INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2002; **38**:1177–1198 (DOI: 10.1002/fld.218)

Dealing with pressure: FEM solution strategies for the pressure in the time-dependent Navier–Stokes equations

Mark A. Christon*

Computational Physics R&D Department, Sandia National Laboratories, M/S 0819, P.O. Box 5800, Albuquerque, New Mexico 87185-0819, U.S.A.

SUMMARY

This paper presents a survey of solution methods for calculating the pressure in the time-dependent Navier–Stokes equations. The primary focus is on the treatment of the pressure-Poisson equation deriving from index-1 DAE formulations of the Navier–Stokes equations. Based on extensive operational experience with a variety of solution strategies, the combination of a stabilized pressure-Poisson operator with an A-conjugate projection and SSOR preconditioned conjugate gradient method has been found to yield the overall best performance relative to the resolve cost of a high performance direct solver. Copyright © 2002 John Wiley & Sons, Ltd.

KEY WORDS: Navier–Stokes; pressure-Poisson; stabilization; projection method; approximate projection; A-conjugate projection

1. INTRODUCTION

Incompressible flow is one of the most frequently encountered flow regimes encompassing problems that range from atmospheric dispersal to food processing, aerodynamic design of automobiles, and manufacturing processes such as chemical vapor deposition, mold filling and casting. The need for scalable time-accurate solution algorithms is growing due to the emerging use of large-eddy simulation (LES) and time-dependent Reynolds averaged Navier–Stokes (RANS) calculations in engineering applications.

In large-eddy simulation, a significant fraction of the energy spectrum must be resolved in order to apply sub-grid scale models that are based on a balance between the production and dissipation of turbulent kinetic energy. Thus, high resolution grids are required even for flows with moderate Reynolds numbers. In addition to the high degree of spatial discretization, the temporal resolution for LES is also demanding, ultimately requiring effective mapping of flow-solution algorithms to massively parallel supercomputer architectures. Although the spatial resolution required for RANS calculations is somewhat relaxed relative

*Correspondence to: M. Christon, Computational Physics R&D Dept., Sandia National Laboratories, M/S 0819, P.O. Box 5800, Albuquerque, NM 87185-0819, U.S.A.

Received 15 November 1999
Revised 24 May 2001

to LES, the time-accurate treatment of the Reynolds averaged Navier–Stokes equations also demands computationally efficient flow solution algorithms.

2. FORMULATION

A brief review of the incompressible Navier–Stokes formulation and projection methods is presented before proceeding with a description of the time-integration methods. To begin, given a flow domain Ω with boundary $\Gamma = \Gamma_1 \cup \Gamma_2$ the incompressible Navier–Stokes equations are

$$\nabla \cdot \mathbf{u} = 0, \tag{1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla P + \nu \nabla^2 \mathbf{u} + \mathbf{f}, \tag{2}$$

where $\mathbf{u} = (u, v, w)$ is the velocity, ν is the kinematic viscosity, $\mathbf{f}$ is the body force, p is the pressure, ρ is the mass density, and $P = p/\rho$.

The system of equations above are subject to boundary conditions that consist of a specified velocity on Γ_1 as in Equation (3), or pseudo-traction boundary conditions on Γ_2 as in Equations (4) and (5).

$$\mathbf{u} = \hat{\mathbf{u}} \quad \text{on } \Gamma_1 \tag{3}$$

$$-P + \nu \frac{\partial \mathbf{u}}{\partial n} = f_n \quad \text{on } \Gamma_2 \tag{4}$$

$$\nu \frac{\partial \mathbf{u}}{\partial \tau} = f_\tau \quad \text{on } \Gamma_2 \tag{5}$$

Here, $\partial \mathbf{u}/\partial n$ and $\partial \mathbf{u}/\partial \tau$ represent the derivative of $\mathbf{u}$ in the normal (n) and tangential (τ) directions respectively. Similarly, f_n and f_τ represent the normal and tangential components of the boundary traction. Homogeneous traction boundary conditions correspond to the well known natural boundary conditions that are typically applied at outflow boundaries.

In addition to the boundary conditions, initial conditions, $\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}^0(\mathbf{x})$, are required. For a well-posed time-dependent incompressible flow problem, the prescribed initial velocity field must satisfy

$$\nabla \cdot \mathbf{u}^0 = 0 \quad \text{in } \Omega, \tag{6}$$

and

$$\mathbf{n} \cdot \mathbf{u}(\mathbf{x}, 0) = \mathbf{n} \cdot \mathbf{u}^o(\mathbf{x}) \quad \text{on } \Gamma. \tag{7}$$

If $\Gamma_2 = 0$, e.g., for enclosure flows with $\mathbf{n} \cdot \mathbf{u}$ prescribed on all surfaces, then global mass conservation enters as an additional solvability constraint as

$$\int_\Gamma \mathbf{n} \cdot \mathbf{u}^o \, d\Gamma = 0. \tag{8}$$

The methods for obtaining the weak-form of the conservation equations are well known and will not be repeated here (see for example, Gresho and Sani [1]). The spatially discrete forms of Equations (1) and (2) are

$$C^T\mathbf{u} = 0, \tag{9}$$

and

$$M\dot{\mathbf{u}} + A(\mathbf{u})\mathbf{u} + K\mathbf{u} + CP = \mathbf{F}, \tag{10}$$

where M is the unit mass matrix, $A(\mathbf{u})$ and K are the advection and the viscous diffusion operators respectively, and $\mathbf{F}$ is the body force. C is the gradient operator, and C^T is the divergence operator. In the subsequent sections, a row-sum lumped, i.e., diagonal, mass matrix, M_L, will be used as well. In order to simplify the nomenclature, $\mathbf{u}$ and P are understood to be discrete approximations to the continuous velocity, and pressure.

FEM projection method

A detailed review of projection methods is beyond the scope of this paper, but a partial list of relevant work is provided. Projection methods, also commonly referred to as fractional-step, pressure correction methods, or Chorin's method [2], have grown in popularity over the past 10 years due to the relative ease of implementation and computational performance. This is reflected by the volume of work published on the development of second-order accurate projection methods, see for example References [3–20].

The philosophy behind projection algorithms is to attempt to provide a legitimate way to decouple the pressure and velocity fields in the hope of providing an efficient computational method for transient, incompressible flow simulations. Exact projection methods decouple the solution of the velocity and pressure fields by first computing an approximate velocity field that is not div-free, and then performing an L^2 projection onto a div-free subspace. The *optimal* Projection-2 (P2) method identified by Gresho [5, 6] forms the starting point for a discussion of the pressure solution algorithms.

Algorithm 1. *Projection-2 (P2)*

1. Given a div-free velocity, $\mathbf{u}^n$, and its corresponding pressure field, P^n, solve

$$[M + \Delta t\theta\hat{K}]\tilde{\mathbf{u}}^{n+1} = [M - \Delta t(1-\theta)\hat{K}]\mathbf{u}^n + \Delta t\{\theta\mathbf{F}^{n+1} + (1-\theta)\mathbf{F}^n - A(\mathbf{u}^n)\mathbf{u}^n - MM_L^{-1}CP^n\} \tag{11}$$

   for an approximate velocity field at time t^{n+1}.
2. Using the approximate velocity, $\tilde{\mathbf{u}}^{n+1}$, solve the global PPE problem for the Lagrange multiplier, λ.

$$[C^TM_L^{-1}C]\lambda = C^T\tilde{\mathbf{u}}^{n+1}. \tag{12}$$

3. Perform the projection step to obtain the final div-free velocity field, $\mathbf{u}^{n+1}$.

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}}^{n+1} - M_L^{-1}C\lambda. \tag{13}$$

4. Update the pressure at time t^{n+1} via

$$P^{n+1} = P^n + \frac{2}{\Delta t}\lambda, \tag{14}$$

5. Repeat steps 1–4 until a maximum simulation time limit or maximum number of time steps is reached.

Remark

1. In Equation (11), $\hat{K}$ represents the usual viscous operator obtained in the weak formulation augmented by balancing tensor diffusivity (BTD) that derives from the second-order, explicit time integrator applied to the advective terms. See Gresho *et al.* [21] or Christon [22] for additional details on BTD. Typically $\theta = 1/2$ is chosen corresponding to a second-order trapezoid method applied to the viscous terms. The inclusion of the BTD contributions in $\hat{K}$ permits stable computations for *CFL* numbers from 5 to 10.
2. Equation (12) is the consistent, discrete form of the pressure-Poisson equation (PPE) for the projection-2 algorithm. The consistent PPE incorporates the effect of the essential velocity boundary conditions from Equation (3), and automatically builds in the boundary conditions from Equations (3) and (4)—see Gresho *et al.* [6]. It represents an algebraic system of equations that is solved for the element-centered Lagrange multiplier during the time-marching procedure.

Turning attention to the explicit time integration algorithm, it is assumed that the explicit algorithm begins with a given velocity field, $\mathbf{u}^0$, and the associated initial pressure field, P^0. For a well-posed initial-boundary value problem, the initial velocity field must be div-free and satisfy the essential boundary conditions.

Algorithm 2. *Explicit Time Integration*

1. Calculate the partial acceleration, i.e., the acceleration neglecting the pressure gradient, at time t^n.

$$\tilde{\mathbf{a}}^n = M_L^{-1}\tilde{\mathbf{F}}^n \tag{15}$$

where

$$\tilde{\mathbf{F}}^n = \mathbf{F}^n - \hat{K}\mathbf{u}^n - A(\mathbf{u}^n)\mathbf{u}^n \tag{16}$$

2. Solve the global PPE for the current pressure field.

$$[C^T M_L^{-1} C]P^n = C^T\tilde{\mathbf{a}}^n \tag{17}$$

3. Update the nodal velocities.

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t\{\tilde{\mathbf{a}}^n - M_L^{-1}CP^n\} \tag{18}$$

4. Repeat steps 1–3 until a maximum simulation time limit or maximum number of time steps is reached.

The saddle point problems

The PPE problem in the projection-2 algorithm, Algorithm (1), arises due to the use of a Helmholtz decomposition of the velocity into div-free and curl-free components. In a finite element context, the Helmholtz velocity decomposition with the concomitant div-free constraint, $C^T\mathbf{u}^{n+1}=0$, is written as

$$\begin{bmatrix} M_L & C \\ C^T & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{u}^{n+1} \\ \lambda \end{Bmatrix} = \begin{Bmatrix} M_L\tilde{\mathbf{u}}^{n+1} \\ 0 \end{Bmatrix}, \tag{19}$$

where, in general, $C^T\tilde{\mathbf{u}}^{n+1}\neq 0$.

The Schur complement of the *projection* operator in Equation (19) yields the consistent pressure-Poisson equation (PPE) in Algorithm (2). In the ensuing discussion, the term *projection* operator is used loosely to describe Equation (19). However, the *projection* operator is not to be confused with the real discrete projection operators, $\mathscr{P}^h = I - M_L^{-1}C[C^TM_L^{-1}C]^{-1}C^T$ and $\mathscr{Q}^h = I - \mathscr{P}^h$ (see Gresho and Chan [6]).

In the explicit time integration algorithm, Algorithm (2), a saddle-point problem analogous to Equation (19) may also be formulated.

$$\begin{bmatrix} M_L & C \\ C^T & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{a}^n \\ P^n \end{Bmatrix} = \begin{Bmatrix} M_L\tilde{\mathbf{a}}^n \\ 0 \end{Bmatrix} \tag{20}$$

In this case, the acceleration and pressure are used rather than velocity and pressure increment highlighting one of the primary differences between the projection-2 and the explicit methods. The explicit method segregates acceleration and pressure which can be done legitimately while the projection-2 method attempts to decouple velocity and pressure. Similar to the projection-2 method, the pressure-Poisson problem in Equation (17) is simply the Schur complement of Equation (20). For both algorithms, the PPE and the saddle-point operators are symmetric and in general singular due in part to the fact that the pressure is known only to within a constant representing the hydrostatic pressure.

3. PRESSURE SOLUTION METHODS

In this section, a survey of solution methods for calculating the pressure (Lagrange multiplier) in the explicit/projection-2 algorithms is presented. The methods used include algebraic multi-grid, a saddle-point conjugate gradient algorithm, and stabilized PPE operators with an A-conjugate projection technique combined with the preconditioned conjugate gradient method. A short review of the less successful approaches is presented first and followed by a presentation of the Q1Q0 stabilization methods and the A-conjugate projection CG method.

Before addressing the solution methods, the pressure-Poisson equation possesses several attributes that are worth noting. First, the PPE, $[C^TM_L^{-1}C]$, is symmetric ($[C^TM_L^{-1}C]_{ij} = [C^TM_L^{-1}C]_{ji}$ regardless of the Reynolds number. The PPE can be singular, i.e., the rows of the matrix sum to zero admitting a hydrostatic pressure mode, although the presence of homogeneous natural boundary conditions weakly sets the hydrostatic pressure to zero. Kim and Ro [23] have noted that for the pressure-correction equation the source data ultimately controls the convergence of the solution method. Similar behavior is exhibited by the PPE

where the right-hand-side data is based on either $C^T\tilde{\mathbf{u}}^{n+1}$ or $C^T\tilde{\mathbf{a}}^n$. In practice, the PPE is much more difficult to solve with iterative techniques than the momentum or scalar transport equations because the right-hand-side varies spatially and is relatively noisy, i.e., has a large spectral content. In addition, the right-hand-side data tends to change rapidly from time-step to time-step making it problematic to use the previous pressure solution to initialize iterative solvers – one of the primary motivations for using the projection CG method.

Q1Q0 stabilization

Although the $Q1Q0$ element has been condemned by theoreticians for its weakly singular modes, this element has long been the workhorse for incompressible flow and continues to be widely used. The unstable modes of the $Q1Q0$ element for incompressible flow have been investigated by Sani *et al.* [24, 25] and more recently by Griffiths and Silvester [26]. Griffiths and Silvester demonstrate that for problems of physical interest, the $Q1Q0$ element will converge to the true solution in the limit as $h \to 0$. In addition, a new convergence proof for the $Q1Q0$ element may be found in Gresho and Sani [1].

The focus here is on pressure stabilization techniques that are compatible with the $Q1Q0$ element and that circumvent the usual Babuška–Brezzi div-stability condition. The global jump stabilization (first proposed by Hughes and Franca [27]) and the local jump stabilization techniques of Silvester and his co-workers [28–30] are applied to the PPE problem. In effect, the jump stabilization techniques provide an *a priori* filter for the weakly unstable pressure modes associated with the $Q1Q0$ element.

The stabilized $Q1Q0$ element yields a regularized saddle-point problem for the projection method,

$$\begin{bmatrix} M_L & C \\ C^T & -S \end{bmatrix} \begin{Bmatrix} \mathbf{u}^{n+1} \\ \lambda \end{Bmatrix} = \begin{Bmatrix} M_L\tilde{\mathbf{u}}^{n+1} \\ 0 \end{Bmatrix}, \tag{21}$$

where S is a symmetric positive semi-definite matrix. Here, pressure stabilization results in an approximate projection method since the concomitant stabilized PPE is no longer constructed using only the discrete div and grad operators. The Schur complement of Equation (21), i.e., the stabilized PPE, is

$$[C^T M_L^{-1} C + S]\lambda = C^T\tilde{\mathbf{u}}^{n+1} \tag{22}$$

where S remains to be defined for the global and local jump stabilization techniques.

The global jump stabilization formulation attempts to control the jump in pressure across element boundaries, and results in a PPE that is perturbed by a 'pressure-diffusion' operator. The off-diagonal entries in the global jump stabilization matrix are defined as

$$S_{IJ} = \beta \frac{|[C^T M_L^{-1} C]_{IJ}|}{\Gamma_{IJ}} \int_{\Gamma_{IJ}} [\![\psi_I]\!][\![\psi_J]\!]\,\mathrm{d}\Gamma, \tag{23}$$

where I and J identify adjacent elements that share a common face as shown in Figure 1(a). Here, Γ_{IJ} represents the shared inter-element boundary, $[\![\cdot]\!]$ is the jump operator, and β is a non-dimensional scaling parameter. For the $Q1Q0$ element, the pressure approximation is piecewise constant with $\psi_I = 1$ inside Ω_e and zero outside. In two dimensions, Γ_{IJ} represents the length of the element edge shared by element I and J, and in three dimensions, it represents the area of

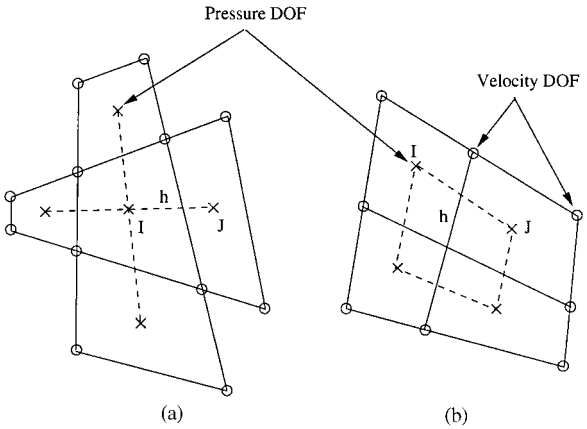

Figure 1. Element configuration for pressure stabilization: (a) global jump; (b) local jump.

the shared face. The inclusion of the PPE term in Equation (23) yields proper dimensionality of the stabilization matrix, accounts for scaling due to irregular elements, and still preserves the symmetry of the original PPE. The diagonal contributions are calculated as

$$S_{II} = \beta \sum_{J} \frac{|[C^T M_L^{-1} C]_{IJ}|}{\Gamma_{IJ}} \int_{\Gamma_{IJ}} [\![\psi_I]\!][\![\psi_J]\!] \, \mathrm{d}\Gamma, \tag{24}$$

where the sum on J indicates a summation over all shared faces of element I. The global jump stabilization is simple to implement when viewed from a linear algebra point of view since the contributions to S are simply augmented row-entries from the original PPE. For alternative scaling procedures, see Chan and Sugiyama [31].

In contrast to the global method, the local stabilization procedure relies on the construction of macro-elements that contain at least one velocity node per edge of the macro element in two dimensions and one velocity node per face in three dimensions. For local pressure stabilization, the entries of the stabilization matrix are calculated according to Equations (23) and (24) but only for the faces shared with elements in the same macro-element as shown in Figure 1(b). In order to use the local jump stabilization formulation, a pre-processing step that identifies the macro-elements is required.

Since similar scaling is used for both the local and global jump stabilization, the only remaining parameter to be determined is β. From the inherited scaling of the stabilization matrix, $\beta = 0$ provides the limit where no-stabilization is applied. For $\beta = 1$, the stabilization matrix will have entries of the same order as the original PPE. In the context of Stokes flow, Norburn and Silvester [30] bounded β by computing the extremal eigenvalues for the stabilized Schur complement as a function of β. They found that $0.01 \leqslant \beta \leqslant 0.1$ minimizes the distance between the extremal eigenvalues. Computational tests have shown that, in general, values of β in this range yield acceptable results for the stabilized PPE as well.

Figure 2 shows the variation of the iteration count for SSOR preconditioned conjugate gradient as a function of β for a 2-D and 3-D flow past a circular cylinder using both local and global jump stabilization. This plot shows that for $\beta \geqslant 0.05$, there is very little variation in the iteration count with respect to β. In addition, the benefit of stabilization

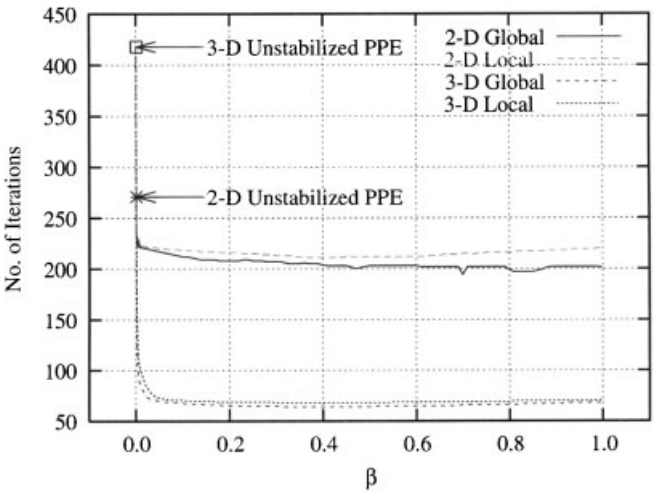

Figure 2. Variation of iteration count vs stabilization parameter for a 2-D $Re = 100$ vortex shedding mesh ($Nel = 11\,200$) and 3-D $Re = 100$ flow past a cylinder ($Nel = 22\,380$).

appears to be considerably larger for 3-D, a factor of six reduction in iteration count for 3-D compared to a factor of about 1.3 for the 2-D case. The marked decrease in iteration count in 3-D is undoubtedly a consequence of the increased size of the null-space for the PPE in three dimensions. Operational experience has shown that $0.01 \leqslant \beta \leqslant 0.1$ works well for most applications while $\beta \geqslant 0.25$ can often yield inaccurate velocity fields.

Figure 3 shows typical convergence histories for the unstabilized and stabilized PPE – here $\beta = 0.05$ with convergence criteria of $\|b - Ax\|/\|b\| \leqslant 10^{-12}$ and $\|x^n - x^{n-1}\|/\|x^n\| \leqslant 10^{-12}$ used for all computations. In two dimensions, the stabilization provides a modest 20% reduction in the iteration count and concomitant computational work. In contrast, the three-dimensional calculations show that the cost of solving the unstabilized PPE yields a method that cannot be scaled to large problem sizes. As demonstrated above, the benefits of stabilization are considerably greater in three-dimensions where a factor 7 to 10 reduction in iteration count is observed for this problem. Since the computational cost of the stabilization occurs only during initialization, the reduction in iteration count translates directly into a reduction in CPU time. The effect of the jump stabilization formulations on the convergence rate for ICCG(0) has been presented in Gresho and Sani [1, *p*.550] although the observed reduction in iteration count was more modest. In both 2-D and 3-D, the gain due to stabilization is considerably more modest when the convergence criteria is relaxed, e.g., $\varepsilon = 10^{-5}$ is acceptable for production computations.

The saddle-point solver

The possibility of solving a saddle-point problem in the context of either the projection-2 or the explicit time integration algorithms was outlined in Section 2. The saddle-point problem in Equation (19) suggests the application of an Uzawa iteration (see for example Fortin and Glowinski [32]). This is not a new idea for the projection method – an early example of an

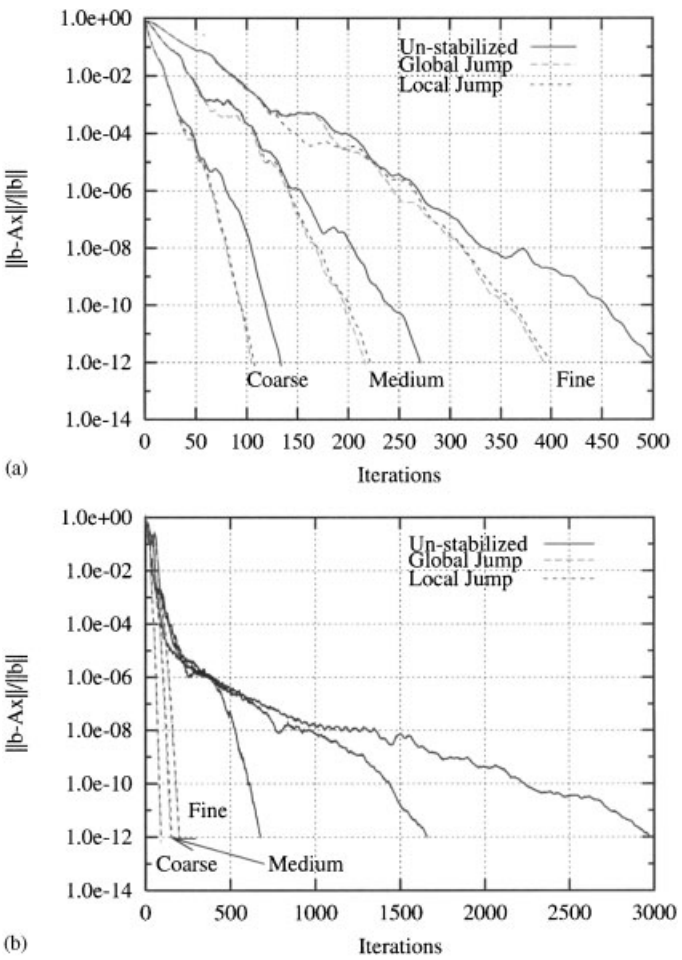

Figure 3. Convergence histories for SSOR preconditioned CG applied to (a) a 2-D $Re = 100$ cylinder flow using coarse (2800 elements), medium (11 200 elements), and fine resolution (44 800 elements) grids; (b) an $Re = 100$ flow past a circular post attached to a flat plate using coarse (7840 elements), medium (62 720 elements), and fine resolution (211680 elements) grids (see Christon [22]).

'Uzawa-like' iteration appeared in Chorin [2] (see Equation 5(a) and (b)) for the solution of the div-free velocity and associated pressure field in Chorin's method.

The saddle-point solver considered here is an adaptation of the gradient-based Uzawa algorithm of Thatcher [33] for the Stokes problem. The saddle-point solver in Algorithm (3) is attractive because it reuses the standard finite element operators (M_L, C, C^T) permitting the saddle-point problem to be solved in a 'matrix-free' mode. In terms of parallelism, the algorithm may be used directly with an element-based domain decomposition requiring communication for steps [3–5] and [8]. In addition, the allowable RMS divergence is used as the convergence criterion linking the algorithm directly to the div-free constraint on the velocity field—a physically appealing feature.

The similarity between the saddle point problem and the Stokes problem is obvious—here the usual viscous diffusion operator is replaced by a lumped mass matrix. In the context of the Stokes problem, there has been considerable effort expended in obtaining fast iterative solutions. Recent work in this area ranges from inexact and preconditioned Uzawa methods [34, 35] to two level pressure correction schemes [36] and stabilization techniques that regularize the Stokes saddle point problem [37]. Vincent and Boyer [38] has identified optimal stabilization parameters for the Stokes problem using the $Q1QO$ element for an Uzawa iteration.

Algorithm 3. *Saddle Point Solver*

1. Set $\mathbf{u}^k = \tilde{\mathbf{u}}^{n+1}$ for Algorithm (1), or $\mathbf{u}^k = \tilde{\mathbf{a}}^n$ for Algorithm (2).
2. Compute the divergence, $C^T\mathbf{u}^k$, and the starting Lagrange multiplier,
$$\lambda^k = C^T\mathbf{u}^k / \mathrm{diag}(C^T M_L^{-1} C)$$
3. Adjust the current velocity (or acceleration) via $\mathbf{u}^k = \mathbf{u}^k - M_L^{-1} C\lambda^k$.
4. Set $g^k = -C^T\mathbf{u}^k$ and compute $z^k = M_L^{-1} C g^k$.
5. $\rho^k = -(\mathbf{u}^k)^T CC^T\mathbf{u}^k / (z^k)^T CC^T\mathbf{u}^k$.
6. $\lambda^{k+1} = \lambda^k - \rho^k g^k$.
7. $\mathbf{u}^{k+1} = \mathbf{u}^k + \rho^k z^k$
8. $\sigma^{k+1} = -(\mathbf{u}^{k+1})^T CC^T\mathbf{u}^{k+1} / (\mathbf{u}^k)^T CC^T\mathbf{u}^k$.
9. $g^{k+1} = -C^T\mathbf{u}^{k+1} + \sigma^{k+1} g^k$
10. Repeat steps 3–9 until $\|C^T\mathbf{u}^{k+1}\|_{\mathrm{RMS}} \leqslant \varepsilon$.

For the projection-2 algorithm, Algorithm (1) the saddle point solver begins with an initial approximate velocity field, $\tilde{\mathbf{u}}$, and terminates with a div-free velocity and associated Lagrange multiplier, $\mathbf{u}^{n+1}$ and λ. In the explicit algorithm, Algorithm (2), the saddle-point solver begins with the partial acceleration, $\tilde{\mathbf{a}}$, and terminates with a div-free acceleration and associated pressure, $\mathbf{a}^n = \{\tilde{\mathbf{a}}^n - M_L^{-1} CP^n\}$ and P^n.

Despite the apparent attractiveness of the saddle-point solver, it does not offer any significant advantages over simply solving the consistent PPE problem. As pointed out by Thatcher [33], Algorithm (3) is equivalent to applying the conjugate gradient method to the Schur complement directly—in this case the consistent PPE. In fact, computational studies have shown that the iteration count (and CPU time) required to solve the PPE and perform the subsequent projection is nearly identical to the iteration count (and CPU time) required to solve a flow problem with the saddle-point solver. Here, the comparison was performed using the PPE solution as a baseline and requiring that the saddle-point solver achieve the same RMS divergence, i.e., $\|C^T\mathbf{u}\|_{\mathrm{RMS}} \leqslant 10^{-8}$. This result suggests a spectral equivalence between the saddle-point algorithm and the conjugate gradient method applied directly to the PPE. The saddle-point algorithm could be improved by the addition of a stabilization operator, S, but this would also defeat its matrix-free form and incur additional computational and communication costs in parallel.

Algebraic multi-grid

The fact that the condition number for the consistent PPE scales as h^{-2} and the right-hand-side, $C^T\mathbf{u}$, is frequently noisy, i.e., it contains short-wavelength information in addition to long-

wavelength information, suggests that the use of multi-grid methods should be very attractive due to the ability to obtain mesh independent convergence rates for elliptic problems. An attempt to apply algebraic multi-grid (AMG) to the PPE using the solver of Ruge and Stuben [39] has yielded promising results on simple problems such as an $Re = 100$ lid driven cavity.

However, AMG has failed to be robust in general application. While an extensive investigation of the failure of the solver has not been undertaken, it is clear that sign changes in the row-entries of the PPE confound the automatic selection of the 'strong connections' used to calculate the coarse-grid operators. This results in the generation of poor approximations of the coarse-grid operators and either a complete failure to converge or such slow convergence that the scheme is not useful.

It was also observed that the algebraic multi-grid-solver seemed particularly sensitive to problems where irregular geometry was introduced—again apparently due to the selection of incorrect 'strong-connections' for the coarse-grid operators. To illustrate this point, for the standard 1760 element vortex shedding mesh [40], the AMG solver was approximately eight times slower than a diagonally scaled conjugate gradient solver with an element-by-element matrix-vector multiply. Relative to the resolve cost of the PVS variable-band solver [41, 42] the AMG solver was 17 times slower per PPE solve. In addition to the performance, the difficulties in extending the AMG initialization phase, i.e., computation of the coarse-grid operators, to a domain-decomposition message-passing implementation appears insurmountable. For this reason, this approach has not been pursued further.

Despite the somewhat disappointing results obtained with AMG it is believed that a native finite element implementation of the multi-grid method would be successful in treating the PPE. This has not been undertaken at this time due to the inherent programming complexity and restrictions on the mesh generation, i.e., the development of multi-level grids.

The projection CG method

The solution of the time-dependent incompressible Navier–Stokes equations requires the repeated solution of the PPE problem where the coefficient matrix is fixed and the right-hand-side changes each time step. To address this aspect of solving the PPE, an A-conjugate projection is integrated with the iterative solution of the PPE in order to use solution information from the previous time steps. In the ensuing discussion, the PPE problem is cast as $Ax = b$ where $A = C^T M_L^{-1} C$ or $A = C^T M_L^{-1} C + S$, $x = \lambda$ and $b = C^T \tilde{\mathbf{u}}$.

The use of an A-conjugate projection as a pre-processing step for the solution of the linear system, $Ax = b$, follows the development presented by Fischer [43] with extensions that permit seeding the A-conjugate vectors. Related work on solving linear systems with multiple right-hand-sides may be found in Saad [44] and Chan and Wan [45].

To begin the development, the idea of a pre-processing A-conjugate projection step relies on minimizing the distance between the solution at a given time step, x^n, and the base vectors Φ in the A-norm. Here, Φ is a set of A-conjugate vectors derived from N prior solutions to $Ax = b$ where $\Phi = \{\phi_i,\ i = 1, N\}$. As will be demonstrated in Section 4, $N = 5$ to $N = 10$ is a reasonable choice.

Let the initial 'guess' for a solution at time-step n be given by

$$\bar{x} = \sum_{i=1}^{N} \alpha_n \phi_n \tag{25}$$

Let the difference between the solution at time-step n and the initial guess, $\bar{x}$, be defined as

$$\|e\|_A = \|x^n - \bar{x}\|_A \tag{26}$$

where $\|\cdot\|_A = \sqrt{(\cdot)^T A(\cdot)}$.

With this definition,

$$\|x^n - \bar{x}\|_A^2 = (x^n)^T A x^n - \bar{x}^T A x^n - (x^n)^T A\bar{x} + (\bar{x})^T A\bar{x} \tag{27}$$

and for A being symmetric positive definite,

$$\|x^n - \bar{x}\|_A^2 = (x^n)^T A x^n - 2(x^n)^T A\bar{x} + (\bar{x})^T A\bar{x} \tag{28}$$

Now, substituting Equation (25) and using the fact that Φ is a set of A-conjugate vectors,

$$\|e\|_A^2 = (x^n)^T A x^n - 2\sum_{i=1}^{N} \alpha_i (\phi_i)^T A x^n + \sum_{i=1}^{N} \alpha_i^2 \tag{29}$$

Minimization of $\|e\|_A$ with respect to α_i yields

$$\alpha_i = (\phi_i)^T A x^n = (\phi_i)^T b^n. \tag{30}$$

Thus, given a set of A-conjugate vectors, Φ, the best approximation to x^n that minimizes the error in the A-norm is obtained by projecting the right-hand-side, b^n, at time-step n onto the set of base vectors, Φ. This suggests the following solution procedure.

Algorithm 4. *A-conjugate projection CG method*

1. $\alpha_i = (\phi_i)^T b^n$.
2. $\bar{x} = \sum_{i=1}^{N} \alpha_i \phi_i$ where N is the number of A-conjugate base vectors.
3. Solve $A\Delta x^n = r^n$ using the conjugate gradient method where $r^n = b^n - A\bar{x}$.
4. Update the solution, $x^n = \bar{x} + \Delta x^n$.
5. Update the base vectors, Φ to include new information from the last solution.

As an aside, the solution at a given time level may be written as

$$x^n = \Delta x^n + \sum_{i=1}^{N} \alpha_i \phi_i \tag{31}$$

Updating Φ

For the initial solution, or when the number of existing base vectors exceeds N, the basis is started by normalizing the solution as

$$\phi_1 = \frac{x^1}{\|x^1\|_A}. \tag{32}$$

For all other cases, a solution vector is a candidate for addition to Φ only when it contains non-trivial information not already present in Φ. Here, the basic idea is that Δx is A-conjugate to $\bar{x}$ as well as to the individual base vectors ϕ_i. Therefore, the addition of a new base vector

should be based on criteria that guarantee that new information is being added to the existing base vectors. Addition of a solution vector (or a part of a solution vector) proceeds by first computing the part of the solution that is not contained in the basis as

$$\psi_{l+1} = x^n - \sum_{i=1}^{l} \alpha_i \phi_i. \tag{33}$$

Now, each ϕ_i is A-conjugate to ψ_{l+1} by construction and is added to the basis as

$$\phi_{l+1} = \frac{\psi_{l+1}}{\|\psi_{l+1}\|_A}. \tag{34}$$

The idea of seeding the A-conjugate vectors in Φ with several initial vectors has been suggested by Hughes [46] as a mechanism for reducing cost of the initial linear solves, i.e., the first PPE solve when Φ is empty. This may be achieved by selecting several global polynomials and even a random vector as seeds for Φ. Construction of the A-conjugate vectors in Φ proceeds according to the algorithm above.

Experimentation with vectors representing a random solution, hydrostatic and simple polynomials have been tested with limited success. In the case of simple flows, e.g., Pouiselle flow, the linear polynomials contribute directly to the initial solution and reduce the number of CG iterates dramatically. However, in moderately complex domains, the simple polynomials provide little reduction in computational cost. It is thought that initial polynomial fields that respect the implied pressure boundary conditions may provide an effective set of seed vectors, but this has not yet been tested.

To illustrate the use of the A-conjugate projection, Plate 1(a) shows a snapshot of the pressure field for an $Re = 100$ vortex shedding computation. In addition, plate 1(b)–(f) shows snapshots of the Φ vectors based on the previous five time steps. It is clear that ϕ_1 provides primarily long wavelength information while the other four vectors provide detailed information about the wake. Thus, the vectors $\phi_2 - \phi_5$ may be viewed as short wavelength corrections to ϕ_1 that yield the best approximation to the current pressure field. The A-conjugate projection procedure, in effect, selects the appropriate information from each ϕ vector in order to minimize the residual in the A-norm before performing any CG iterations.

The A-conjugate projection procedure retains both long and short wavelength information, and in this sense, the procedure may be viewed as an approximate means of deflating the eigenvalue spectrum for the PPE. The combination of the A-conjugate projection method, PPE stabilization and SSOR preconditioning has proved to be the most computationally efficient method for solving the PPE.

4. RESULTS

This section presents the results of several representative computational experiments performed using the A-conjugate projection strategy with several variants of the preconditioned conjugate gradient method. In addition, the effect of pressure stabilization on the iterative solvers is considered for both local and global stabilization. Scaled parallel efficiencies are presented to illustrate the influence of the PPE solution strategy on the parallel performance.

The CG variants used in this study include a Jacobi preconditioned conjugate gradient (JPCG) solver, a symmetric successive over-relaxation preconditioned CG method (SSOR-

PCG), and the Eisenstat-SSOR CG solver (ESSOR-PCG). Each CG solver used a row-compressed storage scheme for the PPE. The SSOR preconditioners were implemented in a stand-alone form (SSOR-PCG) that requires separate matrix-vector and preconditioning functions, and in the ESSOR-PCG form using the transformation first suggested by Eisenstat [47] to reduce the operation count required for the preconditioned conjugate gradient method. (See also Ortega [48] and Eisenstat *et al.* [49].)

The first series of computations used to evaluate the PPE solution methods were carried out for a momentum driven slot jet with Reynolds number $Re=4000$ and Froude number $Fr=326$. The Reynolds number is based on a 15 mm slot width and the Froude number is based on $Fr=v^2/g\beta\Delta TL_c$ where $g=9.81\ \mathrm{m\,s^{-2}}$, $\beta\Delta T=1/3$, $L_c=15$ mm, and $v=4\ \mathrm{m\,s^{-1}}$. In this computation, an energy equation was solved in conjunction with the Navier–Stokes equations using the Boussinesq approximation. The formulation follows that presented by Gresho *et al.* [21, Equation 1(a)–(c)]. The initial conditions consist of a div-free velocity field with a free-field temperature of 300 K. The inlet jet velocity is $4\ \mathrm{m\,s^{-1}}$ and the inlet temperature is 400 K. The working fluid is air with a unit Prandtl number, $Pr=\nu/\alpha$ resulting in $Pe=4000$ where $Pe=RePr$. Here, the relatively large Froude number indicates that the buoyancy forces are small compared to the inertial forces, i.e., a momentum driven jet. Plate 2 shows snapshots of the temperature, vorticity and pressure fields.

Tables I and II show the results of the jet computations in terms of the average iteration count, grind time, and percentage of the CPU time spent solving the PPE averaged over 1000 time steps. Here, the grind-times are normalized with respect to the grind-time for the same computation using the PVS [41, 42] direct solver for the PPE where a single factorization is performed during initialization with a single resolve carried out at each time step. All computations in this comparison were performed using a 500 MHz single processor DEC Alpha with $\varepsilon=10^{-5}$ as the iterative convergence criteria for the PPE solution and $\beta=0.05$ for the stabilization parameter.

With the direct resolve cost of the PVS solver as the baseline metric, the ideal situation would be to have a normalized grind-time using the iterative PPE solvers that matches or even beats the resolve cost for the PVS solver. Note that the CPU time associated with the direct resolve scales roughly as $Nel\times Nb$ where Nb is the half-bandwidth of the PPE.

For both the explicit and semi-implicit projection algorithms, there is little apparent effect of the stabilization parameter for the 2-D jet problem—undoubtedly a consequence of using $\varepsilon=10^{-5}$ for the convergence criteria. However, the SSOR preconditioning reduces the iteration count by nearly a factor of three without the A-conjugate projection. The use of the Eisenstat formulation further reduces the grind-time by a factor of approximately 1.5 relative to the SSOR-PCG solver. The effect of the A-conjugate projection algorithm is seen in the reduction in the iteration count by nearly a factor of three and the grind-time by nearly a factor of three regardless of the preconditioner for the explicit algorithm when five projection vectors are used. For the semi-implicit projection algorithm, this effect is reduced somewhat resulting in a factor of about 2.25 reduction in the iteration count and better than a factor of two reduction in the grind-time.

The combination of the ESSOR-PCG algorithm and 10 projection vectors results in grind-times for the explicit algorithm that are within a factor of three of the grind-times using the PVS solver. For the projection algorithm, the grind-times are within a factor of two emphasizing the additional cost of the semi-implicit treatment of the momentum and scalar transport equations. Relative to the basic JPCG solver the ESSOR-PCG solver with 10 projection vec-

Table I. Effect of the A-Conjugate projection on the PPE solution time for the 2-D momentum driven jet. The grind-times for the explicit algorithm have been normalized with respect to the solution time for 1000 time steps using a direct solver [41, 42]. N_{IT} is the average number of iterations required per time step to solve the PPE problem. ($Nel = 11\,250$, $Nnp = 11\,466$, $N_b = 146$).

Explicit Algorithm—No Stabilization

	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	353	26.52	98.9	120	19.89	98.5	109	12.22	97.3
5	114	8.81	96.8	38	6.65	95.7	35	4.20	92.2
10	82	5.55	94.4	28	4.47	90.4	26	2.97	88.2
25	55	4.48	93.6	19	3.68	91.7	17	2.53	86.7
50	45	3.99	92.7	16	3.58	91.9	14	2.33	85.7

Explicit Algorithm—Global Jump Stabilization ($\beta = 0.05$)

	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	340	24.92	98.8	115	19.23	98.3	107	12.06	97.1
5	109	8.41	96.5	36	6.21	94.9	33	4.02	91.5
10	79	5.33	93.9	26	4.32	92.0	24	2.88	87.3
25	52	4.24	92.7	18	3.47	90.6	16	2.42	85.3
50	42	3.76	91.9	14	3.06	89.4	13	2.21	84.2

Explicit Algorithm—Local Jump Stabilization ($\beta = 0.05$)

	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	291	21.44	98.7	99	16.55	98.1	91	10.35	96.8
5	120	9.27	97.0	40	6.84	95.5	36	4.31	92.5
10	89	5.96	94.7	30	4.80	93.0	27	3.19	88.7
25	62	4.96	94.2	21	3.98	92.1	19	2.73	87.4
50	50	4.31	93.1	17	3.46	90.8	15	2.47	86.1

tors results in a factor of nine reduction in the normalized grind-time for the explicit algorithm and nearly a factor of four reduction for the projection algorithm. It is important to note that in two-dimensions, the storage cost of 10 projection vectors is equivalent to the storage cost of a separate preconditioning matrix. Although further reductions in the grind-times may be had with increasing number of projection vectors, the storage cost and diminishing reduction in the computational costs makes this less attractive. Even when 50 projection vectors are used, the percentage of the CPU time per time step spent on the PPE solution cannot be pushed below $\approx$85% for the explicit algorithm. In contrast, 10 projection vectors reduces the average cost of solving the PPE to be approximately equal to the cost of solving the two

Table II. Effect of the A-Conjugate projection on the PPE solution time for the 2-D momentum driven jet. The grind-times for the projection algorithm have been normalized with respect to the solution time for 1000 time steps using a direct solver [41, 42]. N_{IT} is the average number of iterations required per time step to solve the PPE problem. ($Nel = 11\,250$, $Nnp = 11\,466$, $N_b = 146$).

Projection-2 Algorithm—No Stabilization									
	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	328	6.52	87.4	110	5.05	83.6	100	3.36	74.7
5	148	3.20	75.0	49	2.63	69.4	47	2.02	59.7
10	121	2.53	66.5	41	2.13	59.9	38	1.75	51.1
25	100	2.66	69.4	34	2.18	62.3	31	1.71	51.7
50	87	2.39	66.6	29	2.03	60.4	26	1.60	49.2
Projection-2 Algorithm—Global Jump Stabilization ($\beta = 0.05$)									
	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	309	6.16	86.6	105	4.83	82.8	96	3.23	74.0
5	130	2.91	72.4	43	2.39	66.2	44	1.87	56.2
10	108	2.35	63.7	36	1.98	56.8	37	1.68	48.1
25	88	2.44	66.5	29	2.00	58.8	27	1.65	49.4
50	76	2.20	63.7	26	1.92	57.9	23	1.52	45.9
Projection-2 Algorithm – Local Jump Stabilization ($\beta = 0.05$)									
	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	309	6.52	86.6	105	4.95	82.8	96	3.27	74.0
5	130	3.11	72.4	43	2.53	66.2	44	1.93	56.2
10	108	2.47	63.7	36	2.07	56.8	37	1.73	48.1
25	88	2.61	66.5	29	2.09	58.8	29	1.69	49.4
50	76	2.36	63.7	26	2.04	57.9	25	1.58	46.0

semi-implicit momentum equations and the scalar transport equation, i.e., 50% of the CPU time per step.

The second series of computations were carried out for a three-dimensional channel flow past a circular cylinder with $Re_D = 100$. This problem was motivated by a suite of benchmark problems used as 'round-robin' tests to evaluate incompressible flow solution methods developed under the Deutsche Forschungsgemeinschaft (DFG) Priority Research Program, 'Flow Simulation on High Performance Computers'. In this computation, a parabolic velocity profile is prescribed at the inlet to the channel with no-slip/no-penetration boundaries on the channel and cylinder walls. Natural boundary conditions were used at the channel outflow.

The prescription of $\mathbf{u} \cdot \mathbf{n} = 0$ on the channel walls results in a two-dimensional checkerboard pressure mode that exists in each plane of three-dimensional elements normal to the flow direction. In this situation, pressure stabilization is essential to filter the checkerboard modes. Note that solving the unstabilized PPE is possible, but it requires an extremely large number of iterations, $O(10^2 - 10^3)$, per time step. For this reason, all computations were performed using the pressure stabilization methods outlined in Section 3. Plate 3 shows snapshots of the 3-D z-vorticity field and the corresponding pressure field for this problem.

Table III reports the grind-times, iteration counts, and percentage of CPU time spent solving the PPE per time-step for the channel flow problem. The trends with respect to the preconditioners and number of projection vectors is similar in 3-D to the 2-D results. The most obvious difference here being the increase in the resolve cost of the direct method due to the relatively large half-bandwidth found with a reordering based on the Gibbs–Poole–Stockmeyer bandwidth minimization with reverse Cuthill–McKee numbering. Due to the relatively minimal computational costs associated with solving the momentum equations in the explicit algorithm, the PPE solve requires better than 55% of the CPU time per step. The best balance between memory usage and grind-times are found using the ESSOR-PCG solver combined with 10 projection vectors where a speedup by a factor of 3.6 to 3.8 relative to the basic JPCG solver applied to the stabilized PPE is observed. With 50 projection vectors, the average of the PPE solve requires only about 85% of the CPU time required for the direct resolve cost using the PVS solver. This is undoubtedly a consequence of the relatively large half-bandwidth of the 3-D system.

The projection method exhibits larger computational costs associated with the momentum transport equations as shown by the percentage of time for the PPE in Table III(b). In this case, 10 projection vectors with the ESSOR-PCG solver yield normalized grind times of just over unity with the PPE requiring less than 25% of the CPU time per step. That is to say, the cost of the PPE solve is approximately equal to the cost of solving one component of the momentum equations—here the JPCG algorithm is used for the momentum transport equations.

Now, turning to the parallel situation, a comparison between the element-by-element Jacobi preconditioned CG solver reported by Christon [22] and a sub-domain SSOR preconditioned CG solver with and without 10 projection vectors was conducted. The results of the comparison are shown in Figure 4(a) for an $Re_H = 800$ backward facing step using 250 elements per processor. In Figure 4(b), scaling results are shown for a 3-D flow past a post-plate juncture ($Re_D = 100$) also computed with 250 element per processor. Note that the small number of elements per processor was chosen intentionally in order to show a worst case scenario where the communication costs are large and the computational load per processor is small. All parallel computations were carried out on the ASCI Red TFLOPs computer at Sandia National Laboratories.

For both the 2-D and 3-D computations, a marked reduction in the overhead associated with communication is observed when using 10 projection vectors with either the EBE-JPCG or the SSOR-PCG algorithms. The overall efficiency of the SSOR-PCG algorithm is higher due to the increased and perfectly parallel work introduced by the sub-domain preconditioner. As the number of processors is increased, the effect of the A-conjugate projection becomes more pronounced due in part to the reduced overall iteration count and concomitant communication requirements. With an increasing number of processors, the effect of the A-conjugate projection technique is to remove the long wavelength error components from the solution permitting

Table III. Effect of the A-Conjugate projection on the PPE solution time for the 3-D cylinder vortex shedding problem. The grind-times have been normalized with respect to the solution time for 1000 time steps using a direct solver [41, 42]. N_{IT} is the average number of iterations required per time step to solve the PPE problem. ($Nel = 53\,900$, $Nnp = 58\,916$, $N_b = 2239$).

(a) Explicit Algorithm

Explicit Algorithm—Global Jump Stabilization ($\beta = 0.05$)									
	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	64	4.88	64.8	22	4.00	64.3	20	2.59	62.9
5	37	2.91	63.5	12	2.29	62.6	12	1.61	60.8
10	31	2.49	63.0	10	1.95	61.9	9	1.33	59.5
25	22	1.83	61.5	7	1.43	60.0	7	0.98	57.0
50	17	1.46	60.3	6	1.17	58.5	5	0.83	55.1
Explicit Algorithm—Local Jump Stabilization ($\beta = 0.05$)									
	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	62	4.17	64.7	22	3.86	64.3	19	2.49	62.7
5	38	2.98	63.6	13	2.32	62.5	11	1.50	60.5
10	30	2.38	62.8	10	1.84	61.5	9	1.23	58.9
25	22	1.92	61.5	7	1.40	59.9	6	0.95	56.8
50	18	1.54	60.6	6	1.18	58.8	5	0.85	55.4

(b) Projection Algorithm

Projection-2 Algorithm—Global Jump Stabilization ($\beta = 0.05$)									
	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	63	1.54	43.8	20	1.24	36.4	19	1.12	28.3
5	49	1.34	38.4	16	1.15	31.3	14	1.04	23.9
10	42	1.20	34.9	13	1.10	28.2	12	1.01	21.6
25	36	1.16	31.7	12	1.06	25.7	10	0.99	19.8
50	33	1.14	30.7	11	1.05	24.9	10	0.99	19.4
Projection-2 Algorithm—Local Jump Stabilization ($\beta = 0.05$)									
	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	67	1.43	45.0	21	1.26	37.6	20	1.12	29.5
5	51	1.30	39.0	16	1.16	32.1	15	1.05	24.7
10	44	1.10	35.8	14	1.11	28.9	13	1.02	22.3
25	39	1.06	32.8	12	1.08	26.6	11	1.00	20.5
50	36	1.05	31.9	11	1.07	25.8	10	0.99	20.1

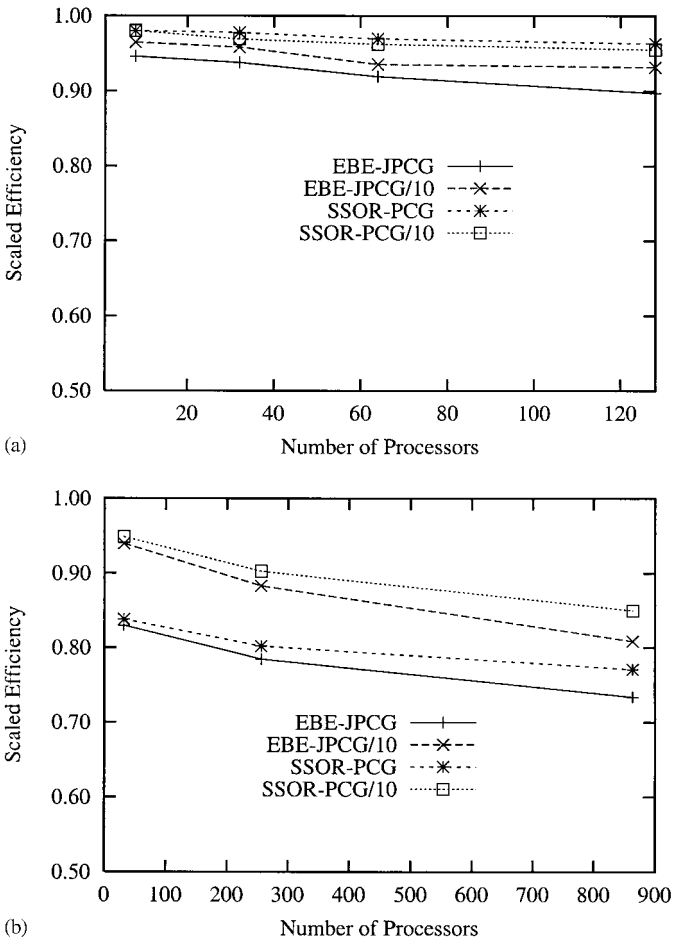

Figure 4. Scaled efficiencies for (a) 2-D $Re = 100$ vortex shedding problem, and (b) $Re = 100$ post and plate juncture flow [22].

the sub-domain preconditioner to operate only on the short wavelength components where it can be effective.

In order to further illustrate the impact of the A-conjugate projection technique in combination with the SSOR preconditioned conjugate-gradient method and a stabilized PPE operator, two large-eddy simulation computations are reported. The first is an $Re = 10\,000$ lid-driven cavity which uses a $32 \times 32 \times 32$ mesh modeled after the grid used by Zang *et al.* [50] and a Smagorinsky model. In this computation, approximately 1500 eddy-turnover times were simulated resulting in an average iteration count of 11 iterations per PPE solve for a PPE with 32 768 equations using the explicit algorithm. Similar iteration counts have been observed for the projection algorithm. The second computation consists of a three-dimensional transitional round jet with $Re_D = 2500$ based on the jet diameter and a computational domain 20 D in diameter extending 30 D downstream from the jet entrance. This computation used 512 processors of the Sandia TFLOPs machine with a total of 1 049 600 elements. In this case,

the use of 10 projection vectors with the global jump stabilization resulted in an average of only 34 iterations per time step demonstrating the scalability of the projection-based CG method.

5. CONCLUSIONS

Based on the results for a broad range of flow problems (too many to report on here), the use of a stabilized consistent pressure-Poisson operator, an A-conjugate projection technique, and SSOR preconditioning with the conjugate gradient method yields the overall best performance relative to the resolve cost of a direct solver. The effect of the A-conjugate projection technique is to introduce both long and short wavelength information which, in effect deflates the eigenvalue spectrum of the PPE. This effect not only reduces the iteration count and concomitant computational cost but is seen to improve the effectiveness of simple sub-domain parallel preconditioners that are effective for wavelengths proportional to the sub-domain size but that cannot smooth error modes that span sub-domains.

Operational experience with both the global and local pressure stabilization formulations has shown that either approach is effective in terms of filtering spurious pressure modes and improving the overall robustness of the computations. Although there is a modest reduction in the PPE solution time when using the pressure stabilization, the larger computational advantage derives from the A-conjugate projection CG algorithm. Ultimately, the combination of the pressure stabilization with preconditioning and the A-conjugate projection technique has proven both robust and computationally efficient making it a reasonable alternative to more complicated approaches based on multi-grid or multi-level algorithms for time-dependent problems.

ACKNOWLEDGEMENTS

The author wishes to acknowledge John Ruge (University of Colorado), for providing the AMG software and consulting on its application to the PPE, Phil Gresho (Lawrence Livermore National Laboratory) for his suggestions and insight into the behavior of the PPE during the many experiments conducted with various solvers over the past eight years, and Tom Voth (Sandia National Laboratories) for his helpful comments on this manuscript.

REFERENCES

1. Gresho PM, Sani RL. *Incompressible Flow and the Finite Element Method, Advection-diffusion and Isothermal Laminar Flow*. John Wiley & Sons: Chichester, 1998.
2. Chorin AJ. Numerical solution of the Navier–Stokes equations. *Mathematics of Computations* 1968; **22**:745–762.
3. Van Kan J. A second-order accurate pressure-correction scheme for viscous incompressible flow. *SIAM Journal of Scientific and Statistical Computing* 1986; **7**:870–891.
4. Bell JB, Colella P, Glaz HM. A second-order projection method for the incompressible Navier–Stokes equations. *Journal of Computational Physics* 1989; **85**:257–283.
5. Gresho PM. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. Part 1: Theory. *International Journal for Numerical Methods in Fluids* 1990; **11**:587–620.
6. Gresho PM, Chan ST. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a newly consistent mass matrix. Part 2: Implementation. *International Journal for Numerical Methods in Fluids* 1990; **11**:621–659.

7. Gresho PM, Chan ST, Christon MA, Hindmarsh AC. A little more on stabilized $q1q1$ for transient viscous incompressible flow. *International Journal for Numerical Methods in Fluids* 1995; **21**:837–856.
8. Gresho PM, Chan ST. Projection 2 goes turbulent—and fully implicit. Technical report, Lawrence Livermore National Laboratory, March 1996. UCRL-JC-123727.
9. Almgren AS, Bell JB, Colella P, Howell LH. An adaptive projection method for the incompressible Euler equations. In *Eleventh AIAA Computational Fluid Dynamics Conference*, pp. 530–539. AIAA, 1993.
10. Almgren AS, Bell JB, Szymcyzk WG. A numerical method for the incompressible Navier–Stokes equations based on an approximate projection. *SIAM Journal for Scientific Computing* 1996; **17**(2):358–369.
11. Almgren AS, Bell JB, Colella P, Howell LH, Welcome ML. A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations. *Journal of Computational Physics* 1998; **142**:1–46.
12. Rider WJ. The robust formulation of approximate projection methods for incompressible flows. Technical Report LA-UR-3015, Los Alamos National Laboratory, 1994.
13. Rider WJ. Filtering nonsolenoidal modes in numerical solutions of incompressible flows. Technical Report LA-UR-3014, Los Alamos National Laboratory, Los Alamos, New Mexico, September 1994.
14. Rider WJ, Kothe DB, Mosso SJ, Cerutti JH, Hochstein JI. Accurate solution algorithms for incompressible multiphase flows. Technical Report AIAA-95-0699, AIAA, Reno, Nevada, January 1995.
15. Rider WJ. Approximate projection methods for incompressible flow: implementation, variants and robustness. Technical Report LA-UR-2000, Los Alamos National Laboratory, Los Alamos, New Mexico, July 1995.
16. Minion ML. A projection method for locally refined grids. *Journal of Computational Physics* 1996; **127**:158–178.
17. Guermond J-L, Quartapelle L. Calculation of incompressible viscous flow by an unconditionally stable projection fem. *Journal of Computational Physics* 1997; **132**:12–23.
18. Puckett EG, Almgren AS, Bell JB, Marcus DL, Rider WJ. A high-order projection method for tracking fluid interfaces in variable density incompressible flows. *Journal of Computational Physics* 1997; **130**:269–282.
19. Sussman M, Almgren AS, Bell JB, Colella P, Howell LH, Welcome ML. An adaptive level set approach for two-phase flows. *Journal of Computational Physics* 1999; **148**:81–124.
20. Knio OM, Najm HN, Wyckoff PS. A semi-implicit numerical scheme for reacting flow. II. stiff operator-split formulation Pre-print submitted to *Journal of Computational Physics*, 1999.
21. Gresho PM, Chan ST, Lee RL, Upson CD. A modified finite element method for solving the time-dependent, incompressible Navier–Stokes equations. Part 1: Theory. *International Journal for Numerical Methods in Fluids* 1984; **4**:557–598.
22. Christon MA. A domain-decomposition message-passing approach to transient viscous incompressible flow using explicit time integration. *Computer Methods in Applied Mechanics and Engineering* 1997; **148**:329–352. (Sandia SAND96-2566J).
23. Kim C-J, Ro ST. Efficient and robust matrix solver for the pressure-correction equations in two- and three-dimensional fluid flow problems. *Numerical Heat Transfer*, *Part B* 1995; **27**:355–369.
24. Sani RL, Gresho PM, Lee RL, Griffiths DF. The cause and cure (?) of the spurious pressures generated by certain fem solutions of the incompressible Navier–Stokes equations: Part 1. *International Journal for Numerical Methods in Fluids* 1981; **1**:17–43.
25. Sani RL. Gresho PM, Lee RL, Griffiths DF, Engelman M. The cause and cure (!) of the spurious pressures generated by certain fem solutions of the incompressible Navier–Stokes equations: Part 2. *International Journal for Numerical Methods in Fluids* 1981; **1**:171–204.
26. Griffiths D, Silvester D. Unstable modes of the q1-p0 element. Technical Report NA-257, University of Manchester/UMIST, Manchester Centre for Computational Mathematics, September 1994.
27. Hughes TJR, Franca LP. A new finite element formulation for computational fluid dynamics: VII. the Stokes problem with various well-posed boundary conditions: symmetric formulations that converge for all velocity/pressure spaces. *Computer Methods in Applied Mechanics and Engineering* 1987; **65**:85–96.
28. Silvester DJ, Kechkar N. Stabilised bilinear-constant velocity-pressure finite elements for the conjugate gradient solution of the stokes problem. *Computer Methods in Applied Mechanics and Engineering* 1990; **79**:71–86.
29. Kechkar N, Silvester D. Analysis of locally stabilised mixed finite element methods for the Stokes problem. *Mathematics of Computation* 1992; **58**:1–10.
30. Norburn S, Silvester D. Stabilised vs. stable mixed methods for incompressible flow. Technical Report NA-307, University of Manchester/UMIST, Manchester Centre for Computational Mathematics, September 1997.
31. Chan ST, Sugiyama G. User's Manual for mc_wind: A new mass-consistent wind model for arac-3. Technical Report UCRL-MA-129067, Lawrence Livermore National Laboratory, Livermore, California 94550, November 1997.
32. Fortin M, Glowinski R. *Studies in Mathematics and its Applications*, Ch. Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems. North-Holland: New York, 1983; 57–75.
33. Thatcher RW. The finite element method for three dimensional incompressible flow. In Max D. Gunzburger and Roy A. Nicolaides (eds). *Incompressible Fluid Dynamics*. Cambridge University Press: New York, 1993; 427–446.

34. Cahouet J, Chabard J-P. Some fast 3d finite element solvers for the generalized Stokes problem. *International Journal for Numerical Methods in Fluids* 1988; **8**:869–895.
35. Elmar HC, Golub GH. Inexact and preconditioned Uzawa algorithms for saddle point problems. *SIAM Journal on Numerical Analysis* 1994; **31**(6):1645–1661.
36. Ramage A, Wathen AJ. Iterative solution techniques for the Stokes and Navier–Stokes equations. *International Journal for Numerical Methods in Fluids* 1994; **19**:67–83.
37. Wathen A, Silvester D. Fast iterative solution of stabilised Stokes systems, Part I: Using simple diagonal preconditioners. *SIAM Journal on Numerical Analysis* 1993; **30**(3):630–649.
38. Vincent C, Boyer R. A preconditioned conjugate gradient Uzawa-type method for the solution of the Stokes problem by mixed q1-p0 stabilized finite elements. *International Journal for Numerical Methods in Fluids* 1992; **14**:289–298.
39. Ruge JW, Stuben K. *Frontiers in Applied Mathematics*, Ch. Algebraic Multigrid. SIAM: Philadelphia, PA, 1987.
40. Gresho PM, Chan ST, Lee RL, Upson CD. A modified finite element method for solving the time-dependent, incompressible Navier–Stokes equations. Part 2: Applications. *International Journal for Numerical Methods in Fluids* 1984; **4**:619–640.
41. Storaasli OO, Nguyen DT, Agarwal TK. A parallel-vector algorithm for rapid structural analysis on high-performance computers. Technical Report 102614, NASA, April 1990.
42. Agarwal TK, Storaasli OO, Nguyen DT. A parallel-vector algorithm for rapid structural analysis on high-performance computers. *AIAA* 1990; 90–1149.
43. Fischer PF. Projection techniques for iterative solution of $ax=b$ with successive right-hand sides. Technical Report 93-90, ICASE, 1993. (Also NASA CR-191571).
44. Saad Y. On the lanczos method for solving symmetric linear systems with several right-hand sides. *Mathematics of Computation* 1987; **48**(178):651–662.
45. Chan TF, Wan WL. Analysis of projection methods for solving linear systems with multiple right-hand sides. *SIAM Journal on Scientific Computing* 1997; **18**(6):1698–1721.
46. Hughes TR. Personal communication. October 1998.
47. Eisenstat SC. Efficient implementation of a class of preconditioned conjugate gradient methods. *SIAM Journal for Scientific and Statistical Computing* 1981; **2**(1):1–4.
48. Ortega JM. Efficient implementations of certain iterative methods. *SIAM Journal for Scientific and Statistical Computing* 1988; **9**(5):882–891.
49. Eisenstat SC, Ortega JM, Vaughan CT. Efficient implementation of a class of preconditioned conjugate gradient methods. *SIAM Journal for Scientific and Statistical Computing* 1990; **11**(5):859–872.
50. Zang Y, Street RL, Koseff JR. A dynamic mixed subgrid-scale model and its application to turbulent recirculating flows. *Physics of Fluids A* 1993; **5**(12):3186–3196.

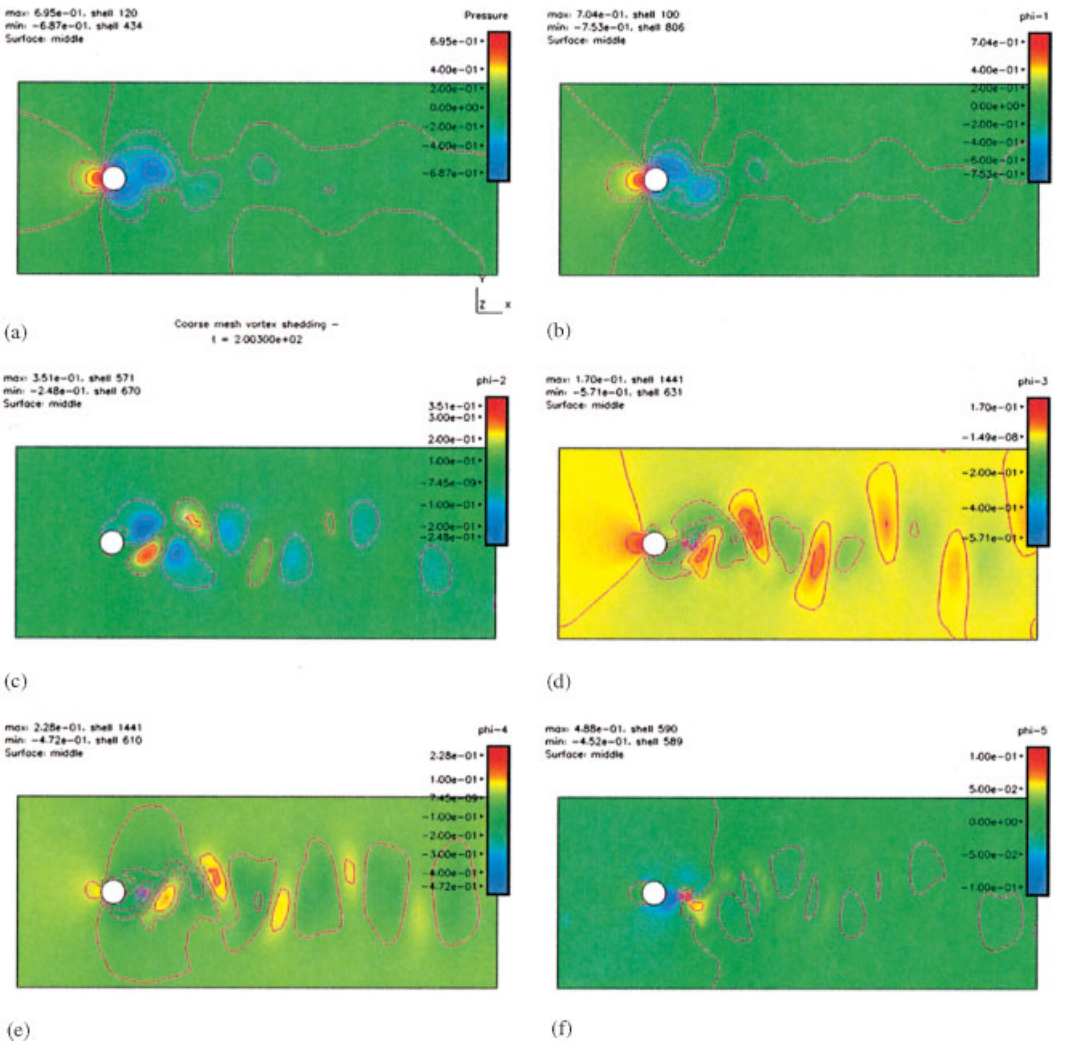

Plate 1. (a) Snapshot of the pressure field during vortex shedding, and five ϕ-fields (b)–(f) based on pressure solution at the five prior time steps.

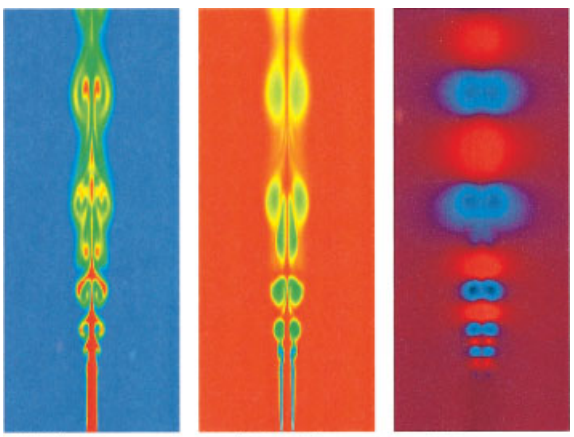

Plate 2. Snapshot of (a) temperature field, (b) z-vorticity field and (c) pressure for the 2-D momentum driven jet, $Re = 4000$, $Fr = 326$. (The temperature, vorticity and pressure fields have been reflected about the vertical centerline for presentation purposes.)

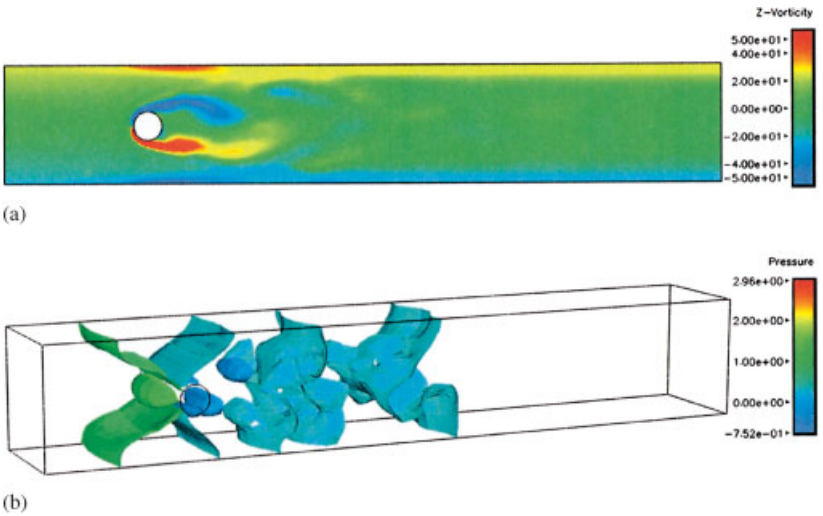

Plate 3. Snapshot of (a) cutplane showing the z-vorticity field and (b) isosurfaces of the pressure for the 3-D cylinder in a channel for $Re_D = 100$.