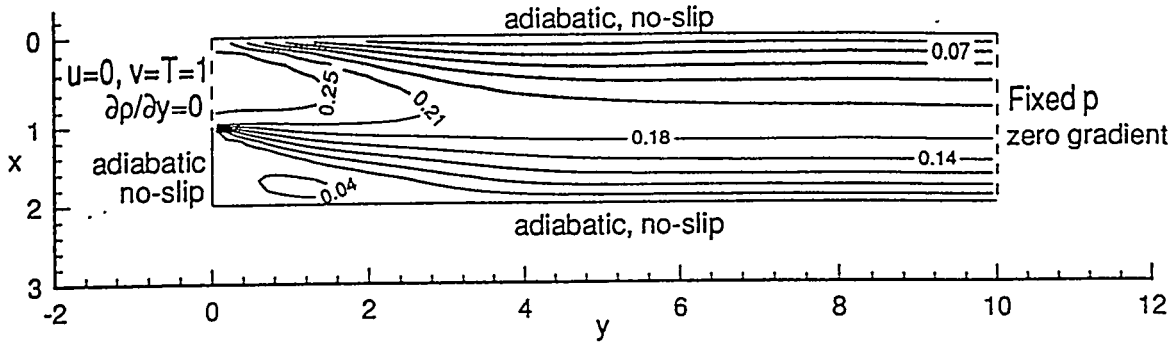

Figure 1. Model problem geometry and Mach number contours for Ma=0.25 and Re=100 on a uniform 16x80 grid.

$$x^{n+1} = x^n + d\delta x^n. \tag{7}$$

The constant, d ($0 < d \leq 1$), in Equation (7) is used to damp the Newton updates. The damping strategy is designed to prevent the calculation of non-physical variable values (i.e., negative temperatures and densities), and to scale large variable updates when the solution is far from the true solution. Note that the elements of the Jacobian appearing in Equation (6) are evaluated numerically using finite difference approximations (Knoll and McHugh, 1992). The convergence criterion for the outer Newton iteration is based upon a relative update defined by the following outer iteration convergence parameter,

$$R_n^o = \underset{all\ i}{Max}\left[\frac{|\delta x_i^n|}{Max\left\{|x_i^n|,1\right\}}\right]. \tag{8}$$

Convergence is then assumed when $R_n^o < 1x10^{-6}$ and $\left\|F(x^n)\right\|_\infty < 1x10^{-6}$.

Krylov Subspace Based Algorithms

Equation (6) requires solution of large linear systems of dimension N. Direct linear solution techniques often become impractical for large linear systems because of high memory and CPU cost. A viable alternative is the use of Krylov subspace based iterative methods, specifically Arnoldi-based and Lanczos-based conjugate gradient-like algorithms.

The Arnoldi-based GMRES algorithm was derived so as to maintain optimality, but at the expense of economical vector recurrences (see Saad and Schultz, 1986). Consequently, the work and storage requirements of GMRES increase with the iteration count. Therefore, practical implementations frequently require use of the restarted version, GMRES(m), where m is the maximum dimension of the Krylov subspace. The restarted algorithm is then only optimal within an m-iteration cycle, and so frequent restarts can lead to slow convergence or even algorithm stall.

In contrast, the Lanczos-based Krylov algorithms are derived so as to maintain economical recurrences, but at the expense of optimality. Note also that the non symmetric Lanczos process itself is susceptible to breakdowns, making algorithms derived based upon this process also susceptible to breakdown. CGS was the first transpose-free algorithm of this type developed (Sonneveld, 1989). It was derived by squaring the BCG (Lanczos, 1952 and Fletcher, 1976) polynomial relations in order to eliminate use of the matrix transpose. CGS can exhibit very rapid convergence compared to BCG, but its convergence is sometimes marred by very wild oscillations, which under certain conditions can lead to inaccurate solutions (van der Vorst, 1992). This difficulty led to the development of both Bi-CGSTAB (van der Vorst, 1992), which uses local steepest descent steps, and TFQMR (Freund, 1993), which uses the quasi-minimization idea, to obtain more smoothly convergent CGS-like solutions.

An iterative linear equation solver allows Equation (6) to be solved less accurately during the initial Newton iterations, and more accurately as the true solution is approached. This is in contrast to the use of a direct solver, which requires the same amount of work each time Equation (6) is solved. An 'inexact' Newton convergence criterion similar to that proposed by Averick and Ortega (1991) and Dembo et al. (1982) is employed. Specifically, the inner Krylov iteration is assumed converged when,

$$R_n^i = \frac{\|J^n \delta x^n + F(x^n)\|}{\|F(x^n)\|} < \varepsilon, \tag{9}$$

The selection of ε is highly empirical. Based on the results of McHugh and Knoll (1993), $\varepsilon = 10^{-2}$ is used. This inner iteration convergence criteria is combined with a limit on the maximum number of inner iterations, set at 200.

An important issue with regard to the efficient implementation of conjugate gradient-like algorithms is preconditioning. In this work two classes of preconditioners are considered. The first class is incomplete lower-upper (ILU) factorization type preconditioners (Meijerink and van der Vorst, 1977). Specifically, ILU preconditioners with various amounts of fill-in are considered based on the level of fill-in idea of Watts (1981). In the current implementation, the banded structure of the Jacobian matrix is exploited and so only non-zero diagonals are stored. Consequently, the ILU(k) preconditioners used in this study are derived from this sparsity pattern. The ILU preconditioning choice is applied from the right. This choice is often used because it ensures that the preconditioner does not influence the residual norm used to evaluate convergence.

The second preconditioning class is the use of domain-based preconditioners. These include preconditioners based upon the additive and multiplicative Schwarz algorithms (see Dryja and Widlund, 1994; and Cai and Saad, 1993). Several different blocking strategies are considered both with and without overlapping domains. Full coupling between variables is retained within each sub-domain and LINPACK banded Gaussian elimination (Dongarra et al., 1979) is used for the required sub-domain solves. These domain-based preconditioners have the additional advantage of being more amenable to parallel implementation compared with ILU-type preconditioners. Note that in this work, the domain-based preconditioners are applied from the left. In this case, the true residual is computed on each inner iteration to ensure the preconditioner choice does not influence the residual norm that is used to monitor convergence.

COMPUTATIONAL RESULTS

An important issue with regard to algorithm performance stems from the pressure dependence in the momentum equations. For low Mach number flows this dependence generates large off-diagonal terms in the Jacobian matrix that are proportional to one over the Mach number squared. Since most other terms are of order unity, Equation (6) represents a poorly conditioned linear system that is difficult to solve. Consequently, the effective use of Krylov iterative algorithms requires very robust yet efficient preconditioning. This section investigates several different Krylov solvers, compares ILU and domain-based preconditioning strategies, and then studies parallel implementations of the domain-based preconditioners.

Comparison of Different Krylov Algorithms

The advantages and disadvantages of the Lanczos-based and Arnoldi-based algorithms are compared and contrasted in this subsection. Figure 2 shows the outer Newton iteration convergence history for three different Newton-Krylov algorithms in solving a 16x80 grid problem from a flat initial guess with an inlet Mach number of 0.25 and a Reynolds number of 100. Note that ILU(2) preconditioning was used with each of the Krylov algorithms. Although convergence was obtained using each algorithm, the Newton-TFQMR algorithm enabled better convergence for this specific problem. Although not shown in Figure 2, convergence behavior similar to the Newton-TFQMR algorithm was also obtained using the CGS and Bi-CGSTAB algorithms. The difference in convergence behavior with GMRES was traced primarily to the second Newton step, which in all cases yielded an especially difficult linear system. The GMRES algorithms experienced stall because of their restricted Krylov subspace dimensions, and so did not converge within the allowed 200 inner iterations. In fact, the GMRES(20) algorithm also did not converge on the first Newton step. TFQMR, although requiring a large number of iterations, was able to converge within this inner iteration limit. Consequently, the Newton-TFQMR algorithm was able to move past this difficult linear system by the next Newton step. In contrast, the GMRES algorithms returned relatively poor Newton updates, which subsequently required damping. Consequently, more iterations were needed to move past this difficult part of the calculation. Damping was initiated for the GMRES(20) algorithm on the first Newton step so that the severity of the poor update returned on the second step was mitigated in comparison with the GMRES(40) algorithm. Consequently, more stringent damping was needed in the case of the Newton-GMRES(40) algorithm, thus explaining why it required the most Newton iterations for this specific case.

The behavior discussed above demonstrates how the Krylov iteration can significantly affect the overall performance of the Newton-Krylov algorithm. Thus, it is instructive to examine the behavior of these different Krylov solvers. Figure 3 presents the convergence history of the different Krylov algorithms in solving the linear system on the first Newton step of this calculation. The first Newton step was selected because then each of the Krylov algorithms are solving the same linear system. Observations indicate that GMRES(m) converges rapidly if the required inner iterations are less than the specified dimension of the Krylov subspace, m. If frequent restarts are necessary, however, the convergence curve may flatten considerably to the point of stall. Specifically, notice that the convergence of GMRES(40) is very strong on this first Newton step, whereas the convergence of GMRES(20) is very poor beyond 20 iterations because of algorithm restarts. Recall from the discussion above, however, that GMRES(40) also encountered stall on the next Newton step. The CGS algorithm works well overall, but does exhibit very erratic convergence behavior as shown in Figure 3. The Bi-CGSTAB exhibits more smoothly converging solutions, but can still exhibit oscillatory behavior. The TFQMR convergence curve is smooth, although somewhat flat during the intermediate iterations.

The TFQMR algorithm was selected for all subsequent calculations because of its smaller storage requirements compared with the GMRES algorithms and its somewhat smoother convergence behavior compared with the other Lanczos based algorithms. Note that the storage requirements for the GMRES(40) algorithm is roughly four times that of the Lanczos-based algorithms and twice that of GMRES(20).

Preconditioner Effectiveness

The efficiency of the overall solution is strongly tied to preconditioner performance. Consequently, the effectiveness of ILU(k) and additive/multiplicative Schwarz preconditioners are investigated in this section for different values of the flow Mach and Reynolds numbers. Preconditioner effectiveness is measured not only by lower inner iteration counts, but also by CPU efficiency and memory cost. The latter two measures also being dependent upon either parallel or serial implementation. This subsection focuses on serial implementations, while the next subsection considers parallel implementation issues.

Table 1 presents the memory requirements for both ILU(k) and domain-based preconditioning on a uniform 16x80 grid. The ILU data assumes a reverse row type ordering, in which cells are numbered sequentially across the channel starting from ($x=2$, $y=10$) proceeding towards the inlet with the x-index, i, running fastest. This ordering scheme performed better than other schemes that started numbering from the inlet and/or that numbered sequentially along the channel. The ILU preconditioner memory requirements are presented as a function of the level of fill-in, k, and represent the storage

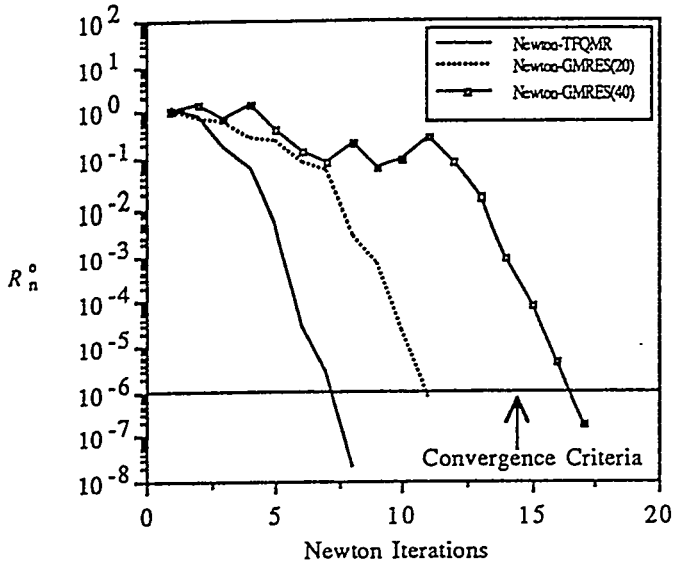

Figure 2. Convergence history of three different Newton-Krylov algorithms.

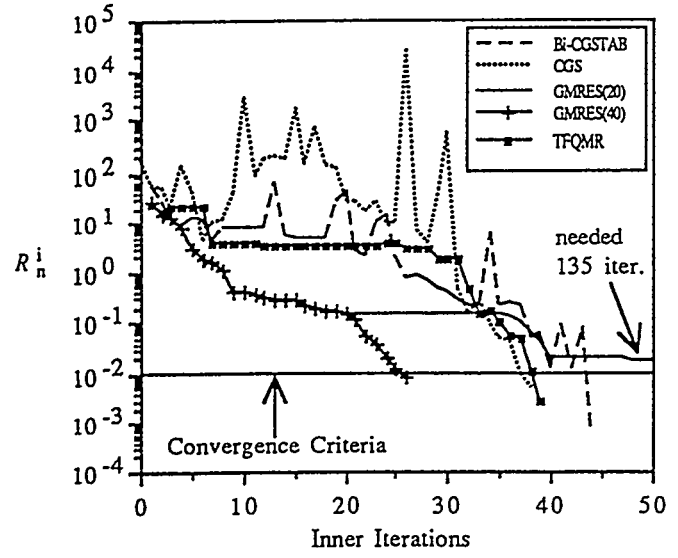

Figure 3. Comparison of convergence behavior of different Krylov solvers.

required for the non-zero diagonals listed in the second column. For the selected cell ordering, ILU(3) is equivalent to a full factorization of the Jacobian without pivoting, and so it can no longer be viewed as an *incomplete* factorization.

The domain-based preconditioner storage requirements in Table 1 are listed as a function of the selected sub-domain blocking strategy and the amount of overlap shared by adjacent sub-domains. The data was computed assuming a full LU factorization for each sub-domain using LINPACK banded Gaussian elimination (Dongarra et al., 1979). The same preconditioner memory is required regardless whether additive or multiplicative Schwarz type preconditioning is selected. Reference names used to identify the different domain-based preconditioner selections in subsequent discussions are listed in column 7 of Table 1. The direct solve memory requirements using the LINPACK routines is given by the limiting 1x1 blocking case (8.3MB). Keep in mind that the per processor storage requirements of the domain-based preconditioners could be reduced considerably in a distributed computing environment.

Table 2 presents performance data for the preconditioners listed in Table 1 in solving 6 different flow conditions identified by 3 different inlet Mach numbers and 2 different flow Reynolds numbers. The required number of Newton iterations (n), the average number of TFQMR iterations per Newton step ($\overline{m}$), and the total CPU time (sec.) are presented for each preconditioner selection. This data was obtained on a single HP Model 735 workstation using a uniform 16x80 grid. The 'NS' abbreviation in Table 2 indicates that no solution was obtained within the allowed 25 Newton iterations, while a superscript over the Newton iterations counter indicates the number of times the inner iterations encountered the upper limit of 200. Note that the preconditioner selections are listed in ascending order with respect to memory cost.

The ILU(k) preconditioners were generally less reliable for lower values of the flow Mach number as seen in Table 2. In fact, solutions using ILU preconditioning were obtained only for an inlet Mach number of 0.25, and only ILU(2) enabled a converged solution for Re=100 at that Mach number. Since the effectiveness of the ILU preconditioners is dependent upon the problem size, the poor performance of these preconditioners will likely worsen as the grid is refined. In contrast to the ILU data, the domain based preconditioners generally performed well at the lower Mach numbers as indicated in Table 2.

Multiplicative Schwarz preconditioning outperformed additive Schwarz preconditioning in most cases, except at the lowest inlet Mach number considered when using the 4x20 and 2x10 blockings without overlap. In both cases, TFQMR encountered difficulty in solving the linear system from the first Newton step. However, the multiplicative Schwarz preconditioned TFQMR algorithm returned a Newton update that resulted in a singular sub-domain matrix on the next Newton step. The use of a 2-cell overlap, however, appeared to remedy this difficulty.

Generally, the fewest number of sub-domains produced the best results on this coarse 16x80 grid. In fact the 1x1 blocking (single sub-domain) produced the best results for all 6 flow conditions. However, on finer grids this selection may be impractical for several reasons: first, the high memory storage cost; second, the full LU factorization becomes more expensive as the grid is refined (see McHugh and Knoll, 1994); and third, a single domain is not amenable to parallel implementation. The last reason highlights another important advantage of the domain-based preconditioners, namely parallel implementation. The additive Schwarz preconditioners can be parallelized readily, while the multiplicative Schwarz algorithms can be parallelized using multi-coloring schemes. In this manner, both preconditioners allow the CPU and memory costs to be distributed over several processors.

Table 1. Preconditioner memory requirements for a uniform 16x80 grid.

ILU(k) Preconditioning (reverse row ordering)			Domain Based Preconditioners [Additive (AS) Schwarz and Multiplicative (MS) Schwarz]				
k	# non-zero diagonals	Memory (MB)	# blocks in x-direction	# blocks in y-direction	# overlap cells	Reference name	Memory (MB)
0	35	1.4	4	20	0	4x20-0-AS & 4x20-0-MS	2.4
1	59	2.4	2	10	0	2x10-0-AS & 2x10-0-MS	4.3
2	94	3.9	4	20	2	4x20-2-AS & 4x20-2-MS	5.3
			2	10	2	2x10-2-AS & 2x10-2-MS	6.8
			1	5	0	1x5-0-AS & 1x5-0-MS	8.3
			1	1	0	1x1	8.3
			1	5	2	1x5-2-AS & 1x5-2-MS	9.3

Table 2. Algorithm performance data (n≡Newton iter., $\overline{m}$≡ave. inner iter. per Newton iter., NS≡No Solution).

		Mach No. = 0.25			Mach No. = 0.025			Mach No. = 0.0025		
Re	Precond. Selection	n	$\overline{m}$	CPU (sec)	n	$\overline{m}$	CPU (sec)	n	$\overline{m}$	CPU (sec)
100	ILU(0)	NS	NS	NS	NS	NS	NS	NS	NS	NS
	4x20-0-AS	8	93	178	7^1	140	222	7^5	184	325
	4x20-0-MS	7	50	124	7	73	168	NS	NS	NS
	ILU(1)	NS	NS	NS	NS	NS	NS	NS	NS	NS
	ILU(2)	8	39	431	NS	NS	NS	NS	NS	NS
	2x10-0-AS	8	41	120	7	62	145	7	110	235
	2x10-0-MS	7	21	81	7	30	103	NS	NS	NS
	4x20-2-AS	8	82	251	7	109	305	8	141	435
	4x20-2-MS	7	28	132	7	44	188	7	72	286
	2x10-2-AS	7	40	134	7	54	169	7	71	210
	2x10-2-MS	7	18	93	7	26	120	7	47	188
	1x5-0-AS	8	19	106	7	26	114	7	39	151
	1x5-0-MS	7	9	70	7	11	76	7	19	103
	1x1	7	0	43	7	0	43	7	0	43
	1x5-2-AS	7	18	96	7	21	107	7	33	142
	1x5-2-MS	7	7	70	7	8	72	7	12	147
10	ILU(0)	9	109	319	NS	NS	NS	NS	NS	NS
	4x20-0-AS	7	85	145	6	140	191	6^2	166	222
	4x20-0-MS	7	48	121	6	64	131	NS	NS	NS
	ILU(1)	7	6	71	NS	NS	NS	NS	NS	NS
	ILU(2)	7	2	105	NS	NS	NS	NS	NS	NS
	2x10-0-AS	7	38	101	6	58	120	6	86	164
	2x10-0-MS	7	22	85	7	22	85	NS	NS	NS
	4x20-2-AS	7	58	179	6	73	184	5	97	197
	4x20-2-MS	7	22	115	7	25	125	5	40	126
	2x10-2-AS	6	30	96	6	39	114	5	49	115
	2x10-2-MS	6	14	69	6	15	74	5	28	92
	1x5-0-AS	7	15	86	6	21	89	5	23	79
	1x5-0-MS	6	8	63	5	9	56	6	12	73
	1x1	6	0	38	4	0	26	5	0	32
	1x5-2-AS	7	13	82	6	11	67	4	16	55
	1x5-2-MS	6	5	56	6	5	54	5	7	51

Table 3. Shared memory parallel performance data (n ≡ Newton iter., $\overline{m}$ ≡ ave. inner iter. per Newton iter.).

No. of Sub-Domains and processors	No. of Blocks, x-dir.	No. of Blocks, y-dir.	n	$\overline{m}$	Serial CPU (sec)	Parallel CPU (sec)	Speed-up $\left(\frac{Serial\ CPU}{Parallel\ CPU}\right)$	Efficiency (%) $\left(\frac{Parallel\ Speedup}{\#\ Processors}\right)$	Precond. Memory (MW)
2	2	1	7	16	426	242	1.8	90	32.3
4	4	1	6	32	326	120	2.7	68	16.5
8	8	1	7	70	424	156	2.7	34	8.7

Parallel Implementation

This subsection investigates two different parallel implementations of the solution algorithm using the additive Schwarz preconditioning selection. Solutions to the model problem with an inlet Mach number of 0.0025 and a Reynolds number of 100 are obtained on finer grids than previously considered. The first implementation uses a shared memory computer, while the second implementation uses a distributed system of workstations in the form of a cluster.

Shared Memory Parallel Implementation. The shared memory parallel implementation consisted of parallelizing the formation of the Jacobian and preconditioner on an eight processor CRAY C-90 computer with 512MW of main memory. Profile data on this computer for a 64x320 grid (81,920 unknowns) solution in a serial mode indicated that the Jacobian evaluation represented the largest portion of the total CPU time. The next significant portion of the calculation was due to the formation and use of the preconditioner. Consequently, these two aspects of the solution algorithm were parallelized. The Jacobian evaluation was parallelized so that Jacobian terms corresponding to equations and variables at a given y-location were computed simultaneously on different processors, while the preconditioner portions of the calculation were parallelized via domain decomposition. Thus, the sub-domain LU factorizations, which make up the global additive Schwarz preconditioner, were each computed on separate processors. Consequently, the number of processors employed was chosen to equal the number of sub-domains. Table 3 presents performance data for these solutions for three different blocking strategies for both serial and parallel implementations. Note that the 2x1 blocking strategy resulted in nearly a two-fold speed-up (1.8) using two processors. However, in the case of the other blockings, the global preconditioner effectiveness was diminished, resulting in larger average inner iteration counts, and a shift in the workload to the Krylov solve, and more specifically to the forward-backward operations with the preconditioner. Unfortunately, these operations have a high parallel overhead cost and so the parallel efficiency was diminished. In addition, the parallel efficiency of the Jacobian evaluation, which is still a significant portion of the calculation, was also diminished. Presumably, this diminished efficiency is also due in part to an increase in the parallel overhead cost. Improvement of the parallel efficiency of the solution algorithm is currently being investigated.

Distributed Memory Parallel Implementation. The distributed parallel implementation used the Parallel Virtual Machine (PVM) software package developed by Geist et al. (1993) to distribute the additive Schwarz preconditioner over a cluster of four HP Model 735 processors, with each processor having 80MB of main memory. This configuration suggested the designation of one processor as the master, which in turn controlled slave processes that were assigned to each of the remaining three processors. In contrast to the CRAY C-90 serial profile data, similar data on a single workstation indicated that the preconditioner formation was the dominant portion of the calculation. Consequently, the distributed parallel implementation concentrated only on the parallel implementation of the preconditioner portions of the solution algorithm. A 39x195 grid (30,420 unknowns) was selected to demonstrate this distributed memory implementation. Since three slave processors were available, this grid was subsequently divided into three sub-domains (1x3 blocking), assigning each slave processor a sub-domain. Each slave processor was required to compute the LU factorization of its sub-domain contribution to the additive Schwarz preconditioner, keeping this LU factorization in local memory. When the additive Schwarz preconditioner was required to act on a global Krylov vector, portions of that vector corresponding to the sub-domains were sent to slave processors by the master. The slave processors in turn computed their local contributions and then sent the results back to the master for assembly. These preconditioner operations were the only ones distributed over the slave processors. However, other portions of the calculation such as matrix-vector products and Jacobian evaluations are amenable to parallelization. Future work will concentrate on distributing these remaining parts of the algorithm.

Note that the single processor memory requirements for storing the preconditioner was approximately 116MB, which prohibited running the calculation on a single processor of the cluster; while the per processor memory requirements for the distributed implementation was one third of this value. In this case, the distributed implementation enabled the three-domain, 39x195 grid computation by amortizing the preconditioner memory cost over three processors. This feature is an important advantage of distributed parallel implementations.

The distributed computation was benchmarked against a singe CPU computation on an identical processor, but one that had over 200MB of main memory. Both computations required 7 total Newton iterations and averaged 22 TFQMR iterations per Newton step. The single CPU computation required 1371.5 seconds, while the distributed computation required 922.8 seconds, representing a 33% speed-up.

SUMMARY AND CONCLUSIONS

Fully coupled Newton-Krylov solution techniques were investigated for solving low Mach number compressible past a backward facing step. Various Krylov solvers and preconditioning strategies were studied. Additionally, parallel implementations using both shared memory and distributed memory environments were considered.

Observations indicate that the Arnoldi-based algorithm, GMRES(m), converges rapidly if the required inner iterations are less than the specified dimension of the Krylov subspace, m. If frequent restarts are necessary, however, the convergence curve may flatten considerably to the point of stall. The CGS algorithm works well in most cases, but sometimes does exhibit very erratic convergence behavior. Bi-CGSTAB and TFQMR exhibit more smoothly converging solutions, but can still exhibit either erratic or stalled convergence.

The ILU(k) preconditioners were generally less reliable for lower values of the flow Mach number, and exhibited sensitivity to both cell ordering and level of fill-in. The domain-based preconditioners outperformed the ILU(k) preconditioners at the lower Mach numbers. Additionally, the domain-based preconditioners can be parallelized readily, allowing the CPU and/or memory costs to be distributed over several processors.

Parallel results were obtained on a shared memory, eight headed C-90 CRAY computer using the additive Schwarz preconditioner without overlap on a 64x320 uniform grid

(81,920 unknowns) with an inlet Mach number of 0.0025 and flow Reynolds number of 100. This work concentrated on parallelizing the Jacobian and preconditioner portions of the calculation. Results indicated that for only two relatively large sub-domains, nearly a two-fold speed-up was obtained on two processors (90% parallel efficiency). However, as the number of sub-domains and processors was increased the parallel efficiency diminished. This diminished efficiency, in part, can be attributed to a shift in the calculation workload so that the inner Krylov solve, which was not fully parallelized, became a more significant portion of the calculation. Improving the parallel efficiency as the number of sub-domains is increased is the subject of ongoing work.

Results for the same flow conditions were also obtained using a parallel implementation on a cluster of four HP Model 735 workstations using the PVM software (Geist, 1983). This solution was obtained using a 39x195 grid (30;420 unknowns). This implementation enabled the per processor memory requirements to be reduced by a factor of three and yielded a 33% speedup in run time compared to a single processor calculation.

ACKNOWLEDGEMENTS

This work was performed under the auspices of the U.S. Dept. of Energy (DOE), Idaho Operations Office, under DOE Contract No. DE-AC07-94ID13223, and supported by the Idaho National Engineering Laboratory Long Term Research Initiative in Computational Mechanics. The authors are grateful to C.H. Chang, P.G. Jacobs, D.E. Keyes, and M.J. Oliver for helpful discussions. The CRAY C-90 computer time supplied by CRAY Research, Inc. is appreciated.

REFERENCES

Ajmani, K. et. al, 1993, "Preconditioned Conjugate-Gradient Methods for Low-Speed Flow Calculations," NASA Tech. Memorandum 105929, AIAA-93-0881, ICOMP-92-22.

Ajmani, K. et. al, 1994, "Preconditioned Implicit Solvers for the Navier-Stokes Equations on Distributed-Memory Machines," NASA Tech. Memorandum 106449, AIAA-94-0408, ICOMP-93-49.

Arnoldi, W.E., 1951, "The Principal of Minimized Iterations in the Solution of Matrix Eigenvalue Problems," *Quart. Appl. Math. 9*, pp. 17-29.

Averick, B M. and Ortega, J.M., 1991, "Solutions of nonlinear Poisson-type equations," *Applied Numerical Mathematics*, Vol. 8, pp. 443-455.

Cai, X.-C., and Saad, Y., 1993, *Overlapping Domain Decomposition Algorithms for General Sparse Matrices*, Army High Performance Computing Research Center, University of Minnesota, Preprint 93-027.

Cai, X.-C., et al., 1993, "Parallel Implicit Methods for Aerodynamics," in *Proc. of the 7th Int. Conf. on Domain Decomposition Methods in Scientific and Engineering Computing*, The Pennsylvania State University, Oct. 27-30.

Dembo, R. et al., 1982, "Inexact Newton methods," *SIAM J. Numer. Anal.*, Vol. 19, pp. 400-408.

Dongarra, J., et al., 1979, *LINPACK User's Guide*, SIAM, Philadelphia.

Dryja, M., and Widlund, O.B., 1994 "Domain Decomposition Algorithms with Small Overlap," *SIAM J. Sci. Comput. 15*, pp. 604-620.

Dutto, L. C., 1993, "The Effect of Ordering on Preconditioned GMRES Algorithm, for Solving the Compressible Navier-Stokes Equations," *Int. J. Numer. Meth. in Eng. 36*, pp. 457-497.

Dutto, L.C., et al., 1994a, "Parallelizable Block Diagonal Preconditioners for the Compressible Navier-Stokes Equations," *Comput. Methods Appl. Mech. and Engrg. 117*, pp.15-47.

Dutto, L.C., et al., 1994b, "A Method for Finite Element Parallel Viscous Compressible Flow Calculations," *Int. J. Numer. Meth. Fluids 19*, pp.275-294.

Fletcher, R., 1976, "Conjugate Gradient Methods for Indefinite Systems," in *Proc. Dundee Conference on Numerical Analysis*, 1975, Lecture Notes in Mathematics 506, G. A. Watson, ed., Springer-Verlag, Berlin, pp. 73-89.

Freund, R. W., 1993, "A Transpose-Free Quasi-Minimal Residual Algorithm for non-Hermitian Linear Systems," *SIAM J. Sci. Comput. 14*, pp. 470-482.

Habashi, W.G. et al., 1994, "Large-Scale Computational Fluid Dynamics by the Finite Element Method," *Int. J. Numer. Meth. Fluids 18*, pp. 1083-1105 .

Knoll, D. A. and McHugh, P. R., 1993, "A Fully Implicit Direct Newton Solver for the Navier-Stokes Equations," *Int. J. Num. Meth. Fluids 17*, pp. 449-461.

Lanczos, C. 1952, "Solution of Systems of Linear Equations by Minimized Iterations," *J. Res. Natl. Natl. Bur. Stand. 49*, pp. 33-53.

McHugh, P. R. and Knoll, D. A., 1994, "Fully Coupled Finite Volume Solutions of the Incompressible Navier Stokes and Energy Equations Using Inexact Newton's Method," *Int. J. Numer. Meth. Fluids 19*, pp. 439-455.

Meijerink, J. A. and van der Vorst, H.A., 1977, "An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric M-Matrix," *Mathematics of Computation 31*, N137, pp. 148-162.

Orkwis, P. and George, J.H., 1993, "A Comparison of CGS Preconditioning Methods for Newton's Method Solvers," *Proc. 11th AIAA CFD Conference*, AIAA-93-3327, Orlando, FL., July 6-9.

Saad, Y. and Schultz, M. H., 1986, "GMRES: A Generalized Minimal Residual Algorithm for Solving Non symmetric Linear Systems," *SIAM J Sci. Stat. Comput. 7*, pp. 856-869.

Saad, Y., 1991, *SPARSKIT, A Basic Tool Kit for Sparse Matrix Computations*, RIACS Technical Report 90.20.

Sonneveld, P., 1989, "CGS, a Fast Lanczos-type Solver for Non symmetric Linear Systems," *SIAM J. Sci. Stat. Comput. 10*, pp. 36-52.

van der Vorst, H. A., 1992, "Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Non symmetric Linear Systems," *SIAM J. Sci. Stat. Comput. 13*, pp. 631-644.

Venkatakrishnan, V. and Mavriplis, D.J., 1993, "Implicit Solvers for Unstructured Meshes," *J. Comput. Phys. 105*, pp. 83-91.

Venkatakrishnan, V., 1991, "Preconditioned Conjugate Gradient Methods for the Compressible Navier-Stokes Equations," *AIAA J. 29*, pp. 1092-1100.

Watts, J. W., 1974, "A Conjugate Gradient-Truncated Direct Method for the Iterative Solution of the Reservoir Simulation Pressure Equation," *Soc. Pet. Eng. J. 21*, pp. 345-353.

White, F.M., 1974, *Viscous Fluid Flow*, McGraw-Hill, Inc., New York.