



## AN IMPROVED PISO ALGORITHM FOR THE COMPUTATION OF BUOYANCY-DRIVEN FLOWS

**Paulo J. Oliveira**

*Departamento de Engenharia Electromecânica, Universidade da Beira Interior, Covilhã, Portugal*

**Raad I. Issa**

*Department of Mechanical Engineering, Imperial College of Science, Technology and Medicine, London, UK*

*A numerical procedure for the calculation of buoyancy-driven flows using the finite-volume approach is presented. It is based on an extension of the operator-splitting procedure PISO of Issa [1] to the specific case in which the coupling between velocity/pressure and temperature is important, as is the case in problems involving free-convection flows. A comparison of the proposed procedure with a standard iterative method shows improvement both in terms of computing speed (a factor of 2.1 to 4.1) and robustness.*

### 1. INTRODUCTION

Buoyancy-driven or natural-convection flows are those generated by density gradients, which in most cases arise from some imposed external heat source. These flows are typical in atmospheric science applications but occur in many engineering applications as well. The challenge is that most (not to say all) numerical methods designed to handle the pressure–velocity coupling, based for example on the SIMPLE procedure of Patankar and Spalding [2] or other segregated method, when applied to buoyant flows result in a considerable increase in the number of iterations for convergence as the strength of buoyancy raises. This difficulty in the computation of free-convection flows has been recognized for a long time (Caretto et al. [3]), but it is fair to acknowledge that the efforts to overcome it have not been very successful. In fact, a survey of the literature reveals that what most authors having to deal with problems involving buoyancy tend to do is just to append the energy equation to the momentum/pressure solution procedure, and hope that the sequential method will eventually converge (e.g., Jang et al. [4]). Important exceptions are the work of Galpin and Raithby [5] and, more recently, that of Sheng et al. [6–7], which are discussed below.

The main purpose of this work is the development of a better numerical method for the computation of buoyant flows. A simple flow geometry is chosen as

Received 8 March 2001; accepted 27 July 2001.

Address correspondence to Dr. P. J. Oliveira, Universidade da Beira Interior, Departamento de Eng<sup>a</sup> Electromecânica, 6201-001 Covilhã, Portugal. E-mail: pjpo@ubi.pt

## NOMENCLATURE

|                 |  |                                    |  |
|-----------------|--|------------------------------------|--|
| $a_P, a_m, a_0$ | coefficients in the discretized equations                        | $\alpha_u$                         | underrelaxation factor for momentum                |
| $c_p$           | specific heat  | $\beta$                            | coefficient of thermal expansion                   |
| $C_\ell, C_L$   | Courant number (local and global)                                | $\delta x$                         | control-volume size                                |
| $g$             | gravity acceleration   | $\delta t$                         | time step  |
| $H$             | convective/diffusive operator<br>[ $H(\phi) = \sum a_m \phi_m$ ] | $\Delta T$                         | temperature difference ( $= T_h - T_c$ )           |
| $k$             | conductivity   | $\Delta_i$                         | gradient operator                                  |
| $L$             | side of square cavity  | $\mu, \nu$                         | dynamic and kinematic viscosity                    |
| $Nu$            | Nusselt number ( $= hL/k$ )                                      | $\rho$                             | density  |
| $p$             | pressure   | $\mathcal{V}$                      | volume of a control volume                         |
| $q_w$           | wall heat flux   | <b>Subscripts and superscripts</b> |  |
| $Pr$            | Prandtl number ( $= \mu c_p / k$ )                               |                                    |  |
| $Ra$            | Rayleigh number<br>( $= g\beta \Delta T L^3 / \alpha \nu$ )      | $h, c, 0$                          | hot, cold, and reference (temperatures)            |
| $S$             | source term in the discretized equations                         | $i, j$                             | Cartesian components                               |
| $t$             | time   | $P, m$                             | main control volume and its neighbors              |
| $T$             | temperature  | $u_i, T$                           | velocity, temperature (coefficients; $H$ operator) |
| $u_i$           | velocity components  | $n, n+1$                           | previous and present time levels                   |
| $x_i$           | spatial coordinates  | $*, **, ***$                       | intermediate values in algorithm                   |
| $\alpha$        | thermal diffusivity  |                                    |  |

test case: the two-dimensional square cavity with vertical walls heated at two different temperatures and horizontal walls adiabatic (see Figure 1). This problem has been extensively investigated because of the following reasons. (1) It models many real situations with interest to different engineering fields: double glazing, glass

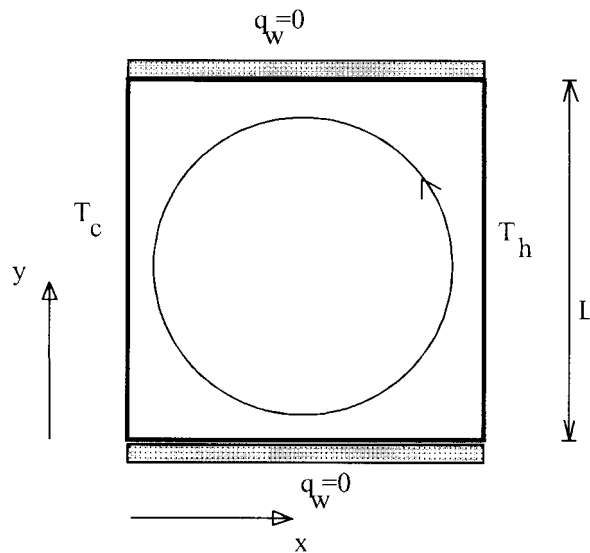


Figure 1. Sketch of the flow geometry.

melting furnaces, motion of magma within tectonic plates (geophysical and astrophysical interest), cooling of some types of nuclear reactors, solar panels, room ventilation by natural convection, etc. (2). The apparent physical simplicity hides the difficult challenge to the numerical methods mentioned above: coupling of the energy and hydrodynamics equations becomes very stiff as the Rayleigh number ( $Ra$ ) increases, resulting in severe numerical instability difficulties.

This article is motivated by the latter reason, and the idea is to extend the operator-splitting algorithm PISO (Issa [1]) to the case of free-convection flows, in which there is a three-variable coupling: velocity, pressure, and temperature. Before formulating and explaining the numerical procedure developed here, a short review of previous relevant numerical work is presented.

Most of the early simulations of natural convection in a rectangular cavity were based on the vorticity/streamfunction formulation, and a classical reference on that is the work of de Vahl Davis [8]; a more up-to-date review of that, and other work with the primitive-variables formulation, is provided in the article of Barakos et al. [9]. During the past 5 years or so, that problem has been subjected to studies using finite-difference methods with parallel computation [10], finite-element methods (FEM) of the coupled Galerkin type [11], or decoupled type with time factorization [12] and pressure-correction techniques [13], and finite-volume methods (FVM) in nonstaggered meshes with various algorithms: SIMPLE [14, 15], SIMPLER [16], and PISO with unstructured meshes [17]. In all these studies the energy equation is either solved in a coupled way with the other equations (in FEM) or it is appended to the iteration loop and solved sequentially at the end of the pressure-velocity algorithm. Emphasis has often been placed on obtaining very accurate results which can be used for benchmarking, e.g., Hortmann et al. [14] for  $Ra = 10^4 - 10^6$ , Le Quere [18] for  $Ra = 10^6 - 10^8$ , Syrjälä [11] for  $Ra = 10^4 - 10^7$ , and Nonino et al. [13] for  $Ra = 10^5 - 10^8$ . Of particular note is the work of Hortmann et al. [14], who used very fine meshes (as fine as  $640 \times 640$ ) together with the full multigrid method. The multigrid method is a numerical technique that can be applied to any fluid flow problem and has nothing specific to free-convection flows.

A close look to the above works reveals a common feature: buoyancy effects lead to considerable requirements in terms of CPU time and computer memory, especially at high  $Ra$ . For example, in [12] it is mentioned that 24 h of CPU time are required to solve the problem with a 166-MHz Pentium computer in a  $16 \times 16$  FEM mesh at the low value of  $Ra = 10^3$  (a similar computation with the present method takes a few seconds). In [13], from 2,500 up to 9,000 iterations are required, for  $Ra$  from  $10^5$  to  $10^8$ , with time steps as small as  $5 \times 10^{-6} L^2/\alpha$  (here some 400 iterations with  $\delta t = 10^{-4}$  are sufficient). In [11] it is stated that difficulty and cost increase considerably when  $Ra$  goes from  $10^6$  to  $10^7$ . It thus appears that a better way of integrating the energy equation into the flow solver is required, so that the interlinkage between the temperature and velocity fields is more effectively accounted for and hence results in a less steep increase in computer time as  $Ra$  increases.

There are not many studies with emphasis on improving the numerical scheme for buoyant flows. The numerical treatment of the temperature-velocity coupling is addressed by Galpin and Raithby [5], who devised a better Newton-like linearization of the advective fluxes in the energy equation which led to a tighter treatment of the coupling between the governing equations. They then used a special coupled solution

method to solve for  $u, p$ , and  $T$  at a given node, without relying on the SIMPLE pressure-correction algorithm. Galpin and Raithby's method compared favorably with others, showing improved convergence rate for fine meshes, but its blocklike nature makes it less attractive for general applications and difficult to implement into existing segregated solvers. In addition to that work, an attempt to improve the SIMPLE algorithm for buoyant flows was recently given by Sheng et al. [6] and [7]. These authors relate the velocity corrections to both pressure and temperature (or density) variations, but while a pressure-correction equation can be derived in the usual way, the temperature variation results from an additional solution of the energy equation, after the momentum prediction. The results presented in [7], and in [6] for high  $Ra$ , show hardly any improvement over the simpler sequential solution of the energy equation at the end of the iterative cycle. A possible reason for this deficient behavior may be related to the fact that the intermediate energy equation is solved with a velocity field which does not satisfy continuity.

As mentioned above, the base method in the present study is the PISO algorithm of Issa [1]. If the energy equation is merely appended to the base algorithm, then there is no improvement in the calculation of free-convection flows, as found by Jang et al. [4], who compared the performance of several algorithms, SIMPLER [19], SIMPLEC [20], and PISO. However, the operator-splitting nature of PISO, with various velocity/pressure correction stages following the momentum prediction, makes it much more amenable to different arrangements for the solution of the energy equation, with the possibility of temperature corrections. A careful choice of the algorithm results in better numerical behavior, as the present results will show.

## 2. FORMULATION OF THE PROBLEM

The objective is to develop an efficient numerical method for the solution of the motion and energy equations of an incompressible Newtonian fluid whose movement is caused by natural convection. The flow is assumed to be laminar and the Boussinesq approximation valid, so that the continuity, momentum, and energy equations become

$$\frac{\partial}{\partial x_i}(\rho u_i) = 0 \quad (1)$$

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_j u_i) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j} \left( \mu \frac{\partial u_i}{\partial x_j} \right) - \rho \beta (T - T_0) g_i \quad (2)$$

$$\frac{\partial}{\partial t}(\rho c_p T) + \frac{\partial}{\partial x_j}(\rho c_p u_j T) = \frac{\partial}{\partial x_j} \left( k \frac{\partial T}{\partial x_j} \right) \quad (3)$$

In these equations the dependent variables are the velocity components,  $u_i$ , pressure,  $p$ , and temperature,  $T$ . The physical properties of the fluid are the density,  $\rho$ , viscosity,  $\mu$ , specific heat at constant pressure,  $c_p$ , thermal conductivity,  $k$ , and the volumetric factor of thermal expansion,  $\beta$ , which are all assumed constant. The flow occurs inside the square cavity represented in Figure 1 and the only nonzero component of gravity is  $g_2 \equiv -g$ . Although the interest is only on steady-state solutions,

the time-dependent terms are retained in Eqs. (2) and (3) because the numerical scheme is based on a time-marching advancement procedure.

Boundary conditions for Eqs. (1)–(3) are as follows: no slip applies on the four surrounding walls; imposed cold and hot temperatures on the vertical walls,  $T = T_c$  ( $\equiv T_0$ ), at  $x = 0$ , and  $T = T_h$  at  $x = L$ ; and a zero heat flux on the horizontal walls,  $\partial T / \partial y = 0$  at  $y = 0$  and  $y = L$ .

This problem is characterized by three nondimensional numbers: Prandtl,  $\text{Pr} = \mu c_p / k$ ; Rayleigh,  $\text{Ra} = g \beta \Delta T L^3 / (\nu \alpha)$ ; and Nusselt,  $\text{Nu} = hL / k = q_w L / (\Delta T k)$ , where  $\Delta T \equiv (T_h - T_c)$ ,  $q_w$  is the heat flux across the vertical walls, and  $h$  is the coefficient of heat transfer. The strength of the coupling between the momentum and energy equations is quantified by the Rayleigh number, as shown by the non-dimensional form of the governing equations. When  $\text{Pr} \geq 1$  the usual practice is to choose  $\alpha / L$  as the characteristic velocity, so that the steady form of the non-dimensional  $y$ -momentum equation becomes

$$\frac{\partial}{\partial X}(UV) + \frac{\partial}{\partial Y}(VV) = -\frac{\partial P}{\partial Y} + \text{Pr} \left( \frac{\partial^2 V}{\partial X^2} + \frac{\partial^2 V}{\partial Y^2} \right) + \text{Ra} \cdot \text{Pr} \cdot \theta$$

It is evident from this equation that for constant  $\text{Pr}$  (here we use  $\text{Pr} = 1$  or  $0.71$ ), the free-convection source term ( $= \text{Ra} \text{Pr} \theta$ ,  $\theta = (T - T_0) / \Delta T$ ) becomes dominant as the Rayleigh number  $\text{Ra}$  is increased, leading to a set of coupled stiff partial differential equations. This term should therefore be treated as implicitly as possible to avoid numerical instabilities; this is the objective of the present work, and to this aim a new arrangement of the operator-splitting procedure PISO [1] has been devised, as explained in the next section.

### 3. NUMERICAL PROCEDURE

The set of equations (1)–(3) is transformed into a set of algebraic equations after application of any standard discretization procedure in a staggered mesh (e.g., Patankar [19]). As mentioned in [1], the numerical algorithm is basically independent of the particular differencing scheme chosen; here the backward, fully implicit scheme is used to represent the time derivatives, and the hybrid and central differencing schemes are used to approximate the convective and diffusive spatial derivatives, respectively. With the operator notation introduced by Issa [1], the finite-difference counterpart of the momentum (2) and energy (3) equations is

$$\left( \frac{\rho \mathcal{V}}{\delta t} + a_0^{u_i} \right) u_{iP} = H(u_i) - \Delta_i p + S^{u_i} + \left( \frac{\rho \mathcal{V}}{\delta t} \right) u_{iP}^n \quad (4)$$

$$\left( \frac{\rho \mathcal{V}}{\delta t} + a_0^T \right) T_P = H^T(T) + S^T + \left( \frac{\rho \mathcal{V}}{\delta t} \right) T_P^n \quad (5)$$

In the above equations, subscript  $P$  denotes values at the central node of the computational molecule and  $H$  stands for the operator which accounts for the influence of neighbor nodes,  $H(u_i) = \sum a_m^{u_i} u_m$  ( $m$  from 1 to number of neighboring cells, 4 in 2-D). The coefficients  $a_m$  are composed by convective and diffusive fluxes across cell faces, its sum is given by  $a_0 = \sum a_m$ , and  $\Delta_i$  is the finite-difference representation of

the gradient operator. The source terms  $S$  include all the remaining terms, which for momentum comprise the free-convection contribution, given by  $\rho g \beta \mathcal{V}(\bar{T}_P - T_0)$ , where  $\bar{T}_P$  is the temperature interpolated at velocity point  $P$ . The computational time step is denoted by  $\delta t$ , previous time-level values are denoted with  $n$ , and  $\mathcal{V}$  represents the volume of a computational cell.

The continuity equation (1) is used to obtain an equation for pressure by taking the divergence of the discretized momentum equation.

In the remainder of this section, we give first the best algorithm to emerge from the numerical experiments (to be discussed in the Results section) and then we briefly describe other variants that have been tested. The influence of the solver on the algorithm performance is discussed at the end of the section.

### 3.1. Proposed Solution Algorithm

A form of the operator-splitting PISO algorithm of Issa [1] is now presented which embeds the temperature variation of the momentum source  $S^{u_i}$  into the velocity–pressure coupling treatment. Following the original presentation, the different predictor–corrector level of variables and coefficients are denoted with superscripts  $n$ ,  $*$ ,  $**$ ,  $***$ . The central coefficient is denoted  $a_P = a_0 + \rho \mathcal{V} / \delta t$ .

#### Step 1. Momentum predictor.

$$(a_P^{u_i})^n u_{i_P}^* = H^n(u_i^*) - \Delta_i p^n + (S^{u_i})^n + \left( \frac{\rho \mathcal{V}}{\delta t} \right) u_{i_P}^n \quad (6)$$

This implicit equation is to be solved for the velocity components at all control volumes,  $u_i^*$ .

#### Step 2. First velocity corrector.

$$(a_P^{u_i})^n u_{i_P}^{**} = H^n(u_i^*) - \Delta_i p^* + (S^{u_i})^n + \left( \frac{\rho \mathcal{V}}{\delta t} \right) u_{i_P}^n \quad (7)$$

Note that velocity in the  $H$  operator and in the term on the left-hand side are evaluated at different iteration levels, hence the terminology “operator splitting.”

**Step 3. First pressure-correction equation.** After subtraction of Eq. (6) from Eq. (7), division by  $a_P^{u_i}$ , and making use of the discretized form of the continuity equation

$$\Delta_i(\rho u_i^{**}) = 0 \quad (8)$$

the first pressure-correction equation is obtained:

$$\Delta_i \left( \frac{\rho}{(a_P^{u_i})^n} \Delta_i(p^* - p^n) \right) = \Delta_i(\rho u_i^*) \quad (9)$$

These first three steps are part of the standard PISO and also correspond to the SIMPLE algorithm.

**Step 4. Temperature predictor.**

$$(a_p^T)^* T_p^* = H^T(T^*) + (S^T)^n + \left(\frac{\rho\mathcal{V}}{\delta t}\right) T_p^n \quad (10)$$

where  $H^T(T)$  is based on coefficients  $(a_m^T)^*$ , which are to be calculated with the corrected velocities  $u_i^*$ , and the equation indicates an implicit solution for the temperature field, thus requiring a linear equation solver.

**Step 5. Second velocity corrector.** Step 5 is based on the following operator-splitting equation:

$$(a_p^{u_i})^n u_{i_p}^{***} = H^n(u_i^{**}) - \Delta_i p^{**} + (S^{u_i})^* + \left(\frac{\rho\mathcal{V}}{\delta t}\right) u_{i_p}^n \quad (11)$$

where it is important to note that the free-convection source term is based on the newly computed temperature  $T^*$ , from Eq. (10). Steps 4 and 5 depart from the standard PISO.

**Step 6. Second pressure-correction equation.** Following a procedure similar to the described above, the second pressure-correction equation is obtained:

$$\Delta_i \left[ \frac{\rho}{(a_p^{u_i})^n} \Delta_i (p^{**} - p^*) \right] = \Delta_i \left\{ \frac{\rho}{(a_p^{u_i})^n} [(S^{u_i})^* - (S^{u_i})^n] + H^n(u_i^{**} - u_i^*) \right\} \quad (12)$$

where the triple-starred velocities are forced to satisfy the continuity equation:

$$\Delta_i (\rho u_i^{***}) = 0 \quad (13)$$

The important point here is that the pressure-correction equation (12) now includes a term which arises from the variation of the free-convection source term with the temperature. Hence a tighter coupling between the velocity and temperature fields is achieved.

**Step 7. First temperature corrector.**

$$(a_p^T)^{**} T_p^{**} = H^{T^{**}}(T^*) + (S^T)^n + \left(\frac{\rho\mathcal{V}}{\delta t}\right) T_p^n \quad (14)$$

where  $H^{T^{**}}(T)$  is based on the coefficients  $(a_m^T)^{**}$ , which are calculated with the corrected velocities  $u_i^{***}$ . This equation is solved pointwise and does not require a call to the solver.

Steps 1–7 define the proposed algorithm, which is an optimized version of the operator-splitting procedure for free-convection flows. These steps are successively applied at each time level and the fields given by  $p^{**}$ ,  $u_i^{***}$ , and  $T^{**}$  correspond, to within a certain truncation error [1], to those of the next time step,  $p^{n+1}$ ,  $u_i^{n+1}$ , and  $T^{n+1}$ . The steady-state solution is thus approached by a sequential time-marching procedure which does not require iteration within each time step. Proximity to the steady-state solution is assessed by controlling the maximum residuals of all equations; these residuals are computed from newly obtained coefficients and previous time-level variables.

### 3.2. Other Algorithms

Numerical experiments revealed that the algorithm described in Section 3.1 yields the best results, in the sense of less computational time and wider range of stability. Its performance will be compared with a standard procedure in Section 4. It is, however, worthwhile to mention briefly other algorithm variants which have been tested, and which are modifications or rearrangements of the above:

**Variant 1.** The temperature predictor is the first step, before the momentum predictor, and the first temperature corrector comes after the first velocity corrector; it may, or may not, have a second temperature corrector at the end of the cycle. The sequence of computed variables is

$T^*$   
 $u^*, v^*$   
 $p^*, u^{**}, v^{**}$   
 $T^{**}$   
 $p^{**}, u^{***}, v^{***}$   
 with, or without,  $T^{***}$

**Variant 2.** The temperature predictor occurs after the velocity predictor, and then follows variant 1 (note that variation of the free-convection source  $S^{u_i}$  is included into the first corrector of  $u_i$ ):

$u^*, v^*$   
 $T^*$   
 $p^*, u^{**}, v^{**}$  [including effect of  $(S^u)^* - (S^u)^n$ ]  
 $T^{**}$   
 $p^{**}, u^{***}, v^{***}$   
 with, or without,  $T^{***}$

**Variant 3.** This variant basically follows the proposed algorithm except the temperature corrector (step 7), which is solved implicitly, therefore being more appropriately called a second temperature predictor.

**Variant 4.** Standard PISO (as in [1]) with the temperature equation solved sequentially at the end of the cycle.

**Variant 5.** All the above arrangements (variants 1–4) have been tested with recalculation of the velocity coefficients; i.e.,  $(a_p^{u_i})^n$  and  $H^n(u_i^{**})$  in Eq. (11) are replaced by  $(a_p^{u_i})^*$  and  $H^*(u_i^{**})$ .

An assessment of these algorithm variants, based on numerical experiments, may be summarized as follows:

- Variant 2 did not perform well, a plausible cause being the fact that the temperature prediction is based on a velocity field which does not satisfy continuity.
- Variant 1 performed better than variant 4, but not as well as the proposed algorithm. In general, variant 4 (the standard PISO) did not offer significant



advantages over SIMPLEC, when both algorithms were used with iterative marching (with underrelaxation factors), thus confirming the findings of Jang et al. [4]. With time marching, the standard PISO would tend to work better than SIMPLEC but, as mentioned above, not up to the proposed algorithm.

- Variant 3 shares some of the good behavior of the proposed algorithm but has a drawback: the point of minimum CPU time presents a stronger dependence on the grid size and Courant number.
- Recalculation of the coefficients  $a^{u_i}$  in variant 5 was always penalized by the time necessary to perform that operation (5 times more than 1 iteration of the conjugate-gradient solver).

### 3.3. Solution of the Linear Sets of Equations

Several sets of linear algebraic equations must be solved at each time step, and the overall performance of the numerical algorithm depends on the method used to solve them. It is well known that the pressure-correction equations should be solved to a tight tolerance if propagation of errors resulting from a velocity field failing to satisfy the divergence-free condition is to be avoided. This requirement, together with the Poisson-like behavior and Neumann-type boundary conditions of the pressure-correction equation, usually result in a great number of inner iterations at each time step, so an efficient solver is required. The conjugate gradient solver for symmetric matrices (CGS) offers a good compromise between simplicity and efficiency. The convergence rate of this method is high, especially when linked to an appropriate preconditioner. In the present study the version of CGS with incomplete-Choleski preconditioner given by [21] was adopted for both the PISO and SIMPLEC. The simpler application of line-by-line iteration with the tridiagonal matrix algorithm was used for the other equation sets, which are much easier to solve since the upwind differencing scheme leads to coefficient matrices in which some of the elements are zero when convection dominates. These matrices are also no longer symmetric, as is the case of the matrix for the pressure equation, and thus conjugate-gradient solvers become more involved.

The number of inner iterations required to solve each of the linear equation sets may be fixed, as recommended by Patankar [19] with the argument that the equations need not be solved very precisely since the overall procedure is itself iterative, or may vary according to a specified tolerance for the relative decay of the residuals (e.g., [20]). Numerical experiments showed that the latter approach is more efficient and so it is adopted here. The number of inner iterations necessary to satisfy a predefined residual tolerance denoted by  $\gamma$  depends on the Rayleigh number and tends to increase for low Rayleigh numbers ( $Ra = 10^3$  and  $10^4$ ), since the importance of diffusion is then accentuated. It is also interesting to mention that the number of inner iterations to solve the energy equation was greater than for the momentum equations. The reason for this is related to the different type of boundary conditions for each variable: Neumann type for  $T$  and Dirichlet for  $u_i$ . The value  $\gamma$  used to stop the inner iterations (when residual/initial residual  $< \gamma$ ) was optimized so as to minimize the computing time; the value  $\gamma = 0.25$  was found to offer the best results for the momentum and energy equations, and values of  $\gamma = 0.1$  and  $0.05$  were found for the pressure-correction equations, respectively, for  $Ra > 10^4$  and  $\leq 10^4$ .

#### 4. RESULTS

Assessment of the proposed method has been carried out by performing a great number of computer runs, where relevant parameters were individually varied, and from analysis of the resulting CPU times in terms of a nondimensional time step. The method chosen for comparison is the iterative SIMPLEC algorithm, implemented as described in its original reference, Van Doormaal and Raithby [20]. All the arrangements of the operator-splitting procedure described in Section 3, and also the comparison method, have been implemented into a single computer code, based on the staggered-mesh arrangement, in such a way that one could easily switch methods by proper choice of a parameter. Hence it is assured that CPU times are not falsified by different programming. A PC Pentium Pro computer with a processor running at 200 MHz was used in all computations.

Runs were made for  $Ra$  ranging from  $10^3$  to  $10^8$ ; higher  $Ra$  were not tested since physical instabilities and transition to turbulent flow are then expected to occur and no turbulence model has been used. Furthermore, we are interested only in determining the numerical behavior of the algorithm to reach a steady-state solution. The results to be presented first correspond to a Prandtl number of unity ( $Pr = 1$ ) and a square cavity. A second case was considered with  $Pr = 0.71$  (valid for air), which allows for a direct comparison with results from other authors. A grid with  $22 \times 22$  nodes was initially adopted, but it was soon realized that proper assessment of the method for high Rayleigh numbers could not be done in such grid since it cannot resolve the flow pattern. This can be inferred from observation of a global parameter characterizing the resulting flow field obtained with different grids. Figure 2 shows the predicted Nusselt number as a function of the number of internal cells in the grid ( $N$ ); it is seen that only the fine grids,  $30 \times 30$  ( $N = 784$ ) and  $40 \times 40$  ( $N = 1,444$ ), are able to yield a Nusselt number independent of  $N$  and with sufficient accuracy. The computational performance of different numerical methods cannot be compared adequately if the flow field is not well resolved, otherwise the relative performance may change when the grid is refined. Higher accuracy can be achieved with non-uniform grid spacing, as in the second case discussed below.

The time-marching behavior of the modified PISO method compared with the iterative SIMPLEC can be seen in Figures 3–6, which correspond to Rayleigh numbers from  $10^4$  to  $10^7$ . Curves of processor time (CPU) against global Courant number are given for two grids,  $30 \times 30$  and  $40 \times 40$ . The global Courant number ( $C_L$ ) is defined with a typical velocity ( $U$ , as in [5]) and the cavity width, that is,

$$C_L = \frac{\delta t U}{L} \quad \text{with } U \equiv \frac{\alpha}{L} \left( \frac{Ra}{1 + 1/Pr} \right)^{1/2} \quad (15)$$

This definition for characteristic velocity is very convenient, since  $U$  can be readily determined for a given  $Ra$  and  $Pr$ ; a more precise, but problem-dependent, characteristic velocity could be defined with the maximum velocity in the resulting solution field. However, it was found that its value correlates well with the above choice for  $U$ , being  $u_{\max} \approx U/3$ . Since the grids used in this first test case were uniform and square, a local Courant number can be obtained from  $C_\ell = \delta t \cdot U / \delta x = N_i \cdot C_L$ , where  $N_i$  is the number of internal cells along  $x$  or along  $y$ .

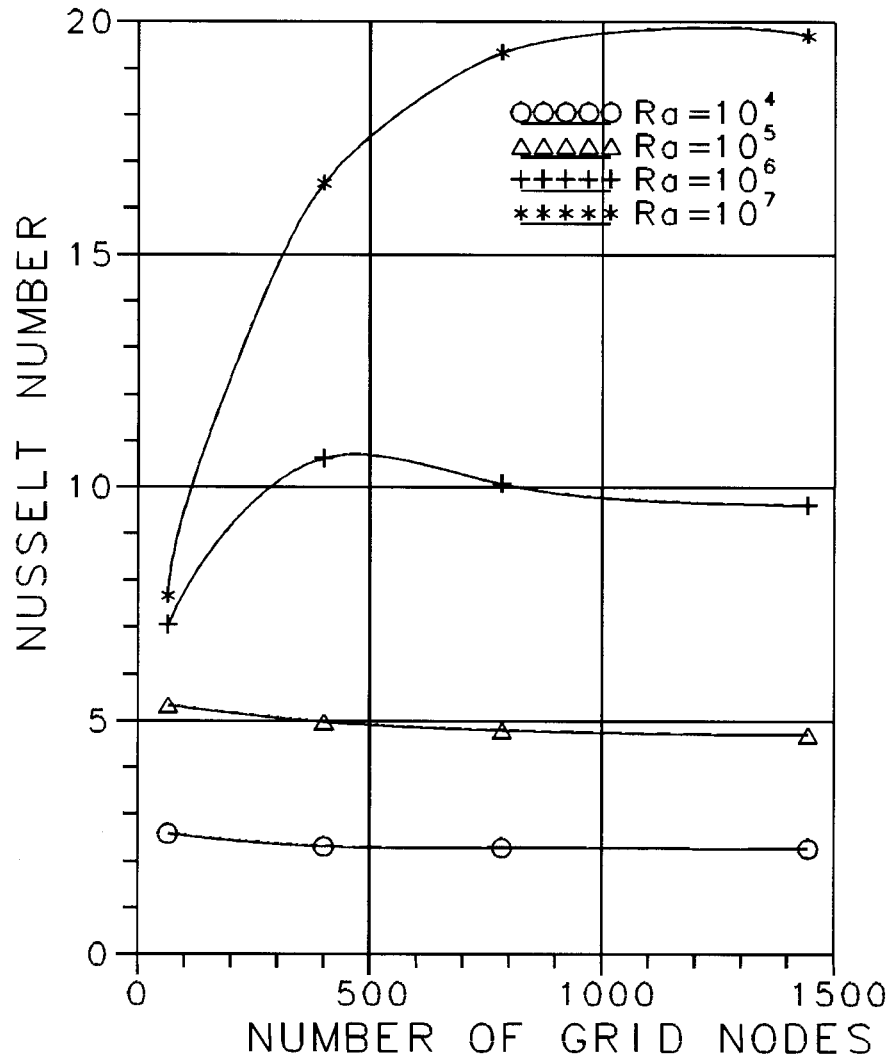


Figure 2. Effect of mesh size on predicted Nusselt number.

For the iterative method the curves were obtained by varying the underrelaxation factor in the momentum equation,  $\alpha_u$ , and determining an equivalent time step from the analogy between time-marching and iterative approaches (see [1] and [20]):

$$\delta t = \frac{\alpha_u}{1 - \alpha_u} \left( \frac{\rho \mathcal{V}}{a_0^u} \right) \quad (16)$$

This is sometimes called a “pseudo” or “distorted” time step because while for a time-marching algorithm  $\delta t$  is constant, the equivalent time step in an iterative marching algorithm varies from point to point, as shown by Eq. (16). In order to

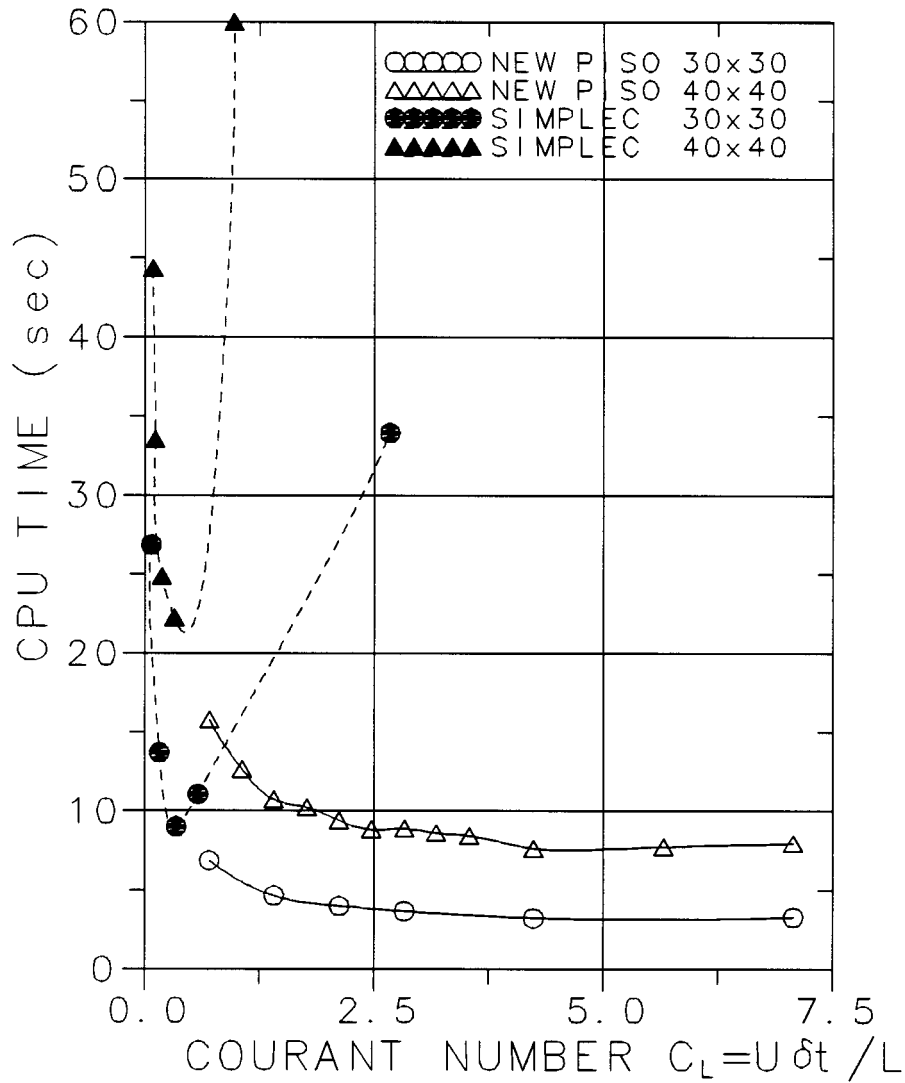


Figure 3. Comparison of time-marching behavior for  $Ra = 10^4$ .

obtain a single value for the equivalent time step for the iterative algorithm, the last term in Eq. (16) is evaluated as  $\text{Min}(\rho V/a_0'')$ .

Figures 3–6 show that the new algorithm is not only faster than SIMPLEC but also presents, in general, a wider range of stability. This range becomes narrower when the Rayleigh number increases, as would be expected. Inspection of Figures 3–6 also shows that the Courant number corresponding to the point of minimum computing time is approximately independent of both the Rayleigh number and the grid size. This optimum Courant number is roughly equal to  $C_L \approx 1.8$ . As a consequence, one is able to start a computation, without any preliminary trials, using an almost optimum time step given by

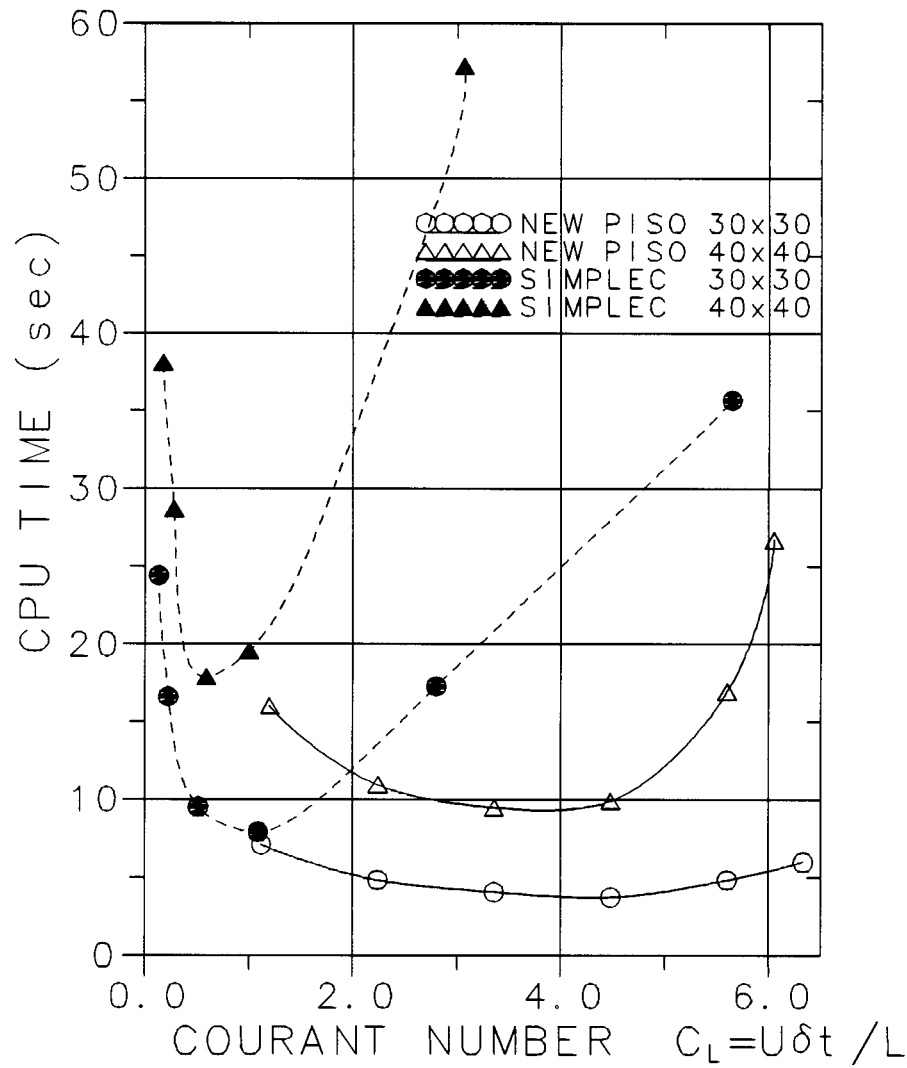


Figure 4. Comparison of time-marching behavior for  $Ra = 10^5$ .

$$(\delta t)_{\text{opt}} \approx 1.8 \left( N_i \frac{\delta x}{U} \right) \quad (17)$$

In terms of a local Courant number defined with the maximum local velocity [ $C_{l\text{max}} = \delta t \cdot \text{Max}(u/\delta x)$ ], which is a more appropriate measure of the stability brought about by the present implicit algorithm, the points of minimum CPU time in Figures 3–6 have values ranging from 45.4 at  $Ra = 10^4$  to 25.3 at  $Ra = 10^7$ .

Figures 7a and 7b show the minimum CPU times achieved with optimized values of  $\delta t$  and  $\alpha_u$  using the modified PISO method and SIMPLEC, as a function of the Rayleigh number for two grids. It is seen that for the finer grid, gains in CPU

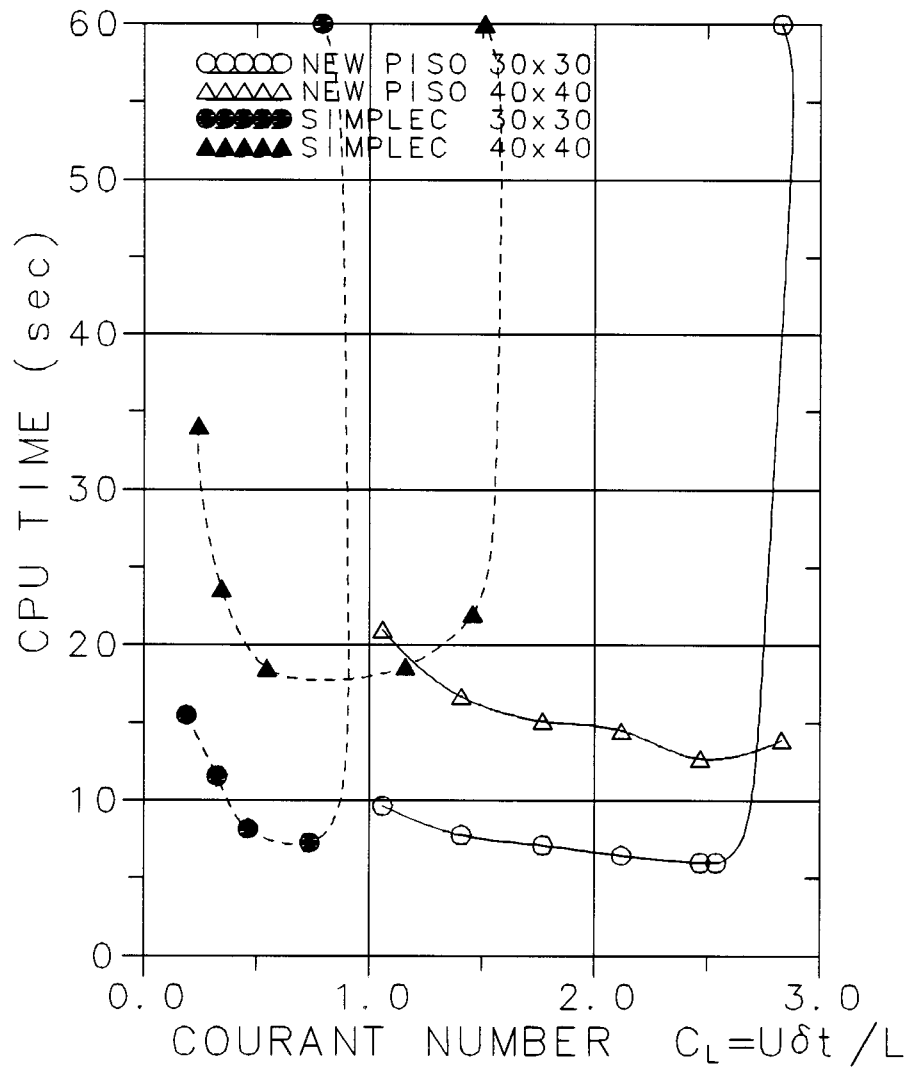


Figure 5. Comparison of time-marching behavior for  $Ra = 10^6$ .

time are quite substantial. In relative terms, the new method is faster than SIMPLEC by a factor ranging from 1.5 to 2.8, when both methods are optimized. In actual computations of free-convection flows this factor will be higher, since it is not practical to optimize the methods before the actual runs, and SIMPLEC shows a very sharp dependence of the optimum point on the underrelaxation factor  $\alpha_u$  (Figures 3–6). With SIMPLEC, any departure of  $\alpha_u$  from the optimum value results in a considerable increase of CPU time from its minimum level. The increase in CPU time seen in Figure 7a when  $Ra$  drops from  $10^4$  to  $10^3$  can be explained as due to the onset of diffusion effects. At low  $Ra$ , diffusion effects are dominant over convection and the energy and momentum equations become like Poisson equations

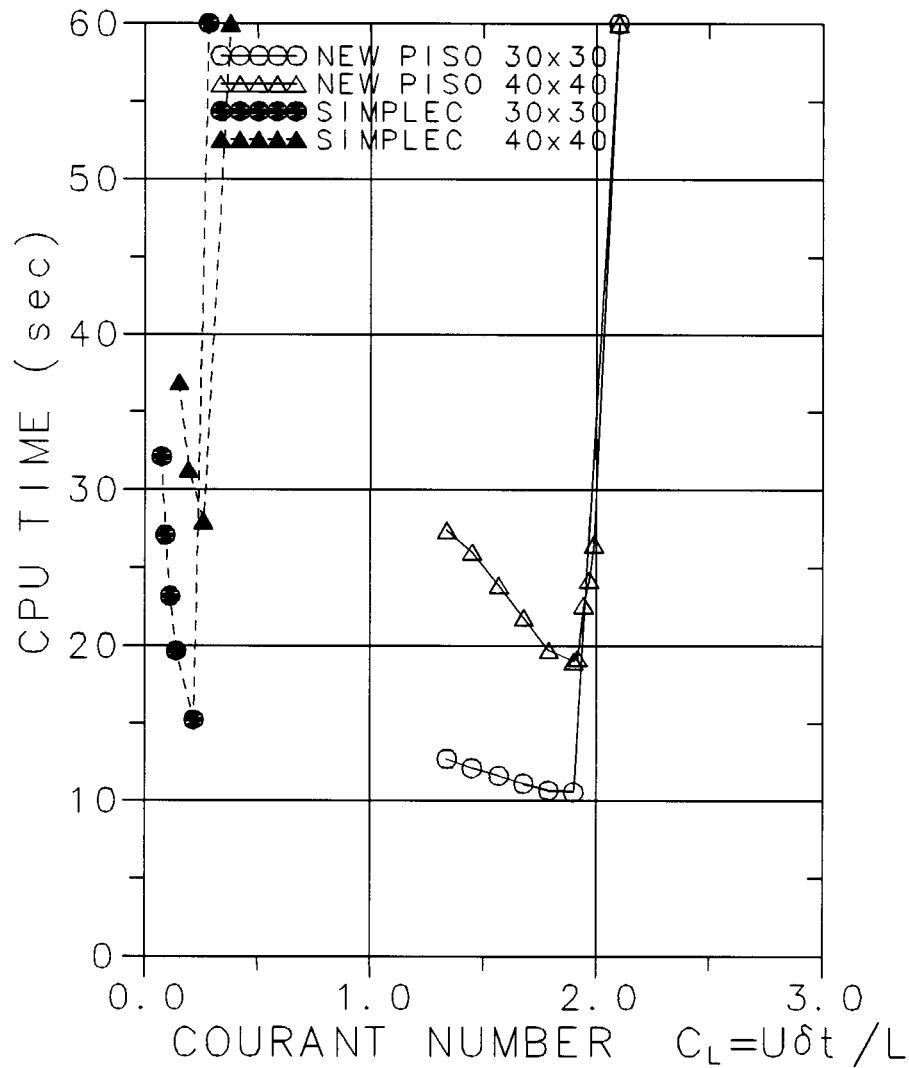
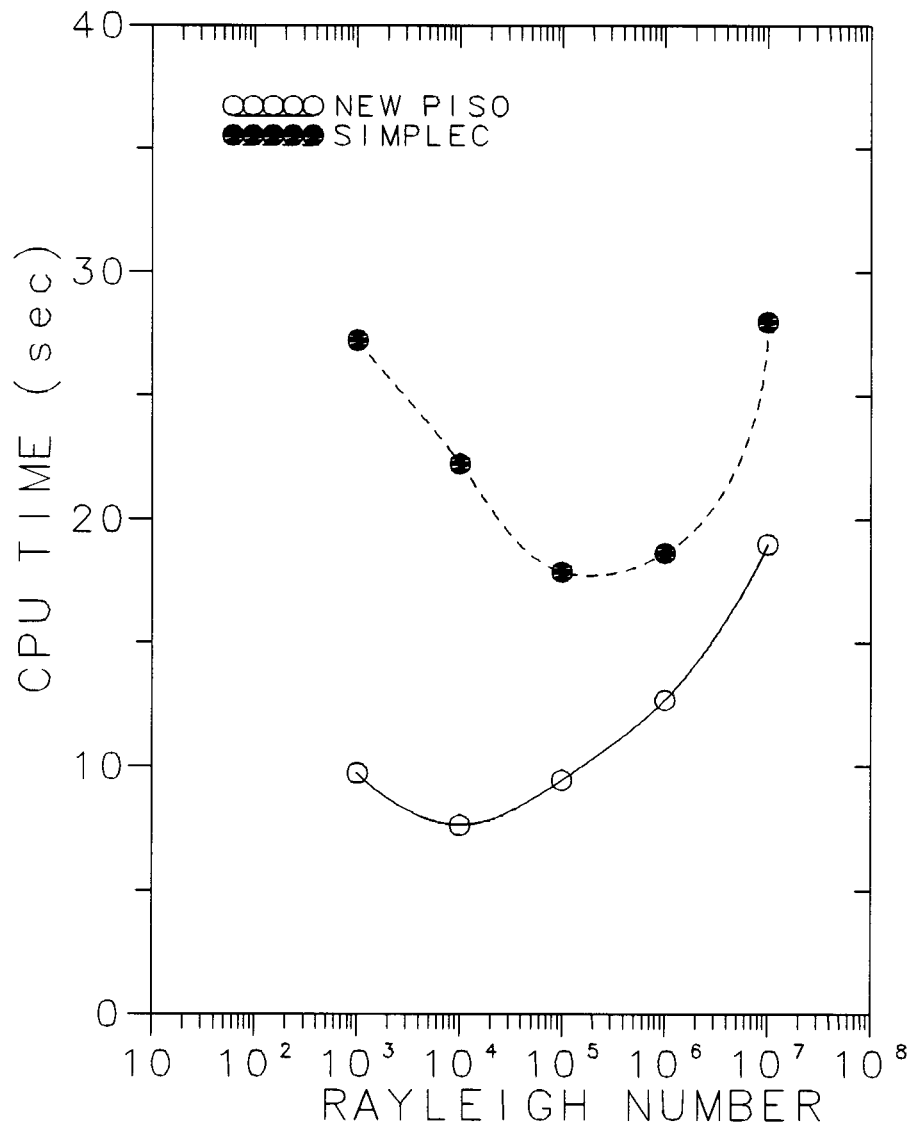


Figure 6. Comparison of time-marching behavior for  $Ra = 10^7$ .

(diffusion-dominated equations), with the consequence that many inner iterations within the solver are required to bring the residuals below the prespecified tolerance.

A second case with  $Pr = 0.71$  was simulated on a nonuniform grid of  $42 \times 42$  nodes ( $40 \times 40$  internal control volumes) designed so that relatively accurate predictions could be achieved. Grid spacing is smallest near the walls, with a minimum value of  $\delta x_{\min} = 0.00488L$ , and expands from there toward the cavity center at a constant geometric rate of 15%. Although the purpose of the present study is not related with “accuracy”—otherwise a better differencing scheme would have been used to represent the convective/advective terms in the equations—there is some interest in documenting the resulting Nusselt numbers for this case:  $Nu = 1.117$ ,



(a)

Figure 7a. Minimum processor time as a function of  $Ra$ : grid  $40 \times 40$ .

2.244, 4.513, 8.755, 16.423, and 30.473 at, respectively,  $Ra = 10^3, 10^4, \dots, 10^8$ . These results compare very favorably with the various benchmark data available in the literature (consistent values in [14, 18, 11, and 13]), with relative differences ranging from 0.04% ( $Ra = 10^4$ ), 0.2% ( $Ra = 10^5$ ), to 0.8% ( $Ra = 10^8$ ). We may thus conclude that the present grid is adequate to provide results up to  $Ra = 10^8$  with sufficient “engineering” accuracy (below 1%) and we turn to the main question of algorithm efficiency. Figure 8 shows the minimum CPU times for those runs as a function of



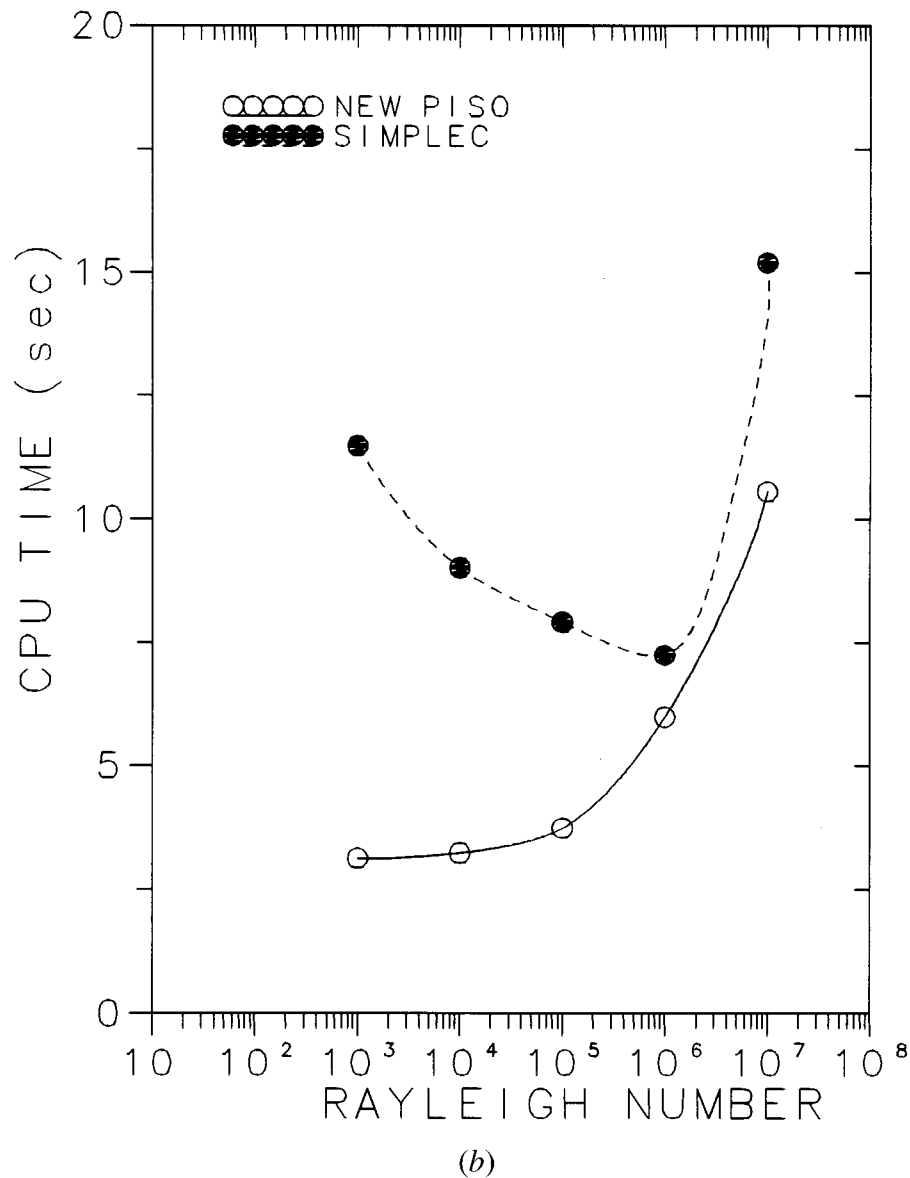


Figure 7b. Minimum processor time as a function of  $Ra$ : grid  $30 \times 30$ .

$Ra$ , for the new PISO algorithm and the standard SIMPLEC. The rate of increase in computer time with the Rayleigh number is much steeper for SIMPLEC (compare with Figure 7a for the uniform grid), and the new PISO is faster than SIMPLEC by a factor ranging from 2.9 at  $Ra = 10^5$  to 5.1 at  $Ra = 10^8$ . The maximum local Courant number for these optimum runs varied from  $C_{lmax} = 19.2$  at  $Ra = 10^5$  to 14.4 at  $Ra = 10^8$ , showing an expected tendency to decrease as  $Ra$  is raised. However, that value for the optimum Courant number is always over 10, reflecting the good stability of the method, and the range of variation is not too wide.

Figure 9 shows the variation of CPU time with the global Courant number for these runs on the nonuniform grid in which good accuracy is achieved. The optimum time step decreases with  $Ra$ , but a time step such that  $C_L \approx 0.9$  yields almost optimum computing times for all cases. Instead of taking a fixed optimum  $C_L$ , as suggested above for the uniform-grid cases, a more refined fitting is given by the correlation

$$C_{L,opt} = 14.8 Ra^{-0.153} \quad (18)$$

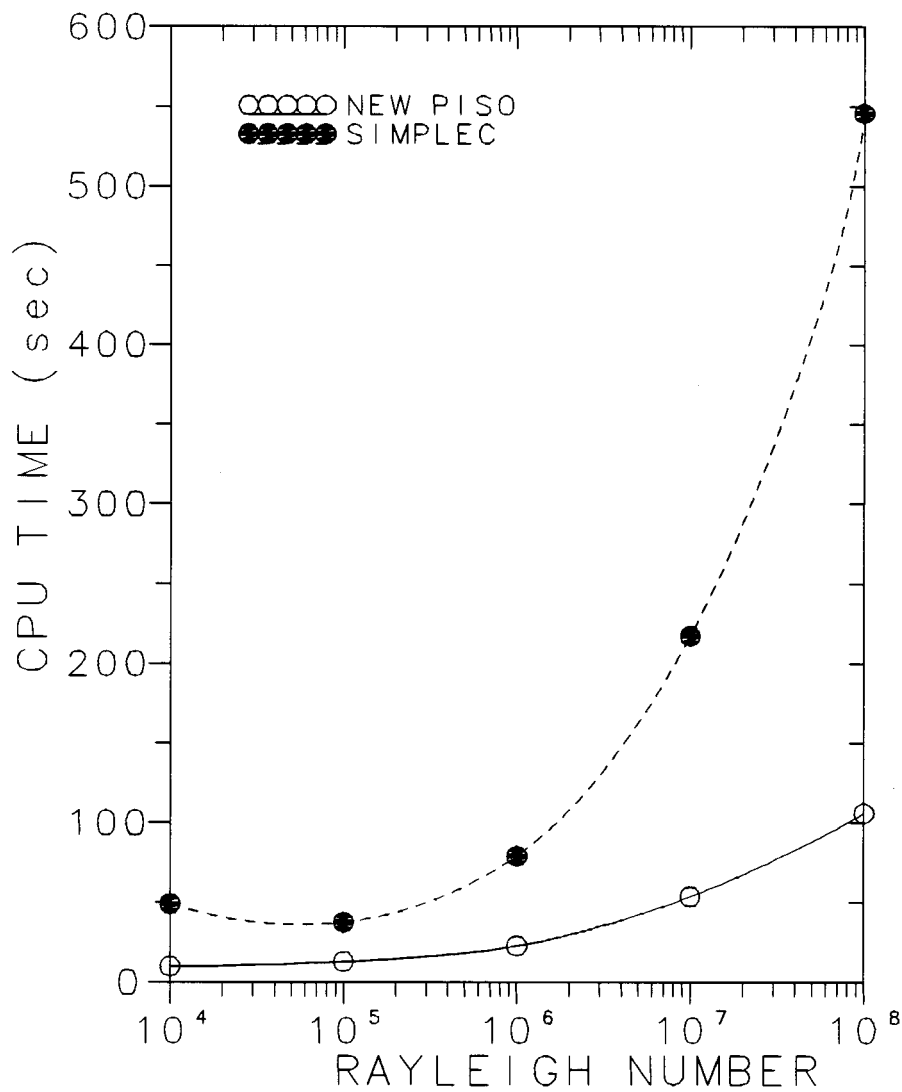
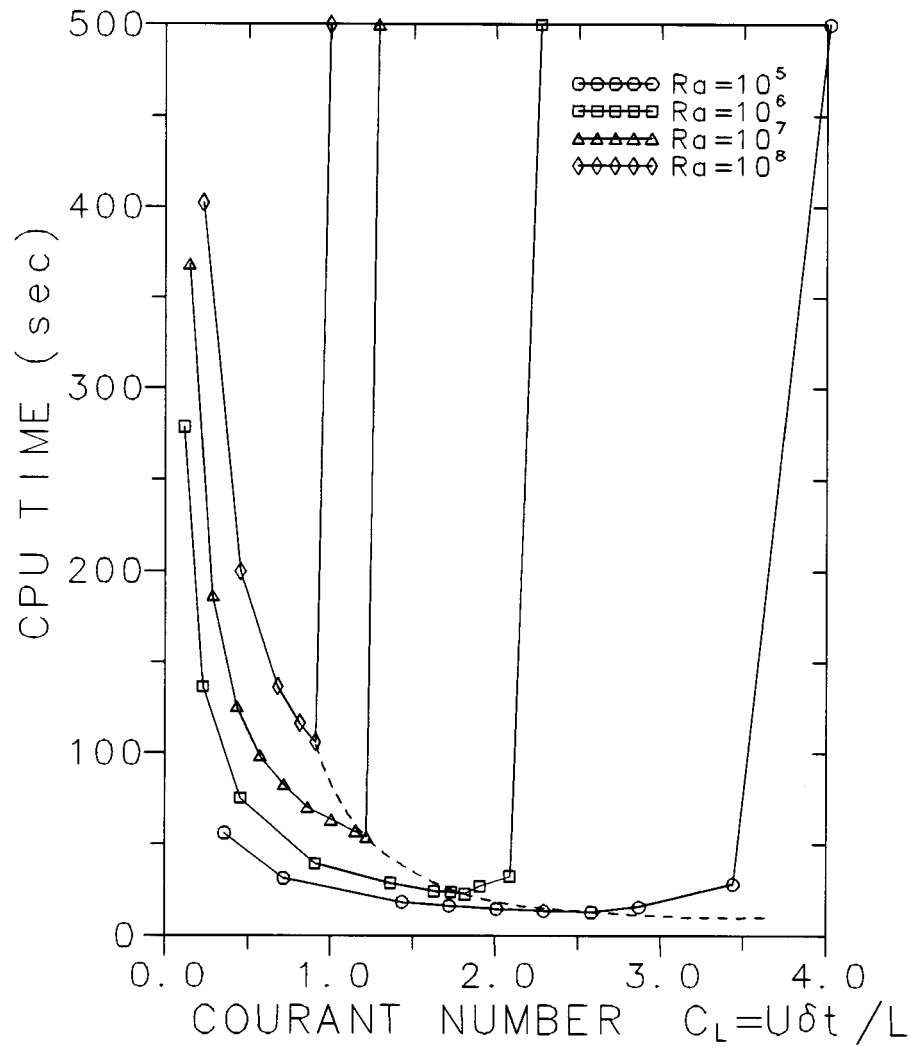


Figure 8. Minimum processor time as a function of  $Ra$  for the case  $Pr = 0.71$  (grid  $42 \times 42$  nonuniform).



**Figure 9.** Time-marching behavior of new PISO for the case  $Pr = 0.71$ , nonuniform grid  $42 \times 42$ . Dashed line correlates points of minimum CPU time.

which can be used to estimate the optimum time steps; this function is represented by the dashed lines in Figure 9.

## 5. CONCLUSIONS

A numerical algorithm especially designed to solve the equations for buoyancy-driven flows has been presented. It is based on the idea of operator splitting, which was originally introduced to deal with the pressure-velocity coupling present in the fluid motion equations, and it is here extended to cater for the additional strong coupling in buoyant flows, between temperature and velocity. The implicit treatment

of the coupling term, essential to dump the numerical instabilities, has been achieved by embedding the calculation of temperature into the predictor-corrector steps of PISO, as follows:

- The temperature equation is solved implicitly after the first velocity-corrector step of PISO.
- The new temperature field changes the buoyancy term in the momentum equations; this change is incorporated into the second velocity corrector and the pressure equation of PISO.
- A final temperature correction is added to the usual PISO so that temperature is adjusted to the new velocity field.

The modified PISO method is assessed by comparison with a standard iterative procedure (SIMPLEC), using the test problem of buoyant flow in a square cavity with differentially heated vertical walls, for several Rayleigh numbers (up to  $10^8$ ) and grid sizes (up to  $42 \times 42$  nonuniform). The results show the new method to be better in terms of both computing speed and stability. There is an improvement in CPU time by a factor ranging from 2.1 (uniform grid) to 4.1 (nonuniform grid, more realistic simulations), on average, when both methods are optimized. This speed-up factor tends to increase sharply when conditions depart from optimal, a sign of improved stability. Indeed, the range of time steps for which the new free-convection PISO enables converged results is much wider than the corresponding equivalent time-step range for SIMPLEC, a measure of improved stability.

Practical usage of the proposed method is facilitated by guidance on the values to assign to the time step in order to have minimum computing times.

## REFERENCES

1. R. I. Issa, Solution of the Implicitly Discretized Fluid Flow Equation by Operator Splitting, *J. Comput. Phys.*, vol. 62, pp. 40–65, 1986.
2. S. V. Patankar and D. B. Spalding, A Calculation Procedure for Heat, Mass, and Momentum Transfer in Three-Dimensional Parabolic Flows, *Int. J. Heat Mass Transfer*, vol. 15, pp. 1787–1806, 1972.
3. L. S. Caretto, A. D. Gosman, and D. B. Spalding, *Removal of an Instability in a Free Convection Problem*, Rep. EF/TN/A/35, Imperial College, London, 1971.
4. D. S. Jang, R. Jetli, and S. Acharya, Comparison of the PISO, SIMPLER, and SIMPLEC Algorithms for the Treatment of the Pressure-Velocity Coupling in Steady Flow Problems, *Numer. Heat Transfer*, vol. 10, pp. 209–228, 1986.
5. P. F. Galpin and G. D. Raithby, Numerical Solution of Problems in Incompressible Fluid Flow: Treatment of the Temperature-Velocity Coupling, *Numer. Heat Transfer*, vol. 10, pp. 105–129, 1986.
6. Y. Sheng, M. Shoukri, G. Sheng, and P. Wood, A Modification of the SIMPLE Method for Buoyancy-Driven Flows, *Numer. Heat Transfer B*, vol. 33, pp. 65–78, 1998.
7. Y. Sheng, M. Shoukri, G. Sheng, and P. Wood, New Version of SIMPLER and Its Application to Turbulent Buoyancy-Driven Flows, *Numer. Heat Transfer A*, vol. 34, pp. 821–846, 1998.
8. G. de Vahl Davis, Natural Convection of Air in a Square Cavity: A Bench Mark Numerical Solution, *Int. J. Numer. Meth. Fluids*, vol. 3, pp. 249–264, 1983.

9. G. Barakos, E. Mitsoulis, and D. Assimacopoulos, Natural Convection Flow in a Square Cavity Revisited: Laminar and Turbulent Models with Wall Functions, *Int. J. Numer. Meth. Fluids*, vol. 18, pp. 695–719, 1994.
10. P. Wang and R. D. Ferraro, Parallel Computation for Natural Convection, *Concurrency: Practice and Experience*, vol. 9, pp. 975–987, 1997.
11. S. Syrjälä, Higher Order Penalty-Galerkin Finite Element Approach to Laminar Natural Convection in a Square Cavity, *Numer. Heat Transfer A*, vol. 29, pp. 197–210, 1996.
12. B. Bermúdez and A. Nicolás, An Operator Splitting Numerical Scheme for Thermal/Isothermal Incompressible Viscous Flows, *Int. J. Numer. Meth. Fluids*, vol. 29, pp. 397–410, 1999.
13. C. Nonino and G. Comini, An Equal-Order Velocity-Pressure Algorithm for Incompressible Thermal Flows, Part 1: Formulation; and Part 2: Validation, *Numer. Heat Transfer B*, vol. 32, pp. 1–15 and pp. 17–35, 1997.
14. M. Hortmann, M. Perić, and G. Scheuerer, Finite Volume Multigrid Prediction of Laminar Natural Convection: Bench-mark Solutions, *Int. J. Numer. Meth. Fluids*, vol. 11, pp. 189–207, 1990.
15. M. Char and Y. Hsu, Computation of Buoyancy-Driven Flow in an Eccentric Centrifugal Annulus with a Non-orthogonal Collocated Finite Volume Algorithm, *Int. J. Numer. Meth. Fluids*, vol. 26, pp. 323–343, 1998.
16. G. Croce, G. Comini, and W. Shyy, Incompressible Flow and Heat Transfer Computations Using a Continuous Pressure Equation and Nonstaggered Grids, *Numer. Heat Transfer B*, vol. 38, pp. 291–301, 2000.
17. Y. G. Lai, An Unstructured Grid Method for a Pressure-Based Flow and Heat Transfer Solver, *Numer. Heat Transfer B*, vol. 32, pp. 267–281, 1997.
18. P. Le Quere, Accurate Solution to the Square Thermally Driven Cavity at High Rayleigh Number, *Comput. Fluids*, vol. 20, pp. 29–41, 1991.
19. S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, McGraw-Hill, New York, 1980.
20. J. P. Van Doormaal and G. D. Raithby, Enhancement of the SIMPLE Method for Predicting Incompressible Fluid Flows, *Numer. Heat Transfer*, vol. 7, pp. 147–163, 1984.
21. J. A. Meijerink and H. A. Van Der Vorst, Guidelines for the Usage of Incomplete Decompositions in Solving Sets of Linear Equations as They Occur in Practical Problems, *J. Comput. Phys.*, vol. 44, pp. 134–155, 1981.