INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 12, 929–946 (1991)

A 3D FINITE ELEMENT METHOD FOR THE SIMULATION OF THERMOCONVECTIVE FLOWS AND ITS PERFORMANCES ON A VECTOR–PARALLEL COMPUTER

PHILIPPE CARRIERE AND DENIS JEANDEL

Laboratoire de Mécanique des Fluides et d'Acoustique, URA CNRS 263, Ecole Centrale de Lyon, 36 Avenue Guy de Collongue, BP 163, F-69131 Ecully Cedex, France

SUMMARY

A three-dimensional finite element method for the simulation of thermoconvective flows is presented. Vector–parallel performances of some preconditioned conjugate gradient methods are compared for solving both large linear systems and the Stokes problem. As significant examples, numerical experiments on the steady two- and three-dimensional Rayleigh–Bénard convection at high Prandtl number are reported.

KEY WORDS Three-dimensional flows Thermoconvective flow Finite element method Preconditioned conjugate gradient methods Rayleigh–Bénard convection

1. INTRODUCTION

During recent years, a large number of computational methods have been proposed for the numerical solution of problems encountered in fluid dynamics. For three-dimensional flows, important difficulties occur when a large number of degrees of freedom are required to obtain accurate approximations of the solution fields. With regard to this major constraint, the problems of dynamical stability such as onset of convection, transition and chaotic dynamics have been investigated in three-dimensional geometries using well adapted pseudo-spectral or finite difference methods.[1–5] In this way it is possible to observe complex chaotic or non-chaotic flow regimes. Nevertheless, the above numerical methods, which take advantage of the use of structured grids, cannot be easily extended to complex industrial geometries.

Recently, the finite element technique has emerged as a new discipline to calculate complex flow patterns. The great success of the finite element method is mainly attributed to its generality and ability to handle complex geometries of industrial problems.[6–9] However, the main drawback of the method, associated with unstructured grids, is that it is time-consuming and less adaptable to vectorization techniques. Nevertheless, recent improvements in supercomputer hardware and software permit increased possibilities of the finite element technique (vectorization, parallelization).

In the present paper a basic finite element method to solve the Navier–Stokes equations is improved in several ways in order to be well adapted to the architecture of the most recent computers. In Section 2 the basic numerical method is described. It is a fractional step method in time and a classical Galerkin approximation in space. The first improvement of the standard

0271–2091/91/100929–18$09.00

Received November 1989
Revised July 1990

method is concerned with the solution of linear systems. Several preconditionings of the iterative conjugate gradient method are analysed; typical economical storages of the matrices appearing in the algorithm are simultaneously considered. The second major improvement concerns the Stokes problem. A preconditioned conjugate gradient version of the Uzawa algorithm is used. Three preconditioners are tested and compared. Finally, the global performances of the optimized algorithm to solve the Navier–Stokes equations are presented and the capabilities of the corresponding programme taking advantage of parallel computer performances are underlined. As an example, the algorithm is applied in Section 6 to simulate typical flow patterns encountered in Rayleigh–Bénard convection. Two-dimensional flow configurations are first obtained and the results are compared with both experiments and analytical asymptotic solutions. Three-dimensional flow patterns are also obtained for typical initial and boundary conditions. The results are discussed and compared with experiments.

2. BASIS OF THE NUMERICAL METHOD

Basic equations

Let us consider a Newtonian fluid flow in a domain Ω of boundary Γ, governed by the standard conservation equations for mass, momentum and energy. These equations are simplified using the classical Boussinesq approximations, such that a linear relation between temperature and density may be assumed and all density variations may be neglected except in the buoyancy force term of the momentum equation. Such an approximation is usually justified for both gases and liquids if small temperature variations are considered (for more details see References 10–12).

Let us denote by p, $\mathbf{u}$ and T the pressure, velocity and temperature fields respectively, and introduce L, U, $T_1 - T_0$ and ρ_0 as length, velocity, temperature and density scales respectively; the set of dimensionless governing equations is then

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{1}{Re}\nabla^2\mathbf{u} - \frac{Gr}{Re^2}T\mathbf{z} + \nabla p = 0, \\ \nabla\cdot\mathbf{u} = 0, \\ \frac{\partial T}{\partial t} + (\mathbf{u}\cdot\nabla)T - \frac{1}{RePr}\nabla^2 T = 0, \end{aligned} \tag{1}$$

where Re, Gr and Pr are the classical Reynolds, Grashof and Prandtl numbers respectively and $\mathbf{z}$ denotes the unit vector in the vertical direction. Equations (1) are solved in a domain Ω with suitable initial and boundary conditions on Γ.

Note that the introduction of the velocity scale U allows the consideration of both free and forced convection; for the buoyancy-driven flows considered here, U was chosen such that $Gr = Re^2$.

Time discretization

For three space dimensions the time discretization is a crucial point of the solution method. On the one hand, a fully implicit formulation may be used (see e.g. Reference 6) which takes advantage of the unconditional stability of the numerical solution field; consequently, an iterative method is required in order to solve the non-linear equations deduced from the discretization. Unfortunately, the convergence rate of such an iterative method is strongly related to the preconditioning method used. In advection-dominated flows an efficient preconditioning is

obtained when a quite 'small' value of the time step is used, increasing the influence of the identity operator. Then, the advantage of the implicit method is somewhat reduced. A similar problem arises when satisfying the continuity constraint.

On the other hand, explicit Euler schemes (see e.g. Reference 13) lead to very stringent stability conditions on the time step, more particularly connected to the spatial discretization. Moreover, the finite element spatial discretization requires a mass-lumping technique to simplify linear system solutions and the continuity constraint cannot be solved explicitly.

In the present paper the applied fractional step time discretization is of a semi-implicit type and has a truncation time error $O(\Delta t^2)$. At time $t+\Delta t$, $\mathbf{u}(t+\Delta t)$, $p(t+\Delta t)$ and $T(t+\Delta t)$ (denoted by $\mathbf{u}^{n+1}$, p^{n+1} and T^{n+1}) are obtained from $\mathbf{u}(t)$, $p(t)$ and $T(t)$ (i.e. $\mathbf{u}^n$, p^n and T^n) through the following three time steps.

Step 1

$$\frac{4}{\Delta t}\mathbf{u}^{n+1/4}-\frac{2}{3Re}\nabla^2\mathbf{u}^{n+1/4}+\nabla p^{n+1/4}=\frac{4}{\Delta t}\mathbf{u}^n+\frac{1}{3Re}\nabla^2\mathbf{u}^n+\frac{Gr}{Re^2}T^n\mathbf{z}-(\mathbf{u}^n\cdot\nabla)\mathbf{u}^n,$$

$$\nabla\cdot\mathbf{u}^{n+1/4}=0,$$

$$\frac{4}{\Delta t}T^{n+1/4}-\frac{2}{3RePr}\nabla^2T^{n+1/4}+(\mathbf{u}^{n+1/4}\cdot\nabla)T^{n+1/4}=\frac{4}{\Delta t}T^n+\frac{1}{3RePr}\nabla^2T^n. \tag{2a}$$

Step 2

$$\frac{2}{\Delta t}T^{n+3/4}-\frac{1}{3RePr}\nabla^2T^{n+3/4}=\frac{2}{\Delta t}T^{n+1/4}+\frac{2}{3RePr}\nabla^2T^{n+1/4}-(\mathbf{u}^{n+1/4}\cdot\nabla)T^{n+1/4},$$

$$\frac{2}{\Delta t}\mathbf{u}^{n+3/4}-\frac{1}{3Re}\nabla^2\mathbf{u}^{n+3/4}+(\mathbf{u}^{n+1/4}\cdot\nabla)\mathbf{u}^{n+3/4}-\frac{Gr}{Re^2}T^{n+3/4}\mathbf{z}$$

$$=\frac{2}{\Delta t}\mathbf{u}^{n+1/4}+\frac{2}{3Re}\nabla^2\mathbf{u}^{n+1/4}+\nabla p^{n+1/4}. \tag{2b}$$

Step 3

$$\frac{4}{\Delta t}\mathbf{u}^{n+1}-\frac{2}{3Re}\nabla^2\mathbf{u}^{n+1}+\nabla p^{n+1}=\frac{4}{\Delta t}\mathbf{u}^{n+3/4}+\frac{1}{3Re}\nabla^2\mathbf{u}^{n+3/4}+\frac{Gr}{Re^2}T^{n+3/4}\mathbf{z}$$

$$-\tfrac{1}{3}[(4\mathbf{u}^{n+3/4}-\mathbf{u}^{n+1/4})\cdot\nabla]\mathbf{u}^{n+3/4},$$

$$\nabla\cdot\mathbf{u}^{n+1}=0,$$

$$\frac{4}{\Delta t}T^{n+1}-\frac{2}{3RePr}\nabla^2T^{n+1}+(\mathbf{u}^{n+1}\cdot\nabla)T^{n+1}=\frac{4}{\Delta t}T^{n+3/4}+\frac{1}{3RePr}\nabla^2T^{n+3/4}. \tag{2c}$$

In Steps 1 and 3 the velocity and pressure fields $(\mathbf{u}^{n+1/4}, p^{n+1/4})$ and $(\mathbf{u}^{n+1}, p^{n+1})$ are solutions of standard Stokes problems. In the momentum equations the advection and buoyancy force terms are in explicit form. In contrast, the advection term in the temperature equation is fully implicit. In Step 2 the advection term of the temperature equation is explicit while that of the momentum equation is semi-implicit. Two main features of the scheme may be emphasized. First, a wide range of efficient methods for solving the Stokes problems are available. Secondly, the

discretized equations are linear and widely decoupled (only connected by the divergence condition), so the discrete spatial approximation will directly lead to linear system solutions. The main deficiency of the time discretization is the fact that the stability condition induced by the explicit contributions is not known exactly (even in the continuous case). Consequently, the stability of the numerical solution has to be verified during the computation. In practice, an adaptive time step obtained from the estimation of the truncation time error of the solution is incorporated in the algorithm.

Finite element discretization

In Steps 1 and 3 (equations (2a) and (2c)) the set of coupled equations governing the velocity and pressure fields may be written in the following form:

$$\left.\begin{aligned} \alpha\mathbf{u}-\nu\nabla^2\mathbf{u}+\nabla p &= \mathbf{f} \\ \nabla\cdot\mathbf{u} &= 0 \\ \mathbf{u}_{|\Gamma} &= \mathbf{g} \end{aligned}\right\} \quad \text{with } \alpha=\frac{4}{\Delta t}, \quad \nu=\frac{2}{3Re}. \tag{3}$$

Using a weak formulation and imposing the integral continuity condition $\int \mathbf{g}\cdot\mathbf{n}\,d\Gamma=0$, a well-posed problem can be obtained in Hilbert space $(H_1(\Omega))^3\times L_2(\Omega)$ for the pair $(\mathbf{u}, p)$.[14] It is also known from the works of Babuška[15] and Brezzi[16] that the discrete formulation of this problem imposes the so-called inf–sup condition to the discrete solution spaces $X_h\times M_h$ for the velocity and pressure fields. A wide range of spaces are available and a lot of them can be found in References 14 and 17.

Here the so-called P_1–P_1 iso P_2 approximation[18] is chosen: the discrete pressure p_h is piecewise linear and continuous on a standard coarse mesh $\mathfrak{T}_h$, and the discrete velocity $\mathbf{u}_h$ is also piecewise linear and continuous, but on a refined mesh $\mathfrak{T}_{h/2}$ obtained from $\mathfrak{T}_h$ by dividing each tetrahedron into eight subtetrahedrons (see Figure 1). In the following, n_h and $n_{h/2}$ denote the number of nodes of $\mathfrak{T}_h$ and $\mathfrak{T}_{h/2}$ respectively. The approximation spaces are thus defined by

$$M_h=\{q_h\in C^0(\Omega);\ q_h\in P_1(e_h^k),\ \forall e_h^k\in\mathfrak{T}_h\}, \tag{4a}$$

$$X_h=\{\mathbf{v}_h\in(C^0(\Omega))^3;\ \mathbf{v}_h\in(P_1(e_{h/2}^k))^3,\ \forall e_{h/2}^k\in\mathfrak{T}_{h/2};\ \mathbf{v}_{h_{|\Gamma}}=\mathbf{g}\}. \tag{4b}$$

Compared to the very closed P_1–P_2 approximation (where p_h is similarly discretized and $\mathbf{u}_h$ piecewise quadratic and continuous on $\mathfrak{T}_h$—tetrahedrons with 10 nodes), the P_1–P_1 iso P_2 approximation leads both to simpler integrations on the tetrahedrons and to sparser matrices. Another advantage is its easy implementation, especially in the three-dimensional case. It may also be noticed that geometrical information on the refined mesh is easily deduced from the coarse one.

The global forms of the coupled finite element equations are derived from the Galerkin residual method, and the discrete Stokes problems become

$$\begin{aligned} &\text{find } (\mathbf{u}_h^{(i)}, p_h^{(i)})\in X_h\times M_h \text{ such that} \\ &\alpha(\mathbf{u}_h^{(i)}, \mathbf{v}_h)_{L_2^3}+\nu(\nabla\mathbf{u}_h^{(i)}, \nabla\mathbf{v}_h)_{L_2^3}+(\nabla p_h^{(i)}, \mathbf{v}_h)_{L_2^3}=(\mathbf{f}_h^{(i)}, \mathbf{v}_h)_{L_2^3}, \quad \forall\mathbf{v}_h\in X_{0h}, \\ &(\nabla\cdot\mathbf{u}_h^{(i)}, q_h)_{L_2}=0, \quad \forall q_h\in M_{0h}, \end{aligned} \tag{5}$$

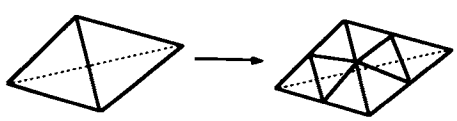

Figure 1. The P_1 and P_1 iso P_2 elements

where

$$M_{0h} = M_h \cap L_0^2(\Omega), \qquad i = \begin{cases} n+1/4, \\ n+1, \end{cases} \tag{6a}$$

$$X_{0h} = \{\mathbf{v}_h \in (C^0(\Omega))^3;\ \mathbf{v}_h \in (P_1(e_{h/2}^k))^3, \forall e_{h/2}^k \in \mathfrak{T}_{h/2};\ \mathbf{v}_{h_{|\Gamma}} = 0\}. \tag{6b}$$

The same procedure applied to the temperature leads to the final form of discretized equations

find $T_h^{(i)} \in Y_h$ such that

$$\alpha(T_h^{(i)}, s_h)_{L_2} + \kappa(\nabla T_h^{(i)}, \nabla s_h)_{L_2} + (\mathbf{u}_h^{(i)} \cdot \nabla T_h^{(i)}, s_h)_{L_2} = (b_h^{(i)}, s_h)_{L_2}, \quad \forall s_h \in Y_{0h}, \tag{7}$$

where

$$i = \begin{cases} n+1/4, \\ n, \end{cases}$$

$$Y_{0h} = \{s_h \in C^0(\Omega);\ s_h \in P_1(e_{h/2}^k),\ \forall e_{h/2}^k \in \mathfrak{T}_{h/2};\ s_{h_{|\Gamma_1}} = 0;\ \partial s_h / \partial n_{|\Gamma_2} = 0\}, \tag{8}$$

$$\Gamma_1 \cup \Gamma_2 = \Gamma$$

and $b_h^{(i)}$ is the right-hand-side term of the temperature equation.

The discrete formulation of Step 2 is deduced in the same manner as in (7), and one has

find $T_h^{n+3/4} \in Y_h$ such that

$$2\alpha(T_h^{n+3/4}, s_h)_{L_2} + 2\kappa(\nabla T_h^{n+3/4}, \nabla s_h)_{L_2} = (b_h^{n+3/4}, s_h)_{L_2}, \quad \forall s_h \in Y_{0h}, \tag{9}$$

find $\mathbf{u}_h^{n+3/4} \in X_h$ such that

$$2\alpha(\mathbf{u}_h^{n+3/4}, \mathbf{v}_h)_{L_2^3} + 2\nu(\nabla \mathbf{u}_h^{n+3/4}, \nabla \mathbf{v}_h)_{L_2^3} + ((\mathbf{u}_h^{n+1/4} \cdot \nabla)\mathbf{u}_h^{n+3/4}, \mathbf{v}_h)_{L_2^3} = (\mathbf{f}_h^{n+3/4}, \mathbf{v}_h)_{L_2^3}, \quad \forall \mathbf{v}_h \in X_{0h}. \tag{10}$$

Equations (7)–(9) lead to one linear system for each equation and each primitive variable.

3. PERFORMANCES OF SOME CONJUGATE GRADIENT METHODS APPLIED TO LINEAR SYSTEM SOLUTIONS

The finite element formulation used here leads to a great number of linear systems of the common form $\mathbf{A}x = b$, where $\mathbf{A}$ is a large sparse matrix; indeed, the average number of non-zero elements in a row of $\mathbf{A}$ (which is the average number of neighbours of a node) is small (about 15 in the present test cases). Consequently, a packed storage of $\mathbf{A}$ which increases linearly with n_h ($n_{h/2}$) can be used. The matrices issued from the discretization of the time derivative and diffusion operators are symmetric and positive definite. Considering Step 2 of equations (2a)–(2c), it appears also that the unsymmetric matrices including the advection term are non-singular.

An important problem is the choice of the solution method. A direct solution, such as lower–diagonal–upper factorization followed by forward elimination and back substitution is clearly not realistic since it requires the storage of a large number of elements owing to the fact that non-zero elements are created in the bandwidth of a row during the factorization. In this case, classical band or skyline forms lead to a very large storage requirement increasing non-linearly with n_h ($n_{h/2}$) and rapidly exceeding available computer capacities. In addition, it is known that effects of round-off errors tend to increase strongly with an increasing number of unknown.[19] Here iterative solution methods are considered; the conjugate gradient method implemented appears to have good convergence properties and can be accelerated by means of convenient preconditioning.

Symmetric linear system solutions

First, considering a symmetric and positive definite matrix **A** and introducing a preconditioning matrix **S** which is symmetric and positive definite close to **A**, the problem of solving $\mathbf{A}x=b$ is equivalent to the following minimization problem:

$$\text{find } v=\mathbf{S}^{-1/2}x \text{ such that } \mathfrak{J}(v)=\min \mathfrak{J}(u), \tag{11}$$

where

$$\mathfrak{J}(u)=\tfrac{1}{2}u^{\mathrm{T}}\mathbf{S}^{-1/2\,\mathrm{T}}\mathbf{A}\mathbf{S}^{-1/2}u-u^{\mathrm{T}}\mathbf{S}^{-1/2\,\mathrm{T}}b, \tag{12}$$

which can be solved by a conjugate gradient method used here as a variant of the original Hesteness–Stiefel[20] algorithm proposed by Takahasi and Nodera.[21]

It is demonstrated that the algorithm converges in less that $n_h(n_{h/2})$ iterations; in practice, owing to round-off errors, the directions h_i are not exactly conjugate so that convergence cannot be obtained in $n_h(n_{h/2})$ iterations. However, one can show that there always exists an initial residual vector g_0 such that the algorithm is stable in terms of round-off error propagation.[20]

The main point is the fact that the convergence properties of conjugate gradient algorithms are strongly related to the conditioning of the matrix $\mathbf{S}^{-1/2\,\mathrm{T}}\mathbf{A}\mathbf{S}^{-1/2}$ or, similarly, to the choice of the preconditioning matrix **S**: the closer **S** is to **A**, the better is the convergence.

Many choices can be found in the literature. The best choice in terms of convergence rate seems to be the incomplete Cholesky factorization 'by value' (i.e. during the factorization keeping only elements exceeding a small given value, as in e.g. Reference 22). Nevertheless, for such a choice the storage requirement becomes non-linearly dependent on the number of nodes of the mesh.

In the present paper, two different preconditionings were implemented. On scalar computers an incomplete lower triangular–diagonal–upper triangular factorization by position ($\mathbf{S}=\mathbf{L}\mathbf{D}\mathbf{L}^{\mathrm{T}}$; only non-zero elements in the initial matrix **A** are kept in **L**) was used. Consequently, storage requirements increase linearly with $n_h(n_{h/2})$ but the convergence rate is slightly dependent on $n_h(n_{h/2})$. Recurrences appearing in such a method do not lead to any problem because the cost of a matrix product is somewhat similar to the cost of a forward elimination–back substitution. On parallel–vector computers it is well known that they inhibit both parallelization and vectorization in such a way that no improvements in the CPU cost can be obtained for either factorization or forward elimination–back substitution. More vectorizable variants were proposed, especially by Van Der Vorst[23] and Axelsson.[19] Both assume that the same matrix is used for several solutions of linear systems and consequently the $\mathbf{L}\mathbf{D}\mathbf{L}^{\mathrm{T}}$ (or similar) factorization may be done once and for all. In this paper, matrices are different from one step to the next since variable Δt and semi-implicit discretization of the advection terms are used. Note that the present programme is also applied to solve problems with variable viscosity.[9,24]

Dubois *et al.*[25] proposed using a truncated Neumann series of $\mathbf{A}^{-1}$ as preconditioner for **A**. In the present approach the diagonal preconditioning was first introduced. Since $\mathbf{A}>0$, then $\mathbf{D}>0$ and $\mathbf{D}^{-1/2}$ exists. The initial problem may then be expressed as

$$\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}u=\mathbf{D}^{-1/2}b, \quad \text{with } x=\mathbf{D}^{-1/2}\mathrm{u}. \tag{13}$$

It is clear that $\mathbf{A}'=\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ is symmetric and positive definite; consequently a conjugate gradient algorithm may be used in order to solve the equivalent problem. If $\mathbf{S}'$ is a symmetric, positive definite matrix close to $\mathbf{A}'$, then the algorithm is as follows:

ε and x_0 given:

$\{ i:=0;\ h_0:=0;$

$$
\begin{aligned}
&u_0 := \mathbf{D}^{1/2} x_0;\\
&g_0 := \mathbf{D}^{-1/2} b - \mathbf{A}' u_0;\\
&\text{while } \left(\frac{(u_i - u_{i-1}, u_i - u_{i-1})}{(u_i, u_i)} > \varepsilon \right)\\
&\quad \Big\{ i := i+1;\\
&\quad \gamma_{i-1} := \frac{1}{g_{i-1}^{\mathrm{T}} \mathbf{S}'^{-1} g_{i-1}};\\
&\quad h_i := h_{i-1} + \gamma_{i-1} \mathbf{S}'^{-1} g_{i-1};\\
&\quad \lambda_i := \frac{1}{h_i^{\mathrm{T}} \mathbf{A}' h_i};\\
&\quad u_i := u_{i-1} + \lambda_i h_i;\\
&\quad g_i := g_{i-1} - \lambda_i \mathbf{A}' h_i \Big\}\\
&x_i = \mathbf{D}^{-1/2} u_i \Big\}.
\end{aligned}
\tag{14}
$$

$\mathbf{D}^{-1/2}$ and $\mathbf{A}'$ are easily computed from $\mathbf{A}$; moreover, such calculations are fully vectorizable and parallelizable. Note that the trivial choice $\mathbf{S}^{-1} = \mathbf{I}$ corresponds also to the zero order of the truncated Neumann series of $\mathbf{A}'^{-1}$. In the following, this preconditioner is denoted $\mathbf{S}_0^{-1}$. Then one introduces the truncated Neumann series of $\mathbf{A}'^{-1}$:

$$
\mathbf{S}_p^{-1} = \sum_{i=0}^{i=p} [-\mathbf{D}^{-1/2}(\mathbf{L}+\mathbf{L}^{\mathrm{T}})\mathbf{D}^{-1/2}]^i. \tag{15}
$$

Unfortunately, the discrete identity operator in three dimensions is not diagonally dominant. It follows that in our case $\mathbf{S}_p^{-1}$ is not necessarily positive definite; more precisely, odd orders of truncation are usually not positive definite while even orders are. Introducing $n > 0$ so that $p = 2n$, then $\mathbf{S}_{2n}^{-1}$ may be a preconditioner of $\mathbf{A}'$. As in Reference 25, $\mathbf{S}_{2n}^{-1}$ is never stored and each iteration requires $2n+1$ matrix–vector products. Then the number of required iterations must be reduced by a factor $2n+1$ to get a better CPU cost than for the zero order.

Storage of $\mathbf{S}_{2n}^{-1}$ is not convenient, even for $n = 1$, since it is largely less sparse than the initial matrix $\mathbf{A}$ owing to fill-in entries during the matrix products. Moreover, since $\mathbf{S}_{2n}^{-1}$ contains many more elements than $\mathbf{A}$, the product of it by any vector is more expensive than those by $\mathbf{A}$ so that the possibility of any improvements in CPU cost is not clearly evident. Lastly, the CPU cost would have to include the computation of $\mathbf{S}_p^{-1}$.

In order to provide a new preconditioner, we introduce the matrix $\mathbf{D}_2$ defined as the diagonal part of $[\mathbf{D}^{-1/2}(\mathbf{L}+\mathbf{L}^{\mathrm{T}})\mathbf{D}^{-1/2}]^2$. The operator

$$
\mathbf{S}_{\mathrm{D}2}^{-1} = [\mathbf{I} - \tfrac{1}{2}\mathbf{D}^{-1/2}(\mathbf{L}+\mathbf{L}^{\mathrm{T}})\mathbf{D}^{-1/2} + \tfrac{3}{8}\mathbf{D}_2]^2 \tag{16}
$$

is symmetric and positive definite. Note that it is obtained from an approximation of the second-order truncated Neumann series of $\mathbf{A}'^{-1/2}$. $\mathbf{D}_2$ is easily computed from its definition and induces a small increase in storage requirement. $\mathbf{S}_{\mathrm{D}2}^{-1}$ requires three matrix–vector products per iteration (as $\mathbf{S}_2^{-1}$) and includes some terms of third and fourth order.

Benchmark tests were performed on a four-processor vector–parallel computer (Alliant FX 80) using automatic microtasking. The test case studied here deals with natural convection inside a cubic box (dimensionless width unity) where vertical boundaries ($x=0$, 1) are differentially heated ($T=0{\cdot}5$ at $x=0$, $T=-0{\cdot}5$ at $x=1$) while other boundaries are adiabatic; no-slip conditions are imposed on all boundaries. The Grashof number is chosen sufficiently high so that advection terms are not too small compared with diffusive terms: $Gr = Re^2 = 10^4$. The initial field is zero for all variables. The grids used contain 11^3 nodes for pressure and 21^3 nodes for velocity and temperature. To minimize the calculation time, the number of steps in time is limited to 10 for a dimensionless step size of unity.

The first test (Table I) deals with the performance of the preconditioners mentioned above on the vector–parallel computer for the solutions of symmetric linear systems appearing in each time iteration of the programme. All preconditioners are compared with the case of no preconditioning, which is chosen as the reference. Comparisons are provided for the convergence rate (i.e the number of required iterations for all the symmetric linear systems having to be solved during each step in time) and for the CPU time, using all the vectorization and parallelization possibilities of the computer.

The best results are given by the diagonal preconditioner ($\mathbf{S}_0^{-1}$). Table I clearly shows that improvement of the convergence rate for $\mathbf{S}_2^{-1}$ and $\mathbf{S}_4^{-1}$ is not sufficient. $\mathbf{S}_{D2}^{-1}$ appears to be a better preconditioner but also has lower performances than the diagonal preconditioner. This result is also related to the greater number of matrix–vector products required. Specifically, the better performance of $\mathbf{S}_{D2}^{-1}$ with respect to $\mathbf{S}_2^{-1}$ is due to the fact that the convergence rate improvement is better for the non-diagonally dominant matrices used here (essentially pressure matrices) while it is somewhat similar for the others (velocity and temperature matrices, in the test case here, with respect to the Reynolds and Peclet number chosen).

Unsymmetrical linear system solutions

In the case of unsymmetric matrices a biconjugate gradient algorithm[26] was implemented. The initial problem

$$\text{find } x \text{ such that } \mathbf{A}x = b \tag{17}$$

is replaced by the equivalent one

$$\text{find } X \text{ such that } \mathfrak{A}X = B, \quad \text{where } \mathfrak{A} = \begin{bmatrix} 0 & \mathbf{A}' \\ \mathbf{A}'^{\mathrm{T}} & 0 \end{bmatrix}, \quad X = \begin{pmatrix} y \\ x \end{pmatrix}, \quad B = \begin{pmatrix} \mathbf{D}^{-1} & b \\ \mathbf{D}^{-1} & c \end{pmatrix}, \tag{18}$$

for any y and c, with $\mathbf{A}' = \mathbf{D}^{-1}\mathbf{A}$.

$\mathfrak{A}$ is symmetric but not positive definite, so a conjugate gradient algorithm may be applied in order to solve (18), but owing to the indefiniteness of $\mathfrak{A}$, one cannot strictly ensure the

Table I. Symmetric linear system solution. Benchmark test. Convergence and CPU rate are given by reference to the standard algorithm without preconditioning

Preconditioner	Convergence rate	CPU rate
No preconditioner	1	1
Neumann $p=0$	0·376–0·897	0·784
Neumann $p=2$	0·365–0·503	1·17
Neumann $p=4$	0·277–0·454	1·46
$\mathbf{D}_2$	0·330–0·538	0·981

convergence of iterations. Let **S** be an unsymmetric matrix close to **A**; the conjugate gradient method applied to (18) leads to

$$
\begin{aligned}
&\varepsilon \text{ and } x_0 \text{ given:}\\
&\left\{ i := 0;\ h_0 := h'_0 := 0;\right.\\
&\quad g'_0 := g_0 := \mathbf{D}^{-1} b - \mathbf{A}' x_0;\\
&\text{while } \left(\frac{(x_i - x_{i-1},\, x_i - x_{i-1})}{(x_i, x_i)} > \varepsilon \right)\\
&\qquad \left\{ i := i + 1;\right.\\
&\qquad\quad \gamma_{i-1} := \frac{1}{g_{i-1}^{\mathrm{T}} \mathbf{S}^{-1\mathrm{T}} g'_{i-1} + g'^{\mathrm{T}}_{i-1} \mathbf{S}^{-1} g_{i-1}};\\
&\qquad\quad h_i := h_{i-1} + \gamma_{i-1} \mathbf{S}^{-1} g_{i-1};\\
&\qquad\quad h'_i := h'_{i-1} + \gamma_{i-1} \mathbf{S}^{-1\mathrm{T}} g'_{i-1};\\
&\qquad\quad \lambda_i := \frac{1}{h'^{\mathrm{T}}_i \mathbf{A}' h_i + h_i^{\mathrm{T}} \mathbf{A}'^{\mathrm{T}} h'_i};\\
&\qquad\quad x_i := x_{i-1} + \lambda_i h_i;\\
&\qquad\quad g_i := g_{i-1} - \lambda_i \mathbf{A}' h_i;\\
&\qquad\quad \left. g'_i := g'_{i-1} - \lambda_i \mathbf{A}'^{\mathrm{T}} h'_i \right\} \right\}.
\end{aligned}
\tag{19}
$$

Note that the sequence $y_0, y_1, \ldots, y_i, \ldots$ is not computed. The free vectors y_0 and c are classically chosen such that

$$g'_0 = g_0 \Rightarrow c - \mathbf{A}'^{\mathrm{T}} y_0 = b - \mathbf{A}' x_0. \tag{20}$$

The main deficiency of the biconjugate gradient algorithm is that it often degenerates. A classical way to ensure convergence is then to reinitialize the algorithm as often as required, the number of reinitializations being strongly dependent on the preconditioning used. In the present calculations the algorithm is always convergent but requires some reinitialization steps. Tests similar to those of the preceding section for symmetric matrices were done; results are given in Table II. The best results are given by the diagonal preconditioner, as in the symmetric case. Note also that $\mathbf{S}_{\mathrm{D}2}^{-1}$ gives a worse convergence rate and CPU cost than $\mathbf{S}_2^{-1}$ and clearly does not constitute a good approach in the case of unsymmetric matrices.

Table II. Unsymmetrical linear system solution. Benchmark test. Convergence and CPU rate are given by reference to the standard algorithm without preconditioning

Preconditioner	Convergence rate	CPU rate
No preconditioner	1	1
Neumann $p=0$	0·601–0·751	0·724
Neumann $p=2$	0·331–0·413	1·04
Neumann $p=4$	0·196–0·256	1·12
$\mathbf{D}_2$	0·343–0·375	1·05

4. PERFORMANCES OF SOME CONJUGATE GRADIENT METHODS APPLIED TO THE STOKES PROBLEM

As mentioned above, some classical methods can be applied to the solution of the Stokes problem. At the present time a preconditioned conjugate gradient version of the Uzawa algorithm is used.

Let us recall that the discrete formulation of the Stokes problem is

$$\begin{aligned} &\text{find } (\mathbf{u}_h, p_h) \in X_h \times M_h \text{ such that} \\ &a(\mathbf{u}_h, \mathbf{v}_h) + b(\mathbf{v}_h, p_h) = (\mathbf{f}, \mathbf{v}_h), \quad \forall \mathbf{v}_h \in X_h, \\ &b(\mathbf{u}_h, q_h) = 0, \quad \forall q_h \in M_h, \end{aligned} \tag{21}$$

where

$$a(.\,,.) = \alpha(.\,,.)_{L_2^3} + \nu(\nabla ., \nabla .)_{L_2^3}, \qquad b(.\,,.) = (\nabla .,.)_{L_2^3}. \tag{22}$$

Defining the operators $\mathbf{A}: X_h \to X_h$, $\mathbf{B}: X_h \to M_h$ and $\mathbf{B}^T: M_h \to X_h$ by

$$\begin{aligned} (\mathbf{Au}, \mathbf{v}) &= a(\mathbf{u}, \mathbf{v}), \quad \forall (\mathbf{u}, \mathbf{v}) \in X_h \times X_h, \\ (\mathbf{Bu}, p) &= b(\mathbf{u}, p), \quad \forall (\mathbf{u}, p) \in X_h \times M_h, \\ (\mathbf{B}^T p, \mathbf{u}) &= b(\mathbf{u}, p), \quad \forall (\mathbf{u}, p) \in X_h \times M_h, \end{aligned} \tag{23}$$

then (21) can be written in the form

$$\begin{aligned} &\text{find } (\mathbf{u}_h, p_h) \in X_h \times M_h \text{ such that} \\ &\mathbf{A}\mathbf{u}_h + \mathbf{B}^T p_h = \mathbf{f}, \\ &\mathbf{B}\mathbf{u}_h = 0, \end{aligned} \tag{24}$$

and is formally expressed by

$$\begin{aligned} \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T p_h &= \mathbf{B}\mathbf{A}^{-1}\mathbf{f}, \\ \mathbf{u}_h &= \mathbf{A}^{-1}(\mathbf{f} - \mathbf{B}^T p_h). \end{aligned} \tag{25}$$

The operator $\mathbf{L} = \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T$ is symmetric, positive definite and continuous;[27] nevertheless, since it is defined via $\mathbf{A}^{-1}$, it is only implicitly known and (24) must be solved through an iterative process. Since $\mathbf{L}$ has all the required properties, a conjugate gradient algorithm may be used. Moreover, as in the preceding sections, improvement of the convergence rate is obtained by introducing a preconditioning operator $\mathbf{C}^{-1}$ close to $\mathbf{L}^{-1}$. The conjugate gradient algorithm is as follows:

$p_h^{(0)}$ and ε given:

$\left\{ i := 0;\ h_h^{(0)} := 0;\ q_h^{(0)} = 0; \right.$

solve: $a(\mathbf{u}_h^{(0)}, \mathbf{v}_h) = (\mathbf{f}, \mathbf{v}_h) - b(\mathbf{v}_h, p_h^{(0)}), \quad \forall \mathbf{v}_h \in X_h;$

solve: $(r_h^{(0)}, t_h) = -b(\mathbf{u}_h^{(0)}, t_h), \quad \forall t_h \in M_h;$

$g_h^{(0)} := -r_h^{(0)};$

$$
\begin{aligned}
&\text{solve: } (\mathbf{C}q_h^{(0)}, \nabla t_h) = -b(\mathbf{u}_h^{(0)}, t_h), \quad \forall t_h \in M_h;\\
&\text{while } \left(\frac{(p_h^{(i)} - p_h^{(i-1)}, p_h^{(i)} - p_h^{(i-1)})}{(p_h^{(i)}, p_h^{(i)})} > \varepsilon \right)\\
&\quad \Big\{ i := i+1;\\
&\quad \gamma^{(i-1)} := \frac{1}{(g_h^{(i-1)}, q_h^{(i-1)})};\\
&\quad h_h^{(i)} := h_h^{(i-1)} + \gamma^{(i-1)} q_h^{(i-1)};\\
&\quad \text{solve: } a(\chi_h^{(i)}, \mathbf{v}_h) = -b(\mathbf{v}_h, h_h^{(i)}), \quad \forall \mathbf{v}_h \in X_h;\\
&\quad \text{solve: } (r_h^{(i)}, t_h) = -b(\chi_h^{(i)}, t_h), \quad \forall t_h \in M_h;\\
&\quad \lambda^{(i)} := \frac{1}{(h_h^{(i)}, r_h^{(i)})};\\
&\quad p_h^{(i)} := p_h^{(i-1)} + \lambda^{(i)} h_h^{(i)};\\
&\quad \mathbf{u}_h^{(i)} := \mathbf{u}_h^{(i-1)} + \lambda^{(i)} \chi_h^{(i)};\\
&\quad g_h^{(i)} := g_h^{(i-1)} - \lambda^{(i)} r_h^{(i)};\\
&\quad \text{solve: } (\mathbf{C}q_h^{(i)}, \nabla t_h) = -b(\chi_h^{(i)}, t_h), \quad \forall t_h \in M_h \Big\} \Big\}.
\end{aligned} \tag{26}
$$

Note that at each iteration the method requires the solution of a linear system of order $n_{h/2}$ for each component of the velocity field (or their associated direction vectors $\chi_h^{(i)}$) and at least two linear systems of order n_h for the pressure (i.e. its direction vectors $q_h^{(i)}$). Each linear system is solved via the method presented in the preceding section.

Recently, using a Fourier analysis in a continuous approach, Cahouet and Chabard[28] proposed the following preconditioning operator:

$$\mathbf{C}_1^{-1} = \nu \mathbf{I}_h^{-1} - \alpha \Delta_h^{-1}, \tag{27}$$

where $\mathbf{I}_h$ and Δ_h are respectively the discrete identity and Laplacian operators defined on M_h. $\mathbf{C}_1^{-1}$ has all the required properties to be a preconditioning operator of $\mathbf{L}$. The two parts of $\mathbf{C}_1^{-1}$ are easily computed once and for all and the preconditioning method requires only two linear system solutions on the coarser grid at each conjugate gradient iteration. A great improvement of the convergence rate is obtained while the cost of preconditioning is low. Nevertheless, the introduction of the Laplacian operator leads one, via a weak formulation, to impose boundary conditions on $q_h^{(i)}$, which are unknown. Classical Neumann boundary conditions are consequently imposed, but this can lead to an ill-conditioned problem.

In order to avoid any problems of ill-conditioning, a discrete approximation of $\mathbf{L}^{-1}$ was tested. Writing

$$
\begin{aligned}
\mathbf{L}^{-1} &= \mathbf{B}^{-1}\mathbf{A}\mathbf{B}^{-1\mathrm{T}} = \mathbf{B}^{-1}(\alpha \mathbf{I}_{h/2} - \nu \Delta_{h/2})\mathbf{B}^{-1\mathrm{T}}\\
&= \alpha \mathbf{B}^{-1}\mathbf{I}_{h/2}\mathbf{B}^{-1\mathrm{T}} - \nu \mathbf{B}^{-1}\Delta_{h/2}\mathbf{B}^{-1\mathrm{T}}\\
&= \alpha(\mathbf{B}\mathbf{I}_{h/2}^{-1}\mathbf{B}^{\mathrm{T}})^{-1} - \nu(\mathbf{B}\Delta_{h/2}^{-1}\mathbf{B}^{\mathrm{T}})^{-1},
\end{aligned} \tag{28}
$$

one can obtain a class of preconditioning operators by approximating the inverse of the operators

$\mathbf{I}_{h/2}$ and $\Delta_{h/2}$ by truncated Neumann series. Besides, as mentioned earlier, only even orders of the truncated Neumann series, which lead to positive definite operators, may be considered; moreover, the higher the order of truncation, the fuller is the corresponding matrix, so the present tests are restricted to the zero order—an inverse diagonal matrix for the first operator and an identity operator (on M_h, as in the Cahouet–Chabard[28] approach) for the second:

$$\mathbf{C}_2^{-1} = \alpha(\mathbf{B}D\{\mathbf{I}_{h/2}\}^{-1}\mathbf{B}^{\mathrm{T}})^{-1} + \nu\mathbf{I}_h^{-1}. \tag{29}$$

As with $\mathbf{C}_1$, $\mathbf{C}_2$ can be computed once and for all. Storage requirement is more important owing to fill-in entries during the two products of the inverse diagonal by $\mathbf{B}$ and $\mathbf{B}^{\mathrm{T}}$; but since $\mathbf{C}_2$ is defined on the coarser grid, the storage requirement is of the order of those for $\mathbf{A}$. Note that $\mathbf{B}$, being calculated once and for all, is also used to compute the integrals of pressure gradient and velocity divergence, so that a great gain in the CPU cost is obtained. A first improvement of $\mathbf{C}_2$ can be obtained by considering $\mathbf{B}D\{\Delta_{h/2}\}^{-1}\mathbf{B}^{\mathrm{T}}$ rather than $\mathbf{I}_h$, but this leads to storing a second matrix, and this possibility was not implemented here. Instead, we tested a preconditioning operator $\mathbf{C}_3$ obtained by a direct calculation from $\mathbf{A}$:

$$\mathbf{C}_3^{-1} = (\mathbf{B}D\{\mathbf{A}\}^{-1}\mathbf{B}^{\mathrm{T}})^{-1}. \tag{30}$$

$\mathbf{C}_3$ requires no more storage than $\mathbf{C}_2$ but becomes dependent on α and ν and must consequently be computed at each time step if α and ν are not constant. However, the computation of $\mathbf{C}_3$ can be vectorized and parallelized and is not too expensive. Moreover, such a preconditioning operator will be well adapted to simulations with variable local viscosity or unsymmetric Stokes problems.

Benchmark tests were done similarly to those in Section 3 and results are shown in Table III. All results are given referring to the case without preconditioning. $\mathbf{C}_3$ gives the best convergence rate; nevertheless, the CPU costs are very close for each preconditioning operator owing to the fact that matrix–vector products are more expensive for $\mathbf{C}_2$ and $\mathbf{C}_3$ since they are fuller than $\mathbf{C}_1$. Moreover, the computation of $\mathbf{C}_3$ at each time step is included in the CPU cost mentioned in the table.

5. GLOBAL PERFORMANCES OF THE PROGRAMME USING VECTOR PARALLEL OPTIMIZATION

Codes are often characterized by the CPU time and the memory space required per iteration in time and per node of the grid. In the preceding test case, using the $\mathbf{C}_2$ preconditioning operator for the Stokes problem and the diagonal preconditioning for linear system solutions, a characteristic CPU time of $2{\cdot}27 \times 10^{-2}$ s per iteration and per node of $\mathfrak{T}_{h/2}$ (with four degrees of freedom on each node of $\mathfrak{T}_{h/2}$ and one for $\mathfrak{T}_h$) was obtained on the Alliant FX80 four processors. The storage requirement is $7{\cdot}23 \times 10^{-4}$ Mbytes per node of $\mathfrak{T}_{h/2}$ in single precision. Since the storage requirement increases linearly with $n_h\,(n_{h/2})$, it represents effectively a characteristic of the

Table III. Comparison of preconditioning methods for the Stokes problem

Preconditioner	Convergence rate	CPU rate
No preconditioner	1	1
$\mathbf{C}_1$	0·34	0·388
$\mathbf{C}_2$	0·272	0·317
$\mathbf{C}_3$	0·236	0·319

programme. On the contrary, a few remarks may be given about the CPU time. Firstly, the intrinsic CPU time is strongly dependent on the computer hardware and software abilities. Secondly, the CPU time is non-linearly dependent on n_h: indeed, on the one hand the convergence rate of the preconditioned conjugate gradient algorithm is dependent on n_h and on the other hand the unknown stability condition on Δt induces a relation between convergence in time and n_h.

A more relevant characteristic of the programme is its ability to take advantage of vectorized and parallelized computations. Table IV gives such performances for the test case of the preceding section. Results are given for the whole code successively considering global optimization (which can be provided on any scalar computer), vectorization only (one processor) and parallel computing with two and four processors. All results are given referring to the case of global optimization. For a better understanding, results are also given separately for linear system solutions, matrix assembly and right-hand-side terms assembly. Linear system solutions, constituting presently the more optimized part of the programme, have very high performances since the CPU time is divided by a factor of more than 10 (note that vectorization is also efficient for these computations). This is due to the fact that only products are done on the matrices. Note also that they are completely stored by row without any symmetry consideration in order to obtain parallel computation of each row. The matrix assembly was less optimized, only local modifications being done in the code. No improvement is obtained by vectorization while some is by parallelization. The right-hand-side terms assembly is not presently optimized except for the pressure and divergence terms (which use the $\mathbf{B}$ and $\mathbf{B}^T$ matrices). Thus this explains why the results are better when using vectorization and less conclusive when using parallelization.

At this stage of the work it is not possible to know exactly how the number of grid nodes will affect the performances of the schemes given above. Further calculations are presently being done in order to investigate more precisely the influence of greater concentrations of nodes (up to 64 000 grid nodes at the present time) and will be presented in a future paper. Vector–parallel performances would not be strongly affected since the techniques used are essentially dependent on the hardware and software abilities of the computer. On the contrary, as already mentioned, the convergence properties of the time scheme and the conjugate gradient methods depend on n_h, so that the loss of performance with an increasing number of nodes has to be estimated to confirm the practical usefulness of the approach.

Table IV. Programme performances

	Whole Programme	Linear system solution	Matrix assembly	Right-hand-side terms assembly
Global optimization 1 processor	1	1	1	1
Vectorization 1 processor	0.457	0·329	0·927	0·613
Vectorization 2 processors	0·261	0·178	0·543	0·402
Vectorization 4 processors	0·157	0·0993	0·318	0·298

6. EXAMPLE: RAYLEIGH–BÉNARD CONVECTION AT HIGH PRANDTL NUMBER

As a check of the validity of our approach, simulations of the so-called Rayleigh–Bénard convection were performed. The test case concerns a thin horizontal layer of fluid confined between two plates and heated from below. It is well known that when the Rayleigh number Ra (characteristic of the temperature difference between the two horizontal boundaries) exceeds a critical value Ra_c ($Ra_c = 1708$ for $Pr > 1$), the flow bifurcates from the static state to a two-dimensional behaviour in the form of parallel rolls. These rolls are stable until a new critical value Ra_1 is reached. This last value as well as the configuration of the flow appearing after this threshold are strongly dependent on the Prandtl number. For small values of Pr a periodic motion takes place, while a steady three-dimensional behaviour is encountered for high values. For the present calculations the studies are limited to high-Prandtl-number fluids for which spatially periodic steady flows may be reached.

Two-dimensional flows

Just above the onset, supercritical convection in large rectangular boxes consists essentially of two-dimensional straight rolls (circular rolls in circular boxes) of wavelength $\lambda_c \approx 2h$, where h is the distance between the two horizontal plates. Such rolls are commonly observed experimentally.[29–31]. Note, however, as a consequence of imperfect boundary conditions, the diameter of the rolls often increases with Ra in experiments. Recent investigations of Kirchatz and Oertel[32] show that such a change does not occur in numerical simulations as long as perfectly conductive horizontal walls are simulated. Consequently, it seems that the modification of the diameter of the rolls is essentially related to the finite conductivity of the walls or to the thermal dependence of the fluid properties.

In the present calculations we tried to reproduce the flow inside a cubic box of dimensionless width unity, with no-slip conditions and given temperature on the horizontal boundaries and free-slip and adiabatic conditions on the vertical ones. The value of the Prandtl number is 930 (see the work of Dubois and Bergé discussed later). Since the static state with a purely conduction field is still a mathematical solution of the problem when $Ra > Ra_c$, the first flow ($Ra = 2700$) was obtained by introducing a small perturbation (of order 1%) in the temperature distribution. Afterwards the converged flow fields were obtained for increasing Rayleigh number values using the preceding state. The grids used contain 11^3 nodes for the pressure and 21^3 nodes for the velocity and the temperature.

An important characteristic of the convection in the box is the Nusselt number Nu, which represents the heat exchange between the two horizontal plates. Accurate measurements of it were made by Koschmieder and Pallas[33] for a circular high-Prandtl-number fluid layer in the domain of roll stability. As previously predicted by Schlüter *et al.*[34] via a perturbation analysis and by Busse[35] via a spectral method in the case of infinite Prandtl number, a change in the slope of the Nusselt number indicates the appearance of convection when Ra increases up to Ra_c ($Nu = 1$ for $Ra < Ra_c$). Moreover, for sufficiently high Pr the Nusselt number becomes independent of it. Comparisons between the two preceding results and ours are given in Figure 2. Note that the two numerical investigations overestimate the heat transfer; this is clearly due firstly to the geometric differences between experiments (circular box) and computations (rectangular box) and secondly to the modification of the roll diameter (13 to 10 rolls) during the experiments.

More local measurements concerning the velocity field were done by Dubois and Bergé.[36] They corroborate the studies of Busse which predict that the velocity field is essentially governed by the first three spatial modes (λ_c, $\lambda_c/2$, $\lambda_c/3$) increasing as powers of the reduced variable $\varepsilon = (Ra - Ra_c)/Ra_c$ (in the range of stability of the rolls, $0 < \varepsilon < 12$). Dubois and Bergé provided

results for the first three modes in the x-horizontal direction. Applying a Fourier decomposition in the x-direction to the components of the velocity field u and w in the x- and z-direction respectively,

$$u(x, z) = u^{(1)}(z)\sin(a_c x) + u^{(2)}(z)\sin(2a_c x) + u^{(3)}(z)\sin(3a_c x) + \ldots, \tag{31a}$$

$$w(x, z) = w^{(1)}(z)\cos(a_c x) + w^{(2)}(z)\cos(2a_c x) + w^{(3)}(z)\cos(3a_c x) + \ldots, \tag{31b}$$

where a_c is the critical wave number ($a_c \approx \pi$), they found that

$$u^{(i)}_{max} \sim \varepsilon^{(0{\cdot}5)^i}, \qquad w^{(i)}_{max} \sim \varepsilon^{(0{\cdot}5)^i}. \tag{32}$$

Trigonometric functions were also used by Busse for his spectral approach in both the x- and z-directions for the temperature field and in the x-direction for the velocity field. Dependence in the z-direction was described by a particular set of hyperbolic functions.

The present results are shown in Figures 3 and 4 and compared with the works of Dubois and Bergé. The results obtained at $y = 0$ are slightly different from those at $y = 0{\cdot}5$ since the mesh has no two-dimensional symmetry. This small deviation can be seen as a measurement of the programme accuracy. Nevertheless, note that the differences are sensitive only for low ε, where $w^{(3)}_{max}$ represents only a few per cent of the whole velocity.

Three-dimensional flows

Two kinds of steady three-dimensional convection are presently known to occur in Rayleigh–Bénard convection of a high-Prandtl-number fluid. The first one is the so-called bimodal convection, appearing when Ra exceeds the critical value of 22 600, according to the

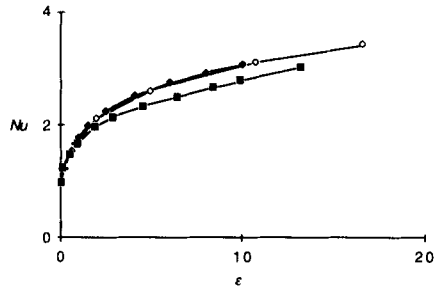

Figure 2. Nusselt number versus $\varepsilon = Ra/Ra_c - 1$: ◆, present calculations, $Pr = 930$, rectangular box; ◇, Busse calculations $Pr = +\infty$, rectangular box; ■, Koschmieder and Pallas measurements, $Pr = 511$, circular box

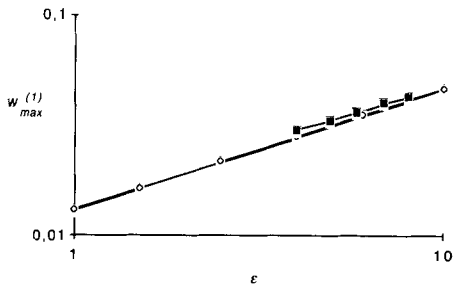

Figure 3. First mode of the vertical component of the velocity, $w^{(1)}_{max}$, versus ε, $Pr = 930$: ◆, present calculations, $y = 0$; ◇, present calculations, $y = 0{\cdot}5$; ■, Dubois and Bergé measurements

stability theory of Busse[35] ($\varepsilon = 12{\cdot}2$). From the original roll pattern of wavelength λ_1, new rolls of wavelength $\lambda_2 \neq \lambda_1$ appear at right angles to the first ones. Experimental observations of this can be found in Reference 37. Previous calculations of Frick *et al.*[38] have shown that a range of λ_2 exists (dependent on both λ_1 and Ra) where bimodal convection is stable. Experimentally, a preferred value of λ_2 is observed and decreases with increasing Rayleigh number.[39] Nevertheless, such a value is strongly related to the conductivity of the horizontal walls and the thermal dependence of the physical properties of the fluid. Here calculations were performed for only one value of λ_2 ($\lambda_2 = 1{\cdot}4$), close to the critical wavelength, permitting comparisons with the works of Frick *et al.* The present computations were done in a box of 21^3 nodes whose dimensions are $0 \leqslant x \leqslant 1, 0 \leqslant y \leqslant 0{\cdot}7, 0 \leqslant z \leqslant 1$. Three-dimensionality is initiated by a small perturbation of the temperature field in the y-direction (1% of ΔT). Calculations were performed for $10 < \varepsilon < 50$. The change in the slope of the Nusselt number (see Figure 5) is smoother than for the onset of convection and does not permit the estimation of the second critical Rayleigh number Ra_1 with sufficient accuracy. Results from Frick *et al.* are also given for different values of λ_2. Calculations were not performed for higher values of Ra since, on the one hand, smaller wavelengths than described by the mesh would have to be taken into account and, on the other hand, bifurcations to periodic motions would occur and these do not agree with the present boundary conditions.

Another kind of three-dimensional motion is presently of interest. Theoretical and numerical works of Busse and Riahi,[40] Proctor[41] and Jenkins and Proctor[42] show that a new pattern, called the square pattern convection, the form of two right-angle rolls of the same wavelength, can be stable above the threshold in the case of small conductive horizontal boundaries. More recently,

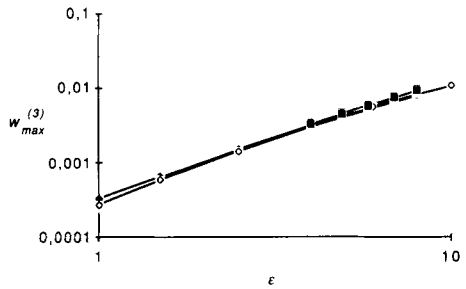

Figure 4. Third mode of the vertical component of the velocity, $w^{(3)}_{max}$, versus ε, $Pr = 930$: ◆, present calculatios, $y = 0$; ◇, present calculations, $y = 0{\cdot}5$; ■, Dubois and Bergé measurements

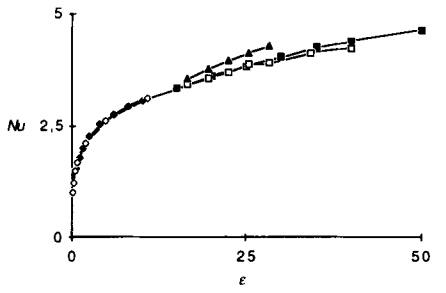

Figure 5. Nusselt number versus ε: ◆, present calculations, $Pr = 930$, roll pattern; ◇, Busse calculations, $Pr = +\infty$, roll pattern; ■, present calculations, $Pr = 930$, bimodal convection, $\lambda_2 = 1{\cdot}4$; □, Frick *et al.* calculations, $Pr = +\infty$, bimodal convection, $\lambda_2 = 1{\cdot}57$; ▲, Frick *et al.* calculations, $Pr = +\infty$, bimodal convection, $\lambda_2 = 1{\cdot}256$

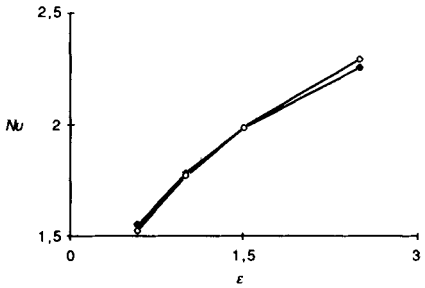

Figure 6. Present calculations. Comparison of Nusselt numbers versus ε for: $\diamond$, the square pattern; $\blacklozenge$, the roll pattern

the experimental study of Le Gal *et al.*[43] shows that square pattern convection occurs just above the onset even if the conductivity is large (but finite). The present computations were done in the same cubic box as for the two-dimensional flows with the same boundary conditions. The initial field for $Ra = 2700$ was the static state, perturbed by a small three-dimensional deviation on the temperature field (2% of ΔT) of dimensionless wavelength $\lambda_x = \lambda_y = 2$, symmetric with respect to the x- and y-directions. Afterwards, fields were obtained from the preceding solutions at lower Rayleigh number. As pointed out by Frick *et al.*, heat exchange resulting from square pattern convection is not related to its stability; Figure 6 gives the Nusselt number versus ε for both roll (Nu_r) and square (Nu_s) patterns. Nu_s is lower than Nu_r for $\varepsilon < 1{\cdot}5$ and greater for $\varepsilon > 1{\cdot}5$, while the square pattern is always found to be unstable for this range of ε. Since the square pattern is unstable at such Rayleigh numbers for the assumed infinite value of the horizontal wall conductivity, it was difficult to obtain a square pattern steady state. For $\varepsilon > 1{\cdot}5$, several runs were done where the flow bifurcated to the roll pattern when too large time steps were used. The fact that the square pattern can numerically be found to be more stable for small values of ε seems to corroborate the idea that the square pattern is only weakly unstable near the threshold.

7. CONCLUSIONS

The ability of a finite element fractional step method to take advantage of vector–parallel computations has been studied. The efficiency of the method is mainly related to the efficiency of the preconditioned conjugate gradient method which is applied to solve both the large linear systems and the continuity constraint. Different approaches have been tested and new preconditioning operators have been introduced to improve the convergence rate of the conjugate gradient version of the Uzawa algorithm. The proposed improvements can be conveniently applied to more complex problems, including variable time steps or variable diffusivity. Further developments are still under way to improve not only the convergence and CPU rate for linear system solutions but also the parallelization of matrices and right-hand-side terms assembly.

The accuracy of the code has also been examined in typical thermoconvective flows. The results concerning the Rayleigh–Bénard convection in an infinite layer of fluid are in good agreement with the literature from the point of view of both the global (heat exchange between the two horizontal walls) and local (Fourier decomposition in the horizontal directions) description of the flows. Further calculations taking into account the finite conductivity of the walls are presently being carried out.

ACKNOWLEDGEMENTS

This work was performed under the auspices of the Lyon–Grenoble Pilot Center for Turbulence (PEPiT). Acknowledgements are also due to Professor J. Wallace (University of Maryland, U.S.A.) for helpful discussions and comments.

REFERENCES

1. J. W. Deardorff, *J. Fluid Mech.*, **41,** 453–480 (1970).
2. J. B. McLaughin, and S. A. Orszag, *J. Fluid Mech.*, **122,** 123–142 (1982).
3. G. Grotzbach, *J. Fluid Mech.*, **119,** 27–54 (1983).
4. J. H. Curry, J. R. Herring, J. Loncaric and S. A. Orszag, *J. Fluid Mech.*, **147,** 1–38 (1984).
5. J. Kim, P. Moin and R. Moser, *J. Fluid Mech.*, **177,** 137–166 (1987).
6. A. Fortin, M. Fortin and J. J. Gervais, *J. Comput. Phys.*, **70,** 295–310 (1987).
7. M. O. Bristeau, R. Glowinski, B. Mantel, J. Périaux and P. Perrier, in R. H. Gallagher *et al.* (eds), *Finite Element Methods in Fluids, Vol. 6*, Wiley, New York, 1985, pp. 1–40.
8. M. Pelletier and R. Lamarho, *Int. j. numer. methods fluids*, **8,** 1563–1586 (1988).
9. Z. Sun, Ph. Carrière, D. Jeandel, J. Bertrand and CI. Bossy, *Dechema-Monographs, Vol. 116*, VCH, Frankfurt, 1989, pp. 351–356.
10. E. A. Spiegel and G. Veronis, *Astrophys. J.*, **131,** 442–447 (1960).
11. J. M. Mihaljan, *Astrophys. J.*, **136,** 1126–1133 (1962).
12. D. D. Gray and A. Giorgini, *Int. J. Heat Mass Transfer*, **19,** 545–551 (1976).
13. P. M. Gresho, S. T. Chan, R. L. Lee and C. D. Upson, *Int. j. numer methods fluids*, **4,** 557–598 (1984).
14. V. Girault and P. A. Raviard, *Finite Element Methods for Navier–Stokes Equations*, Springer, Berlin, 1986.
15. I. Babuška, *Numer. Math.*, **16,** 322–333 (1971).
16. F. Brezzi, *RAIRO Anal. Numer.*, **8,**129–152 (1974).
17. M. Fortin and A. Fortin, in R. H. Gallagher *et al.* (eds), *Finite Element Methods in Fluids, Vol. 6*, Wiley, New York, 1985, pp. 171–187.
18. M. Bercovier and O. Pironneau, *Numer. Math.*, **33,** 211–224 (1979).
19. O. Axelsson, in R. H. Gallagher *et al.* (eds), *Finite Element Methods in Fluids, Vol. 6*, Wiley, New york, 1985, pp. 157–170.
20. M. R. Hesteness and E. Stiefel, *J. Res. NBS*, **49,** 409–436 (1952).
21. H. Takahasi and T. Nodera, in E. Aboi *et al.* (eds), *Numerical Methods for Engineering Vol. 1*, Boutas, Paris, 1980, pp. 209–219.
22. M. A. Ajiz and A. Jennings, *Int. j. numer. methods eng.*, **20,** 949–966 (1984).
23. H. A. Van der Vorst, *SIAM J. Sci. Stat. Comput.*, **3,** 350–356 (1982).
24. G. Brun, M. Buffat, D. Jeandel, J. L. Schultz and M. Desaulty, in T. J. Chang and G. R. Kan (eds), *Finite Element Analysis in Fluids*, UAH Press, Huntsville, 1989, pp. 1592–1597.
25. P. F. Dubois, A. Greenbaum and G. H. Rodrigue, *Computing*, **22,** 257–268 (1979).
26. R. Fletcher, 'Conjugate gradient methods for indefinite systems', in G. A. Watson (ed), *Proc. Dundee Conf. in Numerical Analysis*, Springer, New York, 1975, pp. 73–89.
27. R. Verfurth, *IMA J. Numer, Math.*, **4,** 441–455 (1984).
28. J. Cahouet and J. P. Chabard, *Int. j. numer. methods fluids*, **8,** 869–895 (1988).
29. M. M. Chen and J. A. Whitehead, *J. Fluid Mech.*, **31,** 1–15 (1968).
30. R. Krishnamurti, *J. Fluid Mech.*, **42,** 295–307 (1970).
31. F. H. Busse and J. A. Whitehead, *J. Fluid Mech.*, **47,** 305–320 (1971).
32. K. R. Kirchatz and H. Oertel, *J. Fluid Mech.*, **192,** 249–286 (1988).
33. E. L. Koschmieder and S. G. Pallas, *Int. J. Heat Mass Transfer*, **17,** 991–1002 (1974).
34. A. Schlüter, D. Lortz and F. Busse, *J. Fluid Mech.*, **23,** 129–144 (1965).
35. F. H. Busse, *J. Math. Phys.*, **46,** 140–150 (1967).
36. M. Dubois and P. Bergé, *J. Fluid Mech.*, **85,** 641–653 (1978).
37. F. H. Busse and J. A. Whitehead, *J. Fluid Mech.*, **66,** 67–79 (1974).
38. H. Frick, F. H. Busse and R. M. Clever, *J. Fluid Mech.*, **127,** 141–153 (1983).
39. J. A. Whitehead and G. L. Chan, *Dyn. Atmos. Oceans*, **1,** 33–49 (1976).
40. F. H. Busse and N. Riahi, *J. Fluid Mech.*, **96,** 242–256 (1980).
41. M. R. E. Proctor, *J. Fluid Mech.*, **113,** 469–485 (1981).
42. D. R. Jenkins and M. R. E. Proctor, *J. Fluid Mech.*, **139,** 461–471 (1984).
43. P. Le Gal, A Pocheau and V. Croquette, *Phys. Rev. Lett.*, **54,** 2501–2504 (1985).