INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 18, 1083–1105 (1994)

LARGE-SCALE COMPUTATIONAL FLUID DYNAMICS BY THE FINITE ELEMENT METHOD

W. G. HABASHI

Concordia University, Montreal, Quebec, Canada and Pratt & Whitney Canada, Montreal, Quebec, Canada

M. ROBICHAUD, V.-N. NGUYEN AND W. S. GHALY

Pratt & Whitney Canada, Montreal, Quebec, Canada

M. FORTIN

Université Laval, Québec, Quebec, Canada

AND

J. W. H. LIU

York University, Toronto, Ontario, Canada

SUMMARY

Solution methods are presented for the large systems of linear equations resulting from the implicit, coupled solution of the Navier–Stokes equations in three dimensions. Two classes of methods for such solution have been studied: direct and iterative methods.

For direct methods, sparse matrix algorithms have been investigated and a Gauss elimination, optimized for vector–parallel processing, has been developed. Sparse matrix results indicate that reordering algorithms deteriorate for rectangular, i.e. $M \times M \times N$, grids in three dimensions as N gets larger than M. A new local nested dissection reordering scheme that does not suffer from these difficulties, at least in two dimensions, is presented. The vector–parallel Gauss elimination is very efficient for processing on today's supercomputers, achieving execution rates exceeding 2·3 Gflops the Cray YMP-8 and 9·2 Gflops on the NEC on SX3.

For iterative methods, two approaches are developed. First, conjugate-gradient-like methods are studied and good results are achieved with a preconditioned conjugate gradient squared algorithm. Convergence of such a method being sensitive to the preconditioning, a hybrid viscosity method is adopted whereby the preconditioner has an artificial viscosity that is gradually lowered, but frozen at a level higher than the dissipation introduced in the physical equations. The second approach is a domain decomposition one in which overlapping domain and side-by-side methods are tested. For the latter, a Lagrange multiplier technique achieves reasonable rates of convergence.

1. INTRODUCTION

The simulation of complex aerodynamic fields by means of inviscid and viscous flow equations is rapidly becoming the preferred analysis and design tool in the aerospace industry. There is no shortage of methods for discretizing the Euler and Navier–Stokes equations, with these methods differing in their discretization of the time or pseudo-time term, space terms, linearization and algebraic equation solution method.

The predominant space discretization methods in industrial practice are the finite difference method (FDM) and finite volume method (FVM), with the finite element method (FEM) a distant

CCC 0271–2091/94/111083–23

Received April 1993
Revised December 1993

third in North America. The FEM is often perceived as taxing on computer memory. The situation is more evenly balanced in Europe and elsewhere, where the FEM is recognized as providing flexibility in approximating complex geometries and in the ease of application of boundary conditions. In practice, however, to distinguish between FVM and FEM takes a well-trained eye, since the boundaries between the two are getting fuzzier. During space discretization, methods also differ in applying the dissipation necessary to stabilize the numerical solution. Two approaches are possible: centred schemes, with dissipation introduced through an explicit artificial viscosity, or upwind schemes applied to the convective terms.

For time discretization, explicit and implicit approximations can be used. Explicit schemes trade speed of convergence for simplicity by not requiring matrix solution. They are easily vectorizable and parallelizable. To speed up convergence, various acceleration techniques are used such as local time stepping, residual averaging and multigrid methods. Large-scale problems are therefore more easily amenable to solution on today's computers, with a compromise between large solution times and manageable memory resources. Implicit schemes, on the other hand, allow much larger time steps at the cost of solving some matrices at each step. These range from fully-coupled schemes, to ADI schemes, all the way to schemes that only require the solution of scalar tridiagonal matrices.

It must be appreciated, however, that the hierarchy of simplifications in the solution of the coupled system of equations must be at the cost of additional iterations to obtain the same overall convergence of the non-linear system. While it can be argued that coupled methods of solution are currently impractical because of their memory requirements, it can be pointed out that their convergence is not only much faster but also simpler, since the usual bells and whistles of uncoupled methods are not needed. The slow convergence of explicit methods may also sometimes lead to the temptation of accepting partially converged results under the argument of sufficient engineering accuracy. This is not without danger, as was vividly demonstrated by Pulliam[1]. After experiencing great difficulties with most well-known schemes for the Euler equations, he challenged code developers to predict zero lift for subsonic flow over circular cylinders and ellipses when the mesh is skewed to the freestream direction. In this problem the convergence behaviour of the lift coefficient C_L is such that it starts positive, crosses zero, becomes negative and settles asymptotically to its final artificial-viscosity-dependent value, which may not necessarily be zero. For the ellipse, none of the codes tested by Pulliam yielded a C_L less than 1·545, while for the cylinder the minimum was 13·66. Invariably the response was to propose results from codes stopped after a larger number of iterations at about the time that zero lift was being obtained, using the argument that acceptable engineering accuracy is reached by then. When iteration is resumed in such codes, however, none was able to asymptotically predict zero lift. A year later a second-order Godunov scheme[2] only achieved a reduction of one order of magnitude in lift.

We have presented a simple but effective method for the solution of the fully-coupled system of inviscid (Euler)[3] and viscous (Navier–Stokes)[4] equations. The scheme is a weak-Galerkin formulation with simple Laplacian dissipation terms explicitly added to each of the Euler equations, but only to be the continuity equation for Navier–Stokes formulations. The linearization of the system is carried out via a Newton method, followed by a direct solution of the coupled system of linear equations, at each iteration. This method has proven efficient and accurate for both inviscid and viscous cases, yielding C_L-values of $O(10^{-3})$ for the test cases proposed by Pulliam.

The coupled system of equations resulting from our two-dimensional formulation will include the pressure (or density) and two velocity components and is amenable to a direct solver on a large class of computers and workstations. For example, a flow having three variables (u, v, p)

per cell vertex, discretized on an $N \times M$ structured grid, would require the solution of $3NM$ equations with a bandwidth of $3N$. The number of operations for such a direct solution is estimated as $27MN^3$. For the equivalent three-dimensional problem on an $L \times N \times M$ grid the number of equations increased to $4LMN$ and the bandwidth to $4NL$. The number of operations increases to $64MN^3L^3$, i.e. $2.37L^3$ times the number for the two-dimensional solution.

Preaching the use of coupled methods of solution is therefore not that clear-cut and the present paper is only a step in a sustained effort to address the practical solution of such large nonsymmetric systems of linear equations for three-dimensional flow situations. This includes the development of solvers and techniques for direct or iterative solutions on vector and parallel computers.

2. THE PROBLEM

The steady, three-dimensional compressible, variable viscosity, Navier–Stokes equations can be written as

continuity

$$\nabla \cdot (\rho \mathbf{V}) = \varepsilon \nabla^2 p, \tag{1}$$

momentum

$$\rho(\mathbf{V} \cdot \nabla)\mathbf{V} + \mathbf{V}(\nabla \cdot \rho \mathbf{V}) = -\nabla p + \frac{1}{Re}\left[-\tfrac{2}{3}\nabla(\mu \nabla \cdot \mathbf{V}) + \nabla \times \mu(\nabla \times \mathbf{V}) + 2(\nabla \cdot \mu \nabla)\mathbf{V}\right], \tag{2}$$

energy

$$H_0 = \frac{\gamma}{\gamma - 1}\frac{p}{\rho} + \tfrac{1}{2}\,\mathbf{V} \cdot \mathbf{V}, \tag{3}$$

equation of state

$$p/\rho = RT. \tag{4}$$

It should be remarked that a pressure dissipation term $\varepsilon \nabla^2 p$ has been added to the continuity equation (2) for regularization. Such a Laplacian term with a small coefficient ε prevents odd–even decoupling or checkerboarding from occurring, without the need for unequal-order interpolation of velocity and pressure in the FEM or staggered grids in the FDM.[3,4] This regularization term constitutes, however, a small error in mass that is reflected as a first-order accuracy of the scheme. Second-order accuracy, when needed, can be achieved through the use of a fourth-order operator and has been detailed in Reference 5.

In addition, a simplified energy equation, namely the constancy of total enthalpy, has been used. This is a good approximation for the adiabatic viscous compressible flow of a perfect gas. While the full energy equation can be, and has been, used in our work, our main purpose here is to discuss solution schemes for the large systems of couple equations for three-dimensional flows, presently restricted because of memory limitation to the continuity and momentum system.

After FEM discretization and Newton linearization, the following representative delta form

of the equations can be obtained for three-dimensional flows in terms of the cell vertex unknowns of pressure and velocity components at the eight vertices of a trilinear element:

$$\sum_{e=1}^{E}\left(\sum_{j=1}^{8}([k^p_{i,j}]_p\Delta\hat{p}_j + [k^u_{i,j}]_p\Delta\hat{u}_j + [k^v_{i,j}]_p\Delta\hat{v}_j + [k^w_{i,j}]_p\Delta\hat{w}_j)\right) = -(R_i)_p,$$

$$\sum_{e=1}^{E}\left(\sum_{j=1}^{8}([k^p_{i,j}]_u\Delta\hat{p}_j + [k^u_{i,j}]_u\Delta\hat{u}_j + [k^v_{i,j}]_u\Delta\hat{v}_j + [k^w_{i,j}]_u\Delta\hat{w}_j)\right) = -(R_i)_u,$$

$$\sum_{e=1}^{E}\left(\sum_{j=1}^{8}([k^p_{i,j}]_v\Delta\hat{p}_j + [k^u_{i,j}]_v\Delta\hat{u}_j + [k^v_{i,j}]_v\Delta\hat{v}_j + [k^w_{i,j}]_v\Delta\hat{w}_j)\right) = -(R_i)_v,$$

$$\sum_{e=1}^{E}\left(\sum_{j=1}^{8}([k^p_{i,j}]_w\Delta\hat{p}_j + [k^u_{i,j}]_w\Delta\hat{u}_j + [k^v_{i,j}]_w\Delta\hat{v}_j + [k^w_{i,j}]_w\Delta\hat{w}_j)\right) = -(R_i)_w,$$

which can be written in matrix form as

$$\begin{bmatrix}[K^V]_V & [K^p]_V\\ [K^V]_p & [K^p]_p\end{bmatrix}\begin{Bmatrix}\Delta\mathbf{V}\\ \Delta p\end{Bmatrix} = -\begin{Bmatrix}R_V\\ R_p\end{Bmatrix}. \tag{5}$$

This form could be taken as representative of the fully-coupled solution of an incompressible flow or the fully-coupled solution of a compressible flow with the density lagged. Although the FEM has been selected as the discretization scheme, the remainder of the paper applies equally to the coupled equations resulting from the FDM or FVM, the only difference being in the way the equations are assembled.

For a fully-coupled solution, quadratic convergence has been demonstrated for inviscid transonic two-dimensional flows,[3] with convergence to machine accuracy in about six iterations. For three-dimensional flows, storage limitations dictate that the density be lagged and only linear convergence can be achieved, with machine accuracy reached in about 30 iterations for compressible viscous flows.[4]

The remainder of the paper will concentrate on presenting the schemes developed or tested for the solution of matrices resulting from this coupled approach. These will include the *direct methods* (a) sparse matrix technology and (b) parallel–vector Gauss elimination and the *iterative methods* (a) preconditioned conjugate-gradient-like methods and (b) domain decomposition algorithms.

3. DIRECT SOLVERS

3.1. Sparse matrix technology

Sparse matrix technology has been used for a wide variety of numerical problems requiring the solution of large sparse sets of equations.[6,7] More recently it has attracted interest for the solution of CFD problems by direct methods. Some degree of success has been reported, for example in Reference 8, where large two-dimensional problems have been tackled by such methods. The method starts with a reordering algorithm that scans the matrix topology and restructures it to minimize the storage. The reordered matrix is then decomposed by various methods, often taking advantage of the architecture of modern computers.

Reordering schemes in SPARSPAK. To identify the limits of applicability of this approach to large three-dimensional CFD problems, SPARSPAK, a commercially available sparse matrix package from the University of Waterloo and co-authored by one of the present authors, was used. Three reordering schemes, namely reverse Cuthill–McKee (RCM), multiple minimum degree (MMD) and nested dissection (ND), included in SPARSPAK, were tested and compared with natural ordering.

Test cases were run on a cubic grid of size $N \times N \times N$, with N varying from 5 to 30, and the results obtained are shown in Figures 1 and 2.

The MMD and ND reordering schemes are very efficient in reducing the storage requirements and factorization operations of the natural ordering for large values of N. The RCM algorithm invariably leads to reorderings and operations count, at least for cubic grids, that are worse than natural ordering.

The performance of the reordering schemes changes dramatically, however, when the grid density becomes asymmetric, i.e. increases in one direction, say $M \times M \times N$ with $N > M$. The efficiency of the MMD and ND algorithms suffers drastically. Figures 3 and 4 show that the RCM algorithm recovers somewhat relative to other reordering methods, but never improves

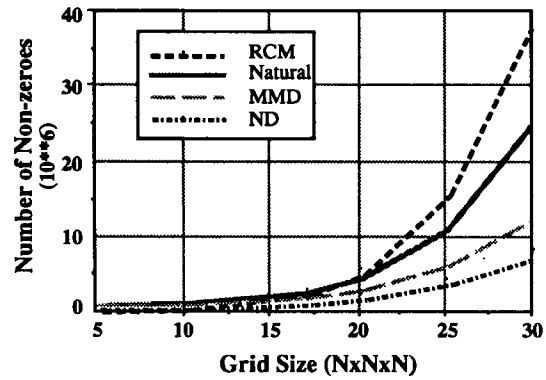

Figure 1. Storage requirements for various reordering schemes, cubic grids

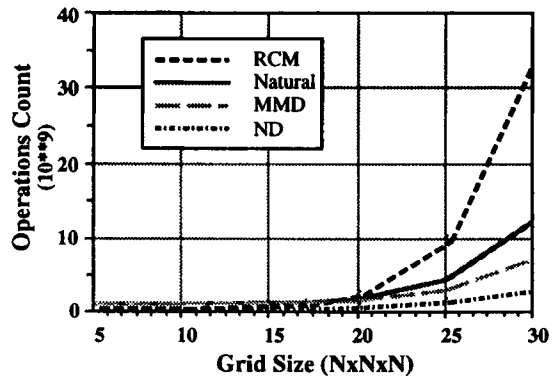

Figure 2. Factorization operations counts for various reordering schemes, cubic grids

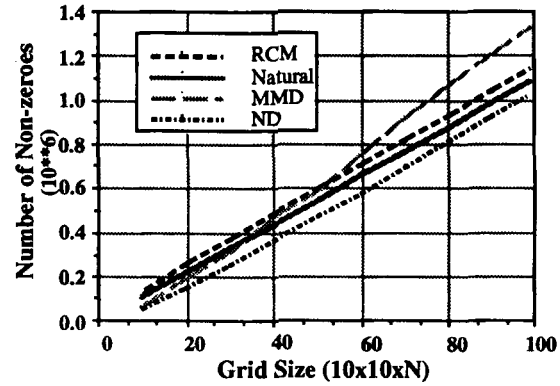

Figure 3. Storage requirements for various reordering schemes, rectangular grids

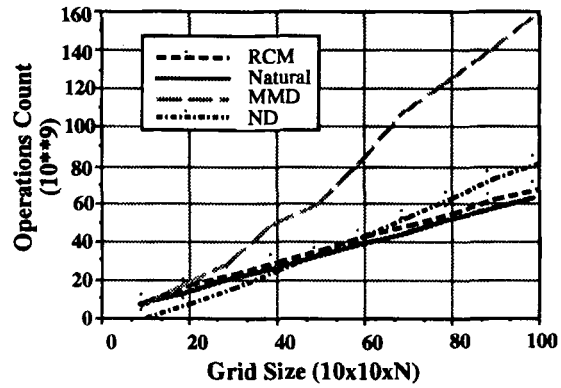

Figure 4. Factorization count for various reordering schemes, rectangular grids

over the natural ordering. The MMD algorithm, on the other hand, deteriorates rapidly with grid density asymmetry, while the ND algorithm shows a consistent improvement in storage requirements but an unfavorable factorization count compared with natural ordering.

An improved method for asymmetric grids: local nested dissection. The above observations provide the impetus for a new ordering scheme which combines the advantages of the natural and ND orderings for asymmetric grid problems.[9] Consider an $N \times M$ grid where $N > M$. The new scheme would order the two rectangular subgrids, consisting of the first $M/2$ rows and the last $M/2$ rows of the grid, using the standard ND ordering. The remaining $N - M$ rows of the grid would be numbered by considering a partitioning factor p, with $1 \leqslant p < M$, and the $(N - M) \times M$ rectangular grid divided into smaller square subgrids of size approximately M/p by a set of horizontal and vertical grid lines. There would be p partitions horizontally and $p(N - M)/M$ vertically. This gives $p^2(N - M)/M$ square subgrids of size M/p. Each square subgrid is ordered by the standard ND ordering. The remaining nodes associated with the partitioning grid lines are then numbered using a scheme similar to the natural ordering, the only difference being that a set of nodes associated with the boundary of a square subgrid is numbered consecutively.

1	7	4	19	10	16	13
3	8	6	20	12	17	15
2	9	5	21	11	18	14
105	104	103	102	101	100	99
43	49	46	95	52	58	55
45	50	48	94	54	59	57
44	51	47	93	53	60	56
98	97	96	92	91	90	89
61	67	64	85	70	76	73
63	68	66	84	72	77	75
62	69	65	83	71	78	74
88	87	86	82	81	80	79
22	28	25	40	31	37	34
24	29	27	41	33	38	36
23	30	26	42	32	39	35

With partitioning factor $p = 2$

1	7	4	19	10	16	13
3	8	6	20	12	17	15
2	9	5	21	11	18	14
105	104	103	102	101	100	99
43	97	44	94	45	91	46
98	96	95	93	92	90	89
47	87	48	84	49	81	50
88	86	85	83	82	80	79
51	77	52	74	53	71	54
78	76	75	73	72	70	69
55	67	56	64	57	61	58
68	66	65	63	62	60	59
22	28	25	40	31	37	34
24	29	27	41	33	38	36
23	30	26	42	32	39	35

With partitioning factor $p = 4$

Figure 5. Located nested dissection for a 15 × 7 grid with partitioning factors $p = 2$ and 4

To illustrate the numbering scheme, Figure 5 shows the orderings of a 7 × 15 grid with partioning factors $p = 2$ and 4.

Since ND ordering is applied locally to a number of smaller subgrids, this ordering can be referred to as a local nested dissection (LND) ordering. It should be remarked that if the grid is square, this ordering becomes the standard ND. Moreover, the LND ordering with a partition value $p = 1$ corresponds to a hybrid nested dissection strategy introduced by Rose and Whitten.[10] At the other extreme, i.e. for a large partition factor p close to M, LND is simply the natural ordering: LND can hence be viewed as a generalization of the hybrid nested dissection strategy of Reference 10.

Choice of partitioning value. In Figure 6 the number of factorization operations is shown versus different values of p for a 500 × 60 grid problem. The minimum operations count of 21×10^6 operations is attained for a partition value of about $p = 5$. For comparison, the standard ND ordering requires $29{\cdot}1 \times 10^6$ operations to factor, while natural ordering requires $58{\cdot}4 \times 10^6$ operations.

The observation that the operations count is a minimum around $p = 5$ can be established

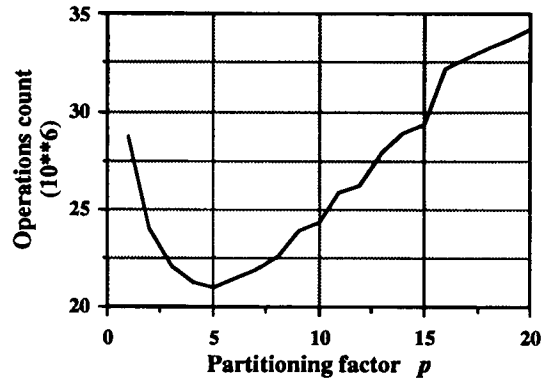

Figure 6. Factorization operations count for different partitioning factors p for a 500 × 60 grid

formally for general rectangular grids. The following bound on the arithmetic operations can be obtained by a simple count:[6]

$$(N-M)M^2\left(p+\frac{7}{2}+\frac{369}{12p}-\frac{125}{6p^2}\right)+\frac{829}{84}M^3. \tag{6}$$

The asymptotic bound is minimized at $p = 4{\cdot}67$. This can be shown by differentiating the coefficient expression for the term NM^2 with respect to p and equating to zero. One obtains

$$12p^3 - 369p + 500 = 0. \tag{7}$$

The roots of this equation are 1·46, 4·67 and −6·13 respectively and $p = 4{\cdot}67$ yields the minimum.

The optimal value of p between 4 and 5 has an intuitive explanation. For square subgrids nested dissection is obviously efficient. For an $S \times S$ subgrid the number of nodes on the boundary is roughly $4S$. When $4S$ reaches M, it pays to switch to a scheme similar to the natural ordering.

Table I gives statistics for various $N \times M$ rectangular grids with a number of unknowns $NM = 30{,}000$. These show that the LND ordering can substantially reduce arithmetic operations for elongated grids.

3.2. Gauss elimination on vector–parallel supercomputers

In the last few years pioneering work has been initiated by Storaasli *et al.*,[11] who, using the computing power of a Cray YMP with eight processors, were able to achieve impressive execution rates for the direct solution of a symmetric set of 54,870 equations for the structural analysis of the Space Shuttle Solid Rocket Booster. In the present work similar ideas are used but with the following important distinctions:

1. The CFD system matrix is non-symmetric, i.e. the method is applied to a general, variable bandwidth matrix.
2. A larger set of equations with a larger bandwidth is solved.
3. No special language other than Fortran is used.
4. The parallel–vector strategy is highly optimized.

In the following the underlying ideas of the vector–parallel Gauss elimination are discussed.

The vector–parallel Gauss elimination. The matrix is stored in a continuous vector containing the entries row-by-row, with a variable bandwidth, i.e. in a skyline mode. Two indices need to

Table I. Factorization statistics, in millions, for $N \times M$ rectangular grids

	Natural ordering		Standard ND		Local ND	
$N \times M$ grid	Non-zeros	Op. count	Non-zeros	Op. count	Non-zeros	Op. count
2000 × 15	0·478	4·523	0·533	7·221	0·430	4·161
1000 × 30	0·928	15·758	0·716	15·351	0·621	9·794
500 × 60	1·826	58·376	0·887	29·098	0·819	21·022
250 × 120	2·615	223·842	1·008	44·679	0·990	39·781
200 × 150	4·507	346·474	1·009	45·148	1·022	45·493

be defined: the first to indicate the start of each row and the second to point out the farthest row above it affecting its elimination.

The classical Gauss elimination procedure starts the elimination from the top of the matrix. The elimination row is first divided by its diagonal and has multiples of it subtracted from all the following rows to eliminate the column corresponding to its diagonal. In the approach of Storaasli *et al.*[11] the procedure is inverted, with a row selected to be operated on and all previous rows affecting it being used to eliminate the corresponding columns of that row. This is obviously more amenable to parallel computing, since at the row being eliminated synchronization is needed only with a number of preceding rows equal to the number of processors. This is much less than the continuous synchronization that would be required by the classical Gauss elimination. In addition, the fact that many rows are available to be used for elimination of the selected row allows loop unrolling, which will be described later.

The decomposition proceeds by assigning an equation to a processor. The elimination is performed in parallel on the processors, using all previously completed factorized rows. As soon as a processor has completed the factorization of a row, it operates on the next unfactorized one. The vectorization is carried our on the row operations using loop unrolling of various levels. The vector length is controlled by the bandwidth and the stride is unity, since all vector components are contiguous.

Dynamic assignment of equations to processors. It should be recognized that each processor must be initiated, taking some finite time to come on stream. Speed is therefore gained by dynamically assigning equations to be operated on to the available processors. This is illustrated in Figure 7 for a three-processor case.

Let us assume that equation 1 has been assigned to the first processor; it would be divided by its diagonal in preparation for the elimination of subsequent rows. In the figure it is shown that rows 2 and 3 are successively assigned to processor 1 as long as processors 2 and 3 are not fully active yet. As soon as processor 2 is fully operational, the next equation to be eliminated is assigned to it, in this case equation 4; processor 1 will then host equation 5. When processor 3 is activated, it is shown that it hosts equation 8, and so on. In addition, the figure illustrates

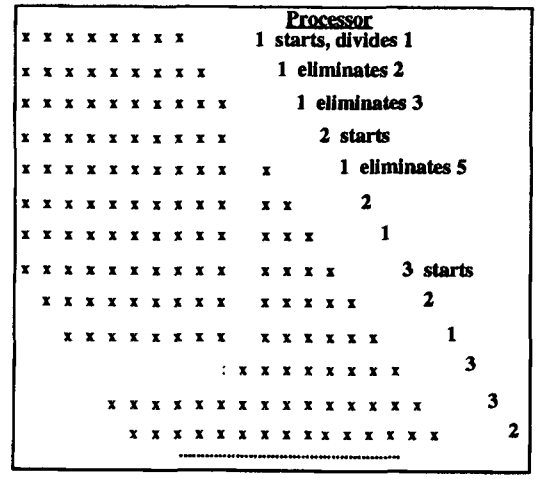

Figure 7. Dynamic assignment of equations to processors

that in the second access of processor 3 no elimination is required, since no rows affect the current one, and processor 3 immediately operates on the next available equation.

Dynamic loop unrolling. Loop unrolling is a technique to minimize the fetching and storing of data to and from memory in a compute-intensive application. It consists of explicitly writing out portions of a DO-loop to minimize the number of times data are stored back to memory. As an example of a level-3 loop unrolling, consider the following:

```
      DO 100 I = 1, M
      DO 100 J = 1, N
      A(J) = A(J)+B(I)*C(J,I)
100   CONTINUE
```

A level-3 unrolling of the above DO-loop can be written as

```
      DO 100 I = 1, M, 3
      DO 100 J = 1, N
      A(J) =  A(J)+B(I)*C(J,I)+B(I+1)*C(J,I+1)+B(I+2)*C(J,I+2)
100   CONTINUE
```

Since memory access is costly, substantial savings are obtained by the unrolled form, since data for A(J) remain in the vector register without having to be repeatedly stored back. On the current Cray computers the optimal level of unrolling is found to be 7 for this particular type of application.

Static and dynamic loop unrolling are illustrated in Figures 8(a) and 8(b) for a level-3 loop unrolling. Storaasli *et al.*[11] use level-9 loop unrolling and divide the matrix into blocks of nine rows each. It can be seen for the level-3 example of Figure 8(a) that when a particular row is being eliminated (shown by an arrow), normally two special blocks each having less than three rows will occur. This means that special loop-unrolling statements must be written for both blocks, with a lower level of unrolling and hence less efficiency.

By dynamically sizing blocks to start at the first row affecting the elimination, it is clear from Figure 8(b) that only one special short block can occur. Over the large number of operations involved in the matrix decomposition, this can translate into a sizable saving.

Dynamic elimination. In a static elimination procedure a block is only processed if all information within that block is ready, i.e. the whole block has already been operated on, as indicated for $L-1$ blocks of Figure 9. This implies a wait state where a processor is idle if the entire block operations are not yet complete, as shown in the same figure for block L.

On the other hand, in a dynamic elimination, instead of spending CPU cycles in an idle state, the processor, in this case processor 1, is allowed to start operating on the completed portion of the equation block L even if the whole block is not yet complete. While this partial block operation affects the level of loop unrolling, the penalty is less than allowing a processor to remain idle. At the termination of a partial block operation it is not unusual for the previously uncompleted rows to have been operated on, and the row being eliminated can then be completely processed.

Gauss elimination for Navier–Stokes. The Navier–Stokes solutions have been obtained on an NEC SX-3/44 with four processors and 128 Mwords of memory and are compared with the 1990 results obtained on a Cray Research YMP-8 computer with eight processors and similar

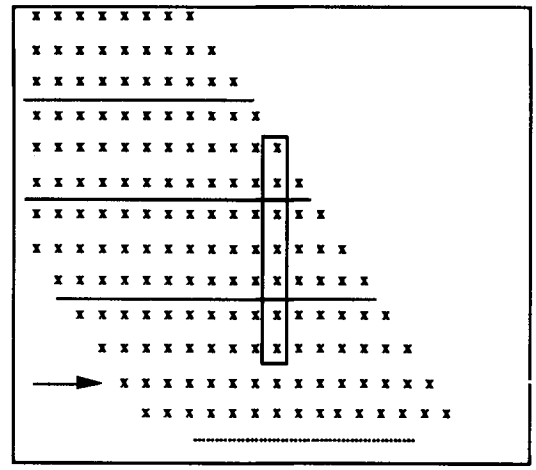

Figure 8(a). Static block unrolling

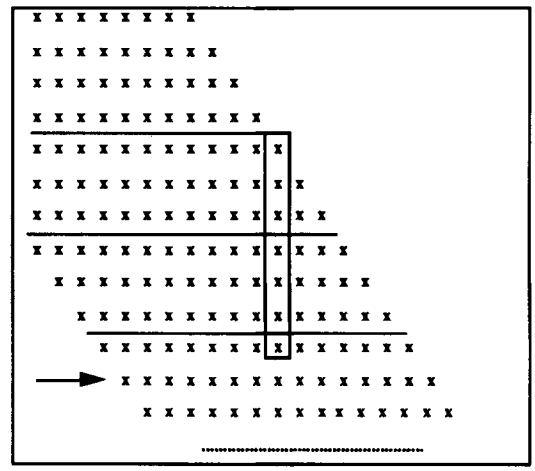

Figure 8(b). Dynamic block unrolling

memory. The equation sets solved are from the discretization of the Navier–Stokes equations for a three-dimensional flow in a gas turbine pipe diffuser,[14] shown in Figure 10. Access to the Cray permitted its use in a dedicated model, while the operation of the NEC machine did not permit us to use it as single users.

Although the physical non-linear problem of this paper is the solution of the Navier–Stokes equations, the discussion in the rest of this section pertains to the application of the Gauss elimination algorithm to the solution of the matrix at a single Newton step. Test case size information is included in Table II. The number of equations in the test cases ranges from about 6800 to over 100,000 and the bandwidth from 476 to 2877.

Table III presents the results obtained on the eight-processor YMP. Results indicate that the speed-up from one to two processors is nearly 99% efficient. With eight processors the efficiency

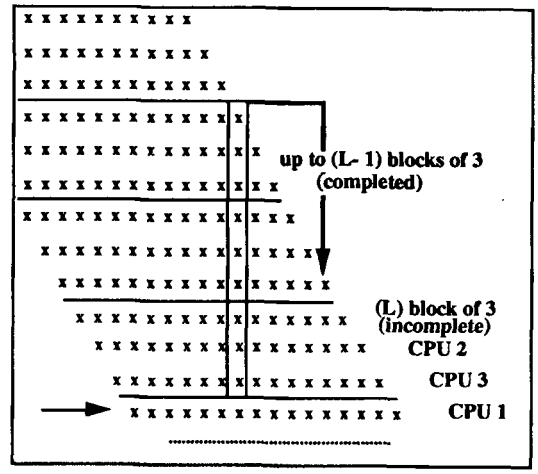

Figure 9. Static and dynamic elimination

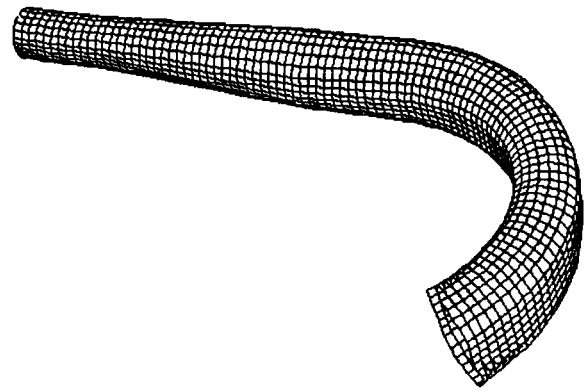

Figure 10. Gas turbine pipe diffuser: surface grid

Table II. 3D Navier–Stokes Gauss solver test case information

Test case	No. of nodes	No. of elements	No. of equations	Maximum bandwidth	Storage (Mwords)
1	2112	1785	6804	476	5·5
2	2550	2208	8496	492	7·5
3	3625	3168	12192	688	14·7
4	4240	3762	14630	964	25·1
5	14000	12936	50470	1269	116
6	28860	26078	100334	620	110
7	12050	7500	24021	2258	91·7
8	6280	5445	21420	2877	103

Table III. 3D Navier–Stokes Gauss solver on the Cray YMP-8 in dedicated mode

Test case	1 CPU (Mflops)	2 CPU (Mflops)	3 CPU (Mflops)	4 CPU (Mflops)	Speed-up 1 to 8
1	224	443	869	1577	88%
2	227	449	884	1635	90%
3	238	469	933	1796	94%
4	247	490	973	1899	96%
5				2276	
6				1980	
7				2307	

Table IV. 3D Navier–Stokes Gauss solver on the NEC SX-3/44 in non-dedicated mode

Test case	1 CPU (Mflops)	2 CPU (Mflops)	3 CPU (Mflops)	4 CPU (Mflops)	Speed-up 1 to 4
1	1632	2924	3898	4389	67%
2	1708	3057	4085	4577	67%
3	1984	3567	4774	5274	66%
4	2360	4280	5674	6188	66%
5	2624	4733	6025	6451	61%
6	1846	3323	4420	5074	69%
8	3119	5628	7335	9155	73%

is of the order of 96%. Results indicate that the larger the problem, the better the parallel efficiency. The largest test cases attempted used nearly all the memory. Test case 5 has a large number of unknowns and a moderate bandwidth, while test case 6 has half the unknowns but double the bandwidth. The ratio between the two speeds also works out exactly to be proportional to NB^2, where N is the number of equations and B is the bandwidth. The fastest execution rate of 2·307 Gflops was obtained in test case 7, selected for its large bandwidth, which improves vectorization by increasing the vector length. For the largest test case, i.e. test case 7, the solution time on the Cray was 77 s.

The results of Table IV are from the tests run on the NEC machine. Test case 7 was not run on the NEC and was replaced by test case 8. As mentioned previously, there was no provision for its usage in dedicated mode and therefore the comparisons of speed-up from one to four processors and the effect of problem size may not be accurately reflected in the table. Each processor of the NEC can produce 16 floating point operations per clock cycle and hence requires a longer vector length than the Cray to take full advantage of its capabilities. This is reflected in test case 8, chosen for its large bandwidth. For the largest test case, i.e. test case 8, the solution time on the NEC was 28 s.

4. ITERATIVE SOLVERS

4.1. Preconditioned conjugate-gradient-like algorithms

Direct methods such as Gauss elimination require $O(\mathrm{NEQ}^{2\cdot33})$ operations (where NEQ is the number of equations) for the factorization step and $O(\mathrm{NEQ}^{1\cdot67})$ operations for the substitution

step, while storage requirements are proportional to $O(\mathrm{NEQ}^{1.67})$ on an $N \times N \times N = \mathrm{NEQ}$ mesh. Iterative methods for the Newton correction, on the other hand, can offer the advantage of $O(\mathrm{NEQ})$ storage and, under certain conditions, preconditioned conjugate gradient methods can produce a machine-accurate solution in $O(\mathrm{NEQ}^{1.17})$ operations.[12] These estimates must, however, be tempered by the sensitivity of iterative methods to matrix conditioning and by the difficulty of vectorization for many preconditioning schemes. Some progress is being achieved in adapting iterative methods to the capabilities of new architecture computers.[13]

The choice of iterative methods for the systems arising from the linearization of the Navier–Stokes equations is limited by the non-symmetry and non-positive definiteness of the matrix. Classical conjugate gradient methods, highly efficient for symmetric problems, become inapplicable. One must use variants based either on minimization of the residuals, such as the generalized minimum residual (GMRES) method,[14] or on extensions of the biconjugate gradient, such as the conjugate gradient squared (CGS) method.[15]

Experience with minimum residual methods indicates that they are not robust for non-positive systems, often stagnating at some non-zero value of the residual. The only cure seems to be a better preconditioning that makes the real part of the eigenvalues positive and redistributes them better. Standard conjugate gradient methods, on the other hand, are more robust for systems having eigenvalues with negative real part.[16] CGS therefore seems a good choice, since in addition it has moderate storage requirements (six vectors) compared with GMRES, for which it is often necessary to keep up to 20 of the previous solutions.[17] Both approaches may necessitate the solution of the unsteady system, which is positive definite for small time steps.

Preconditioning for CGS. Preconditioning is an important issue and a good preconditioner not only makes the difference between fast or slow convergence but between convergence and divergence. Here a CGS method preconditioned by incomplete factorization (PCGS) has been applied to the non-symmetric tangent matrix arising from the Newton linearization. Given a matrix $[K]$, one computes $[S] = [L][U]$, an approximation of the factorization of $[K]$, and transforms the system

$$[K]\{\Delta x\} = -\{R\} \tag{8}$$

into the preconditioned one (PCGS)

$$[L]^{-1}[K][U]^{-1}\{\Delta z\} = -[L]^{-1}\{R\} \tag{9}$$

where

$$\{\Delta z\} = [U]\{\Delta x\} \tag{10}$$

This might be described as an equilibrated preconditioning, different from the left preconditioning

$$[S]^{-1}[K]\{\Delta x\} = -[S]^{-1}\{R\} \tag{11}$$

generally employed in the implementation of the GMRES algorithm. It can be verified that (9) implicitly sets on the finite-dimensional spaces new scalar products associated with $[L]^{\mathrm{T}}[L]$ and $[U]^{\mathrm{T}}[U]$, which are positive definite, while a left preconditioning brings in a scalar product defined by $[S]^{-1}$, which lacks this property when used with iterative methods such as CGS. Indeed, a preliminary study showed that such left preconditioning for CGS yielded very poor convergence properties that can probably be attributed to the non-positive definiteness of the scalar product.

It should also be mentioned that an attempt to increase the accuracy of the incomplete factorization by permitting one level of fill-in showed no real gains for the problems tested. In addition, the effect of ordering, considered to be important, was investigated. For symmetric positive definite systems it has been shown[18] that a good ordering can significantly improve the efficiency of the modified incomplete factorization in which discarded non-zero terms are lumped into the diagonal. Our preliminary experience with this method, however, was that reordering led to no tangible improvement for the problems under consideration.

PCGS for Navier–Stokes: time marching and hybrid artificial viscosity. Iterative methods are found to converge slowly, if they converge at all, for the 3D steady incompressible and compressible Navier–Stokes equations.[19] To improve the conditioning of the matrix system, a time-marching procedure has been used here. The introduction of a time-dependent term in the equations improves the conditioning of the matrix by the addition of a mass matrix $[M]/\Delta t$ on the diagonal:

$$\begin{bmatrix} [M] + \Delta t[K^{\mathrm{V}}]_{\mathrm{V}} & \Delta t[K^{p}]_{\mathrm{V}} \\ \Delta t[K^{\mathrm{V}}]_{p} & [M] + \Delta t[K^{p}]_{p} \end{bmatrix} \begin{Bmatrix} \Delta \mathbf{V} \\ \Delta p \end{Bmatrix} = - \begin{Bmatrix} R_{\mathrm{V}} \\ R_{p} \end{Bmatrix}. \tag{12}$$

In addition, for coarse grids a streamline diffusion is added to the problem on both sides of the equation. This can be represented in the condensed form

$$[K(\varepsilon, \mu_{\mathrm{art}})] \begin{Bmatrix} \Delta \mathbf{V} \\ \Delta p \end{Bmatrix} = - \begin{Bmatrix} R_{\mathrm{V}}(\mu_{\mathrm{art}}) \\ R_{p}(\varepsilon) \end{Bmatrix}. \tag{13}$$

This leads to an algorithm where the iteration matrix $[K]$ is computed with progressively lower values of the parameters ε and μ_{art}, referred to as $\varepsilon^{\mathrm{LHS}}$ and $\mu_{\mathrm{art}}^{\mathrm{LHS}}$, but higher than those in the residual denoted by $\varepsilon^{\mathrm{RHS}}$ and $\mu_{\mathrm{art}}^{\mathrm{RHS}}$. The residual is therefore computed with the smallest possible values of these parameters for which the outer Newton iteration converges. This hybrid artificial viscosity algorithm can be described as follows.

1. Set

$$\mu_{\mathrm{art}}^{\mathrm{RHS}} = \mu_{\mathrm{art}}^{\mathrm{LHS}}, \qquad \varepsilon^{\mathrm{RHS}} = \varepsilon^{\mathrm{LHS}}.$$

2. $\mathbf{V}$ and p being given, compute $\| R_{\mathrm{V}}, R_{p} \|_{0}$.

Newton iteration

3. Solve $\Delta \mathbf{V}_i$ and Δp_i with PCGS at each Newton iteration:

$$[K(\varepsilon^{\mathrm{LHS}}, \mu_{\mathrm{art}}^{\mathrm{LHS}})] \begin{Bmatrix} \Delta \mathbf{V} \\ \Delta p \end{Bmatrix} = - \begin{Bmatrix} R_{\mathrm{V}}(\mu_{\mathrm{art}}^{\mathrm{RHS}}) \\ R_{p}(\varepsilon^{\mathrm{RHS}}) \end{Bmatrix}. \tag{14}$$

4. Update $\mathbf{V}$ and p:

$$\begin{Bmatrix} \mathbf{V}_{i+1} \\ p_{i+1} \end{Bmatrix} = \begin{Bmatrix} \mathbf{V}_{i} \\ p_{i} \end{Bmatrix} + \begin{Bmatrix} \Delta \mathbf{V}_{i} \\ \Delta p_{i} \end{Bmatrix}, \tag{15}$$

till $\| R_{\mathrm{V}}, R_{p} \|_{i+1} / \| R_{\mathrm{V}}, R_{p} \|_{0} < 10^{-m}$; repeat from Step 3.

5. Lower $\varepsilon^{\mathrm{RHS}}$ and $\mu_{\mathrm{art}}^{\mathrm{RHS}}$ and repeat from Step 2 if necessary ($m = 3$ for the first cycle and $m = 6$ for the second and subsequent cycles).

Table V. Convergence properties of PCGS as a function of time step; $\varepsilon^{RHS} = \varepsilon^{LHS} = 0{\cdot}05$; $\mu_{art}^{RHS} = 0{\cdot}025$; $\mu_{art}^{LHS} = 0{\cdot}05$

Time step	Newton iterations	Average PCGS iterations per Newton step	Total CPU hours (SGI 4D/310)
0·5	152	10	3·14
1·0	75	17	2·13
5·0	16	47	1·14

Normally two such cycles are sufficient. Numerical results are presented for the gas turbine pipe diffuser of Figure 10. The solutions are obtained for a Reynolds number of 1000 on a structured grid consisting of 25 planes in the flow direction, each with 135 nodes, leading to a total of 11,521 unknowns.

Solutions have been obtained using this algorithm with various time steps. Table V shows the effect of the time step on the convergence properties of the Newton method and PCGS. It can be seen that increasing the time step from 0·5 to 5·0 reduces the number of Newton iterations by a factor of nearly 10, while the number of PCGS iterations per Newton step increases by a factor of nearly 5. The net outcome, however, demonstrates clearly that the highest time step for which PCGS converges will lead to the smallest computational effort. It must be noted that the PCGS method could not converge for the steady state equations, even with very high damping values of ε and μ_{art} on the left-hand side.

Figure 11 shows the effect of the time step on the PCGS convergence. The method is rapid and uniform for small time steps. At large time steps convergence is slower and its behaviour is non monotonic.

The effect of μ_{art} is shown in Table VI. The convergence rate is significantly improved with an artificial viscosity of 0·05. This is especially true for large time steps, where the convergence rate can be increased by as much as 50%.

Figure 12 presents the storage requirements for a direct method with skyline storage compared with PCGS. It should be remarked, however, that the various meshes used had the same number of nodes per plane, which favours direct methods, since the bandwidth remains constant for all meshes. This explains why the storage requirement is only $O(\text{NEQ}^{1\cdot3})$ rather than $O(\text{NEQ}^{1\cdot67})$.

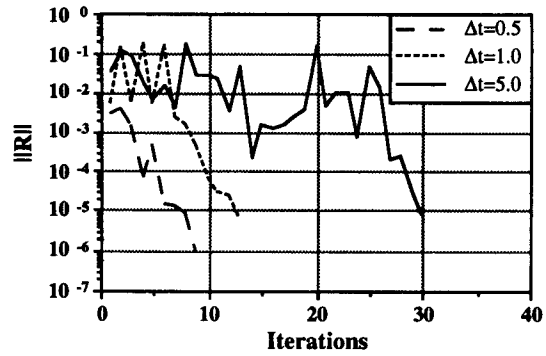

Figure 11. Convergence of PCGS as a function of time step

Table VI. Influence of μ_{art} and time step on convergence properties; $\varepsilon^{RHS} = \varepsilon^{LHS}$, $\mu_{art}^{RHS} = \mu_{art}^{LHS}$

Time step	PCGS convergence rate ($\varepsilon = 0{\cdot}05$, $\mu_{art} = 0{\cdot}00$)	PCGS convergence rate ($\varepsilon = 0{\cdot}05$, $\mu_{art} = 0{\cdot}05$)
0·5	0·335 (30*)	0·346
1·0	0·166 (15*)	0·201
5·0	0·050 (1*)	0·073

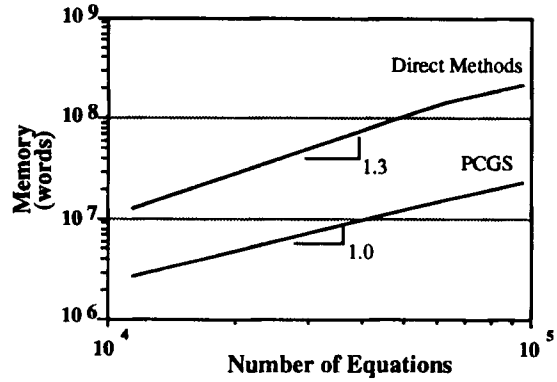

Figure 12. Memory requirement of PCGS and direct method as a function of number of equations (bandwidth constant)

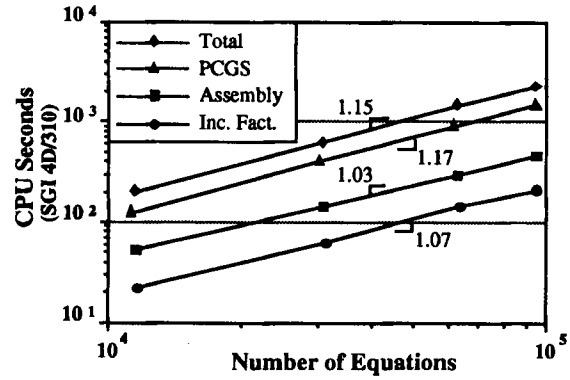

Figure 13. Solution time requirement of PCGS per Newton step as a function of number of equations

For the largest problem tested, involving about 100,000 variables, the ratio of storage is around 10, confirming the advantage of PCGS over direct methods in terms of memory requirements.

Figure 13 shows that the theoretical optimal rate of $O(\text{NEQ}^{1\cdot17})$ for the PCG method on symmetric problems is also attained by the PCGS method for the non-symmetric test problems presented here. It also shows that the matrix integration and assembly is $O(\text{NEQ})$ and relatively important when compared with the incomplete factorization.

It should be mentioned finally that the parallelism offered by a four-CPU Silicon Graphics machine has not yet been utilized. Proper implementation of parallelism in the PCGS algorithm could lead to some future gains in execution time.

4.2. Domain decomposition

Domain decomposition encompasses a wide range of approaches. For example, the Schwarz decomposition has been used in its classical overlapping domain form by many, typified by References 20 and 21, while more recently an effort has been made in the direction of non-overlapping domains.[22–26] In References 22–24 the non-overlapping approach is interpreted as an augmented Lagrangian method. Akay and Ecer,[25] on the other hand, alternate between specifying Dirichlet and Neumann conditions at the interfaces: in the first iteration the domains are solved with a Dirichlet boundary condition and in the following iteration the numerical Neumann condition is imposed, yielding the next Dirichlet condition, and so on. Dacles and Hafez[26] present a hybrid scheme solving the fully-coupled equations on subdomains and communicating by solving the segregated variables on the global domain. In the following, only two approaches are investigated and the augmented Lagrangian methods developed in References 22–24 are applied to the Navier–Stokes equations.

Overlapping domains. The first method tested is the classical Schwarz method, with a solution sought on overlapping subdomains, using the latest values on the overlapping boundaries as boundary conditions. This method is well understood for elliptic boundary value problems and convergence proofs and estimates of the rate of convergence have been obtained in References 21 and 22. Results have also been obtained for incompressible viscous flows.[17,22]

In the present work the approach is applied to the incompressible flow in the gas turbine pipe diffuser of Figure 10 at a Reynolds number of 200. Figure 14 shows convergence results with the number of blocks varying from two to six and the number of planes of overlap from three to five. As expected, convergence depends on both parameters. The top two curves demonstrate that convergence is slowed down almost proportionally to the number of blocks used, for the same number of planes of overlap, but that the convergence rate is relatively unaffected. The bottom two curves demonstrate the effect of the degree of overlap: when this is increased from three to five on two subdomains, the convergence rate is substantially improved.

It has not yet been attempted to compare between the solution times of the different

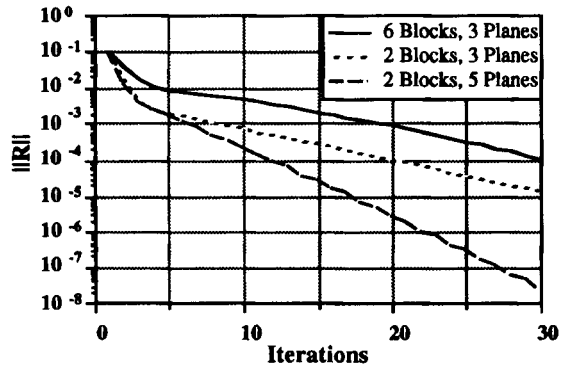

Figure 14. Convergence behaviour of Schwarz overlapping domain method

partitioning methods. For example, a renumbering of the nodes over each subdomain can lead to a greatly reduced bandwidth and dramatically reduced solution times when using a direct solver.

Non-overlapping or side-by-side domains. For complex geometries, the overlapping subdomain method introduces a difficult management of data structures and some redundant computations. A variant of the method was proposed by Lions[23] and Glowinski and Le Tallec[24] for simple elliptic problems on non-overlapping domains. The approach is to solve on side-by-side subdomains Robin-type problems, with an adjustment of the fluxes at the interfaces in order to obtain convergence of the global problem.

The method is applied here to the incompressible Navier–Stokes equations

$$\nabla \cdot (\mathbf{V} - \varepsilon \nabla p) = 0, \tag{16a}$$

$$\mathbf{V} \cdot \nabla \mathbf{V} = -\frac{1}{\rho} \nabla p + \frac{1}{Re} \nabla^2 \mathbf{V}, \tag{16b}$$

the boundary conditions being that at the inlet to the global domain

$$\mathbf{V} = \mathbf{V}_0 \tag{17a}$$

and at the exit

$$\frac{\partial \mathbf{v}}{\partial n} = 0, \qquad p = 0. \tag{17b}$$

A weak-Galerkin weighted residual formulation yields for the continuity and x-momentum equations

$$\sum_{i=1}^{N} \int_{\Omega_i} [(\varepsilon p_x - u)W_x + (\varepsilon p_y - v)W_y + (\varepsilon p_z - w)W_z] \, d\Omega_i - \int_{\partial\Omega_i} \left(\varepsilon \frac{\partial p}{\partial n} - \mathbf{V} \cdot \mathbf{n} \right) W \, ds = 0, \tag{18a}$$

$$\sum_{i=1}^{N} \int_{\Omega_i} [ReW\mathbf{V} \cdot \nabla u + (u_x - Rep)W_x + (u_y - Rep)W_y + (u_z - Rep)W_z] \, d\Omega_i$$
$$- \int_{\partial\Omega_i} \left(\frac{\partial \mathbf{V}}{\partial n} - Rep\mathbf{n} \right) W \, ds = 0, \tag{18b}$$

N being the number of non-overlapping subdomains. The above equations are then solved on each subdomain with the following boundary conditions at nodes on the interfaces:

$$-\varepsilon \partial p / \partial n + r_p p = \lambda_p + r_p p_{\text{adj}}, \tag{19a}$$

$$\partial \mathbf{V} / \partial n - Rep\mathbf{n} + r_{\mathbf{V}} \mathbf{V} = \lambda_{\mathbf{V}} + r_{\mathbf{V}} \mathbf{V}_{\text{adj}}, \tag{19b}$$

where r is a relaxation or penalty and $(\lambda_p, \lambda_{\mathbf{V}})$ are Lagrange multipliers for the continuity and momentum equations respectively.

Introducing (19) into the Galerkin formulation, one has for the continuity equation,

$$\sum_{i=1}^{N} \int_{\Omega_i} [(\varepsilon p_x - u)W_x + (\varepsilon p_y - v)W_y + (\varepsilon p_z - w)W_z]\, \mathrm{d}\Omega_i + \int_{\partial\Omega_i} [(\lambda_p + r_p p_{\mathrm{adj}} - r_p p) + \mathbf{V}\cdot\mathbf{n}]W\, \mathrm{d}s = 0 \tag{20a}$$

and for the x-momentum equation

$$\sum_{i=1}^{N} \int_{\Omega_i} [ReW\mathbf{V}\cdot\nabla u + (u_x - Rep)W_x + (u_y - Rep)W_y + (u_z - Rep)W_z]\, \mathrm{d}\Omega_i - \int_{\partial\Omega_i} (\lambda_u + r_u u_{\mathrm{adj}} - r_u u)W\, \mathrm{d}s = 0, \tag{20b}$$

Therefore for each block one would be solving the following set of equations for continuity and x-momentum:

$$\mathrm{LHS} - \int_{\partial\Omega_i} r_p p W\, \mathrm{d}s = \mathrm{RHS} - \int_{\partial\Omega_i} (\lambda_p + r_p p_{\mathrm{adj}})W\, \mathrm{d}s, \tag{21a}$$

$$\mathrm{LHS} + \int_{\partial\Omega_i} r_u u W\, \mathrm{d}s = \mathrm{RHS} - \int_{\partial\Omega_i} (\lambda_u + r_u u_{\mathrm{adj}})W\, \mathrm{d}s, \tag{21b}$$

where LHS and RHS correspond to those of the original equations on each subdomain before domain decomposition.

The initial values of $(\lambda_p, \lambda_{\mathbf{V}})$ can be taken as

$$\lambda_p = -\varepsilon \partial p/\partial n, \tag{22a}$$

$$\lambda_{\mathbf{V}} = \partial \mathbf{V}/\partial n - Rep\mathbf{n} \tag{22b}$$

and are updated as follows:

$$\lambda_p^{n+1} = \lambda_p^n + r_p(p_{\mathrm{adj}} - p)^n, \tag{23a}$$

$$\lambda_{\mathbf{V}}^{n+1} = \lambda_{\mathbf{V}}^n + r_{\mathbf{V}}(\mathbf{V}_{\mathrm{adj}} - \mathbf{V})^n. \tag{23b}$$

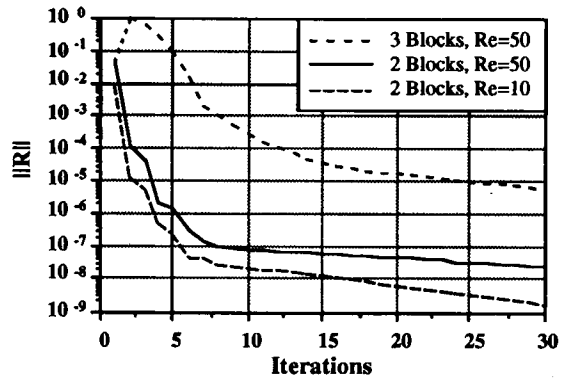

figure 15. Augmented Lagrangian method for domain decomposition

Results for the Navier–Stokes equations have been obtained for the gas turbine pipe diffuser of Figure 10. Convergence depends, as expected, on the Reynolds number as shown in Figure 15 and this requires a good choice of the relaxation parameter r. Values of $r_p = 1$ and $r_v \approx Re$ have proven to be a reasonably successful choice for the Reynolds numbers tested here.

These results demonstrate the potential of the method. Its robustness and convergence rates could be improved by using PCGS as a preconditioner for a minimum residual algorithm applied to the fully-coupled update of the Lagrange multipliers to minimize the jumps at the domain interfaces.

5. CONCLUSIONS

It can be concluded that advanced supercomputers such as the Cray YMP or NEC SX-3 make the use of large-scale direct solvers for the solution of CFD problems possible today. In this respect it has been demonstrated that direct solution of the coupled sets of equations resulting from the Euler and Navier–Stokes analysis of fluid problems is practical and that Gaussian elimination procedures vectorize and parallelize well on such computers. Execution rates of over 2·3 Gflops can be attained and work is continuing on an out-of-core version of the method to handle even larger CFD problems.

On the other hand, the performance of sparse matrix algorithms seems somewhat reduced for three-dimensional problems in which the grid is more dense in one direction. The development of algorithms such as the LND partially improves such technology, but not yet to the point of making it competitive with other techniques. Further work is needed in three dimensions to demonstrate the appropriateness of an LND sparse matrix technology.

The present application of preconditioned iterative methods such as PCGS to the solution of linearized problems associated with the incompressible and compressible Navier–Stokes equations is quite successful. The introduction of time-dependent terms in the equations improves the matrix conditioning. When this is coupled with a hybrid artificial viscosity method, i.e. higher in the iteration matrix than in the physical matrix, a robust scheme is obtained.

Overlapping domain methodology has been investigated and an interesting side-by-side technique has been introduced which, although not yet fully verified for high-Reynolds-number problems, seems promising enough for the breakdown of large problems into subdomains manageable by either direct or iterative solvers. An implicit method for solving for the Lagrange multipliers at each global field update should considerably improve the speed of the method.

Undoubtedly, the continuing evolution of hardware will make it possible to take increasing advantage of the properties of fully-coupled strategies for fluid dynamics problems. It is also evident that none of the above four methods can single-handedly provide the solution for large problems, but that combinations of them seem promising enough.

ACKNOWLEDGEMENT

This work was partially supported under Strategic and Operating Grants of the Natural Sciences and Engineering Research Council of Canada (NSERC) and of the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR) of the Quebec Government. The continuous support of Pratt & Whitney Canada is gratefully acknowledged.

The authors would like to thank Cray Research (Canada) Inc. and Cray Research Inc. for their generosity in providing all necessary YMP access times, in dedicated mode, on their computers.

The authors would also like to acknowledge the support of SiliconGraphics Computer Systems

(Canada) Inc. for graciously providing the Concordia–Pratt & Whitney team with the 4D/340 platform on which all iterative methods were developed.

APPENDIX; NOMENCLATURE

e	element index
E	total number of elements
$[k]$	element influence matrix
$[K]$	global influence matrix
$[L]$, $[U]$	lower/upper triangular components of $[K]$
L_2	energy norm for residual $= \Sigma(R_i^2)$
$[M]$	global mass matrix
$\mathbf{n}$	normal to interface between subdomains
n	outward normal to a domain
N	total number of subdomains
p	pressure
r	relaxation factor
$\{R\}$, $\|R\|$	residual vector at nodes, residual norm
Re	Reynolds number
t	time
$[S]$	global preconditioning matrix
u, v, w	velocity components
$\mathbf{V}$	velocity vector
W	Galerkin weight function

Greek letters

Δ	change in a variable
ε	pressure dissipation parameter
λ	Lagrange multiplier
μ	viscosity
ρ	density
Ω_i, $\partial\Omega_i$	volume, surface area of a subdomain i

Subscripts

adj	values from interface of adjacent block
art	artificial viscosity
i, j	nodal indices
p	related to continuity equation
u	related to x-momentum equation
v	related to y-momentum equation
$\mathbf{V}$	pertaining to momentum equations
w	related to z-momentum equation
x, y, z	differentiation with respect to x, y, z

Superscripts

LHS	left-hand-side

n	iteration number
p	contribution to pressure term
RHS	right-hand side
u	contribution to u-velocity term
v	contribution to v-velocity term
w	contribution to w-velocity term

REFERENCES

1. T. H. Pulliam, 'A computational challenge: Euler solution for ellipses', *AIAA J.* **28,** 1703–1704 (1990).
2. I. Lottati, S. Eidelman and A. Drobot, 'A fast unstructured grid second-order solver', *AIAA Paper 90-0699*, 1990.
3. G. Baruzzi, W. G. Habashi and M. M. Hafez, 'Finite element solutions of the Euler equations for transonic external flows', *AIAA J.* **29,** 1886–1893 (1991).
4. M. F. Peeters, W. G. Habashi, B. Q. Nguyen and P. L. Kotiuga, 'Finite elements solutions of the Navier–Stokes equations for compressible internal flows'. *AIAA J. Propul. Power*, **8,** 192–198 (1992).
5. G. S. Baruzzi, W. G. Habashi and M. M. Hafez, 'A second order method for the finite element solution of the Euler and Navier–Stokes equations', *Proc. 13th Int. Conf. on Numerical Methods in Fluid Dynamics*, Rome, July 1992, Lecture Notes in Physics no. 414, Springer-Verlag, pp. 509–513.
6. A. George and J. W. H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
7. C. Ashcroft, R. Grimes, J. Lewis, B. Peyton and H. Simon, 'Progress in sparse matrix methods for large linear systems on vector supercomputers', *Int. J. Supercomput. Appl.*, **1,** (4), 10–30 (1987).
8. L. B. Wigton, 'Applications of MACSYMA and sparse matrix technology to multielement airfoil calculations', *AIAA Paper 87-1142-CP*, 1987.
9. M. V. Bhat, W. G. Habashi, J. W. H. Liu, V.-N. Nguyen and M. F. Peeters, 'A note on nested dissection for rectangular grids', *SIAM J. Matrix Anal. Appl.*, **14,** 253–258 (1993).
10. D. J. Rose and G. F. Whitten, 'A recursive analysis of dissection strategies', in J. R. Bunch and D. J. Rose (eds), *Sparse Matrix Computations*, Academic, New York, 1976.
11. O. O. Storaasli, D. T. Nguyen and T. K. Agarwal, 'Parallel–vector solution of large-scale structural analysis problems on supercomputers', *AIAA J.*, **28,** 1211–1216 (1990).
12. O. A. Axelsson and V. A. Barker, *Finite Element Solution of Boundary Value Problems*, Academic, New York, 1984.
13. H. A. van der Voorst, 'The performance of FORTRAN implementations for preconditioned conjugate gradients on vector computers', *Parallel Comput.* **3,** 49–58 (1986).
14. Y. Saad and M. H. Schultz, 'GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems', *SIAM J. Sci. Stat. Comput.*, **17,** 856–869 (1986).
15. P. Sonneveld, 'CGS: a fast Lanczos-type solver for nonsymmetric linear systems', *SIAM J. Sci. Stat. Comput.*, **20,** 36–52 (1989).
16. M. Fortin, 'Some iterative methods for incompressible flow problems, *Comput. Phys. Commun.*, **53,** 393–399 (1989).
17. L. B. Wigton, N. J. Yu and D. P. Young, 'GMRES acceleration of computational fluid dynamic codes', *AIAA Paper 85-1494*, 1985.
18. E. F. D'Azavedo, P. A. Forsyth and W.-P. Tang, 'Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems', *Res. Rep. CS-90-04*, Department of Computer Science, University of Waterloo, 1990.
19. G. F. Carey, K. C. Wang and W. D. Joubert, 'Performance of iterative methods for Newtonian and generalized Newtonian flows', *Int. j. numer. methods fluids*, **9,** 127–150 (1989).
20. G. Rodrigue, 'Some ideas for decomposing the domain of elliptic partial differential equations in the Schwarz process', *Commun. Appl. Numer. Methods*, **2,** 245–249 (1986).
21. P. L. Lions, 'On the Schwarz's alternating method', *Proc. First Int. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1987.
22. R. Aboulaich and M. Fortin, 'Schwarz's decomposition method for incompressible flow problems', *Proc. First Int. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1988, pp. 333–349.
23. P. L. Lions, 'On the Schwarz alternating method III: a varient for nonoverlapping domains', *Proc. Third Int. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1990, pp. 202–223.
24. R. Glowinski and P. Le Tallec, 'Augmented Lagrangian interpretation of the nonoverlapping Schwarz alternating method'. *Proc. Third Int. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1990, pp. 224–231.
25. H. U. Akay and A. Ecer, 'A block-structured finite element solution of viscous internal flows', *Proc. Int. Conf. on Computational Methods in Flow Analysis*, Okayama, 1988, pp. 328–333.
26. J. Dacles and M. M. Hafez, 'Numerical methods for 3-D viscous incompressible flow using velocity–vorticity', *AIAA Paper-90237*, 1990.