Journal of Computational Physics **181,** 617–638 (2002)
doi:10.1006/jcph.2002.7144

Krylov-Based Uzawa Algorithms for the Solution of the Stokes Equations Using Discontinuous-Pressure Tetrahedral Finite Elements

François Bertrand and Philippe A. Tanguy

URPEI, Department of Chemical Engineering, Ecole Polytechnique, P.O. Box 6079, Station Centre-ville, Montreal, Quebec, H3C 3A7, Canada
E-mail: bertrand@urpei.polymtl.ca

Received June 19, 2001; revised June 3, 2002

The objective of this work is to develop an efficient and robust Krylov-based Uzawa algorithm for the solution of the 3D steady-state and unsteady-state Stokes equations with discontinuous-pressure tetrahedral finite elements. To this end, a class of preconditioned conjugate gradient Uzawa algorithms are presented and compared to those of the literature for other finite element types. A comparison is also made with an inexact Uzawa algorithm for the steady-state case. An analysis of the results obtained for various problems leads to an iterative scheme that is well adapted to the solution of Stokes problems of all sizes and complexities such as those found in industrial applications. © 2002 Elsevier Science (USA)

Key Words: Stokes equations; finite element method; 3D; iterative solver; Uzawa algorithm; Krylov subspace methods.

1. PRELIMINARIES

1.1. *Introduction*

Over the years, many different computer algorithms based on the finite element method have been developed for the solution of the Navier–Stokes equations

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\operatorname{grad}\mathbf{v}\right) - \mu\Delta\mathbf{v} + \operatorname{grad} p = \mathbf{f}, \quad \text{in } \Omega, \tag{1}$$

$$\operatorname{div}\mathbf{v} = 0, \quad \text{in } \Omega. \tag{2}$$

Classical ways to cope with the nonlinearity owing to the inertia term include the method

0021-9991/02 $35.00
© 2002 Elsevier Science (USA)
All rights reserved.

of successive substitution (Picard iteration) and Newton's method. These transform the Navier–Stokes equations into a series of Stokes-like problems, the discretization of which leads to the linear system

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^{\mathbf{T}} \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{V} \\ \mathbf{P} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{0} \end{bmatrix}, \tag{3}$$

where $\mathbf{A}$ contains a mass matrix term to account for the time derivative. In the case of the steady-state Stokes equations, $\mathbf{A}$ stands for the diffusion matrix only. It thus comes as no surprise that the overall efficiency of these fixed-point methods is closely related to the performance of the Stokes solver used. Other situations that involve as a substep the solution of Stokes-like problems include the simulation of non-Newtonian fluid flows, where some linearization process is required to handle the nonlinear diffusion term [1], and operator-splitting strategies for the solution of the Navier–Stokes equations, where inertia and incompressibility are decoupled [2].

1.2. *Iterative Methods*

In the context of three-dimensional problems, a large number of degrees of freedom are generally required for a good approximation, so that solving the Stokes equations implies dealing with huge sparse linear systems. For this purpose, iterative methods are definitely more attractive than direct methods, as they require much less memory and CPU time [3] and, to a certain extent, are more easily amenable to parallelism. They are, however, very sensitive to matrix conditioning and, as a result, the number of iterations necessary for their convergence increases with the matrix system size. Consequently, iterative methods are normally preconditioned. In other words, matrix system $\mathbf{Ax} = \mathbf{b}$ is replaced by $\mathbf{S}^{-1}\mathbf{Ax} = \mathbf{S}^{-1}\mathbf{b}$, where $\mathbf{S}$ is a preconditioning matrix, the inverse of which should be easy to build and as close as possible to $\mathbf{A}^{-1}$. In the field of computational fluid dynamics (CFD), the search for efficient preconditioners has been the subject of active research for more than 20 years [4]. We can cite, in particular, the preconditioners based on incomplete factorization (see, e.g., [5–7]), which perform very well even on three-dimensional unstructured meshes. We can also mention the multilevel preconditioners, which are of two types: the multigrid methods [8] that rely on a hierarchy of discretizations and the algebraic multilevel methods [9, 10] that only require the matrix of the linear system and have been shown to cope quite well with the unstructured grids of industrial CFD problems [11].

Choosing an iterative method (also called acceleration method) for the solution of a linear system cannot be done without some *a priori* knowledge of the properties of the related matrix. Furthermore, it is known that there is no such thing as a best overall iterative method. This fact is confirmed in the paper by Nachtigal *et al.* [12] that clearly shows that one can construct examples for which any given method outperforms others by some reasonable factor. For these reasons, numerous iterative methods have been developed over the years, including Krylov subspace methods such as the conjugate gradient method for symmetric positive definite systems and more recently methods such as CGS [13], Bi-CGSTAB [14], TFQMR [15], and GMRES [16] for nonsymmetric systems. A good survey of these methods can be found in [4].

1.3. *Solution Strategies for the Stokes Equations*

There exist different strategies for solving the Stokes equations, that is, system (3), where **A** stands for diffusion and inertia is neglected.

First, the mixed method consists of solving this system in a coupled manner, that is, for **V** and **P** at the same time. In such a case, the associated matrix is singular, which reflects the fact that the pressure in the momentum equation (1) is unique up to a constant. Since the matrix is symmetric indefinite, the conjugate gradient method is not applicable. As an alternative several authors such as Robichaud *et al.* [17], Rusten and Winther [18], Atanga and Wathen [19], and Ramage and Wathen [20] have shown the rather good ability of residual-type methods to solve the Stokes equations.

On the basis that (coupled) mixed methods are often impractical for the treatment of three-dimensional problems, mainly because of their memory requirements and their propensity to increase matrix bandwidth and spoil conditioning, considerable effort has been devoted to the development of efficient decoupled methods. One of them is the penalty method, which amounts to solving the modified momentum equation

$$(\mathbf{A} + r\mathbf{B}^T\mathbf{B})\mathbf{V} = \mathbf{F}, \tag{4}$$

where r is a penalty parameter. The pressure is eliminated from the formulation and can be retrieved through the relation

$$\mathbf{p} = -r\mathbf{B}\mathbf{V}. \tag{5}$$

When used in conjunction with direct methods for the solution of (4), the penalty method has proven to be both robust and accurate provided that the penalty parameter is large enough [21]. However, when the problem size dictates the choice of iterative solvers, its efficiency is known to decay significantly, the reason being that the condition number of the related linear system increases with the value of the penalty parameter. To alleviate this shortcoming, one can resort to the Uzawa algorithm [22] with $0 < \lambda < 2r$, where λ is a descent parameter:

0. Given $\mathbf{P}^{(0)}$
1. For $n = 0, 1, 2, \ldots$, until convergence
   1.1 Given $\mathbf{P}^{(n)}$, solve for $\mathbf{V}^{(n+1)}$ the primal problem
   $$(\mathbf{A} + r\mathbf{B}^T\mathbf{B})\mathbf{V}^{(n+1)} = \mathbf{F} - \mathbf{B}^T\mathbf{P}^{(n)}. \tag{6}$$
   1.2 Compute $\mathbf{P}^{(n+1)}$
   $$\mathbf{P}^{(n+1)} = \mathbf{P}^{(n)} - \lambda\mathbf{B}\mathbf{V}^{(n+1)}. \tag{7}$$

Step 1.2 in the Uzawa algorithm is nothing but a descent method for the solution of the so-called dual problem (also called the Schur complement)

$$\mathbf{B}\mathbf{V} = \mathbf{0} \Leftrightarrow \mathbf{B}(\mathbf{A} + r\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{P} = \mathbf{B}(\mathbf{A} + r\mathbf{B}^T\mathbf{B})^{-1}\mathbf{F}, \tag{8}$$

where the right-hand side of the equivalence results from the elimination of **V** using Eq. (6). One can then show that this linear system in pressure is symmetric positive definite and that its condition number decreases as r increases [22]:

$$\lim_{r\to\infty} \mathrm{cond}(\mathbf{B}(\mathbf{A} + r\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}) = 1. \tag{9}$$

In other words, choosing $r > 0$ can be viewed as one way to precondition the dual problem (8). Unfortunately, as mentioned before, the effect on the convergence of iterative solvers

for the solution of primal problem (6) is quite the opposite since

$$\lim_{r\to\infty} \text{cond}(\mathbf{A} + r\mathbf{B}^T\mathbf{B}) = +\infty. \tag{10}$$

In practice, it turns out that the selection of an optimal trade-off value for the penalty parameter is a rather delicate operation. One possibility is to simply drop this penalty parameter and rely upon other forms of preconditioning. Atanga and Silvester [23] even consider that preconditioning the dual problem is of rather limited interest since matrix $\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T$ has a condition number that is independent of mesh size h, if one uses a stable mixed finite element [24] or a stabilized finite element such as the locally stabilized $Q_1 - P_0$ element introduced by Silvester and Kechkar [25]. Along the same line, Verfürth [26] developed a robust and proficient nonpreconditioned Uzawa solver based on a multigrid method for the solution of the underlying primal problem. We will show in this paper that preconditioning the dual matrix can lead to a significant decrease in the number of Uzawa iterations.

To speed up the convergence of the Uzawa algorithm, Cahouet and Chabard [27] proposed solving both the primal and dual problems by means of a preconditioned conjugate gradient method. In their work, an incomplete factorization was used to precondition the primal problem and an efficient preconditioner, based on a Fourier analysis of the divergence operator, was devised for the dual problem. Developed in the context of the 3D continuous-pressure $P_2 - P_1$ Taylor–Hood element, this dual preconditioner has also been employed by Carriere and Jeandel [28] for the solution of nonisothermal fluid flow problems with the 3D continuous-pressure $P_1 - P_1$ *iso* P_2 element. It has since been extended to the case of the 3D discontinuous-pressure $Q_2 - P_0$ and $Q_2 - P_1$ brick elements by Zhou [29] and to the case of the 2D locally stabilized $Q_1 - P_0$ element by Vincent [30].

1.4. *Objective of This Work*

Despite the greater flexibility of tetrahedra over bricks for the meshing of complex geometries and the risk inherent to the use of nonlocally mass-conserving continuous-pressure finite elements as shown by Pelletier *et al.* [31], no work has been reported on the extension of the Krylov-based Uzawa algorithm proposed by Cahouet and Chabard [27] to the case of 3D discontinuous-pressure tetrahedral elements. The objective of this paper then consists of introducing an efficient and robust Krylov-based Uzawa algorithm for the solution of the 3D Stokes equations with discontinuous-pressure tetrahedral finite elements, following along the lines of Cahouet and Chabard [27]. In particular, we will show that not preconditioning the dual problem can result in a significant reduction of the convergence speed of the iterative method and, in some cases, to the divergence of this method. To this end, many dual preconditioners will be presented and compared through various steady-state and unsteady-state problems.

The outline of the remainder of the paper is as follows. In Section 2, the discontinuous-pressure tetrahedral finite elements used in this work are presented. In Section 3, two Stokes solvers are described: the *preconditioned conjugate gradient Uzawa algorithm* (PCGU), as introduced by Cahouet and Chabard [27], and the seemingly very fast *incomplete Uzawa algorithm* (IU) of Robichaud *et al.* [32], a Stokes solver that has been used by the authors for more than 10 years and implemented in the commercial finite element program POLY3D™ from Rheotek Inc. Next, Section 4 is divided into two subsections. In the first, which relates to the steady-state Stokes equations, the convergence properties of the PCGU algorithm are analyzed and compared, through the selection of benchmark problems, with those

obtained in the literature for other finite element types. Comparison is also made with the IU algorithm and shows the superiority of the PCGU algorithm in many cases. In the second subsection, which relates to the unsteady-state Stokes equations, the convergence properties of the PCGU algorithm are investigated. In particular, the efficiency of three different preconditioners for the dual problem is assessed with respect to the size of the time step via a dimensionless group called the mesh Reynolds number.

2. DISCRETIZATION

We consider the solution of the Stokes equations. Discretization is achieved using discontinuous-pressure tetrahedral finite elements. In this work, three such elements will be considered [33]: the 8-node linear $P_1^+ - P_0$ element, which comprises 24 velocity degrees of freedom and 1 pressure degree of freedom at the element centroid, the 14-node linear $P_2^+ - P_0$, which is composed of 42 velocity degrees of freedom and 1 pressure degree of freedom, and the 15-node quadratic $P_2^+ - P_1$ Crouzeix–Raviart element, which contains 45 velocity degrees of freedom and 4 pressure degrees of freedom located at the element centroid (Fig. 1). In this latter case, a classical static condensation can be applied to eliminate

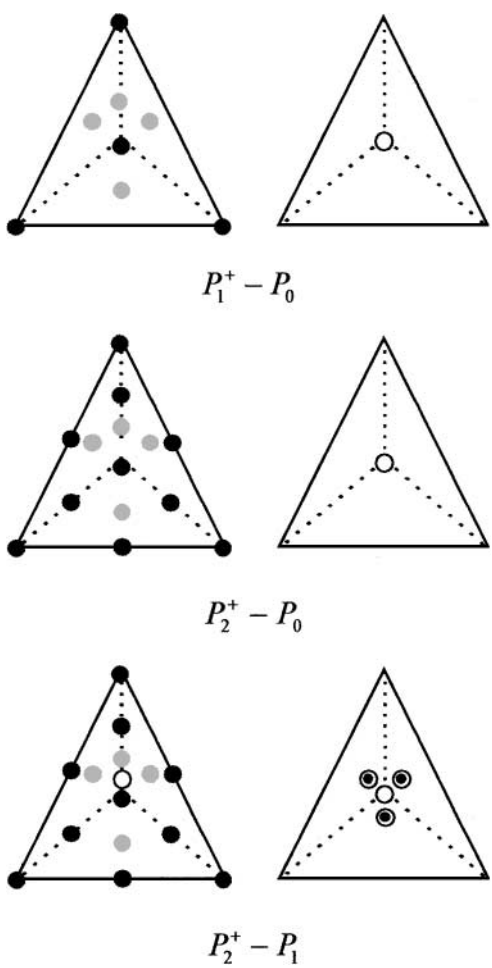

FIG. 1. Discontinuous pressure tetrahedral finite elements. ●, vertex or edge node; ●, face node; ○, centroid node; ◉, centroid node (gradient components).

the three internal velocity degrees of freedom. Alternatively, one can opt for the trick of Fortin and Fortin [34] to eliminate the three internal velocity degrees of freedom as well as the three degrees of freedom related to the pressure gradient. As this procedure incurs almost no extra cost, computational effort then becomes equivalent to that for the constant-pressure 14-node $P_2^+ - P_0$ element.

3. CONJUGATE-GRADIENT-BASED UZAWA SOLVERS

3.1. *Preconditioned Conjugate-Gradient Uzawa Algorithm*

Following along the lines of Cahouet and Chabard [27], we propose to use for the solution of the Stokes equations the following preconditioned conjugate-gradient Uzawa algorithm:

0. Given $\mathbf{P}^{(0)}$
1. Solve the primal problem
$$\mathbf{A}\mathbf{V}^{(0)} = \mathbf{F} - \mathbf{B}^T\mathbf{P}^{(0)}. \tag{11}$$
2. Solve the dual problem (8) using the preconditioned conjugate-gradient method
   2.1 Initializations
   $b^{(0)} = 0,$
   $\mathbf{D}^{(0)} = 0.$
   2.2 For $n = 1, 2, \ldots,$ until convergence
   2.2.1 $\mathbf{D}^{(n)} = \mathbf{C}^{-1}\mathbf{B}\mathbf{V}^{(n-1)} + b^{(n-1)}\mathbf{D}^{(n-1)},$
   2.2.2 $\mathbf{Z}^{(n)} = \mathbf{A}^{-1}\mathbf{B}^T\mathbf{D}^{(n)},$
   2.2.3 $a^{(n)} = \dfrac{\left(\mathbf{B}\mathbf{V}^{(n-1)}, \mathbf{C}^{-1}\mathbf{B}\mathbf{V}^{(n-1)}\right)}{\left(\mathbf{C}^{-1}\mathbf{B}\mathbf{V}^{(n-1)}, \mathbf{B}\mathbf{Z}^{(n)}\right)},$
   2.2.4 $\mathbf{P}^{(n)} = \mathbf{P}^{(n-1)} + a^{(n)}\mathbf{D}^{(n)},$
   2.2.5 $\mathbf{V}^{(n)} = \mathbf{V}^{(n-1)} + a^{(n)}\mathbf{Z}^{(n)},$
   2.2.6 $b^{(n)} = \dfrac{\left(\mathbf{B}\mathbf{V}^{(n)}, \mathbf{C}^{-1}\mathbf{B}\mathbf{V}^{(n)}\right)}{\left(\mathbf{B}\mathbf{V}^{(n-1)}, \mathbf{C}^{-1}\mathbf{B}\mathbf{V}^{(n-1)}\right)}.$

In this algorithm, the primal problems, that is, step 2.2.2 and Eq. (11), which is nothing but Eq. (6) with $r = 0$, are solved classically using a conjugate-gradient method preconditioned by an incomplete factorization (ILU). Note that this preconditioner does not appear explicitly in the algorithm shown here for the sake of brevity and that other preconditioners such those based on (algebraic) multilevel methods could be used.

The solution of dual problem (8) is also obtained by means of a preconditioned conjugate-gradient method. No penalty term is used to enhance the convergence speed of the dual iterations because of the detrimental effect it might have on the convergence speed of the primal iterations. Instead, preconditioning can be based on a discretization of the operator proposed by Cahouet and Chabard [27],

$$\mathbf{C}^{-1} = \frac{\mu}{\rho}\mathbf{I}^{-1} - \frac{1}{dt}\Delta^{-1}, \tag{12}$$

where dt represents the time step (for the time integration scheme), $\mathbf{I}$ is the identity operator, and Δ is the pressure Laplacian. With such a choice, one can prove that if the divergence of the velocity is taken to be a harmonic function, then the descent parameter in the Uzawa algorithm becomes independent of the frequency. Consequently, it should

be effective at getting rid of both the (high-frequency) oscillatory and the (low-frequency) smooth components of the residual, a well-known property of multilevel methods [35].

For steady-state problems, preconditioner (12) simply becomes

$$\mathbf{C}^{-1} = \frac{\mu}{\rho}\mathbf{I}^{-1}, \tag{13}$$

whereas for strongly unsteady-state cases, that is, whenever the mesh Reynolds number,

$$\mathrm{Re}_h = \frac{\rho h^2}{\mu\, dt}, \tag{14}$$

is much larger than 1, it simplifies to

$$\mathbf{C}^{-1} = -\frac{1}{dt}\Delta^{-1}. \tag{15}$$

The latter is nothing but the preconditioner proposed by Labadie and Lasbleiz [36] in the case of unsteady fluid flow problems.

The discretization of the first term in (12) is straightforward as it amounts to assembling a pressure mass matrix $\mathbf{M_p}$. The same cannot be said of the second term, which is a pressure Laplacian. One possibility consists of directly discretizing it to obtain the so-called classical Laplacian along with its nongeneral and debatable homogeneous Neumann boundary conditions. This is, for instance, the approach that is advocated by Zhou [29] when dealing with the $Q_2 - P_0$ brick. As the pressure is approximated by one degree of freedom at the centroid of each element, this author suggests resorting to the finite volume method for the computation of the terms $\mathbf{C}^{-1}\mathbf{B}\mathbf{V}^{(n)}$ and $\mathbf{C}^{-1}\mathbf{B}\mathbf{V}^{(n-1)}$ that appear in the algorithm. This strategy is, however, limited to constant pressure approximations and it does not extend easily to the case of linear (discontinuous- or continuous-) pressure approximations. Of course, in the discontinuous case, one may employ the trick of Fortin and Fortin [34] to condense the (nonconstant) Hermitian part of the pressure. In any case, there is no doubt that using the finite volume method in the context of unstructured meshes of $P_1^+ - P_0$ finite elements would be more cumbersome.

Alternatively, the second term in (12) can be discretized by means of what is commonly referred to as the compatible Laplacian, leading to

$$\mathbf{C}^{-1} = \frac{\mu}{\rho}\mathbf{M}_\mathbf{p}^{-1} - \frac{1}{dt}\left(\mathbf{B}\mathbf{M}_\mathbf{v}^{-1}\mathbf{B}^T\right)^{-1}, \tag{16}$$

which has the appealing advantage of entailing pressure boundary conditions that conform to the velocity field. However, this necessitates the computation of the inverse of velocity mass matrix $\mathbf{M_v}$, a full matrix that makes this approach impracticable in 3D for storage reasons. As a result, many researchers have sought alternative expressions similar to (16). In particular, Zhou [29] suggested replacing $\mathbf{M_v}$ by its lumped version $\mathbf{M}_\mathbf{v}^{\mathrm{lumped}}$:

$$\mathbf{C}^{-1} = \frac{\mu}{\rho}\mathbf{M}_\mathbf{p}^{-1} - \frac{1}{dt}\left(\mathbf{B}\left(\mathbf{M}_\mathbf{v}^{\mathrm{lumped}}\right)^{-1}\mathbf{B}^T\right)^{-1}. \tag{17}$$

He reported a slightly better convergence rate than that obtained with a preconditioner based on the diagonal of $\mathbf{M_v}$, that is,

$$\mathbf{C}^{-1} = \frac{\mu}{\rho}\mathbf{M}_\mathbf{p}^{-1} - \frac{1}{dt}(\mathbf{B}\,\mathrm{diag}(\mathbf{M_v})^{-1}\mathbf{B}^T)^{-1}, \tag{18}$$

as suggested by Cahouet and Chabard [27] and Carriere and Jeandel [28].

It is clear that the second term in (18) dominates for large values of the mesh Reynolds number. In such a case, $\mathbf{A} \sim \mathbf{M_v}$ so that an alternative to preconditioner (18) is given by

$$\mathbf{C}^{-1} = \frac{\mu}{\rho}\mathbf{M_p}^{-1} - \frac{1}{dt}(\mathbf{B}\,\mathrm{diag}(\mathbf{A})^{-1}\mathbf{B}^T)^{-1}, \tag{19}$$

which can be easily built because the matrix $\mathbf{A}$ is readily available. This preconditioner can be viewed as an extension of the preconditioner [28]

$$\mathbf{C}^{-1} = (\mathbf{B}\,\mathrm{diag}(\mathbf{A})^{-1}\mathbf{B}^T)^{-1}, \tag{20}$$

which, albeit rather inefficient for steady-state flows [29], has proven to be superior in the case of unsteady problems [27, 29] to the preconditioners (16)–(18) and the one involving the classical discretization of the pressure Laplacian. In fact, as will be shown later, the extra term in (19) will make this preconditioner most efficient in extremal cases (steady state and strongly unsteady) as well as in intermediate cases, the proper amount of weighting for the two terms being adjusted as a function of the mesh Reynolds number (14) to ensure rapid convergence in all situations.

3.2. *Incomplete Uzawa Algorithm*

As was just seen in Section 3.1, each iteration of the preconditioned conjugate gradient Uzawa algorithm requires at step 2.2.2 the iterative solution of a linear system with coefficient matrix $\mathbf{A}$. The following question then arises: How accurate should the solution to this linear system be so as to maintain the overall convergence of the Uzawa algorithm? In other words, for each outer iteration, how many inner iterations should be performed?

A first answer may be based upon observation. First, let us denote by $\mathbf{R}_p$ the primal residual, that is, the residual corresponding to the inner loop 2.2.2 (as well as to that of Eq. (11)), and by $\mathbf{R}_d = \mathbf{B}\mathbf{V}^{(n+1)}$ the dual residual. Next, let us consider in the case of the primal and dual problems respectively that the iterative methods are converged when

$$\left\|\mathbf{R}_p^{(k)}\right\| \le \varepsilon_p \left\|\mathbf{R}_p^{(0)}\right\| \tag{21}$$

and

$$\left\|\mathbf{R}_d^{(k)}\right\| \le \varepsilon_d \left\|\mathbf{R}_d^{(0)}\right\|, \tag{22}$$

where k stands for the kth iterate. The issue is then to decide how to choose tolerance values ε_p and ε_d that will ensure the convergence of the overall algorithm. For instance, Ramage and Wathen [20] and Wille [37] reported that ε_p should be smaller than 10^{-3} and that

$$\frac{\varepsilon_d}{\varepsilon_p} \ge 10 \tag{23}$$

proved to be a good choice in many circumstances. Our experience has revealed that condition (23) is normally sufficient provided ε_d is not too small; otherwise, greater accuracy is generally required for the solution of the primal problem. The results presented in Section 4 were obtained with $\varepsilon_p = 10^{-8}$ and $\varepsilon_d = 10^{-6}$.

It is interesting to note that, as far back as in the early 1980s, Fortin and Glowinski [22] had observed that using a relatively small number of iterations was sufficient to maintain the same convergence rate for the Uzawa algorithm as that obtained with an exact solve of

the inner primal problem. This observation has paved the way to the development of what is now referred to as *inexact Uzawa algorithms*. For instance, Elman and Golub [38] suggested a method in which, for each outer (dual) iteration, inner iterations are performed until

$$\|\mathbf{R}_p\| \leq \tau \|\mathbf{R}_d\|, \tag{24}$$

where τ is a constant parameter. In other words, the accuracy of the solution to the inner (primal) problem is gradually improved as the dual residual decreases. Such an approach that approximates $\mathbf{A}^{-1}$ through an iterative process is called a *nonlinear inexact Uzawa algorithm* [39], in contrast to *linear inexact Uzawa algorithms*, which replace $\mathbf{A}^{-1}$ by the inverse of a linear preconditioner. An example of this latter method is given by the *incomplete Uzawa algorithm* of Robichaud *et al.* [32], which we recall here since its efficiency will be compared in the next section with that of the preconditioned conjugate gradient Uzawa algorithm:

0. Given $\mathbf{P}^{(0)}$
1. For $m = 1, 2, \ldots,$ until convergence
   - 1.1 Primal problem
     - 1.1.1 $\mathbf{R}^{(m)} = \mathbf{F} - \mathbf{B}^T\mathbf{P}^{(m)} - (\mathbf{A} + r\mathbf{B}^T\mathbf{B})\mathbf{V}^{(m)}$,
     - 1.1.2 $c^{(m)} = \dfrac{\left(\mathbf{R}^{(m)}, \mathbf{S}^{-1}\mathbf{R}^{(m)}\right)}{\left((\mathbf{A} + r\mathbf{B}^T\mathbf{B})\mathbf{S}^{-1}\mathbf{R}^{(m)}, \mathbf{S}^{-1}\mathbf{R}^{(m)}\right)}$,
     - 1.1.3 $\mathbf{V}^{(m+1)} = \mathbf{V}^{(m)} + c^{(m)}\mathbf{S}^{-1}\mathbf{R}^{(m)}$.
   - 1.2 Dual problem

     For $n = 1, 2, \ldots, N_d$,
     - 1.2.1 $\mathbf{V}^{(n)} = \mathbf{V}^{(n+1)}$, if $n = 1$,
     - 1.2.2 $\mathbf{P}^{(n)} = \mathbf{P}^{(n+1)}$, if $n = 1$,
     - 1.2.3 $\mathbf{D}^{(n)} = \begin{cases} \mathbf{BV}^{(n)}, \text{ if } n = 1 \\ \mathbf{BV}^{(n)} + b^{(n)}\mathbf{D}^{(n-1)}, \text{ else,} \end{cases}$

       with $b^{(n)} = \dfrac{\left(\mathbf{BV}^{(n)}, \mathbf{BV}^{(n)}\right)}{\left(\mathbf{BV}^{(n-1)}, \mathbf{BV}^{(n-1)}\right)}$,
     - 1.2.4 $\mathbf{Z}^{(n)} = \mathbf{S}^{-1}\mathbf{B}^T\mathbf{D}^{(n)}$, (25)
     - 1.2.5 $a^{(n)} = \dfrac{\left(\mathbf{BV}^{(n)}, \mathbf{BV}^{(n)}\right)}{\left(\mathbf{BV}^{(n)}, \mathbf{BZ}^{(n)}\right)}$,
     - 1.2.6 $\mathbf{P}^{(n+1)} = \mathbf{P}^{(n)} + a^{(n)}\mathbf{D}^{(n)}$,
     - 1.2.7 $\mathbf{V}^{(n+1)} = \mathbf{V}^{(n)} + a^{(n)}\mathbf{Z}^{(n)}$.
   - 1.3 Updates
     - 1.3.1 $\mathbf{V}^{(m+1)} = \mathbf{V}^{(n+1)}$,
     - 1.3.2 $\mathbf{P}^{(m+1)} = \mathbf{P}^{(n+1)}$.

In this algorithm, $\mathbf{S}^{-1}$ represents an incomplete factorization of matrix $\mathbf{A}$ while N_d stands for the number of dual iterations performed after each primal iteration; the results presented in Section 4 were obtained with $N_d = 2$. As just mentioned, the conjugate gradient method that is used for the solution of the dual problem is modified; step 1.2.4 should indeed read as

$$\mathbf{Z}^{(n)} = (\mathbf{A} + r\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{D}^{(n)}, \tag{26}$$

which, even if solved with an iterative solver, remains a much more time-consuming task than the forward and backward substitutions inherent to (25). From another perspective, the incomplete Uzawa algorithm belongs to the class of Arrow–Hurwicz algorithms, which

can be considered as nonlinear, inexact Uzawa algorithms [40] since the primal problem (step 1.1) is solved incompletely by one iteration of a descent method. It is thus inexact (or incomplete) for two reasons.

4. NUMERICAL RESULTS

To better appreciate the convergence properties of the iterative methods, we introduce the convergence rate for the dual problem,

$$\aleph = \left[\frac{\left\|R_d^{(n)}\right\|}{\left\|R_d^{(0)}\right\|}\right]^{\frac{1}{n}} = \left[\frac{\left\|\mathbf{BV}^{(n)}\right\|}{\left\|\mathbf{BV}^{(0)}\right\|}\right]^{\frac{1}{n}}, \tag{27}$$

where n is the iterate number.

4.1. *Steady-State Flow*

In this section, the efficiency of both the preconditioned conjugate gradient Uzawa (PCGU) method and the incomplete Uzawa (IU) algorithm for the solution of the steady-state Stokes equations is assessed by way of the following three-dimensional benchmark problems:

- lid-driven cavity flow problem;
- 4 : 1 contraction flow problem;
- Poiseuille flow problem.

We first consider a unit cubic cavity with the upper wall moving at a constant velocity of 1 m/s. A no-slip boundary condition is imposed on the other walls. Three meshes are used, the characteristics of which are summarized in Table I. These meshes are similar to those used by Cahouet and Chabard [27] and Zhou [29].

Figure 2 shows a graph of the convergence behavior of the conjugate gradient Uzawa algorithm (preconditioned with preconditioner (13) or nonpreconditioned) with respect to mesh size for the $P_1^+ - P_0$ and $P_2^+ - P_1$ finite elements. The following properties can be noted:

- for the $P_1^+ - P_0$ and $P_2^+ - P_1$ elements, the convergence speed does not depend upon the mesh size, a behavior that is theoretically justified [24] since both elements satisfy the Brezzi–Babuska stability condition;

TABLE I
Characteristics of the Meshes Used for the Lid-Driven Cavity Problem

Mesh	Element type	Mesh size (h)	Number of elements	Number of nodes	Number of velocity equations
Coarse	$P_1^+ - P_0$	0.2	936	2280	5430
Intermediate	$P_1^+ - P_0$	0.1	7792	17,806	47,472
Refined	$P_1^+ - P_0$	0.075	18,144	41,119	113,037
Coarse	$P_2^+ - P_1$	0.2	936	4559	10,863
Intermediate	$P_2^+ - P_1$	0.1	7792	35,611	94,947
Refined	$P_2^+ - P_1$	0.075	18,144	82,237	226,077

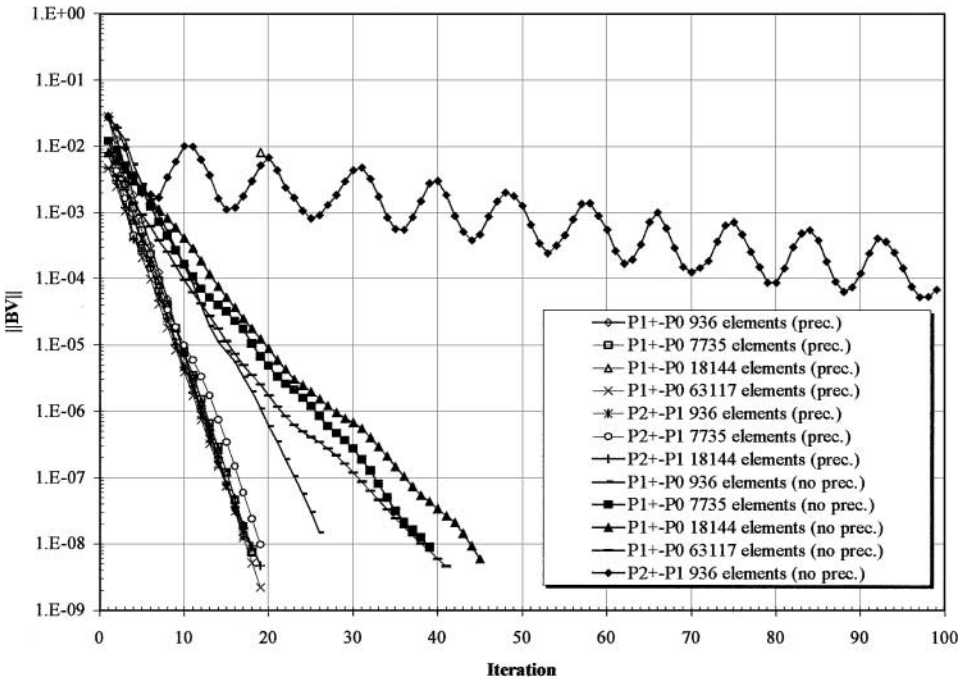

FIG. 2. Convergence of the conjugate gradient Uzawa algorithm with and without preconditioning, for the lid-driven cavity flow problem.

- the use of a preconditioner results in a significant reduction of the number of iterations, all the more so in the case of the $P_2^+ - P_1$ element;
- preconditioning is necessary for the $P_2^+ - P_1$ element.

Remark. The convergence speed of the dual problem does not depend on the mesh size for finite elements satisfying the Brezzi–Babuska condition. For the primal problem, it can be shown [3] that the number of iterations is proportional to h^{-1}.

The rates of convergence are presented in Table II and are compared to those obtained by Cahouet and Chabard [27] and Zhou [29] with the PCGU algorithm. Since all the elements under investigation are stable, the rates of convergence barely vary with mesh size, except for the case of the coarse mesh of $Q_2 - P_0$ elements. It appears that the rates of convergence for the continuous-pressure $P_2 - P_1$ finite element, $0.56 < \aleph < 0.58$, are not as good as those obtained for the discontinuous-pressure $P_1^+ - P_0$ and $P_2^+ - P_1$ finite elements used in this work, $0.42 < \aleph < 0.48$, and those reported by Zhou [29] for the discontinuous-pressure

TABLE II
Convergence Rates for the Lid-Driven Cavity Flow Problem

	This work			Cahouet and Chabard (27)	Zhou (29)	
Mesh	$P_1^+ - P_0$	$P_2^+ - P_1$	$P_2^+ - P_0$	$P_2 - P_1$	$Q_2 - P_0$	$Q_2 - P_1$
Coarse	0.43	0.42	0.33	0.58	0.28	0.41
Intermediate	0.45	0.48	0.35	0.56	0.37	0.44
Refined	0.46	0.47	0.32	0.57	0.40	0.45

$Q_2 - P_0$ and $Q_2 - P_1$ finite elements, $0.28 < \aleph < 0.45$. In the former case, the pressure degrees of freedom are interrelated, which means that more work is needed to satisfy the weak form of the continuity equation since updating the pressure is a task that must be done in a global manner. On the other hand, in the latter cases, the pressure degrees of freedom are independent of one another: Each Lagrange multiplier, that is, each pressure degree of freedom, has for its support a single finite element, which means that the discretized continuity equation can be worked out elementwise.

We were intrigued by the somewhat better rates of convergence obtained by Zhou [29] for the $Q_2 - P_0$ finite element, $0.28 < \aleph < 0.40$, as opposed to $0.41 < \aleph < 0.45$ for the $Q_2 - P_1$ finite element. But as shown in Table II, simulations carried out for the $P_2^+ - P_0$ finite element, which is the tetrahedral counterpart of the $Q_2 - P_0$ element, revealed a similar trend. A plausible explanation is that, in the case of the $P_2^+ - P_0$ and $Q_2 - P_0$ finite elements, the number of velocity degrees of freedom, n_v, is large with respect to the number of pressure degree of freedom, n_p. For these two element types, the *constraint ratio* defined as [41]

$$r = \frac{n_v}{n_p} \tag{28}$$

takes on the values $51/10$ and 21 respectively. As these two values are greater than 3, the number of dimensions, it follows that the incompressibility may be poorly approximated, in which case one says that these elements are "too soft" [42]. Projecting the velocity field onto a divergence-free subspace is then easier to accomplish, hence leading to a faster convergence of the PCGU algorithm.

Figure 3 shows a graph of the convergence behavior of the IU algorithm with respect to mesh size for the $P_1^+ - P_0$ finite element. Let us mention that, in almost all the simulations that were attempted, the IU algorithm failed to converge with the $P_2^+ - P_1$ element. This

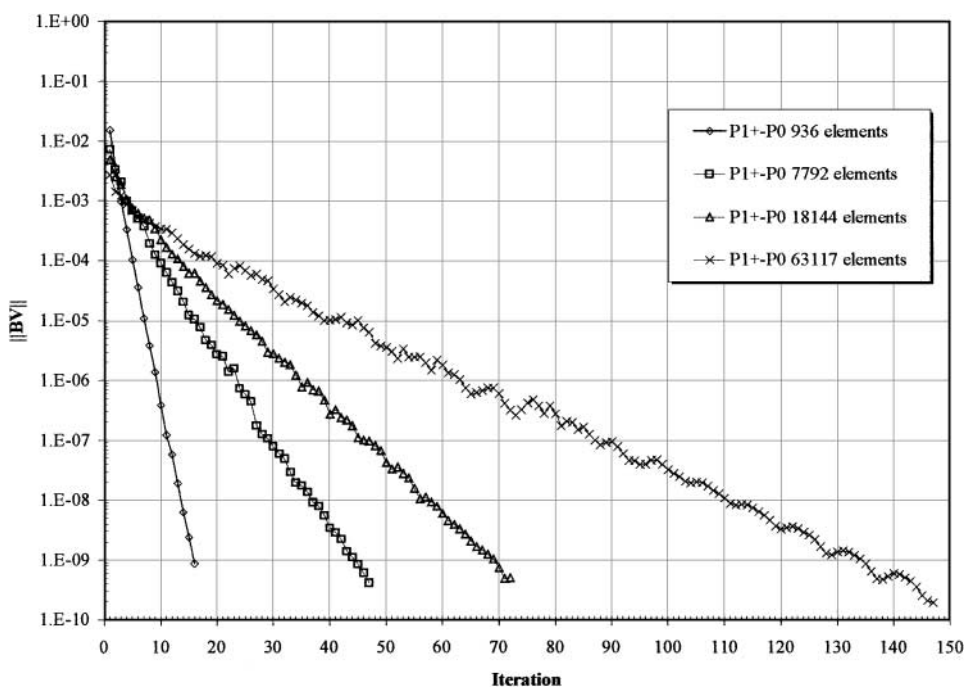

FIG. 3. Convergence of the IU algorithm for the lid-driven cavity flow problem.

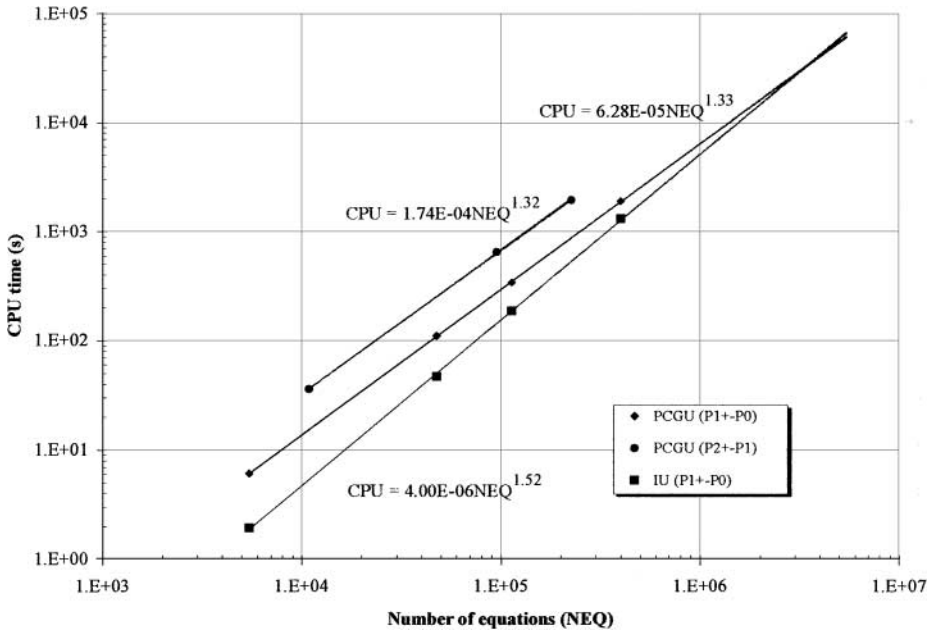

FIG. 4. Graph of CPU time vs number of velocity equations for the lid-driven cavity flow problem.

lack of robustness is related to the use of approximation (25), which, however, appears to be sufficient for the $P_1^+ - P_0$ element. One possible explanation might be the poor conditioning resulting from the use of fourth-degree polynomials in the case of the $P_2^+ - P_1$ element [33].

One may also observe that, for the $P_1^+ - P_0$ element, the number of dual iterations depends on the number of equations. The reason for this is that each time the pressure is modified the velocity is not fully updated as with the PCGU algorithm. Instead, at every other dual iteration ($N_d = 2$), one single primal iteration is carried out. As a result, the average cost of one dual iteration is less than that with the PCGU algorithm. The following question then naturally arises: In terms of CPU time, which of the two methods is the more efficient? The answer to this question can be found in Fig. 4, which shows a graph of the CPU time versus the number of velocity equations (NEQ) for the IU and the PCGU algorithms. Similar exponents, that is, $\sim$1.33, are obtained with the PCGU algorithm for both element types. It is interesting to note that in spite of an exponent of 1.52, the IU algorithm outperforms the PCGU algorithm whenever the number of velocity equations is less than 3,000,000.

As mentioned before, the convergence of the IU algorithm is erratic when $P_2^+ - P_1$ elements are used. It appears that its performance is also unpredictable with the $P_1^+ - P_0$ element when it is used to simulate fluid flow inside an open geometry. To illustrate this fact, we consider the 4 : 1 contraction problem. One single mesh was generated, the characteristics of which are summarized in Table III.

The following boundary conditions were specified with $\mathbf{v} = (v_x, v_y, v_z)$, where z represents the main flow direction:

- $v_z = 1$ m/s (Dirichlet boundary condition) at the inlet;
- $\mathbf{v} = 0$ m/s (Dirichlet boundary condition) on the solid wall;
- $v_u = v_v = 0$ m/s (Dirichlet boundary condition) and free v_z (homogeneous Neumann boundary condition) at the outlet.

TABLE III
Characteristics of the Meshes Used for the 4 : 1 Contraction Problem

Mesh	Element type	Number of elements	Number of nodes	Number of velocity equations (Dirichlet boundary conditions at inlet)	Number of velocity equations (Neumann boundary conditions at inlet)
Intermediate	$P_1^+ - P_0$	8298	19,377	50,163	50,470
Intermediate	$P_2^+ - P_1$	8298	38,753	100,328	100,941

In Fig. 5, one may readily note the deficiency of the IU algorithm and the fast convergence of the PCGU algorithm for this problem.

As can be seen in Fig. 6, the convergence of the PCGU algorithm is in fact *superlinear*, which means that its convergence rate decreases during the iterative process. Such a phenomenon, which has been studied by Van der Sluis and Van der Vorst [43], is related to the behavior of the extremal Ritz values and how quickly these converge to the corresponding eigenvalues, thereby reducing the "apparent" condition number of the system matrix.

We believe that the imposition of Dirichlet boundary conditions at the inlet is responsible for the divergence of the IU algorithm. Such a choice combined with a zero-velocity field inside the domain initially leads to a non-zero-divergence velocity field as an initial solution for the algorithm. For reasons not completely understood (approximation (25) is undoubtedly a key factor), the IU algorithm may fail to converge in these situations. It has been observed that one way to alleviate this problem consists of replacing the Dirichlet boundary conditions at the inlet by equivalent Neumann boundary conditions. Assuming that the flow

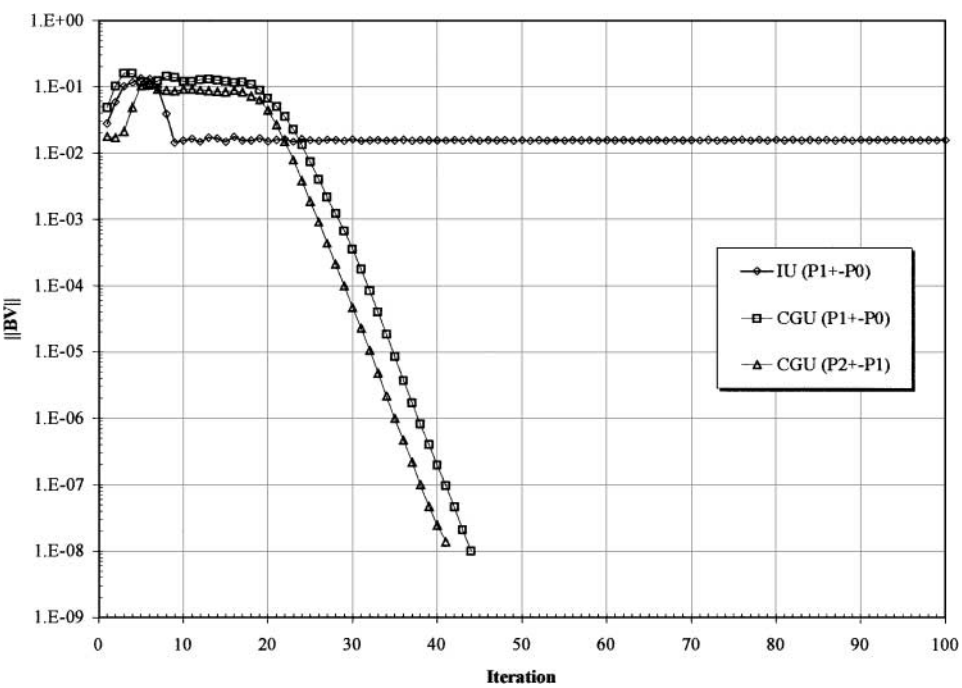

FIG. 5. Graph of the convergence of the PCGU and IU algorithms for the 4 : 1 contraction problem (with Dirichlet boundary conditions at the inlet).

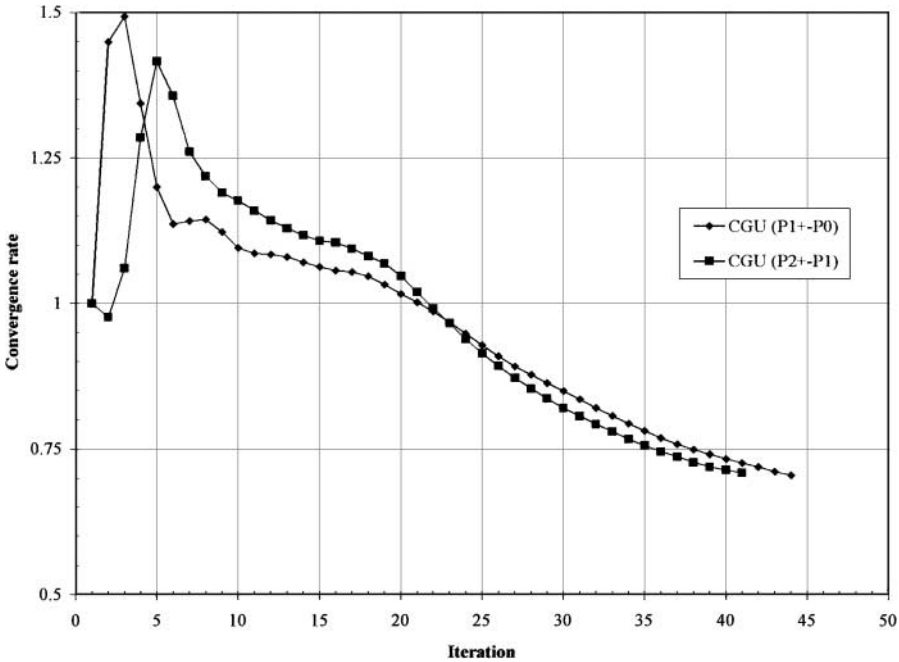

FIG. 6. Graph of the convergence rate $\aleph$ versus the iterations for the PCGU algorithm.

is fully developed there, this amounts to imposing a value for the pressure. Such a choice leads to a zero-divergence velocity field as an initial solution for the algorithm. As shown in Fig. 7, the convergence of the IU is indeed better when Neumann boundary conditions are imposed at the inlet; if iterations end up in a crawl for the $P_2^+ - P_1$ element (the algorithm

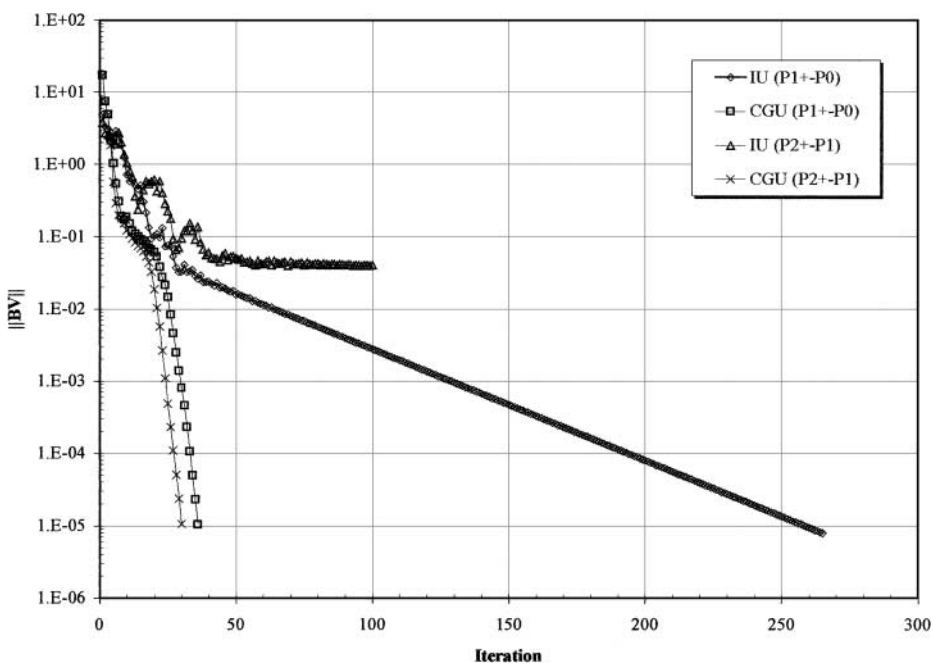

FIG. 7. Graph of the convergence of the PCGU and IU algorithms for the 4 : 1 contraction problem (with Neumann boundary conditions at the inlet).

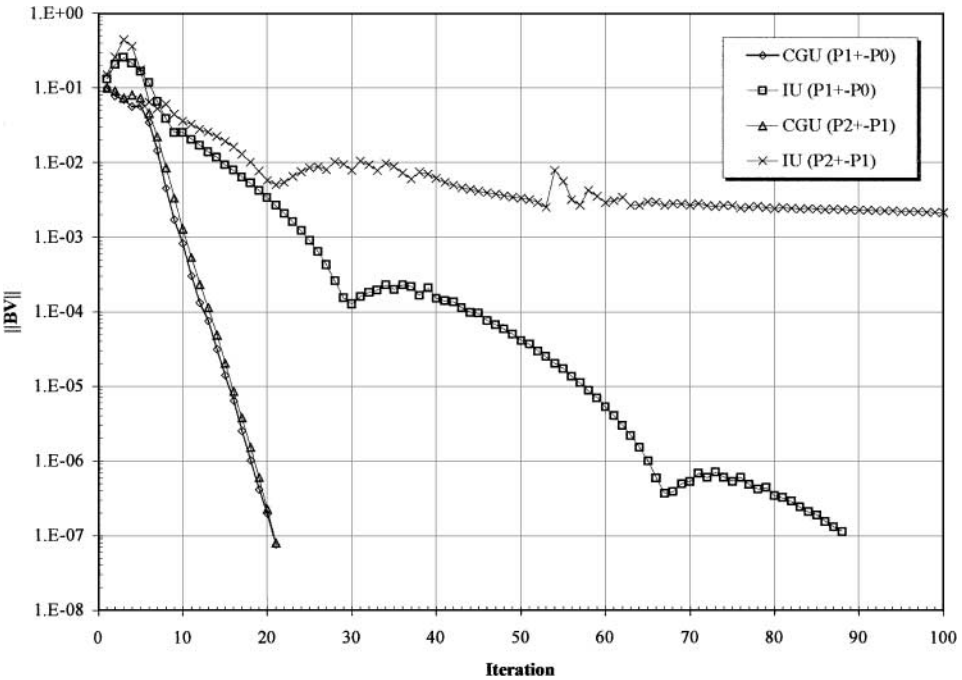

FIG. 8. Graph of the convergence of the PCGU and IU algorithms for the Poiseuille flow problem (with Dirichlet boundary conditions at the inlet).

was stopped after 100 iterations), a rather smooth albeit slow convergence is obtained for the $P_1^+ - P_0$. Nevertheless, in this latter case, the PCGU algorithm is much faster than the IU algorithm, the CPU times being 136 (41 iterations) and 206 (265 iterations) seconds respectively.

It must be emphasized that for some open geometry problems, convergence has been observed for the $P_1^+ - P_0$ element with inlet Dirichlet boundary conditions. Figure 8, which shows a graph of the convergence behavior of the IU and PCGU algorithms for a Poiseuille flow problem, illustrates this fact. The characteristics of the meshes used for the simulations can be found in Table IV. The boundary conditions are similar to those of the 4 : 1 contraction problem. One may note that, for the $P_2^+ - P_1$ element, the IU algorithm is ineffective with a rate of convergence $\aleph \sim 1$. Even though fewer iterations are required for convergence, it appears that the PCGU algorithm is slower (79 s) than the IU algorithm in the case of the $P_1^+ - P_0$ element (68 s).

In summary, the previous results have assessed the efficiency of the preconditioned conjugate-gradient Uzawa algorithm for the solution of the steady-state Stokes equations

TABLE IV
Characteristics of the Meshes Used for the Poiseuille Flow Problem

Mesh	Element type	Number of elements	Number of nodes	Number of velocity equations
Intermediate	$P_1^+ - P_0$	8298	19,377	50,163
Intermediate	$P_2^+ - P_1$	8298	38,753	100,328

with discontinuous-pressure tetrahedral finite elements. We conclude that

- the PCGU algorithm is robust;
- the number of dual iterations is independent of h (for elements satisfying the Brezzi–Babuska stability condition); and
- the execution time varies as $NEQ^{1.33}$, where NEQ is the number of velocity equations.

Alternatively, we can recommend the use of the incomplete Uzawa algorithm in the following situations where it is more efficient than the preconditioned conjugate-gradient algorithm:

- with the $P_1^+ - P_0$ element only;
- in confined geometries, when the number of velocity equations is not too large, that is, when there are fewer than 3,000,000.

4.2. *Unsteady-State Flow*

The solution of the unsteady-state Stokes equations with the preconditioned conjugate-gradient Uzawa algorithm is now considered. Consequently, matrix $\mathbf{A}$ includes a mass term to account for the time derivative in the momentum equation. It was observed in the previous section that, for the steady-state case, the number of dual iterations is independent of the number of velocity equations. Our goal is now to assess the efficiency of the preconditioners introduced in Section 3 with respect to the mesh Reynolds number Re_h defined by (14). For each of the benchmark problems that follow, the kinematic viscosity μ/ρ and the mesh size h will be constant so that increasing (resp. decreasing) Re_h is equivalent to decreasing (resp. increasing) the size of the time step dt. Consequently, our goal will also be to verify to what extent these preconditioners lead to schemes for which the number of dual iterations is independent of dt.

The following preconditioners, discussed in Section 3.1, will be investigated:

- $\mathbf{C}^{-1} = \frac{\mu}{\rho}\mathbf{M}_\mathbf{p}^{-1}$, hereafter called the *diagonal preconditioner*;
- $\mathbf{C}^{-1} = \frac{1}{dt}(\mathbf{B}\,\mathrm{diag}(\mathbf{A})^{-1}\mathbf{B}^T)^{-1}$, hereafter called the *compatible Laplacian preconditioner*;
- $\mathbf{C}^{-1} = \frac{\mu}{\rho}\mathbf{M}_\mathbf{p}^{-1} - \frac{1}{dt}(\mathbf{B}\,\mathrm{diag}(\mathbf{A})^{-1}\mathbf{B}^T)^{-1}$, hereafter called the *optimal preconditioner*; and
- none.

The efficiency of these preconditioners will be assessed for the $P_1^+ - P_0$ finite element using the lid-driven cavity flow problem (structured and unstructured meshes; simple geometry) and an open-channel contraction flow problem arising from some extrusion process (unstructured mesh; more complex geometry). The characteristics of the meshes used for these problems are given in Table V.

Figure 9 shows the influence of Re_h on the number of dual iterations for the PCGU algorithm to converge in the case of the lid-driven cavity flow problem and a structured mesh. One may observe that the number of dual iterations increases with the value of Re_h and that this number varies but slightly with the type of preconditioner used. A closer look reveals the following:

- at all values of Re_h, not using a preconditioner results in more dual iterations;
- at low values of Re_h, that is, for weakly unsteady-state flows, the diagonal preconditioner is better than the compatible Laplacian preconditioner;

TABLE V
Characteristics of the Meshes Used for the Lid-Driven and the Open-Channel Contraction Flow Problems

Problem	Element type	Number of elements	Number of nodes	Number of velocity equations
Lid-driven cavity flow (structured mesh)	$P_1^+ - P_0$	5000	11,931	30,387
Lid-driven cavity flow (unstructured mesh)	$P_1^+ - P_0$	7792	17,806	47,472
Open-channel contraction flow	$P_1^+ - P_0$	1256	3961	7964

- at large values of Re_h, that is, for strongly unsteady-state flows, the compatible Laplacian preconditioner is better than the diagonal preconditioner;
- at all values of Re_h, the optimal and the compatible Laplacian preconditioners are equivalent.

All these properties comply with the theory discussed in the previous section. The fact that the differences in terms of the number of dual iterations needed to reach convergence with each preconditioner are small are due to the regularity of the geometry and the use of a structured mesh. To prove this, we solved the lid-driven cavity flow problem using an unstructured mesh. In this case, conditioning is not as good and one would expect more significant differences in the performance of the various preconditioners. As can be seen in Fig. 10, the differences are indeed more important. In particular, one may

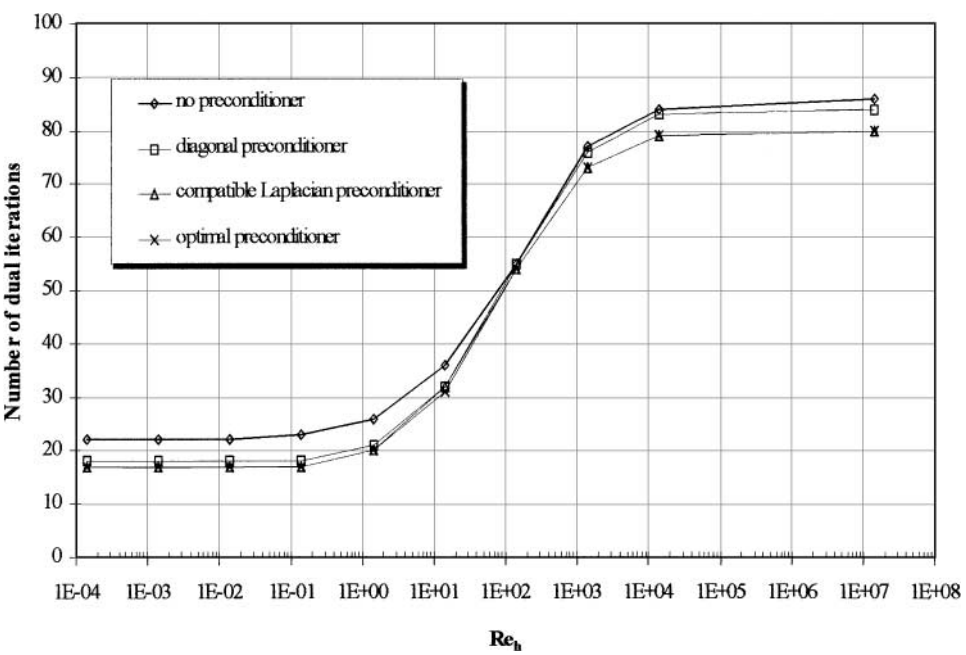

FIG. 9. Graph of the convergence of the PCGU algorithm for the cavity flow problem and a structured mesh of 5000 $P_1^+ - P_0$ elements.

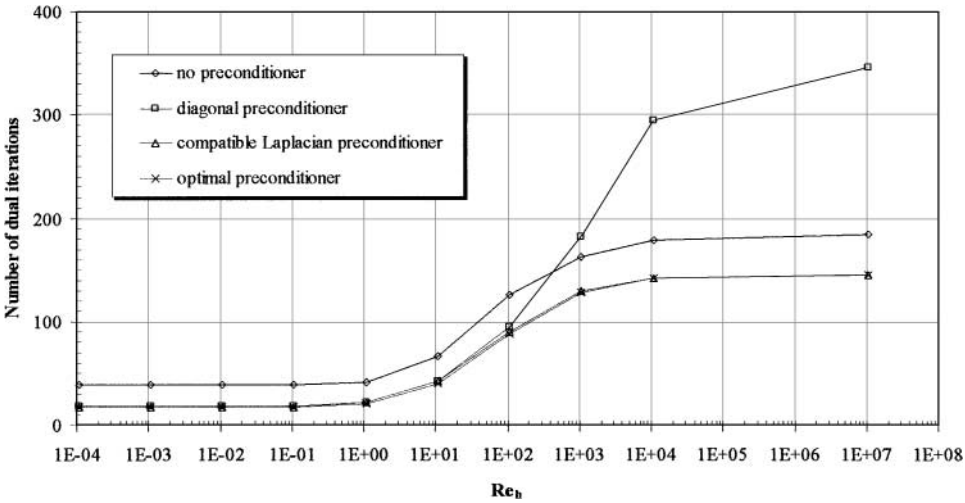

FIG. 10. Graph of the convergence of the PCGU algorithm for the cavity flow problem and an unstructured mesh of 7792 $P_1^+ - P_0$ elements.

notice that

- at large values of Re_h, that is, for strongly unsteady-state flows, the diagonal preconditioner performs poorly, as expected, and is even outperformed by the nonpreconditioned PCGU algorithm and
- as with the structured mesh, at all values of Re_h, the optimal and the compatible Laplacian preconditioners are equivalent.

This last property is a bit surprising since, theoretically, the optimal preconditioner should perform better than the compatible Laplacian preconditioner, at least at small values of Re_h. We believe that this phenomenon is related to the geometric regularity of the cavity. To investigate the influence of the regularity, we solved an open-channel contraction flow problem arising from some extrusion process. The boundary conditions for this problem are similar to those for the 4 : 1 contraction considered in the previous section. The influence of Re_h on the number of dual iterations is shown in Fig. 11. As can be seen, the superiority of the optimal preconditioner over the diagonal and the compatible Laplacian preconditioners is quite clear for this problem. One may also observe that, at small values of Re_h, the optimal preconditioner is equivalent to the diagonal preconditioner and that at large values of Re_h, it is equivalent to the compatible Laplacian preconditioner. As mentioned in the previous section, the optimal preconditioner is most efficient in extremal cases (steady and strongly unsteady) as well as in intermediate cases, the proper amount of weighting for the two terms of (19) being adjusted as a function of Re_h. In fact, Fig. 11 shows that, for values of Re_h between 0 and 2.84, which corresponds to values of dt between ∞ (steady state) and 10^{-4}, the number of dual iterations is between 327 and 613 without preconditioning, 27 and 234 for diagonal preconditioner, 31 and 132 for the compatible Laplacian preconditioner, and 27 and 31 for the optimal preconditioner. These numbers show that, for a wide range of values of dt, the optimal preconditioner outperforms the other preconditioners and that the number of dual iterations required by this preconditioner to reach convergence barely varies with dt. This latter property complies with the recent findings of Kobelkov and Olshanskii [44] in the case of the continuous Stokes equations in regular domains.

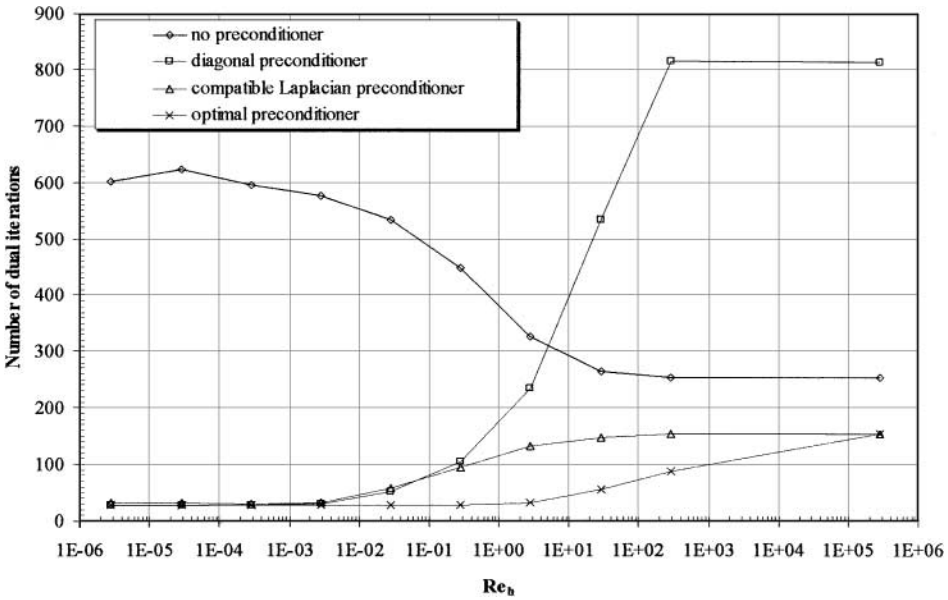

FIG. 11. Graph of the convergence of the PCGU algorithm for the open-channel contraction flow problem and an unstructured mesh of 1256 $P_1^+ - P_0$ elements.

Finally, we mention that the preconditioned conjugate gradient Uzawa algorithm has been tested in our group on a fairly large amount of problems and that, each time, the use of the optimal preconditioner proposed in this paper has led to convergence in a robust and efficient manner. This solver has also been implemented in the commercial finite element program POLY3D from Rheotek Inc. The superiority of the optimal preconditioner over the other preconditioners in all the cases presented in this work does not comply with the findings of Carriere and Jeandel [28], who reported (for a different element type) that the compatible Laplacian preconditioner gave the best performance. As for the work by Zhou [29], comparisons are diffcult to make since he tested the variant (17) of the optimal preconditioner (18) advocated by Cahouet and Chabard [27].

5. CONCLUDING REMARKS

The objective of this work consisted of developing an efficient and robust Krylov-based Uzawa algorithm for the solution of the three-dimensional steady-state and unsteady-state Stokes equations with discontinuous-pressure tetrahedral finite elements. To this end, a class of preconditioned conjugate-gradient Uzawa algorithms were presented and their convergence properties were analyzed and compared through the solution of a selection of benchmark problems. A comparison was also made with an inexact Uzawa algorithm for the steady-state case. It was shown that the preconditioned conjugate gradient Uzawa algorithm represents a robust and efficient solver for the Stokes equations and that in particular:

- the PCGU algorithm is more robust than the IU algorithm and more efficient in many situations;
- preconditioning the dual problem is indeed very important for the convergence of the PCGU algorithm;

• the optimal preconditioner (18), which is an extension to discontinuous-pressure tetrahedral finite elements of the preconditioner proposed by Cahouet and Chabard [27], outperforms the diagonal and the compatible Laplacian preconditioners for all values of Re_h in the case of problems with complex geometries; and

• the use of the optimal preconditioner leads to a version of the PCGU algorithm for which the number of dual iterations is independent of the time step.

Finally, a forthcoming paper will be devoted to the development of Krylov-based Uzawa algorithms for the solution of the unsymmetric Oseen equations, such as those that arise from the linearization of the Navier–Stokes equations through Newton's method.

ACKNOWLEDGMENTS

The financial contributions of NSERC and Paprican are gratefully acknowledged.

REFERENCES

1. P. A. Tanguy, M. Fortin, and L. Choplin, Finite element simulation of dip coating, II: Non-Newtonian fluids, *Int. J. Numer. Methods Fluids* **4**, 459 (1984).
2. R. Glowinski and O. Pironneau, Finite element methods for Navier–Stokes equations, *Annu. Rev. Fluid. Mech.* **24**, 167 (1992).
3. O. Axelsson and V. A. Barker, *Finite Element Solution of Boundary Value Problems* (Academic Press, San Diego 1984).
4. R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods* (Soc. for Industr. & Appl. Math., Philadelphia, 1984).
5. H. P. Langtangen, Conjugate gradient methods and ILU preconditioning of non-symmetric matrix systems with arbitrary sparsity patterns, *Int. J. Numer. Methods Fluids* **9**, 213 (1989).
6. O. Dahl and S. Wille, An ILU preconditioner with coupled node fill-in for iterative solution of the mixed finite element formulation of the 2D and 3D Navier–Stokes equations, *Int. J. Numer. Methods Fluids* **15**, 525 (1992).
7. F. F. Campos and N. R. C. Birkett, An efficient solver for multi-right-hand-side linear systems based on the CCCG(η) method with applications to implicit time-dependent partial differential equations, *SIAM J. Sci. Comput.* **19**(1), 126 (1998).
8. W. Hackbush, *Multigrid Methods and Applications* (Springer-Verlag, Berlin, 1985).
9. O. Axelsson and P. Vassilevski, Algebraic multilevel preconditioning methods, I, *Numer. Math.* **56**, 157 (1989).
10. O. Axelsson and P. Vassilevski, Algebraic multilevel preconditioning methods, II, *Numer. Math.* **57**, 1569 (1990).
11. G. Poole and Y. C. Liu, Advancing analysis capabilities in ANSYS through solver technology, presented at the 10th Copper Mountain Conf. on Multigrid Methods, Copper Mountain, Colorado, 2001.
12. N. Nachtigal, S. Reddy, and L. Trefethen, How fast are nonsymmetric matrix iterations, *SIAM J. Matrix Anal. Appl.* **13**, 778 (1992).
13. P. Sonneveld, CGS, a fast Lanczos-type solver for non-symmetric linear systems, *SIAM J. Sci. Stat. Comput.* **10**, 36 (1989).
14. H. Van der Vorst, A fast and smoothly converging variant of bi-CG for the solution of nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.* **13**, 631 (1992).
15. R. W. Freund, A transpose-free quasi-minimum residual algorithm for non-Hermitian linear systems, *SIAM J. Sci. Comput.* **14**, 470 (1993).
16. Y. Saad and M. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.* **7**, 856 (1986).
17. M. Robichaud and P. A. Tanguy, Finite element solution of three-dimensional incompressible fluid flow problems by a preconditioned conjugate residual method, *Int. J. Numer. Methods Eng.* **24**, 447 (1987).

18. T. Rusten and R. Winther, A preconditioned iterative method for saddle point problems, *SIAM J. Matrix Anal. Appl.* **13**(3), 887 (1992).
19. J. Atanga and A. Wathen, Iterative methods for stabilized mixed velocity–pressure finite elements, *Int. J. Numer. Methods Fluids* **14**, 71 (1992).
20. A. Ramage and A. Wathen, Iterative solution techniques for the Stokes and Navier–Stokes equations, *Int. J. Numer. Methods Fluids* **19**, 67 (1994).
21. J. N. Reddy, On penalty function methods in the finite element analysis of fluid flow, *Int. J. Numer. Methods Fluids* **2**, 151 (1982).
22. M. Fortin and R. Glowinski, *Augmented Lagrangian Methods* (North-Holland, Amsterdam, 1983).
23. J. Atanga and D. Silvester, Iterative methods for stabilized mixed velocity–pressure finite elements, *Int. J. Numer. Methods Fluids* **14**, 71 (1992).
24. U. Langer and W. Queck, On the convergence factor of Uzawa's algorithm, *J. Comput. Appl. Math.* **15**, 191 (1986).
25. D. J. Silvester and N. Kechkar, Stabilized bilinear-constant velocity–pressure finite elements for the conjugate gradient solution of the Stokes problem, *Comput. Methods Appl. Mech. Eng.* **79**, 71 (1990).
26. R. Verfürth, A combined conjugate gradient–multigrid algorithm for the numerical solution of the Stokes problem, *IMA J. Numer. Anal.* **4**, 441 (1984).
27. J. Cahouet and J. P. Chabard, Some fast 3D finite element solvers for the generalized Stokes problem, *Int. J. Numer. Methods Fluids* **8**, 869 (1988).
28. P. Carriere and D. Jeandel, A 3D finite element method for the simulation of thermoconvective flows and its performances on a vector-parallel computer, *Int. J. Numer Methods Fluids* **12**, 929 (1991).
29. R. Q. N. Zhou, Preconditioned finite element algorithms for 3D Stokes flow, *Int. J. Numer. Methods Fluids* **17**, 667 (1993).
30. C. Vincent, The influence of the stabilization parameter on the convergence factor of iterative methods for the solution of the discretized Stokes problem, *Int. J. Numer. Methods Fluids* **20**, 1237 (1995).
31. D. Pelletier, A. Fortin, and R. Camarero, Are FEM solutions of incompressible flows really incompressible? (Or how simple flows can cause headaches!), *Int. J. Numer. Methods Fluids* **9**, 99 (1989).
32. M. P. Robichaud, P. A. Tanguy, and M. Fortin, An iterative implementation of the Uzawa algorithm for 3D fluid flow problems, *Int. J. Numer. Methods Fluids* **10**, 429 (1990).
33. F. Bertrand, M. Gadbois, and P. A. Tanguy, Tetrahedral elements for fluid flow problems, *Int. J. Numer. Methods Fluids* **33**, 1251 (1992).
34. M. Fortin and A. Fortin, Experiments with several elements for viscous incompressible flows, *Int. J. Numer. Methods Fluids* **5**, 911 (1985).
35. W. L. Briggs, *A Multigrid Tutorial* (Soc. for Industr. & Appl. Math., Philadelphia, 1987).
36. G. Labadie and P. Lasbleiz, *Quelques méthodes de résolution du problème de Stokes en éléments finis*, EDF Report HE41/83.01 (1983).
37. S. O. Wille, A preconditioned alternating inner–outer iterative solution method for the mixed finite element formulation of the Navier–Stokes equations, *Int. J. Numer. Methods Fluids* **18**, 1135 (1994).
38. H. C. Elman and G. H. Golub, Inexact and preconditioned Uzawa algorithms for saddle point problems, *SIAM J. Numer. Anal.* **31**(6), 1645 (1994).
39. J. H. Bramble, J. E. Pasciak, and A. T. Vassilev, Analysis of the inexact Uzawa algorithm for saddle point problems, *SIAM J. Numer. Anal.* **34**(3), 1072 (1997).
40. W. Queck, The convergence factor of preconditioned algorithms of the Arrow–Hurwicz type, *SIAM J. Numer. Anal.* **26**(4), 1016 (1989).
41. T. J. R. Hughes, *The Finite Element Method* (Prentice Hall, New York, 1987).
42. M. Fortin and A. Fortin, Newer and newer elements for incompressible flow, in *Finite Elements in Fluids 6*, edited by R. H. Gallagher, G. F. Carey, J. T. Oden, and O. C. Zienckiewicz (Wiley, New York, 1985).
43. A. Van der Sluis and H. A. Van der Vorst, The rate of convergence of conjugate gradients, *Numer. Math.* **48**, 543 (1986).
44. G. M. Kobelkov and M. A. Olshanskii, Effective preconditioning of Uzawa type schemes for a generalized Stokes problem, *Numer. Math.* **86**, 443 (2000).