Contemporary Mathematics
Volume 157, 1994

Parallel Domain Decompositon for Incompressible Fluid Dynamics

PAUL F. FISCHER

ABSTRACT. We consider the implementation of a domain decomposition scheme for spectral element solution of the Navier-Stokes equations in three-dimensional domains. Calculations performed on the 512 node Intel Delta machine illustrate the effectiveness of improved strategies for the coarse grid solve, subdomain edge update, and local block-preconditioner when solving large problems. In addition, a projection technique is presented which significantly reduces the number of iterations for solution of the elliptic sub-problems arising from this class of time-dependent problems.

1. Introduction

Domain decomposition provides a mechanism for parallel solution of PDE's by establishing a two-level data hierarchy naturally suited to distributed memory architectures. In the context of iterative solution of linear systems, domain decomposition also provides a framework for the development of efficient multi-level methods based upon the combined use of local and global solvers which effect rapid information transfer at their respective scales. In this paper, we discuss several key components for a domain decomposition approach to solution of the incompressible Navier-Stokes equations on large scale multi-computers. The scheme is based upon the spectral element method [1,2] coupled with the domain decomposition based pressure iteration introduced by Ronquist [3,4]. Calculations have been performed on the Intel Delta machine at Caltech, which is a mesh-connected multiprocessor employing 512 Intel i860 microprocessors.

Our current development work has focussed upon the particular baseline problem shown in Fig. 1, which is the case of a flat plate boundary layer interacting with a hemispherical roughness element. The discretization consists of 512

1991 *Mathematics Subject Classification.* Primary 65M70,65Y05, 65M55, 76D05.
The author was supported in part by NSF Grants #ASC-9107674 and #CCR-8809615.
Time on the Intel Delta was provided by Concurrent Supercomputing Consortium.
This paper is in final form and no version of it will be submitted for publication elsewhere.

0271-4132/94 $1.00 + $.25 per page

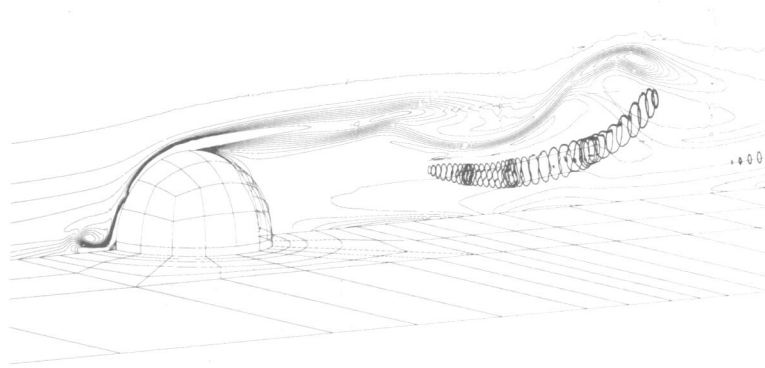

FIGURE 1. Computed interaction of a flat plate boundary layer with a hemispherical roughness element at $Re = 500$. $K = 512$, $N = 9$.

spectral elements of order 9. The hemisphere of radius $R = 1$ is centered at $x = (0, 0, 0)$. A Blasius profile with $\delta_{.99} = 1.15$ and $U_\infty = 1$ is specified for the x-component of velocity both as an initial condition and inlet profile at $x = -8.4$. Symmetry boundary conditions are specified at $y = 0$, $y = -6.4$, and $z = 6.5$, and Neumann outflow boundary conditions are imposed at $x = 25.6$. Fig. 1 shows contours of spanwise vorticity in the symmetry plane ($y = 0$) along with isosurfaces of axial vorticity in the foreground for a Reynolds number of $Re = \frac{RU_\infty}{\nu} = 500$. The hairpin vortices downstream of the hemisphere are observed experimentally [5].

The hemisphere problem of Fig. 1 is of moderate size; when running on all 512 nodes on the Delta, the distribution of one spectral element per processor yields fewer than 1000 points per processor. At this relatively fine granularity, parallel overhead can be significant. In this paper we address two sources of overhead generic to many domain decomposition solvers - the communication required for the residual update of grid points residing on the "wire-frame" comprised of subdomain edges and vertices, and the coarse-grid solve, which is typically not amenable to parallelism because of the few degrees-of-freedom involved. In addition, we present improvements to the local preconditioner, and a projection technique which improves the initial guess for the pressure solver, resulting in a significant reduction in pressure iterations.

2. Domain Decomposition

In order to underscore the key components of our Navier-Stokes solver, we briefly review the spectral element discretization and domain decomposition based solvers introduced in [3].

The spectral element method is based upon a decomposition of the computational domain into K hexahedral elements which are locally mapped to $[-1,1]^d$ in $\mathcal{R}^d$. Within each element, the geometry, solution, and data are expanded in terms of high-order tensor-product Lagrangian bases in each coordinate direction. Variational projection operators are used to discretize the elliptic PDE's arising from time split treatment of the Navier-Stokes equations, and a consistent variational formulation is used for the pressure/divergence treatment. The velocity is represented by Nth-order polynomials on the Gauss-Lobatto-Legendre quadrature points, $\xi_j \in [-1,1]$, $j = 0,...,N$, in each coordinate direction, with C^0 continuity enforced at element interfaces. The pressure is represented by polynomials of degree $N-2$ based upon the Gauss-Legendre quadrature points, $\eta_j \in]-1,1[$, $j = 1,...,N-1$, in each coordinate direction, with continuity and boundary conditions imposed only through the divergence-free constraint in the Stokes operator. Further details of spectral element discretizations for the Navier-Stokes equations may be found in [2].

Temporal discretization is based upon an operator splitting in which the nonlinear convective terms are treated explicitly via a characteristic/sub-cycling scheme, and the viscous and divergence operators are treated implicitly. Spatial discretization of the resultant unsteady Stokes problem leads to the following linear system to be solved at each time step:

$$\begin{aligned} \underline{H}\,\underline{u}_i - \underline{D}_i^T\,\underline{p} &= \underline{B}\,\underline{f}_i, \qquad i = 1,...,d \quad , \\ \underline{D}_i\,\underline{u}_i &= 0 \quad . \end{aligned} \tag{2.1}$$

Here, $\underline{H}$ is the discrete equivalent of the Helmholtz operator, $\{-\frac{1}{Re}\nabla^2 + \frac{1}{\Delta t}\}$; $\underline{B}$ is the mass matrix associated with the velocity mesh; $\underline{D} = (\underline{D}_1,...,\underline{D}_d)$ is the discrete gradient operator; and underscore refers to basis coefficients. The solution of (2.1) is simplified by a Stokes operator splitting which decouples the viscous and pressure/divergence constraint [6]. This splitting leads to the solution of a standard Helmholtz equation for each velocity component, while the resulting system for the pressure is similar to (2.1) save that $\underline{H}$ is replaced by $\frac{1}{\Delta t}\underline{B}$. The resulting system can be efficiently treated by formally carrying out block Gaussian elimination (Uzawa decoupling) for $\underline{p}$, leading to:

$$\underline{E}\underline{p} = \underline{g}, \tag{2.2}$$

where

$$\underline{E} = -\sum_{i=1}^{d} \underline{D}_i\,\underline{B}^{-1}\underline{D}_i^T, \tag{2.3}$$

and $\underline{g}$ is the inhomogeneity resulting from the time-split treatment of (2.1). The advantage of the Stokes splitting is that no system solves are required when applying $\underline{E}$, as $\underline{B}$ is diagonal.

$\underline{E}$ corresponds to a consistent Poisson operator for the pressure and, though symmetric-positive definite, is less well conditioned than the Helmholtz problems for the velocity components. Consequently, solution of (2.2) dominates the Navier-Stokes solution time. An effective multi-level iterative solver for this problem was developed by Rønquist [3] in which the pressure is split into two components, $\underline{p}_0 \in \mathcal{R}^K$ - which is a vector containing the average pressure within each element, and $\underline{p}_N$ - which accounts for the local pressure fluctuation within each element. With this decomposition, equation (2.2) becomes:

$$\underline{E}(\underline{I}\underline{p}_0 + \underline{p}_N) = \underline{g}\,, \tag{2.4}$$

where $\underline{I}$ is a local operator that maps a constant onto each node within a given element. Using the transpose operator, $\underline{I}^T$, which takes the local average of values within an element, we define the following operators:

$$\underline{E}_0 = \underline{I}^T \underline{E}\, \underline{I}\,, \tag{2.5}$$

$$\underline{E}_N = \underline{E} - \underline{E}\, \underline{I}\, \underline{E}_0^{-1} \underline{I}^T \underline{E}\,, \tag{2.6}$$

which can be used to recast (2.4) as:

$$\underline{E}_0 \underline{p}_0 = \underline{I}^T \underline{g} - \underline{I}^T \underline{E} \underline{p}_N\,, \tag{2.7}$$

$$\underline{E}_N \underline{p}_N = \underline{g} - \underline{I}\, \underline{E}_0^{-1} \underline{I}^T \underline{g}\,. \tag{2.8}$$

Once $\underline{p}_N$ is known, $\underline{p}_0$ can be readily computed, as $\underline{E}_0$ is a relatively small $K \times K$ system which can be computed explicitly and factored in a preprocessing step. $\underline{E}_N$ is a much larger system which is solved by preconditioned conjugate gradient iteration, necessitating repeated global solves in $\underline{E}_0$ due to (2.6).

As noted in [3], the advantages of the decomposition of (2.2) into (2.7-2.8) are: (i) low-wavenumber components in $\underline{p}_N$ are effectively eliminated in (2.8) by the presence of $\underline{E}_0^{-1}$, making the required number of iterations to reach a specified tolerance relatively insensitive to the number of elements, and (ii) the system (2.8) is sufficiently decoupled such that an effective block preconditioner, $\tilde{\underline{E}}_N$, can be developed by using the local $\underline{E}$-matrices corresponding to imposing homogeneous Dirichlet velocity boundary conditions on all the external *and* internal elemental (or subdomain) interfaces. As a result of the decoupling, the preconditioning step can be performed using direct solvers and is readily parallelized as it involves no communication.

In summary, the basic elements of the two-level pressure iteration are:

- Forward Operator Application - $\underline{E}$
- Coarse Grid Solve - $\underline{E}_0^{-1}$
- Local Preconditioner Solve - $\tilde{\underline{E}}_N^{-1}$

In the following sections, we present approaches to each of these components which are appropriate for three-dimensional domain decomposition implementations on a large number of processors.

2.1. Operator Evaluation/Communication. Our parallel implementation follows standard domain decomposition/distributed memory multicomputer programming paradigms in which K elements are distributed amongst P processors, iteration and time advancement proceeds in a loosely synchronous manner, and interprocessor communication serves to synchronize the computation [7,8]. Vector reductions (e.g., inner-products) are performed in $O(\log_2 P)$ communication cycles and face-face exchanges are employed to invoke direct stiffness summation.

Spectral element operator evaluation proceeds in two stages. Locally, derivative operators are applied in a tensor product fashion, leading to an operation count for each element of $O(N^{d+1})$ in d space dimensions. The evaluation can be expressed as a matrix-matrix product, which is an extremely fast operation on most vector and RISC architectures. Inter-element continuity is applied via direct stiffness summation in which intermediate residual values corresponding to gridpoints shared by two or more elements are summed. Because it is non-local, direct stiffness summation requires interprocessor communication.

Direct stiffness summation is organized into two distinct phases. The first involves a face-face exchange and sum of data between adjacent pairs of elements. Edges and vertices shared by more than two elements may require communication amongst more than just two processors. However, if the elements are arrayed in a tensor-product structure, it is possible to update all vertices and edges correctly by organizing the face-face exchange into a d-directional exchange and sum sequence [7]. For irregular element configurations, it is not always possible to find such a sequence. Thus, the second phase of the direct stiffness summation is to employ a global combine operation to update the remaining $\mathcal{N}_{spec}$ "special" nodes not correctly updated in the first phase.

Our original implementation of the special node combine operation proceeded by replicating a vector of length $\mathcal{N}_{spec}$ on each processor which was initialized to zero. Residuals at each special node were added to the appropriate location in the vector, which was then summed via a global combine operation requiring $O(\mathcal{N}_{spec} \log_2 P)$ communication. This approach is easy to code, and very effective when $\mathcal{N}_{spec}$ and $\log_2 P$ are small.

On currently available parallel supercomputers, we face the situation where we can solve very large problems, implying that $\mathcal{N}_{spec}$ and $\log_2 P$ are no longer small. We have therefore adopted a generalized combine operation based on the static crystal router concept of Fox, *et. al.* [8]. The procedure recursively subdivides groups of processors into two new subgroups, and at each stage copies (with summation) data from one subgroup to the other for vertices represented in both subgroups. The procedure requires establishing a destination table for intermediate traffic through each processor, but does result in significant reduction in the overall amount of data traffic for large three-dimensional problems. For example, in the baseline problem of Fig. 1, the total number of points on the "wire-frame" comprised of edges and vertices is roughly 15000, out of 375,000

total. The use of the organized face-face exchange reduces the number of points requiring special treatment to $\mathcal{N}_{spec} = 2000$. Finally, use of the crystal-router based summation scheme results in a maximum message length of only 100 words in any stage of the $log_2 P$ exchange and sum routine.

2.2. Coarse Grid Solve. The presence of the global inverse, $\underline{E}_0^{-1}$, in (2.8) is central to improving the condition of $\underline{E}_N$. However, its presence does imply the possibility of significant serial and communication overhead in a parallel implementation. Because $\underline{E}_0$ is a relatively small $K \times K$ system, a common approach to computing the repeated action of its inverse is to solve the coarse grid system redundantly on each processor. For static geometries, the LU decomposition of $\underline{E}_0$ can be computed once, and subsequent forward-back sweeps are in principle fairly inexpensive. However, for large values of P and K the coarse grid solve will dominate the calculation, and some level of parallelism must be introduced. As the LU solves are inherently serial, an alternative approach is to explicitly compute and store $\underline{E}_0^{-1}$, so that the coarse grid solve is effected by a matrix-vector product (see, e.g., [9]). After a $\log_2 P$ gather operation to construct the right-hand side, the coarse grid solve is completed for each element by computing a *local* inner-product of the right-hand side and the corresponding row of $\underline{E}_0^{-1}$ which is stored on the appropriate processor.

2.3. Finite Element Preconditioner. The original implementation of the two-level pressure iteration employed a local spectral approximation to $\underline{E}$ as a block preconditioner for $\underline{E}_N$ [3]. This approach works well when the order of the approximation is small, e.g., $N \leq 6$. Unfortunately, the memory requirements scale as $O(KN^{2d})$ in $\mathcal{R}^d$, which is prohibitive when $d = 3$ and N is large. An alternative to the full spectral operator is to use a low-order finite element discretization based upon the same Gauss-Legendre collocation points. There have been extensive analyses of FEM based preconditioners for spectral operators in the past, e.g., [10,11], with the general result that the condition number of $L_{FEM}^{-1} L_{sp}$ is bounded and that nine point finite element stencils (in $\mathcal{R}^2$) tend to work better than five point finite difference stencils. As we are considering general geometries, we have chosen the FEM approach, which is capable of capturing more information about local deformation due to the additional off-diagonal terms in the stiffness matrix. However, a sparse finite difference operator coupled with a fast Poisson solver may be of interest if further reduction in memory is required at the expense of preconditioner performance. In the present case, the FEM preconditioner provides an $O(N)$ reduction in memory and operation count for the preconditioning step. In practice, the savings must be weighed against other costs in the Navier-Stokes solution.

In the context of the domain decomposition scheme (2.7-2.8), the FEM preconditioning strategy is quite simple. We replace the *local* consistent Poisson operator, $\underline{\tilde{E}}_N$, with a set of K local standard Poisson operators, $\underline{\tilde{E}}_{FEM}$, discretized using linear finite element bases within each subdomain. Each Pois-

son problem is solved independently, owing to the decoupling afforded by the functional decomposition (2.4). This is a significant improvement over previous FEM preconditioning strategies which result in large FEM problems which must be solved, e.g., approximately by ILU schemes. Our experience has been that introduction of approximate factorization at the FEM level can degrade the preconditioner performance [2]. However, as the domain decomposition results in a set of small local problems, a direct factorization is possible. Each Poisson operator, $\tilde{\underline{E}}_{FEM}$, is a Neumann operator having a null space of dimension unity corresponding to the average pressure level in each element. Thus, when factoring $\tilde{\underline{E}}_{FEM}$ we eliminate the last row and column.

For the baseline problem of Fig. 1, the FEM based preconditioner yielded no appreciable change in iteration count which was 125 when using $\tilde{\underline{E}}_N$ *vs.* 126 when using $\tilde{\underline{E}}_{FEM}$, to reach the same tolerance. Further details of the $\tilde{\underline{E}}_{FEM}$ implementation can be found in [4].

2.4. Performance. We compare the relative impact of each of the algorithm modifications discussed in this section by plotting in Fig. 2 the solution time required for the first step of the hemisphere calculation of Fig. 1 *vs.* number of processors on the Intel Delta. Curve 1 is the time required for the original domain decomposition solver (in 32 bit arithmetic), which clearly does not attain linear speedup. Curve 2 shows the improved performance obtained with the new wire frame update strategy, while Curve 3 shows the additional gain derived from the distributed coarse grid solve. Note that these changes have little impact when

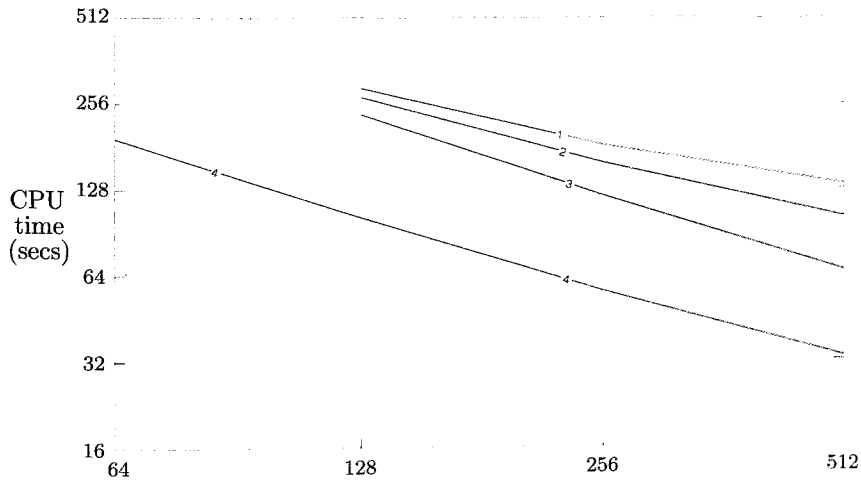

FIGURE 2. Delta performance for first step of hemisphere problem of Fig. 1: Curve 1 - original two-level iteration scheme: Curve 2 - same as 1, with improved wire-frame update scheme: Curve 3 - same as 2, with parallel coarse grid solve; Curve 4 - same as 3, with FEM based preconditioner.

the amount of work on each processor is large, i.e., when $P = 128$. Curve 4 shows the benefit of the FEM based preconditioner, which yields an additional two-fold reduction in solution time, and sufficient memory reduction to allow this problem to run on 64 processors. Overall, a four-fold reduction in CPU time on 512 processors is realized for this particular problem.

3. Orthogonolization

We have recently developed a technique which significantly reduces the required number of iterations for the preconditioned E-solver which can be generalized to a broad class of time dependent problems. We consider repeated solution of the SPD system for the pressure at time level n:

$$\underline{E}\,\underline{p}^n = \underline{f}^n \quad . \tag{3.1}$$

When employing iterative solvers in the advancement of evolution problems it is common to use $\underline{p}^{n-1}$ as an initial guess for $\underline{p}^n$, thereby solving only for the perturbation, $\Delta\underline{p}^n \equiv \underline{p}^n - \underline{p}^{n-1}$. In fact, it is possible to substantially improve upon this initial guess by recognizing that one has available a sequence of inhomogeneities, $\underline{f}^k$, and associated solutions, $\underline{p}^k$, satisfying (3.1). The underlying idea is to remove from $\underline{f}^n$ all components for which we already know the solution, and to solve only for the resulting perturbed problem.

We begin by assuming that we have stored a set of vectors $\tilde{F} = \{\underline{\tilde{f}}^1, \ldots, \underline{\tilde{f}}^l\}$ and solution vectors $\tilde{P} = \{\underline{\tilde{p}}^1, \ldots, \underline{\tilde{p}}^l\}$ satisfying:

$$\underline{E}\underline{\tilde{p}}^k = \underline{\tilde{f}}^k \quad k = \{1, \ldots, l\} \tag{3.2}$$

$$< \underline{\tilde{f}}^i, \underline{\tilde{f}}^j > = \delta_{ij} \quad , \tag{3.3}$$

where δ_{ij} is the Kroenecker delta, and $<>$ is an appropriately weighted inner-product. The algorithm is based upon the following Gram-Schmidt procedure:

(3.4) *At time level* n, *input* $\underline{f}^n$:

$$\begin{aligned}
&\alpha_k =< \underline{f}^n, \underline{\tilde{f}}^k >, \qquad k = 1, \ldots, L' \\
&\underline{\bar{f}} \longleftarrow \underline{f}^n - \textstyle\sum \alpha_k \underline{\tilde{f}}^k \\
&\textit{solve } \underline{E}\underline{\bar{p}} = \underline{\bar{f}} \text{ to tolerance } \epsilon \\
&\underline{p}^n \longleftarrow \underline{\bar{p}} + \textstyle\sum \alpha_k \underline{\tilde{p}}^k \\
&\textit{update } \{\tilde{P}, \tilde{F}\} \\
&\textit{return } \underline{p}^n \quad .
\end{aligned}$$

To complete the procedure, we require a mechanism to update $\{\tilde{P}, \tilde{F}\}$. Initial trials have shown the following approach to be successful. If L is taken to be the maximum number of vector pairs to be stored, i.e., $l \leq L$, then at each time level:

(3.5)
$$
\begin{aligned}
&\textit{If } (l = L) \textit{ then:} && \tilde{\underline{p}}^1 \longleftarrow \underline{p}^n / ||\underline{E}\underline{p}^n|| \\
& && \tilde{\underline{f}}^1 \longleftarrow \underline{E}\underline{p}^n / ||\underline{E}\underline{p}^n|| \\
& && l = 1 \\
&\textit{else:} && \hat{\underline{f}} \longleftarrow \underline{E}\tilde{\underline{p}} \\
& && \alpha_k = < \hat{\underline{f}}, \tilde{\underline{f}}^k >, \qquad k = 1, \ldots, l \\
& && \tilde{\underline{f}}^{l+1} \longleftarrow (\hat{\underline{f}} - \textstyle\sum \alpha_k \tilde{\underline{f}}^k) / ||\hat{\underline{f}} - \sum \alpha_k \tilde{\underline{f}}^k|| \\
& && \tilde{\underline{p}}^{l+1} \longleftarrow (\tilde{\underline{p}} - \textstyle\sum \alpha_k \tilde{\underline{p}}^k) / ||\hat{\underline{f}} - \sum \alpha_k \tilde{\underline{f}}^k|| \\
& && l = l + 1 \\
&\textit{endif}\,.
\end{aligned}
$$

Here, $||.|| = < . >^{\frac{1}{2}}$. The procedure re-initializes $\{\tilde{P}, \tilde{F}\}$ with the most recent solution pair when the memory limits are exceeded, and then reconstructs a set which satisfies (3.2-3.3). Notice that the intermediate variable $\hat{\underline{f}}$ is used in place of $\tilde{\underline{f}}$, as the vector pair $\{\tilde{\underline{p}}, \tilde{\underline{f}}\}$ does not satisfy (3.2) exactly, owing to error due to incomplete iteration.

The effectiveness of the orthogonalization technique depends upon the extent to which $\underline{f}^n$ can be represented by the basis $\tilde{\underline{f}}_k$. In flows devoid of dynamics, the pressure at time t^n will be well represented by $\underline{p}^{n-1}$. Hence, little improvement can be expected for $L > 1$. For problems having a richer dynamical structure, there is greater potential for savings. This is illustrated in Fig. 3, in which we plot the number of pressure iterations, $\mathcal{N}_E$, per step to compute flow past a cylinder at $Re_D = 200$, for $L = 1$, 3, and 21. The discretization consists of $K = 116$ spectral elements of degree $N = 9$, with time step $\Delta t = .0168$. At each

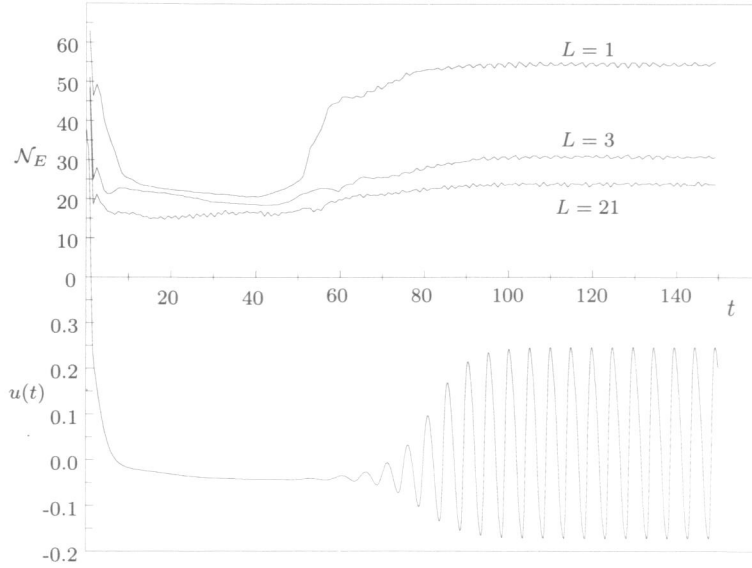

FIGURE 3. Pressure iteration count and time history of velocity for impulsively started flow past a cylinder at $Re = 200$.

step the iteration was carried out until the $\mathcal{L}^2$ norm of the residual was reduced to 10^{-5}. For clarity, a 50 step windowed average of the data is presented. The flow passes through three transient regimes: symmetric wake formation, wake destabilization, and periodic vortex shedding. The first and third regimes are characterized by a high level of dynamic activity, while the second is relatively quiescent, as illustrated in the lower half of Fig. 3 by the time trace of u at a point in the near wake region of the cylinder. Increasing L to 30 brought about no additional reduction in iteration count.

We have also applied the orthogonalization technique with $L = 11$ to the hemisphere problem of Fig. 1. For $K = 512$, $N = 9$ (260,000 pressure d.o.f.'s), the average CPU time/step on the Delta ($P = 512$) was reduced from 15 to 10 seconds. For the same problem with $N = 11$ (512,000 pressure d.o.f.'s), the time was reduced from 50 to 25 seconds/step.

Acknowledgements

The author would like to thank Dr. Einar Rønquist of Nektonics, Inc. for providing access to his iterative solvers and for many helpful discussions during the course of this work.

References

1. A.T. Patera, *J. Comput. Phys.***54** (1984) 468.
2. Y. Maday, and A.T. Patera, "Spectral element methods for the Navier-Stokes equations", in *State of the Art Surveys in Computational Mechanics*, edited by A.K. Noor, ASME, New York, 1989.
3. E.M. Rønquist, "A Domain Decomposition Method for Elliptic Boundary Value Problems: Application to Unsteady Incompressible Fluid Flow," in *Fifth Conference on Domain Decomposition Methods for Partial Differential Equations* (T.F. Chan, D.E. Keyes, G.A. Meurant, J.S. Scroggs, and R.G. Voigt, eds.), SIAM, 1992.
4. P.F. Fischer and E.M. Rønquist, "Spectral Element Methods for Large Scale Parallel Navier-Stokes Calculations" in *Proceedings of the International Conference on Spectral and High Order Methods for Partial Differential Equations '92* (ed. C. Bernardi and Y. Maday), (1993) to appear.
5. M.S. Acarlar and C.R. Smith, *J. Fluid Mech.* **175** (1987) 1-41.
6. Y. Maday, A.T. Patera, and E.M. Rønquist, "An operator-integration-factor splitting method for time-dependent problems: application to incompressible fluid flow." *J. Sci. Comput.*, **5**, (4) (1990) 263-292.
7. P.F. Fischer and A.T. Patera, "Parallel Spectral Element Solution of the Stokes Problem." *J. Comput. Phys.* **92** (1991) 380-421.
8. G. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon, D. Walker *Solving Problems on Concurrent Processors. Volume 1 : General Techniques and Regular Problems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1988.
9. W.D. Gropp and D.E. Keyes, "Domain decomposition with local mesh refinement," *SIAM J. Sci. Statist. Comput.* **13** (1992).
10. S.A. Orszag, *J. Comput. Phys.* **37** (1980) 70.
11. M.O. Deville and E.H. Mund, "Fourier analysis of finite element preconditioned collocation schemes", *SIAM J. Sci. Statist. Comput.*, **13** (1992).

Division of Applied Mathematics, Brown University, Providence, RI 02912
E-mail address: pff@cfm.brown.edu