SIAM J. SCI. COMPUT.
Vol. 21, No. 4, pp. 1387–1400

ELEMENT-BY-ELEMENT PARALLEL COMPUTATION OF INCOMPRESSIBLE NAVIER–STOKES EQUATIONS IN THREE DIMENSIONS*

TONY W. H. SHEU†, MORTEN M. T. WANG†‡, AND S. F. TSAI†

Abstract. Development of a stable finite element model for solving steady incompressible viscous fluid flows in three dimensions is the main theme of the present study. For stability reasons, weighting functions are designed in favor of field variables on the upstream side. For accuracy reasons, it is required that weighting functions be equipped with the streamline operator so that false diffusion errors can be largely suppressed. In the steady-state analysis of Navier–Stokes equations, we adopt the mixed formulation to preserve mass conservation on quadratic elements which accommodate the Ladyzhenskaya–Babŭska–Brezzi (LBB) stability condition. To resolve difficulties arising from asymmetry and indefiniteness in the resulting large-size matrix equations, we abandon the elimination-like solution solver because the storage demand exceeds the ability of our hardware to solve for three-dimensional problems. A modern iteration solver, known as the biconjugate gradiant stabilized (BICGSTAB) solution solver, is thus implemented in an element-by-element fashion to effectively alleviate the problem. For performance reasons, the finite element code developed here should be implemented in a hardware environment which is suited to the use of an iterative solver. To this end, our analysis is implemented in shared memory parallel architectures, CRAY C-90 and J-90. We benchmark the parallel computing performance through a lid-driven cavity flow problem and a problem amenable to analytic solution.

Key words. incompressible viscous flow, three dimensions, asymmetry, indefiniteness, BICGSTAB, element-by-element, parallel

AMS subject classifications. 65F10, 65N30, 76D05

PII. S1064827598335416

1. Introduction.

The study of incompressible Navier–Stokes equations is of practical interest with respect to transport phenomena and also helps research engineers gain a better understanding of the processes which occur in many industrial applications. Investigating flow convection in incompressible viscous fluid flows poses a formidable challenge to numerical methods. First among the complication factors is the fact that the pressure does not appear explicitly in the continuity equation even though the pressure field has a direct influence on the divergence of the velocity. A pressure Poisson equation (PPE) can be derived from the momentum and continuity equations [10]. A fundamental problem with the PPE approach for incompressible flow analyses is to implement numerically the proper pressure boundary conditions. A mixed formulation, which solves for the continuity equation directly rather than the Poisson equation for pressure, is presented as a remedy for the difficulty. In doing so, the ambiguity in imposing the proper boundary condition for the Poisson pressure equation is eliminated. However, the absence of pressure in the continuity equation is problematic with regard to the choice of a finite element space for working variables and poses a challenge to apply an iterative solver to solve for matrix equations. In the first category, the guideline concerns whether or not the finite element is en-

*Received by the editors March 9, 1998; accepted for publication (in revised form) May 10, 1999; published electronically February 2, 2000. This paper was supported by the National Science Council of the Republic of China under grant NSC84-2212-E-002-060.
http://www.siam.org/journals/sisc/21-4/33541.html
†Department of Naval Architecture and Ocean Engineering, National Taiwan University, 73 Chou-Shan Rd., Taipei, Taiwan, R.O.C. (sheu@indy.na.ntu.edu.tw, S.F.Tsai@cfd.na.ntu.edu.tw).
‡Tatung Co. Ltd., Taipei, Taiwan R.O.C. (morten@indy.na.ntu.edu.tw).

dowed with the Ladyzhenskaya–Babŭska–Brezzi (LBB) stability condition [3,13]. One major hurdle in the calculation of large-sized matrix equations using the Gaussian-elimination–based direct solver involves dealing with the continued fill-in while finding solutions. Researchers have no choice but to employ iterative solvers. This prompts us to use a modern iterative solver to resolve difficulties in association with matrix indefiniteness and asymmetry, and this is the main theme of this study.

The nonlinear coupling between advective fluxes and the divergence-free velocity presents difficulties in the numerical solution of a convection-dominated flow field and raises the possibility of oscillatory velocities. On physical grounds, there is considerable incentive to adopt the upwind model to suppress velocity oscillations. Although useful for stabilizing the differential system under investigation, the conventional upwind method is not sufficient for predicting the transport phenomena in a domain of multiple dimensions. The prediction accuracy deteriorates due to the introduction of false diffusion errors into the multidimensional discretization of advective fluxes. Therefore, it is desirable to add artificial viscosity to the primary flow direction. The streamline upwind finite element model [19] has formed the basis of the high Reynolds number flow model in the present three-dimensional analysis. Although our finite element model has been used with varying degrees of success, the need to solve large-sized indefinite and unsymmetric algebraic equations places limitations on the use of the computational technique as a rigorous tool. The present work is part of a research effort aimed at reducing the computing time required for finite element study of three-dimensional Navier–Stokes equations. With the advent of software such as PVM and MPI, it is now possible to run codes in parallel on workstations and mainframes. Parallel calculation of coupled algebraic equations will be proposed as a likely approach to reducing the elapsed time requirements for three-dimensional simulations.

The outline of the rest of this paper is as follows. In this paper, we present the working Navier–Stokes equations together with the constraint condition to preserve mass conservation. For the problem to be well-posed, boundary conditions are prescribed along the entire boundary. In section 2, we introduce the concept of the Petrov–Galerkin finite element model on quadratic grids and show how the high-order advection-diffusion scheme is constructed. This is followed by a brief overview of the iterative solvers in the literature, and we also give reasons in section 3 to explain why we adopt the biconjugate gradient stabilized (BICGSTAB) iterative solver to deal with indefinite and unsymmetric matrix equations. In section 4, we point out that there is growing interest in exploring the rapid development of parallel computers. For this study, we ran code in parallel on a shared memory hardware architecture. In section 5, we present numerical results and benchmark the present parallel computation. Finally, we draw conclusions in section 6.

2. Working equations. In a domain of three dimensions, we consider working equations which concern mass and momentum conservations in the incompressible flow field as follows:

$$\nabla \cdot \underline{u} = 0, \tag{2.1}$$

$$\underline{u} \cdot \nabla \underline{u} + \bigtriangledown p - \frac{1}{Re} \nabla^2 \underline{u} = 0 \ . \tag{2.2}$$

It is noted that for incompressible flow, the continuity equation becomes a constraint on the velocity field $\underline{u}$, and the pressure p is implicit in nature.

The above elliptic differential system is subject to the following Dirichlet-type boundary condition at $\partial\Omega = \Gamma$:

$$\underline{u} = \underline{g}, \tag{2.3}$$

where

$$\int_{\Gamma} \underline{n} \cdot \underline{g} \, \mathrm{d}\Gamma = 0 \, . \tag{2.4}$$

The Reynolds number $\mathrm{Re} \equiv \rho u_{\mathrm{ref}} L_{\mathrm{ref}}/\mu$ is the direct result of normalization of dependent as well as independent variables. Despite the trend of adopting the velocity-vorticity formulation in the analysis of incompressible viscous flows, we stick to working equations cast in a primitive-variable form because a formulation involving use of primitive velocities and pressure accommodates closure boundary conditions [14].

There are several possible avenues for attacking the differential system of interest here. One may be tempted to resort to a segregated approach to solving (2.1)–(2.2). The analysis that follows decouples the calculation of the continuity equations and the equations of motion. While the segregated approach fits the algorithmic idea of parallel computation, this approach has been hindered by the lack of mathematically rigorous boundary conditions. The mixed formulation is, thus, considered to be a logical choice between two major classes of approaches, namely, the segregated and mixed approaches. Given the above reason for using the mixed formulation, solutions to (2.1)–(2.2) are obtained in the weak form from the following weighted residuals statement: Given admissive functions $\underline{w} \in \mathcal{H}_0^1(\Omega) \times \mathcal{H}_0^1(\Omega)$ and $q \in \mathcal{L}_0^2(\Omega)$, primitive variables in the simply connected domain Ω subject to the essential boundary condition $\underline{u} = \underline{g}$ on $\partial\Omega$, we seek $\underline{u} \in \mathcal{H}_0^1(\Omega)$ and $p \in \mathcal{L}_0^2(\Omega)$ from

$$\int_{\Omega} (\underline{u} \cdot \nabla)\underline{u} \cdot \underline{w} \, \mathrm{d}\Omega + \frac{1}{\mathrm{Re}} \int_{\Omega} \nabla \underline{u} : \nabla \underline{w} \, \mathrm{d}\Omega - \int_{\Omega} p \nabla \cdot \underline{w} \, \mathrm{d}\Omega = \int_{\Gamma/\Gamma_n} r \, \underline{w} \cdot \underline{n} \, \mathrm{d}\Gamma + \int_{\Gamma/\Gamma_\gamma} \underline{s} \cdot \underline{w} \times \underline{n} \, \mathrm{d}\Gamma, \tag{2.5}$$

$$\int_{\Omega} (\nabla \cdot \underline{u}) \, q \, \mathrm{d}\Omega = 0 \, . \tag{2.6}$$

In (2.5) above, $\Gamma/\Gamma_{n,r}$ denotes the complement of $\Gamma_{n,r}$ in $\Gamma = \partial\Omega$. By definition, $\underline{\phi} \in \Gamma/\Gamma_i (i = n, r)$ implies that $\underline{\phi} \in \Gamma$ but $\underline{\phi} \notin \Gamma_i$. As to $\underline{n}$ and $\underline{s}$, they denote the outward normal and the tangent to Γ, respectively. Corresponding to the present choice of a bilinear form, namely, $\frac{1}{\mathrm{Re}} \int_{\Omega} \nabla \underline{u} : \nabla \underline{w} \, \mathrm{d}\Omega$, shown in equation (2.5), the following natural boundary conditions are invoked in the weighted residuals statement:

$$-p + \frac{1}{\mathrm{Re}} \nabla \underline{u} \cdot \underline{n} = r \qquad \text{on} \quad \Gamma/\Gamma_n, \tag{2.7}$$

$$\frac{1}{\mathrm{Re}} \underline{n} \cdot \nabla \underline{u} \times \underline{n} = \underline{s} \qquad \text{on} \quad \Gamma/\Gamma_r \, . \tag{2.8}$$

These physically irrevelant boundary conditions are suggested to be used in conjunction with the following essential boundary conditions [11]:

$$\underline{u} \cdot \underline{n} = g_n \qquad \text{on} \quad \Gamma_n, \tag{2.9}$$

$$\underline{n} \times \underline{u} \times \underline{n} = g_r \qquad \text{on} \quad \Gamma_r \, . \tag{2.10}$$

In the above weak statement which constituted the basis for modeling incompressible viscous fluid flows, the choice of basis spaces for primitive variables is critical to the success of the mixed finite element formulation. An outstanding feature of the weak statement given in (2.5)–(2.6) is that pressure appears only in the momentum equations and acts as a source term in driving the velocity transport. The lack of a scalar equation for pressure in our primitive-variable working equations is a concern in that there is evidence of checkerboard pressure oscillations as discussed by Gunzburger [11]. Recognizing the dual role of the pressure, both as an enforcer of the continuity constraint and as a force in the mechanical balance law for momentum conservation, shape functions for the velocity vector and the scalar pressure warrant careful consideration if we are to avoid spurious node-to-node pressure oscillations. The guiding principle in the choice of basis spaces for primitive variables to obtain the discrete problem which is free of the pressure mode is the accommodation of the LBB (or *inf-sup*) condition [3, 13]. In this light, finite elements chosen here fall within the uni-variant framework [20]. One way to satisfy the *inf-sup* div-stability condition is to employ triquadratic polynomials, $N^i (i = 1, 27)$, for velocities while using trilinear polynomials, $M^i (i = 1, 9)$, for the pressure

(2.11)
$$N^i = \left(\frac{3}{2}\bar{\xi}^2 + \frac{1}{2}\bar{\xi} + 1 + \xi^2 - \xi_i^2\right)\left(\frac{3}{2}\bar{\eta}^2 + \frac{1}{2}\bar{\eta} + 1 + \eta^2 + \eta_i^2\right)\left(\frac{3}{2}\bar{\zeta}^2 + \frac{1}{2}\bar{\zeta} + 1 + \zeta^2 - \zeta_i^2\right),$$

$$M^i = \frac{1}{8}(1 + \bar{\xi})(1 + \bar{\eta})(1 + \bar{\zeta}), \tag{2.12}$$

where

$$\bar{\zeta} = \zeta\zeta_i, \qquad \bar{\eta} = \eta\eta_i, \qquad \bar{\xi} = \xi\xi_i\ . \tag{2.13}$$

In the above, we denote ξ_i, η_i, and ζ_i as the normalized coordinates for the ith node.

Consideration must now be given to the approximation of spatial derivatives. Prediction of high Reynolds number flows requires careful selection of a test space to enhance the discrete problem. Stability enhancement is achieved through the addition of a biased polynomial to the shape function so that the upwind information is favorably considered. Upwind schemes are, however, prone to numerical contamination due to the introduction of false diffusion errors, which are typical of multidimensional analyses [17]. To avoid this type of prediction error without sacrificing of stability, we have designed a streamline operator so that the stabilized terms are mainly added in the primary flow direction to enhance the discrete stability. As a result, the stiffness matrix equation which has association with the convection term is derived as follows:

$$C^{il} = \left[N^i + \tau\left(N^l \tilde{V}_j^l\right)\right]\frac{\partial N^l}{\partial X_j} N^l \tilde{V}_j^l \frac{\partial N^l}{\partial X_j} + \frac{1}{Re}\frac{\partial N^l}{\partial X_j}\frac{\partial N^l}{\partial X_j}, \tag{2.14}$$

where

$$\tau = \frac{\alpha_\xi V_\xi h_\xi + \alpha_\eta V_\eta h_\eta + \alpha_\zeta V_\zeta h_\zeta}{2V_j V_j}, \tag{2.15}$$

$$\alpha_{Y_i} = \delta\left(V_{Y_i} h_{Y_i} \frac{\mathrm{Re}}{2}\right), \tag{2.16}$$

$$V_{Y_i} = \hat{e}_{Y_i} \cdot \underline{u}. \tag{2.17}$$

The parameter δ in (2.16) determines the degree of upwinding and is the key to success of application of the streamline operator. In the present three-dimensional analysis, formulated on quadratic elements, we define δ as follows:

$$\delta(\gamma) = \begin{cases} \frac{1}{2}\cosh(\frac{\gamma}{2} - \frac{1}{\gamma}) & \text{at center-nodes,} \\ \frac{\cosh(\gamma)-2}{\sinh(\gamma)-4\tanh(\frac{\gamma}{2})} - \frac{1}{\gamma} & \text{at end-nodes.} \end{cases} \tag{2.18}$$

Insofar as the one-dimensional analysis is concerned, the above choice of δ provides nodally exact solutions in quadratic elements.

3. Element-by-element BICGSTAB (EBE-BICGSTAB) iterative solver. The matrix equations formulated here are unsymmetric in form. Furthermore, this matrix equation is intrinsically devoid of real value. This further complicates the calculation of finite element solutions for primitive variables with as many zeros as continuity equations in the diagonal in that eigenvalues become poorly distributed. This poses no problem for the use of a Gaussian elimination direct solver. For very large-size problems, the storage requirement becomes prohibitive in the course of fill-in processes. As a result, computer time and storage requirements greatly exceed the capacity of today's computers. For this reason, there is a strong need to use iterative solvers to circumvent this problem. As noted earlier, there is a trend toward using robust iterative solvers as research moves toward three-dimensional analysis of heat and fluid flows.

It is generally accepted that iteration numbers needed for an iterative method to converge increase dramatically with the increase in the number of grid points. Thus, iterative solvers of the nonstationary type are considered preferable. The reason for using nonstationary solvers instead of their stationary counterparts is that the computations involve information that changes in each iteration. A typical example of a nonstationary iterative method is the conjugate gradient method of Hestenes and Stiefel [12]. The idea behind the construction of nonstationary iterative solvers is to replace the matrix equation problem, $\underline{\underline{A}}\underline{x} = \underline{b}$, with the minimization problem. Let $\underline{x}_0$ denote a starting vector and let $\underline{r}_0 \equiv \underline{b} - \underline{\underline{A}}\underline{x}_0$; the method generates a sequence $\underline{x}_1, \underline{x}_2, \underline{x}_3, \ldots$ given by

$$\underline{x}_i \in \underline{x}_0 + K_{ryi}\Big(\underline{\underline{A}}; \underline{r}_0\Big), \tag{3.1}$$

where $K_{ryi}(\underline{\underline{A}}; \underline{r}_0)$ is known as the Krylov subspace given by

$$K_{ryi}\Big(\underline{\underline{A}}; \underline{r}_0\Big) = \text{span}\Big(\underline{r}_0; \underline{\underline{A}}\underline{r}_0, \ldots, \underline{\underline{A}}^{i-1}\underline{r}_0\Big). \tag{3.2}$$

In the conjugate gradient (CG) method, the search direction vector is chosen optimally to update the iterative $\underline{x}_i$ in each iteration. This is fundamental to the efficiency of the CG method. When $\underline{\underline{A}}$ is not symmetric, we lose the three-term recurrence property for the residual vector $\underline{r}_i \equiv (\underline{b} - \underline{\underline{A}}\underline{x}_{i-1})$. Development of a Krylov subspace method to cope with matrix asymmetry is thus an active field of study, and new methods are still emerging. In the literatures, there exist two classes of nonstationary iterative solvers which are often used to overcome the problem of matrix asymmetry. They are the Chebyshev method [9], which works effectively only

for positive definite matrix equations, and the Krylov subspace methods, which are designed using the idea of orthogonalization. Our review of the iterative methods will focus primarily on the Krylov subspace methods.

The biconjugate gradient (BICG) method [7] is representative of the Krylov subspace methods and is designed to handle the matrix asymmetry. In the BICG method, the approximations are constructed using two three-term recurrence relations for rows $\{\underline{\mathrm{r}}_i\}$ and $\{\hat{\underline{\mathrm{r}}}_i\}$. The residual vector $\underline{\mathrm{r}}_i$ is orthogonal with respect to one row of vectors $\hat{\underline{\mathrm{r}}}_0, \hat{\underline{\mathrm{r}}}_1, \ldots, \hat{\underline{\mathrm{r}}}_{i-1}$, and vice versa, $\hat{\underline{\mathrm{r}}}_j$ is orthogonal with respect to $\underline{\mathrm{r}}_0, \underline{\mathrm{r}}_1, \ldots, \underline{\mathrm{r}}_{i-1}$. As with the CG method, the BICG method terminates in at most n steps. The disadvantage is that there is no minimization property as in CG for the intermediate steps. Other deficiencies inherent in the BICG method are the irregular convergence behavior and the need to perform a transpose operation on the coefficient matrix. To solve these problems, the Lanczos and Arnoldi methods can be used. The generalized minimized residuals (GMRES(k)) method [18] is most often referred to in the class Arnoldi methods. GMRES accommodates a self-orthogonal sequence so that the residuals are minimized optimally. No more than n steps are needed to reach convergence of the solution. It is noted that k is the maximum number of vectors used. While GMRES works effectively in regularizing the convergence behavior, this method suffers from a severe storage requirement. This problem can be circumvented through use of a restart capability programmed in the code.

In the Lanczos category, product methods are featured by having dual orthogonal vector sets. Methods in this class which are often referred to are the conjugate gradient squares (CGS) [21], quasi-minimal residual (QMR) [8], and BICGSTAB [4] methods. They are all regarded as effective in solving matrix asymmetry problems and distinguish themselves in their combination of construction polynomials. The QMR method of Freund and Nachtigal [8] avoids irregular convergence behavior but still suffers from the necessity of transposing the stiffness matrix. In contrast, the CGS method dispenses with the transpose of the matrix equation but inhibits the irregular convergence behavior since the contraction polynomial involved in the CGS method is the same as that in the BICG method [21]. The BICGSTAB method of Van der Vorst [4] was developed within the Lanczos framework to solve unsymmetric matrix equations without sacrificing irregular convergence. The main idea behind choosing BICGSTAB to solve unsymmetric and indefinite matrix equations for incompressible Navier–Stokes equations is rooted in its accommodation of local minimization of residuals through GMRES (1). Another advantage in BICGSTAB is that mathematical manipulation of the transpose matrix is avoided. Nevertheless, there is much work to be done to refine the BICGSTAB iterative solver to avoid pivoting breakdown and Lanczos breakdown because this method still inherits the essence of BICG. In the literature other iterative solvers, such as ORTHOMIN [2], generalized conjugate residual (GCR) [6], and conjugate gradient method on normal residuals (CGNR) [16] are viable remedies for matrix asymmetry.

In the finite element stiffness matrix equations, the profile of nonzeros is crucial to effective use of iterative solvers. With this in mind, our strategy for ordering nodal points and allocating primitive working variables is to avoid unnecessary storage of voids. This helps reduce the bandwidth of the matrix equations and, thus, aid efficient application of the BICGSTAB iterative solver. To achieve the goal of compressing the matrix equations, we invoke the element-by-element capability together with BICGSTAB. The resulting main steps involved in the EBE-BICGSTAB solution algorithm are summarized as follows:

Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ for some initial guess $\mathbf{x}_0$
Choose $\bar{\mathbf{r}}$, such that $(\bar{\mathbf{r}}, \mathbf{r}_0) \neq 0$
For $i = 1, 2, \ldots$
  $\rho_{i-1} = (\bar{\mathbf{r}}, \mathbf{r}_{i-1})$
  if $\rho_{i-1} < \epsilon_1$ [**near break down**]
  if $i = 1$
    $\mathbf{r}_i = \mathbf{r}_{i-1}$
  else
    $\beta_{i-1} = (\rho_{i-1}/\rho_{i-2})(\alpha_{i-1}/\omega_{i-1})$
    $\mathbf{r}_i = \mathbf{r}_{i-1} + \beta_{i-1}(\mathbf{r}_{i-1} - \omega_{i-1}\mathbf{v}_{i-1})$
  endif
  solve $\mathbf{K}\bar{\mathbf{r}} = \mathbf{r}_i$
  $\mathbf{v}_i = \sum_{elem}(\mathbf{A}_{elem}\bar{\mathbf{r}})$ ⟵ element-by-element procedure
  $\alpha_i = \rho_{i-1}/(\bar{\mathbf{r}}, \mathbf{v}_i)$
  if $(\bar{\mathbf{r}}, \mathbf{v}_i) < \epsilon_2$ [**near break down**]
  $\mathbf{s} = \mathbf{r}_{i-1} - \alpha_i\mathbf{v}_i$
  solve $\mathbf{K}\bar{\mathbf{s}} = \mathbf{s}$
  $\mathbf{t} = \sum_{elem}(\mathbf{A}_{elem}\bar{\mathbf{s}})$ ⟵ element-by-element procedure
  if $\|\mathbf{s}\|_2 < \epsilon$
    $\omega_i = 0$
  else
    $\omega_i = (\mathbf{t}, \mathbf{s})/(\mathbf{t}, \mathbf{t})$
  endif
  $\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_i\bar{\mathbf{r}}_i + \omega_i\bar{\mathbf{s}}$
  $\mathbf{r}_i = \mathbf{s} - \omega_i\mathbf{t}$
  check convergence; continue if necessary ($\omega_i \neq 0$)
End

The present investigation attempts to resolve matrix asymmetry and indefiniteness with no specific emphasis placed on the preconditioner, although we realized that preconditioning the present matrix equation is an essential prerequisite of the three-dimensional computation. However, there is still a lack of detailed knowledge about the appropriate way to choose a preconditioner. As a result, we did not take this theoretically challenging subject into the present consideration. Further studies are necessary to accelerate the convergence through introduction of a suitable preconditioner.

4. Parallel implementation of finite element code. The motivation for this study was derived from the need for efficient implementation of the finite element code so far developed for three-dimensional flow simulations. Parallel computation has been proposed as a way to gain substantial improvement in computational throughput in solving large-scaled problems in general and fluid dynamics problems, especially on parallel architectures. To achieve this goal, the computer code has to be adapted to enable execution in a multiprocessor computational environment. There exist several platforms which take advantage of concurrent processing. The parallel architectures now range from desktop workstations with few processors to massively parallel machines.

Owing to the speed-up potential offered, the use of multiprocessor architectures in which many computer processors work simultaneously on a flow problem has become an integral part of our code development. As a first step toward increasing

TABLE 1
Features of machines CRAY C-90 *and* J-90.

Machine	J-90	C-90
CPU Speed (MFloaps)	200	1000
Memory Bandwidth (Gbs/s)	25.6	250*
I/O Bandwidth (Gbs/s)	6.4	13.6
CPU Clock Speed (ns)	10	4.2
Maximal no. of CPU	32	16

* Peak value

the execution speed of a computer program, it is best to identify which portions of the code use the majority of the execution time. In the programming stage, we can then pay more attention to those parts of the code where programming will have the greatest impact on computing performance. To this end, we use utility tools, such as FLOWTRACE, PERFTRAC, and PROF, which are installed on a CRAY UNICOS system [5]. Through such performance analysis on our incompressible Navier–Stokes code, it has been found that the iterative solver used to obtain solutions from unsymmetric and indefinite matrix equations constitutes the most computationally intensive part of the code. In this regard, much work has been expended on the computationally intensive EBE-BICGSTAB iterative solver in order to optimize the performance of the analysis code.

Presented here are results of our preliminary parallel studies, which were carried out on a sixteen-processor CRAY C-90 system. Each processor of the C-90 had a 4,167 nanosecond cycle time and dual vector pipes. The chaining of addition and multiplication was allowed in each processor, thus delivering a 15.36 Gflop peak performance rate. On the C-90, vectorization of the code was automatically done by a compiler, named "cft77," which was installed on the standard CRAY UNIONS system. For parallel compilation, we have the options of microtasking and autotasking. Of those, microtasking is much simpler to use because it uses directives rather than cumbersome synchronization subroutine calls in the Fortran-callable library. Beyond this, microtasking uses special parallel features of the CRAY C-90. Much improved overall speed-up results are produced as compared with macrotasking. A further advantage can be gained through use of autotasking, which places an enriched set of microtasking directives in the original Fortran code. Based on the above, a combination of autotasking and microtasking was used here to achieve better performance in the numerical simulation of incompressible flow problem. For comparison purposes, analyses were also carried out on another CRAY platform named "J-90." The features of J-90 and C-90 are summarized in Table 1 for the reader's reference.

5. Computed results. In our previous article [22] the justification for applying the iterative solver BICGSTAB to incompressible Navier–Stokes codes was presented. It was reported here that very good convergence was obtained. The results compare favorably with the analytic solutions. In a cube which is covered with 8^3 uniformly distributed quadratic elements, more than 20% of the CPU time can be saved by using the presently employed BICGSTAB iterative solver, as compared with the frontal direct solver. When the problem size increases to 16^3, the analysis is no longer amenable to the frontal solver due to the prohibitive size of the matrix equations. This study has shown that the demand for finer resolution continues to increase the need to carry out calculations using iterative solvers.

We can now focus our attention on the performance gain in the parallel imple-

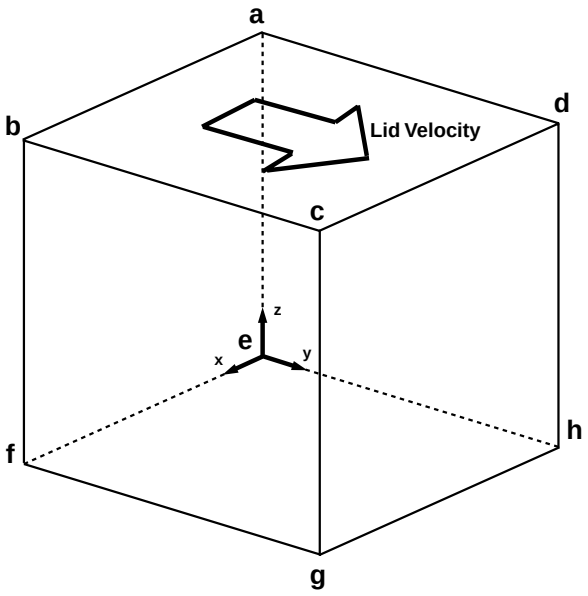

FIG. 1. *Illustration of the three-dimensional lid-driven cavity problem.*

mentation of the finite element code described in this study. In the current study, we benchmarked the parallelized BICGSTAB iterative solver by considering a well-documented lid-driven cavity flow problem. Analyses were carried out in a cube schematically in Figure 1. For the Reynolds number under consideration, Re = 400 is small enough to keep the flow laminar. Prior to discussion of the results in a unit square driven by a roof lid, it is instructive to note that the present steady-state finite element code has been 86% parallelized on a Cray C-90 platform which has 16 CPUs, as shown in Figure 2. A fixed point iteration is used in the linearization of differential equations. The convergence criteria used for the inner iteration is 10^{-1} and 10^{-6} for the outer iteration. Figure 3 presents the computed velocity profiles along the vertical and horizontal center lines on the mid-plane of the cavity. The computed solutions on the parallel platform show close agreement with those of Babu and Korpela [1]. This verifies the utility of the BICGSTAB iterative solver implemented in a parallel environment.

Advances in computational environment have made three-dimensional finite element flow analyses feasible in our computational fluid dynamics laboratory. This poses a serious threat to us to provide, in depth, the flow structure inferred from an enormous amount of three-dimensional data. To explore into the flow details we adopted a theoretically rigorous theory of topology. Of vector fields which can be chosen in the topological study of three-dimensional data, we considered limiting streamlines which are, by definition, streamlines passing immediately above the body surface [15]. Since the kinematic nature of limiting streamlines are best described topologically by singular points, in Figure 4 we plot nodes, foci, and saddles on the no-slip walls of the cavity. Also given in Figure 4 are half-nodes which arise at the intersection of the plane with the cavity. Thanks to the lines of separation, the flow structure pertinent to the flow can be faithfully depicted. Although the problem is very simple in geometry, the flow structure is by no means simple in nature; especially noteworthy is the prediction of a three-dimensional saddle in the interior of the cavity,

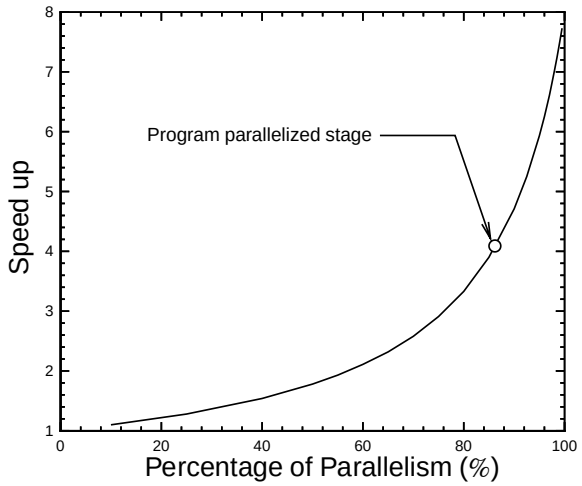

FIG. 2. *Speed-up against percentage parallelism.*

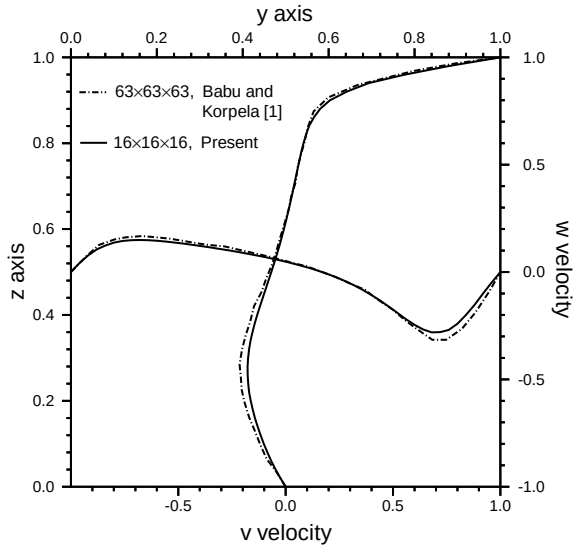

FIG. 3. *Comparison of velocity profiles along the vertical and horizontal centerlines on the symmetry plane,* $x = 0.5$, *at* $\mathrm{Re} = 400$.

as is shown in Figure 5. The vortical flow structure also manifests itself by presence of two vortical core lines shown in Figure 6. By definition, the vortical core line is a collection of three-dimensional foci. They emanate from the two end walls *bfgc* and *aehd* and proceed toward the symmetry plane and, finally, meet at the saddle point. For particle tracers adjacent to the vortical core line, they wrap the vortical core line and spiral towards the plane of symmetry.

There remains a discussion on how much computational efficiency can be gained from parallel implementation of the finite element code. To this end, we benchmarked the parallel performance of the CRAY C-90 and CRAY J-90 supercomputers for the lid-driven cavity flow problems conducted on three refined elements, namely, 4^3, 8^3,

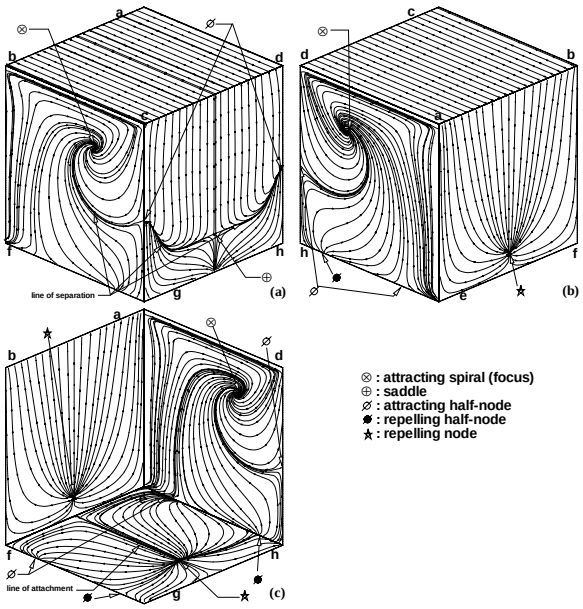

FIG. 4. *Three-dimensional plot of limiting streamlines from different views.* (a) *abcd-bfgc-cghd planes;* (b) *cdab-dhea-aefb planes;* (c) *abfe-efgh-aehd planes.*

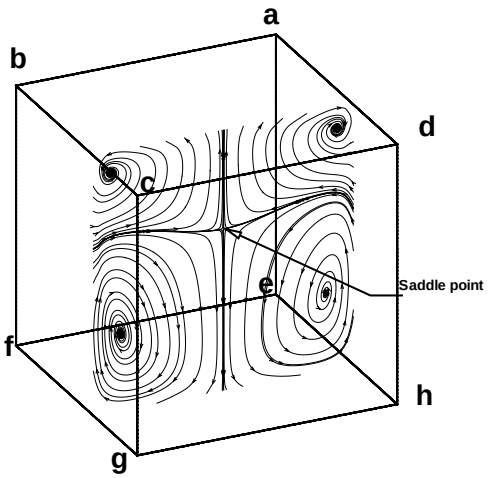

FIG. 5. *Secondary flow pattern and vortical flow structure in the cavity.*

and 16^3. The performance was measured by means of the CPU time used against the number of processors. Included in Table 2 are also the memory sizes (ms) for the reader's reference. We summarize the parallel performance results in Table 2, from which the corresponding speed-up can be obtained, as shown in Figure 7. It is noted that as the problem size increased to 16^3, C-90 outperformed J-90 in the percentage of parallelism.

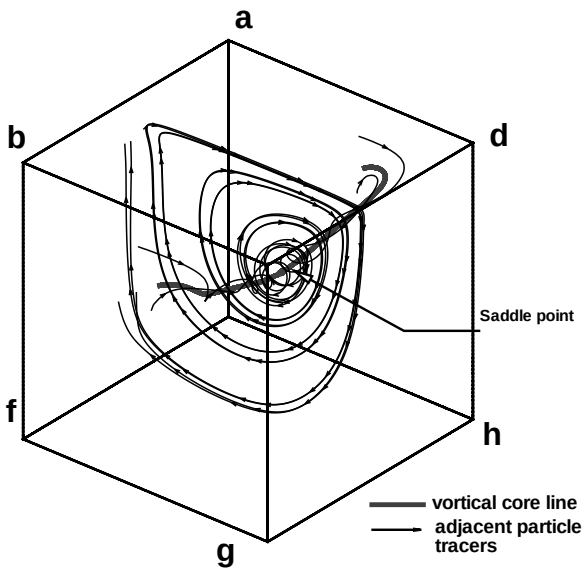

FIG. 6. *An illustration of a three-dimensional saddle point and the adjacent streamlines*

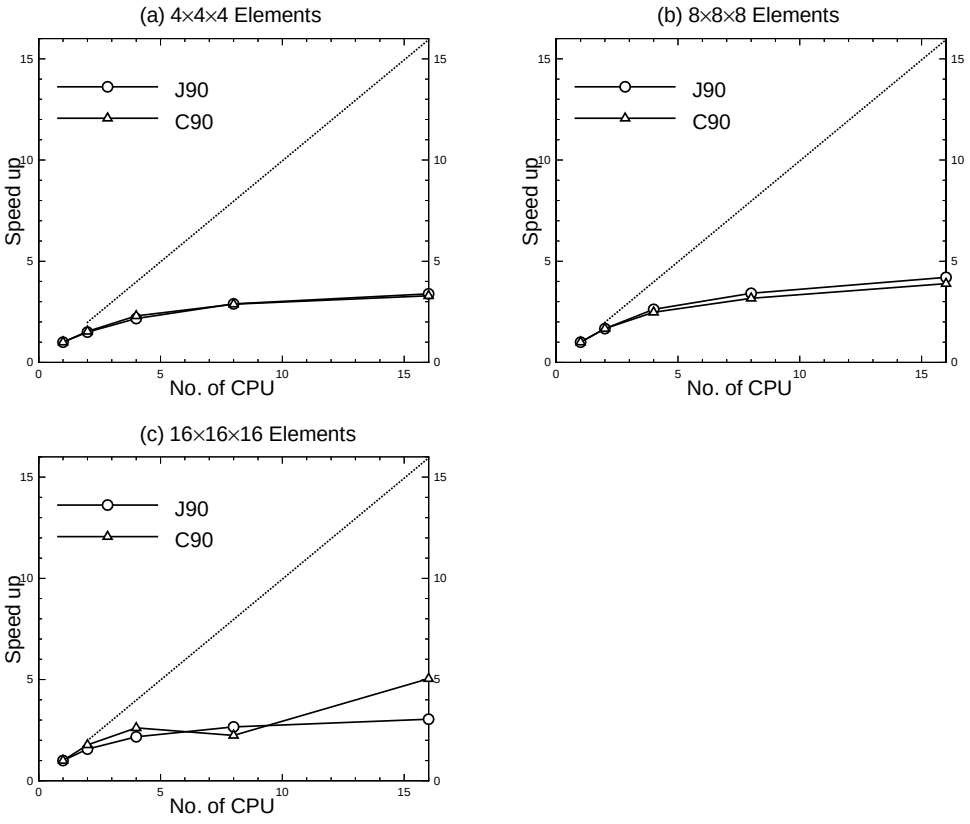

FIG. 7. *Speed-up for problems conducted on different meshes using the parallelized finite element code.* (a) $4 \times 4 \times 4$ *elements;* (b) $8 \times 8 \times 8$ *elements;* (c) $16 \times 16 \times 16$ *elements.*

TABLE 2

Program test on multi-CPU; ne: number of elements, ms: memory size in megawords, ni: number of outer/inner iterations.

(a) CPU times and memory size on J-90

Ne	No. of CPU					Ms	Ni
	1	2	4	8	16		
4^3	78	52	36	27	(23)	1.2	22/3210
8^3	986	591	377	288	(235)	7.2	24/7890
16^3	10639	(6800)	(4900)	(4000)	(3500)	52.8	38/10036

(b) CPU times and memory size on C-90

Ne	No. of CPU					Ms	Ni
	1	2	4	8	16		
4^3	23	15	10	8	(7)	1.2	22/3620
8^3	241	144	97	76	(62)	7.2	31/6039
16^3	3753	2124	1346	947	(745)	52.8	67/11475

* CPU time is measured in second

† Estimates are in parentheses "( )"

6. Concluding remark. We have presented parallel computation of steady-state Navier–Stokes equations. The fluid flow under investigation has the incompressibility property. The results presented here are based on the streamline upwind formulation. It is addressed that the finite element model is implemented on quadratic elements using combinations of interpolants for velocity and pressure variables which accommodate the *inf-sup* div-stability condition. This finite element model is effective in suppressing numerical oscillations under high Reynolds number conditions and has the ability to avoid crosswind diffusion errors. The finite element formulation gives rise to a system of coupled, nonlinear equations, which fall within the unsymmetric and indefinite context. Due to the prohibitive size of the algebraic system in the present three-dimensional simulations, it is advisable to compute solutions using an iterative solver which can resolve matrix asymmetry and indefiniteness. We have used the BICGSTAB update technique to achieve this goal and have implemented it in an element-by-element format to improve the computational performance. It is the computationally intensive nature of solving a large set of indefinite and unsymmetric matrix equations which has motivated us to further increase the execution speed of programs. One solution is the use of multiprocessor architectures because of the speed-up potential offered. In this study, we have focused on the parallel performance for code run on two parallel platforms, CRAY J-90 and C-90, using the three-dimensional lid-driven problem to benchmark the parallel performance.

Acknowledgments. The authors are indebted to C. M. Grassl, CRAY Research Inc., Mountain View, CA, and S. T. Tang and C. H. Lee, who provided facilities as well as useful consultations during the course of this study.

REFERENCES

[1] V. BABU AND S. A. KORPELA, *Numerical solution of the incompressible three-dimensional Navier–Stokes equations*, Comput. and Fluids, 23 (1994), pp. 675–691.

[2] A. BEHIE AND P. A. FORSYTH, *Comparison of fast iterative methods for symmetric system*, IMA J. Numer. Anal., 3 (1983), pp. 41–63.

[3] F. Brezzi and J. Douglas, *Stabilized mixed methods for the Stokes problem*, Numer. Math., 53 (1988), pp. 225–235.
[4] H. A. van der Vorst, *BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.
[5] K. Dowd, *High Performance Computing*, O'Reily, Sebastopol, CA, 1993.
[6] S. C. Eisenstat, H. C. Elman, and M. H. Schultz, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20 (1983), pp. 345–357.
[7] R. Fletcher, *Conjugate gradient methods for indefinite systems*, in Numerical Analysis, Lecture Notes in Math., Springer, Berlin, 1976, pp. 73–89.
[8] R. Freund and N. Nachtigal, *QMR: A quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.
[9] G. H. Golub and C. F. van Loan, *Matrix Computations, 2nd ed.*, The Johns Hopkins University Press, Baltimore, 1989.
[10] P. M. Gresho and R. L. Sani, *On pressure boundary conditions for the incompressible Navier–Stokes equations*, Int. J. Numer. Methods Fluids, 7 (1987), pp. 1111–1145.
[11] M. D. Gunzburger, *Finite Element Methods for Viscous Incompressible Flows, A Guide to Theory, Practice, and Algorithms*, Academic Press, New York, 1989.
[12] M. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436.
[13] I. Babuška, *Error bounds for finite element methods*, Numer. Math., 16 (1971), pp. 322–333.
[14] O. Ladyzhenskaya, *The Mathematical Theory of Viscous Incompressible Flow*, Gordon and Breach, New York, 1969.
[15] R. Legendre, *Séparation de courant Íéconlment laminaire tridimensional*, Rech. Aérospat., 54 (1956), pp. 3–8.
[16] N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl, 13 (1992), pp. 778–795.
[17] S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, Hemisphere, Washington, D.C., 1980.
[18] Y. Saad and M. H. Schultz, *GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear system*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–569.
[19] T. W. H. Sheu, S. F. Tsai, and M. M. T. Wang, *A Petrov-Galerkin formulation for incompressible flows at high Reynolds numbers*, J. Comput. Fluid Dynamics, 5 (1995), pp. 213–230.
[20] T. W. H. Sheu and M. M. T. Wang, *A comparison study on multivariant and univariant finite elements for three dimensional incompressible viscous flows*, Int. J. Numer. Methods Fluids, 21 (1995), pp. 683–696.
[21] P. Sonneveld, *CGS, A fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.
[22] M. M. T. Wang and T. W. H. Sheu, *An element-by-element BICGSTAB iterative method for three-dimensional steady Navier–Stokes equations*, J. Comput. Appl. Math., 79 (1997), pp. 147–165.