INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 8, 283–303 (1988)

PRECONDITIONED CONJUGATE GRADIENTS FOR SOLVING THE TRANSIENT BOUSSINESQ EQUATIONS IN THREE-DIMENSIONAL GEOMETRIES

S. DUPONT AND J. M. MARCHAL

Unite de Méchanique Appliquée, Université Catholique de Louvain, Bâtiment Simon Stevin, Place du Levant 2, B-1348 Louvain-la-Neuve,

SUMMARY

In this paper we present a new version of the 'modified finite element method' (MFEM) presented by Gresho, Chan, Lee and Upson.[1] The main modification of the original algorithm is the introduction of a cost-effective and memory-saving iterative solver for the discretized Poisson equation for the pressure. The vectorization of the preconditioner has been especially considered. For low Prandtl number problems we also split the advection–diffusion operator of the energy equation into explicit and implicit parts. In that sense the present approach is related to the recent implicitization of the diffusive terms introduced by Gresho and Chan[2] and by Gresho.[3] The algorithm is applied to the study of buoyancy-driven flow oscillations occuring in a horizontal crucible of molten metal under the action of a horizontal temperature gradient.

KEY WORDS Finite elements Transient flow Three-dimensional flow Natural convection Incomplete Choleski conjugate gradients Iterative solver Vectorization Crystal growth Gallium arsenide

INTRODUCTION

In the present paper we present a modified form of the algorithm developed by Gresho *et al.*[1] for solving the time-dependent incompressible Boussinesq equations. The original algorithm, labelled a 'modified finite element method' (MFEM) by their authors, has been changed in two ways: firstly to allow for a pressure update at each time step without input/output (I/O) cost (on CRAY-1), and secondly to increase the stability-controlled time step of the energy equation in the case of low Prandtl number fluids. A semi-implicit method has recently been introduced by Gresho and Chan[2] and Gresho.[3] The present scheme differs from these two papers in the following points: (i) we have introduced the implicitization of the diffusive terms for the energy equation only, and our scheme is fully implicit for the heat diffusion; (ii) the preconditioner of the conjugate gradient solver generalizes an incomplete Choleski decomposition introduced by Van der Vorst[4] for its use with finite elements.

We have used most of the modifications to the conventional Galerkin finite element method (GFEM) introduced in Reference 1, i.e., mass lumping, one-point quadrature and explicit time integration with a balancing tensor diffusivity (BTD). An iterative solver, based on the incomplete Choleski conjugate gradient (ICCG) algorithm, has been implemented for solving the discretized consistent Poisson equation for the pressure at each time step without any disk storage on large three-dimensional problems (typically 10000 nodes).

0271–2091/88/030283–21$10.50 *Received 23 May 1986*
Revised 30 January 1987

In the case of low Prandtl number fluids (order of 10^{-2}) the forward (explicit) Euler scheme is cost-ineffective since the stability-limited time step tends to zero together with the Prandtl number; moreover, the BTD (efficient in advection-dominated problems) is then helpless. The advection–diffusion (AD) operator of the energy equation has been split into an explicit part for advection and an implicit part for diffusion. This time-marching scheme requires the solution of a linear system (constant in time) at each time step; a direct solver with prefactorization of the matrix, storage on disk and forward reduction/back substitution would make it unaffordable (because of the I/O cost) on the CRAY-1. The ICCG solver has once more revealed itself as a powerful tool which allows for an in-core algorithm together with reasonable CPU times. Our iterative solver has been implemented on the basis of a vectorizable scheme developed by Van der Vorst[4] for a finite-difference five-point stencil.

We apply the method to the calculation of the transient three-dimensional flow which occurs in a horizontal crucible of molten metal under the action of a horizontal temperature gradient. Such a configuration is typical of the Bridgman crystal growth process, where it is known that thermal oscillations appear beyond a critical value of the temperature gradient. In the present paper we would like to confirm the value of the critical temperature gradient found for gallium arsenide with a steady state code[5] and to investigate the nature of the three-dimensional flow beyond this critical value.

SPATIAL DISCRETIZATION AND MODIFIED GALERKIN FINITE ELEMENT METHOD

The non-dimensional form of the Navier–Stokes equations with the Boussinesq approximation taken into account is given by

$$\begin{aligned} \partial \mathbf{v}/\partial t + \mathbf{v}\cdot\nabla\mathbf{v} &= -\nabla p + \Delta\mathbf{v} - Gr\,T\mathbf{e},\\ \partial T/\partial t + \mathbf{v}\cdot\nabla T &= Pr^{-1}\,\Delta T, \\ \nabla\cdot\mathbf{v} &= 0. \end{aligned} \tag{1}$$

The meaning of the symbols is given in Table I. The non-dimensional form of the equations has been obtained with the use of a characteristic dimension L and a characteristic temperature deviation δT from a reference temperature T_0, while reference values for velocity, time and pressure are given by the groups ν/L, L^2/ν and $\rho(\nu/L)^2$ respectively. We note that, for a given geometry, the solutions of (1) will depend upon the Grashof and the Prandtl numbers (defined in Table I). In our applications Pr is low and typically of the order of 0.07, while we wish to obtain solutions for high values of Gr of the order of 10^6.

The spatial discretization is based on the Galerkin finite element method, where the velocity, temperature and pressure are interpolated as follows;

$$\begin{aligned} \mathbf{v}^h(\mathbf{x},t) &= \sum_{i=1}^{N} \mathbf{V}^i(t)\phi_i(\mathbf{x}),\\ T^h(\mathbf{x},t) &= \sum_{i=1}^{N} T^i(t)\phi_i(\mathbf{x}), \\ p^h(\mathbf{x},t) &= \sum_{i=1}^{M} P^i(t)\pi_i(\mathbf{x}). \end{aligned} \tag{2}$$

The finite element mesh has N nodes and M elements. The ϕ_i are the C^0-trilinear shape functions defined on the isoparametric brick-like elements, and the π_i are discontinuous piecewise-constant shape functions.

Table I.

$\mathbf{v}$	velocity vector
p	pressure
T	temperature
t	time
$\mathbf{e}$	unit vector oriented along the gravity field
Gr	Grashof number, defined as
$\alpha(T_1 - T_0)gL^3/\nu^2$, where	
α	coefficient of volumetric expansion
g	acceleration due to gravity
$T_1 - T_0 = \delta T$	typical temperature deviation
L	characteristic length
ν	kinematic viscosity
Pr	Prandtl number, given by
$\rho\nu c_p/k$, where	
c_p	heat capacity per unit of mass
k	thermal conductivity
ρ	specific mass

Applying the GFEM to the set of equations (1), and using the same notation as Gresho *et al.*,[1] we obtain the following set of ordinary differential equations:

$$\begin{aligned} \mathbf{M}\dot{\mathbf{V}} + \mathbf{K}(\mathbf{V})\mathbf{V} + \mathbf{C}\mathbf{P} &= \mathbf{F}, \\ \mathbf{M}_s\dot{\mathbf{T}} + \mathbf{K}_s(\mathbf{V})\mathbf{T} &= \mathbf{F}_s, \\ \mathbf{C}^T\mathbf{V} &= \mathbf{0}, \end{aligned} \tag{3}$$

with the initial conditions

$$\begin{aligned} &\mathbf{V}(0) = \mathbf{V}_0, \quad \text{where} \quad \mathbf{C}^T\mathbf{V}_0 = \mathbf{0}, \\ &\mathbf{T}(0) = \mathbf{T}_0. \end{aligned} \tag{4}$$

In (3) $\mathbf{V}$, $\mathbf{T}$ and $\mathbf{P}$ represent respectively the vectors of nodal velocities, temperatures and pressures; $\mathbf{F}$ is the vector of generalized forces taking the buoyancy terms into account and $\mathbf{F}_s$ is the vector of generalized fluxes. The subscript 's' refers to the energy equation. $\mathbf{M}$ and $\mathbf{M}_s$ are the mass matrices, $\mathbf{K}(\mathbf{K}_s)$ is the sum of the advection matrix $\mathbf{N}(\mathbf{N}_s)$ and the diffusion matrix $\mathbf{K}^*(\mathbf{K}_s^*)$, while $\mathbf{C}$ is the gradient matrix whose transpose $\mathbf{C}^T$ denotes the divergence matrix.

Applying the operator $\mathbf{C}^T\mathbf{M}^{-1}$ to the first equation (3), it is easy to obtain, in view of the third equation (3), the discretized consistent Poisson equation for the pressure; i.e.,

$$(\mathbf{C}^T\mathbf{M}^{-1}\mathbf{C})\mathbf{P} = \mathbf{A}\mathbf{P} = \mathbf{C}^T\mathbf{M}^{-1}[\mathbf{F} - \mathbf{K}(\mathbf{V})\mathbf{V}]. \tag{5}$$

We note that the symmetric matrix $\mathbf{A}$ defined in (5) is positive-definite provided one has removed the possible indeterminacy of the pressure and spurious pressure modes.

Several approximations have been brought to the GFEM towards an efficient algorithm:

1. The mass matrices $\mathbf{M}$ and $\mathbf{M}_s$ have been diagonalized by mass lumping. A preprocessor evaluates the coefficients of the full mass matrices by means of a 2^3-point quadrature, and the diagonalization is obtained by means of the true row sum technique.

2. The gradient matrix is evaluated by means of a one-point quadrature and stored element by element.
3. The matrices **K**(**V**) and $\mathbf{K}_s(\mathbf{V})$ as well as the vectors **F** and $\mathbf{F}_s$ are evaluated by means of a one-point quadrature rule. However, the **K**(**V**) and $\mathbf{K}_s(\mathbf{V})$ matrices are only used for forming the products **K**(**V**)**V** and $\mathbf{K}_s(\mathbf{V})\mathbf{T}$. With the one-point quadrature rule it is easy to calculate these products through the evaluation of **V** at the centre of the element and of the coefficients of **C**. Such a method, used in Reference 1, is cost-efficient and allows for a highly vectorized code.

Gresho *et al.*[1] note that the one-point quadrature rule for calculating **K** and $\mathbf{K}_s$ leads to their singular behaviour with respect to '$2\Delta x$' waves under purely natural boundary conditions; the singular behaviour may also occur when natural boundary conditions are imposed on a part of the boundary. A correction (called the hour-glass matrix) has been suggested in Reference 1 for avoiding that difficulty. The present paper is concerned with the buoyancy-driven flow at a low Prandtl number in a horizontal furnace; a part of the boundary is subject to natural boundary conditions. We have compared the one-point and the 2^3-point quadrature rules for calculating the matrices **K** and $\mathbf{K}_s$. We found that the results obtained with both rules were essentially the same and that small wiggles, whenever present, were not caused by the one-point integration. We have concluded that, for our type of problem characterized by a low value of *Pr*, the hour-glass correction of Reference 1 was unnecessary, the reason being that the flow is not advection-dominated and the '$2\Delta x$' waves are not excited.

TIME INTEGRATION

Gresho *et al.*[1] integrate the first two equations (3) by means of a forward Euler scheme; a correction term called the balancing tensor diffusivity (BTD) is added to the **K** matrices for improving the stability of the scheme. More precisely, let $\mathbf{V}_n$, $\mathbf{T}_n$ denote the nodal velocities and temperatures at time t_n, and let Δt be the time increment from t_n to t_{n+1}. The solution at t_{n+1} is then obtained as follows:

$$\begin{aligned}\mathbf{V}_{n+1} &= \mathbf{V}_n + \Delta t\mathbf{M}^{-1}[\mathbf{F}_n - \mathbf{K}(\mathbf{V}_n)\mathbf{V}_n - \mathbf{C}\mathbf{P}_n],\\ \mathbf{T}_{n+1} &= \mathbf{T}_n + \Delta t\mathbf{M}_s^{-1}[\mathbf{F}_{s_n} - \mathbf{K}_s(\mathbf{V}_n)\mathbf{T}_n],\end{aligned} \tag{6}$$

where $\mathbf{P}_n$ has been calculated from

$$\mathbf{A}\mathbf{P}_n = \mathbf{C}^{\mathrm{T}}\mathbf{M}^{-1}[\mathbf{F}_n - \mathbf{K}(\mathbf{V}_n)\mathbf{V}_n]. \tag{7}$$

A balancing tensor

$$\tau_{ij} = v_i v_j \Delta t/2 \tag{8}$$

is added to the diffusive coefficients in (6).

It is important at this stage to draw attention to the stability of the forward Euler scheme and the associated time step. Let us consider the one-dimensional advection–diffusion equation with constant coefficients,

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} = \kappa\frac{\partial^2 T}{\partial x^2} \tag{9}$$

The combined use of the Galerkin method, linear elements on a uniform mesh and the forward Euler scheme leads us to a maximum time step given by

$$\Delta t \leqslant \Delta x^2/2k \tag{10}$$

and

$$\Delta t \leqslant 2k/u^2. \tag{11}$$

When the BTD is used together with the forward Euler scheme, i.e., when k is replaced by $k+u^2\,\Delta t/2$ in (9), one finds an upper bound for Δt given by

$$\Delta t \leqslant \frac{\Delta x^2}{k[1+(1+u^2\,\Delta x^2/k^2)^{1/2}]}. \tag{12}$$

While the BTD is especially efficient for advection-dominated flows, it is much less useful for low *Pr* flows, since (12) becomes equivalent to (10) when *Pr* is small, and Δt tends to vanish when k becomes large.

It is clear that the advantages of the fully explicit scheme are lost for low Prandtl number flows because the stability condition (10) leads to a time step which is much too small. The same situation has been encountered in Reference 6, where it has been found economical to split the advection–diffusion operator into an implicit part for the diffusion and an explicit part for the advection. An important problem is then to solve the implicit equation for the diffusion equation in an efficient manner.

Since the matrix of the system is constant in time, one approach is to factorize the matrix in a preprocessor code and to perform a forward reduction/back substitution at each time step. This technique is not costly in CPU time and very cost-efficient when the lower–upper matrices can be stored in core. However, the large problems solved on vector computers would require a storage of the factorized matrix on disk, and the I/O time would then exceed the CPU time by one or two orders of magnitude. Noting that the time integration of the diffusion operator leads to a self-adjoint system, we found that the incomplete Choleski conjugate gradient technique would be a good candidate since the convergence is then guaranteed. The performance of the ICCG solver will be considered later.

We may now review the organization of the algorithm which we have eventually adopted. Starting with a velocity field $\mathbf{V}_n$ satisfying

$$\mathbf{C}^{\mathrm{T}}\mathbf{V}_n = \mathbf{0}, \tag{13}$$

we evaluate the right-hand sides $\mathbf{F}_n$ and $\mathbf{F}_{\mathrm{s}_n}$ as well as $\mathbf{K}(\mathbf{V}_n)\mathbf{V}_n$ and $\mathbf{N}_{\mathrm{s}}(\mathbf{V}_n)\mathbf{T}_n$ by a loop on the elements. Then we solve the consistent Poisson equation for the pressure,

$$\mathbf{A}\mathbf{P}_n = \mathbf{C}^{\mathrm{T}}\mathbf{M}^{-1}[\mathbf{F}_n - \mathbf{K}(\mathbf{V}_n)\mathbf{V}_n], \tag{14}$$

by means of the ICCG technique with $\mathbf{P}_{n-1}$ as a first guess. We compute $\mathbf{C}\,\mathbf{P}_n$ and then $\mathbf{V}_{n+1}$ as follows:

$$\mathbf{V}_{n+1} = \mathbf{V}_n + \Delta t\,\mathbf{M}^{-1}[\mathbf{F}_n - \mathbf{K}(\mathbf{V}_n)\mathbf{V}_n - \mathbf{C}\mathbf{P}_n]. \tag{15}$$

Next we use an explicit operator similar to (15) for updating the temperature field on the basis of the advective terms; i.e.,

$$\mathbf{T}^* = \mathbf{T}_n + \Delta t\,\mathbf{M}_{\mathrm{s}}^{-1}[\mathbf{F}_{\mathrm{s}_n} - \mathbf{N}_{\mathrm{s}}(\mathbf{V}_n)\mathbf{T}_n]. \tag{16}$$

Equations (15) and (16) are calculated in a single loop on the elements. The diffusive contribution to the temperature field is then calculated by means of an implicit method; i.e.,

$$\mathbf{K}_{\mathrm{s}}^*\mathbf{T}_{n+1} = -\mathbf{M}_{\mathrm{s}}(\mathbf{T}_{n+1} - \mathbf{T}^*)/\Delta t \tag{17}$$

or

$$\mathbf{B}\mathbf{T}_{n+1} = (\mathbf{K}_{\mathrm{s}}^* + \mathbf{M}_{\mathrm{s}}/\Delta t)\mathbf{T}_{n+1} = \mathbf{M}_{\mathrm{s}}\mathbf{T}^*/\Delta t. \tag{18}$$

No BTD has been added to the energy equation in the implicit scheme. The matrix **B** is positive-definite since the sum of two positive-definite matrices maintains the same character. We may thus use the ICCG routine for solving (18). In References 2 and 3 all the diffusive terms (the physical diffusivity and the BTD) of the energy and momentum equations have been integrated via a semi-implicit trapezoid rule, the other terms being integrated via a forward Euler scheme. The stability of the scheme is then increased, at the cost of a new diffusion matrix in core (this would be a disadvantage in our problem since the number of variables is limited by memory).

The matrices **A** and **B** appearing in (14) and (18) respectively are evaluated by a preprocessor code, scaled and stored in core together with the associated matrices required by the ICCG algorithm.

The system (13)–(17) guarantees mass conservation in the discrete sense. This is easily seen by calculating the product of (15) on the left by $\mathbf{C}^{\mathrm{T}}$, from which we get, in view of (14),

$$\mathbf{C}^{\mathrm{T}}\mathbf{V}_{n+1} = \mathbf{C}^{\mathrm{T}}\mathbf{V}_{n}. \tag{19}$$

However, (19) holds only if (14) is solved exactly. Using the iterative ICCG technique, (14) is only solved up to an imposed level of accuracy ε, so that (19) is written more correctly as

$$\mathbf{C}^{\mathrm{T}}\mathbf{V}_{n+1} = \mathbf{C}^{\mathrm{T}}\mathbf{V}_{n} + \mathbf{R}_{n+1}, \tag{20}$$

where $\mathbf{R}_{n+1}$ tends to zero when the required accuracy is higher. Since the sum of residues $\mathbf{R}_n$ over a large number of time steps could possibly cause a perturbation which significantly affects the solution, we found it necessary to implement a 'return to the discretized mass consistency' after a given number of time steps n_c depending upon ε. The projection of the calculated velocity field $\tilde{\mathbf{V}}$ on a solenoidal space is performed by the technique introduced by Gresho *et al.*[1] within the framework of the subcycling of the pressure. Defining a vector $\mathbf{\Lambda}$ by means of the equation

$$\mathbf{A}\mathbf{\Lambda} = \mathbf{C}^{\mathrm{T}}\tilde{\mathbf{V}}, \tag{21}$$

we calculate the divergence-free velocity field as

$$\mathbf{V} = \tilde{\mathbf{V}} - \mathbf{M}^{-1}\mathbf{C}\mathbf{\Lambda} \tag{22}$$

The system (21) is again solved by means of the ICCG technique, with $\mathbf{\Lambda} = \mathbf{0}$ as a first guess.

THE INCOMPLETE CHOLESKI CONJUGATE GRADIENT TECHNIQUE

We have seen in the previous section that the ICCG technique is used for solving the systems (14) and (18). Our choice has been guided by the following reasons:

1. For the size of problem that we wanted to solve (10000 nodes, 47000 variables), the systems (14) and (18) do not require with this technique any I/Os on a 9×10^5 words CRAY-1S computer. The cost of the I/Os would be prohibitive with a direct Choleski solver.
2. The convergence of the iterations is guaranteed since **A** and **B** are symmetric positive-definite matrices.
3. The algorithm can be highly vectorized, and the CPU time is not prohibitive as compared with a back substitution as long as the iterations are initiated with a good approximation of the solution; i.e., the solution at the previous time step.

Although it is not the case in the present paper, the technique is advantageous when the **A** or **B** matrices are not constant in time; for example, on a moving mesh or when the thermal diffusivity is temperature-dependent. A factorization of **A** or **B** would therefore not be viable, whereas the additional cost of the conjugate gradients would be limited to re-evaluating the coefficients of the matrices.

To describe our ICCG algorithm, let us introduce a generic linear system

$$\mathscr{A}\mathbf{x} = \mathbf{b}, \tag{23}$$

with $\mathscr{A}$ being an $N \times N$ sparse-symmetric positive-definite matrix. The conjugate gradients are a semi-iterative algorithm in the sense that the solution of (23) will be found, in the absence of rounding errors, in at most N iterations. However, this property is not useful when N is large, since the algorithm will be stopped when a precision ε has been reached. One proves (see, for example, Reference 7) that the number of iterations necessary to make the error less than ε times the initial error is roughly bounded by $(1/2)\sqrt{[\mathscr{K}(\mathrm{A})]}\ln(2/\varepsilon)$, where $\mathscr{K}(\mathscr{A})$ is the spectral condition number of $\mathscr{A}$. In most practical applications, the condition number $\mathscr{K}(\mathscr{A})$ is too high and does not allow for a fast convergence of the algorithm. Moreover, the rounding errors may then become prohibitive and limit the convergence in an unacceptable way. For these reasons the conjugate gradients are almost always used with a preconditioner.

Preconditioning the system (23) means solving

$$\mathscr{M}^{-1}\mathscr{A}\mathbf{x} = \mathscr{M}^{-1}\mathbf{b}, \tag{24}$$

where $\mathscr{M}$ is a symmetric positive-definite matrix; the preconditioning is efficient when the condition number $\mathscr{K}(\mathscr{M}^{-1}\mathscr{A})$ is significantly smaller than $\mathscr{K}(\mathscr{A})$. The product $\mathscr{M}^{-1}\mathscr{A}$ is never formed explicitly, and the preconditioned conjugate gradient method requires solving a system in $\mathscr{M}$ at each iteration. Formally, this system will be written as

$$\mathscr{M}\tilde{\mathbf{x}} = \mathbf{x}', \tag{25}$$

where $\tilde{\mathbf{x}}$ and $\mathbf{x}'$ are some vectors taking place in the algorithm. For reducing $\mathscr{K}(\mathscr{M}^{-1}\mathscr{A})$ the best choice is obviously $\mathscr{M} = \mathscr{A}$. The preconditioned conjugate gradient method then becomes a direct method requiring the factorization of $\mathscr{A}$; this is precisely the operation to avoid. A good preconditioner will present the following qualities:

1. The computational effort for solving (25) is small compared with the direct resolution of (23).
2. $\mathscr{K}(\mathscr{M}^{-1}\mathscr{A})$ is as small as possible, and significantly smaller than $\mathscr{K}(\mathscr{A})$, i.e., $\mathscr{M}$ is close to $\mathscr{A}$.
3. The storage for $\mathscr{M}$ is not excessive.

The class of incomplete Choleski preconditioners is restricted to $\mathscr{M}$ matrices of the form

$$\mathscr{M} = \mathscr{L}\mathscr{L}^{\mathrm{T}}, \tag{26}$$

where $\mathscr{L}$ is lower triangular; solving (25) is then equivalent to solving two triangular systems. In the version we have implemented, the $\mathscr{L}^{\mathrm{T}}$ matrix has the same sparsity as the upper triangular part of $\mathscr{A}$; this algorithm has been labelled ICCG (1, 1) by Meijerink and Van der Vorst.[8] This preconditioner has the advantage of not requiring extra memory for the storage of $\mathscr{L}$, since by an appropriate scaling of $\mathscr{A}$ the coefficients of $\mathscr{L}^{\mathrm{T}}$ are identical to the coefficients of the upper part of $\mathscr{A}$.

On a CRAY-1 the possibility of vectorizing must also be considered. The conjugate gradient method itself is vectorizable if an appropriate storage of $\mathscr{A}$ allows for vectorizing the matrix vector product $\mathscr{A}\mathbf{x}$. For storing the A and B matrices of (14) and (18), we opted for a structure privileging the diagonal direction rather than the rows or the columns. This structure allows for operations on long vectors if the mesh numbering has a good regularity, and reduces the size of the pointer tables. The pattern of non-vanishing diagonals is detected by a preprocessor, which also establishes the system of pointers minimizing the memory requirement. This 'multidiagonal' structure is not limited to brick-like meshes, although optimum vectorization is achieved on highly regular grids. The incomplete Choleski preconditioner unfortunately presents a feature inhibiting the vectorization: solving triangular systems implies first-order recurrences (the first upper diagonal of $\mathscr{L}^{\mathrm{T}}$ is non-vanishing), and the FORTRAN compiler will generate serial code. The preconditioner is then the bottle-neck of the algorithm. This problem has been examined by several authors;[4,9] we opted for an algorithm developed by Van der Vorst[4] for a five-point finite difference stencil. The basic idea of the algorithm is to split $\mathscr{L}^{\mathrm{T}}$ into $\mathscr{L}'^{\mathrm{T}}$ and $\mathscr{L}''^{\mathrm{T}}$, $\mathscr{L}'^{\mathrm{T}}$ being the restriction of $\mathscr{L}^{\mathrm{T}}$ to the main and first upper diagonal, and

$$\mathscr{L}^{\mathrm{T}} = \mathscr{L}'^{\mathrm{T}} + \mathscr{L}''^{\mathrm{T}}. \tag{27}$$

The vector of unknowns is then partitioned into blocks having a dependence via $\mathscr{L}'^{\mathrm{T}}$ but *not* via $\mathscr{L}''^{\mathrm{T}}$. The length of a block is the distance between the first non-vanishing diagonal of $\mathscr{L}''^{\mathrm{T}}$ and the main diagonal. The inverse matrix $\mathscr{L}'^{-\mathrm{T}}$ is then approximated by a truncated series, and the approximation of $\mathscr{L}'^{-\mathrm{T}}$ is stored in core. The first-order recurrence introduced by the first upper diagonal of $\mathscr{L}'^{\mathrm{T}}$ has been avoided, and the parallelism of the algorithm increases from 1 to the size of the block. In the application presented in this paper, the length of the vectors is approximately 50. The quality of the preconditioner is slightly affected by the truncation, and for achieving a given precision the number of conjugate gradient iterations will be superior as compared with the original ICCG (1, 1). However, the computer time per iteration is strongly reduced, and on a CRAY-1 the parallel algorithm performs much better globally than the serial one.

APPLICATION: THREE-DIMENSIONAL OSCILLATORY FLOW IN A BRIDGMAN HORIZONTAL FURNACE

We wish to apply the method described in the third section to the transient periodic flow taking place in a horizontal Bridgman furnace, where a crucible filled with molten metal is subjected to a horizontal temperature gradient. The crucible, shown in Figure 1, has the shape of a half cylinder of length L and radius h with rounded ends. We consider here a simplified parallelepiped geometry, also shown in Figure 1, and we will *a priori* limit ourselves to solutions which are symmetric with respect to the plane of symmetry of the crucible. The flow domain is then restricted to an $L \times h \times h$ box, where $z = 0$ is a plane of symmetry. We choose h as a characteristic length, while the aspect ratio L/h is here equal to 4.

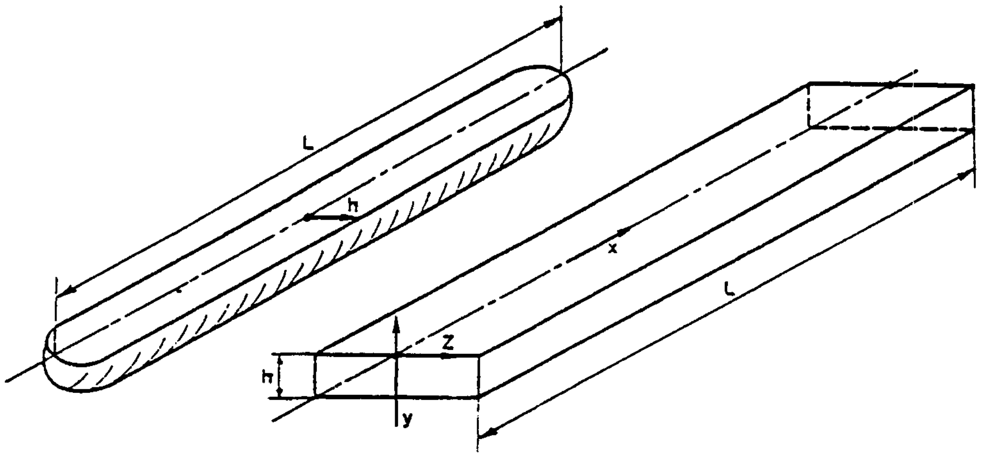

Figure 1. Perspective view of the crucible and idealized geometry

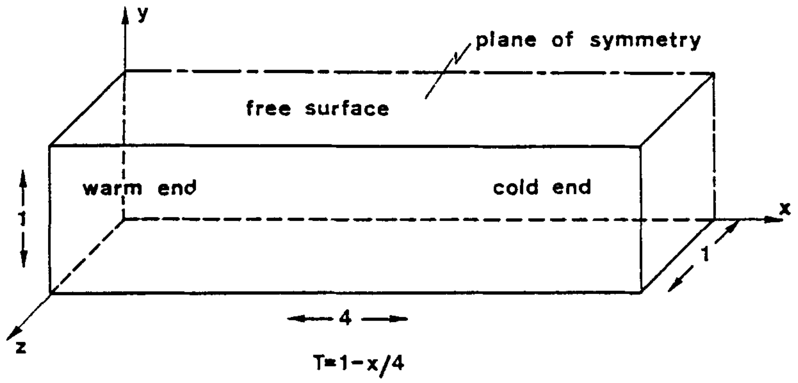

Figure 2. Boundary conditions

The boundary conditions are shown in Figure 2. The fluid does not slip along the walls while the upper plane is a horizontal free surface; the normal velocity and the tangential forces vanish identically on the free surface and on the plane of symmetry. The thermal boundary conditions of the actual process are very complex, since radiation dominates the heat exchange between the furnace and the crucible. Here we will adopt a simpler set of thermal boundary conditions: we assume that the (non-dimensional) temperature is imposed on all the faces with a value which varies linearly between 1 on the plane $x = 0$ and 0 on the plane $x = 4$. The direction of gravity is $(0, -1, 0)$ with the co-ordinate system shown in Figure 1. The actual fluid would be molten gallium arsenide, with $Pr = 0{\cdot}069$.

A two-dimensional representation of the horizontal Bridgman growth has been studied by Crochet *et al.*,[6] with the assumption that the z component (see Figure 1) of the velocity field vanishes while the velocity components and the temperature are z-independent. Using both finite difference and finite element techniques, these authors predicted the onset of periodic oscillations in the crucible of molten metal. The bifurcation from a stationary solution to a periodic flow occurs at a critical value of the Grashof number Gr which was also identified as the limit of convergence of the Newton–Raphson iterations proper to a steady finite element code. With a 4:1 aspect ratio and $Pr = 0{\cdot}069$ the critical value of Gr is 7.1×10^5.[5] The order of magnitude of the periods of oscillation found in Reference 6 agreed with experimental data.

Using the same domain and boundary conditions indicated above, Dupont *et al.*[5] have extended the analysis of Reference 6 to three-dimensional stationary flows. They used a Galerkin finite element method with bilinear shape functions for the velocity components and the temperature and a penalty formulation for the pressure. Newton–Raphson iterations were used on the complete set of equations. They showed that the three-dimensional effects are important and that the convective pattern is different in two and three dimensions. The limit of convergence with steady codes is also lower in three- as compared to two-dimensional flows.

In the present paper we wish to address the important question as to what happens at a Grashof number higher than the limit of convergence of the three-dimensional stationary code. In particular we wish to discover whether the flow bifurcates towards an oscillating solution.

First we have generated transient solutions on a finite element mesh called 3D2 which was used in Reference 5 and which is shown in Figure 3. It contains $40 \times 14 \times 10$ elements, 6765 nodes and 32660 variables with the present formulation. The limit of convergence of the stationary code with this mesh was $Gr = 5 \times 10^5$. For monitoring the progress of the transient behaviour, we define a non-dimensional kinetic energy K as

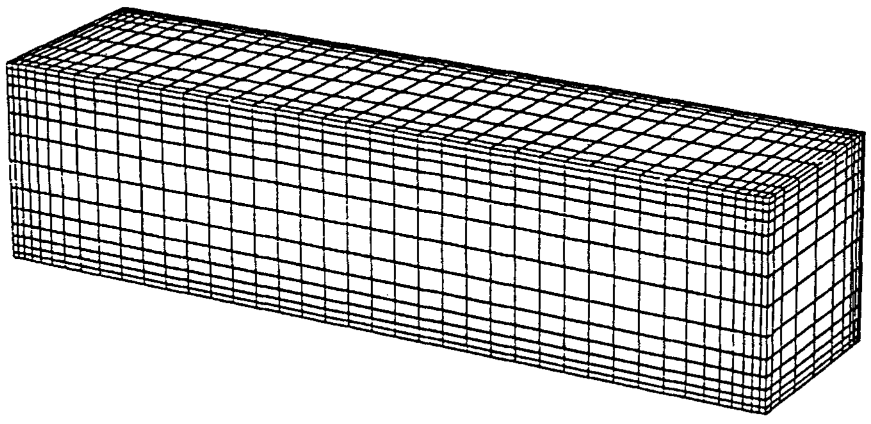

Figure 3. Perspective view of the three-dimensional finite element mesh 3D2

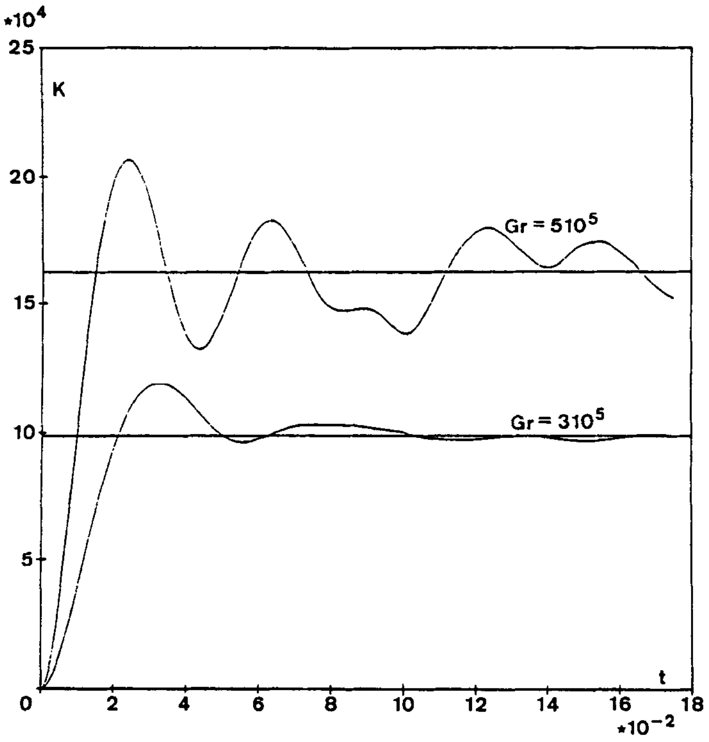

Figure 4. Kinetic energy versus non-dimensional time and levels of kinetic energy of the stationary solutions

$$2K = \int_{\Omega} (u^2 + v^2 + w^2)\,\mathrm{d}\Omega, \tag{28}$$

where Ω is the flow domain and u, v, w denote the velocity components.

Figure 4 shows the evolution of K as a function of time at $Gr = 3 \times 10^5$ (where a steady state was calculated) and at $Gr = 5 \times 10^5$ (where we lost convergence). For this calculation the experimentally determined time step is 10^{-4} and 1750 steps have been calculated. The horizontal lines in Figure 4 denote the levels of the kinetic energy which have been obtained in Reference 5 for the

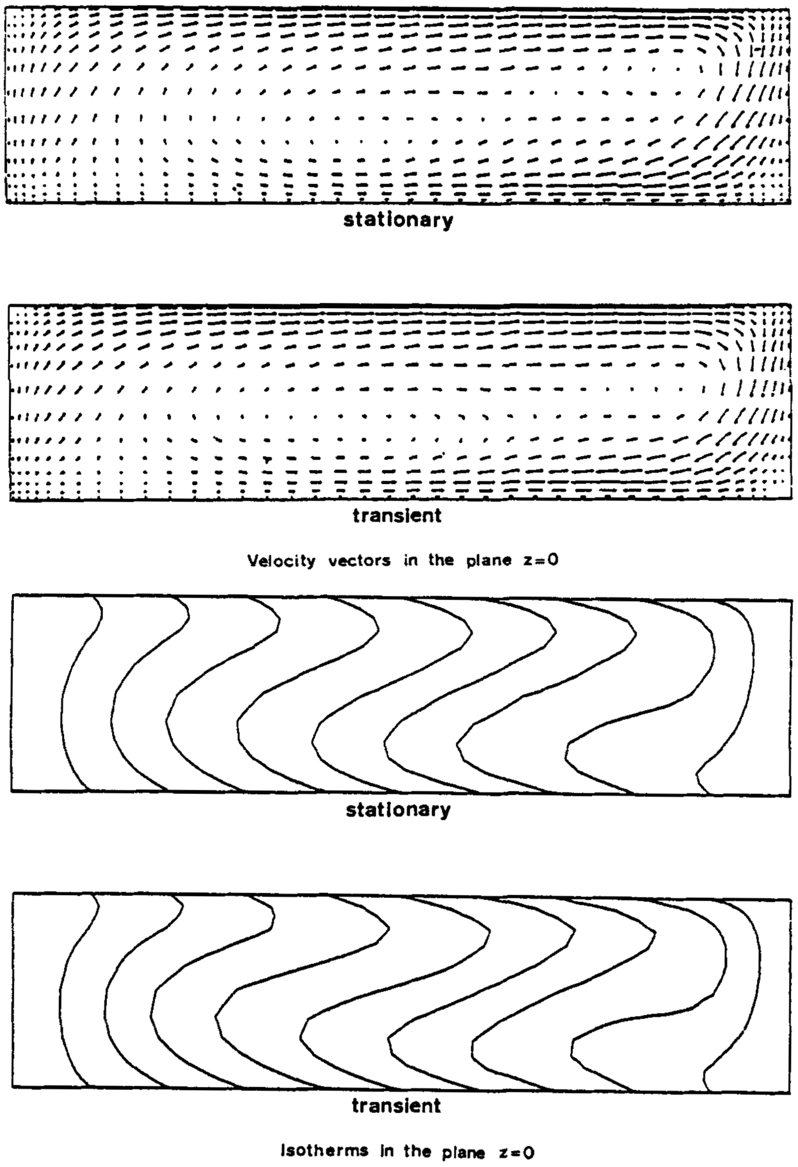

Figure 5. Comparison of the transient and stationary solutions

steady solutions. It is clear in Figure 4 that at $Gr = 3 \times 10^5$ the transient solution evolves towards a stationary flow with a value of K consistent with our earlier findings. At $Gr = 5 \times 10^5$ the damping is less severe but still present. In Figure 5 we compare the velocity vectors in the plane of symmetry obtained at $Gr = 3 \times 10^5$ with the steady state solution and the transient code at $t = 0.175$. In Figure 5 we also show the isotherms in the plane of symmetry. In Figure 6 we show the velocity profiles obtained at the intersection of the plane of symmetry and the free surface. While the transient solution is still oscillatory at the time used for the comparison, we find good agreement between the earlier stationary results and the transient ones obtained with our present code.

We have then studied the nature of the flow at the higher value of $Gr = 7 \times 10^5$, where we could

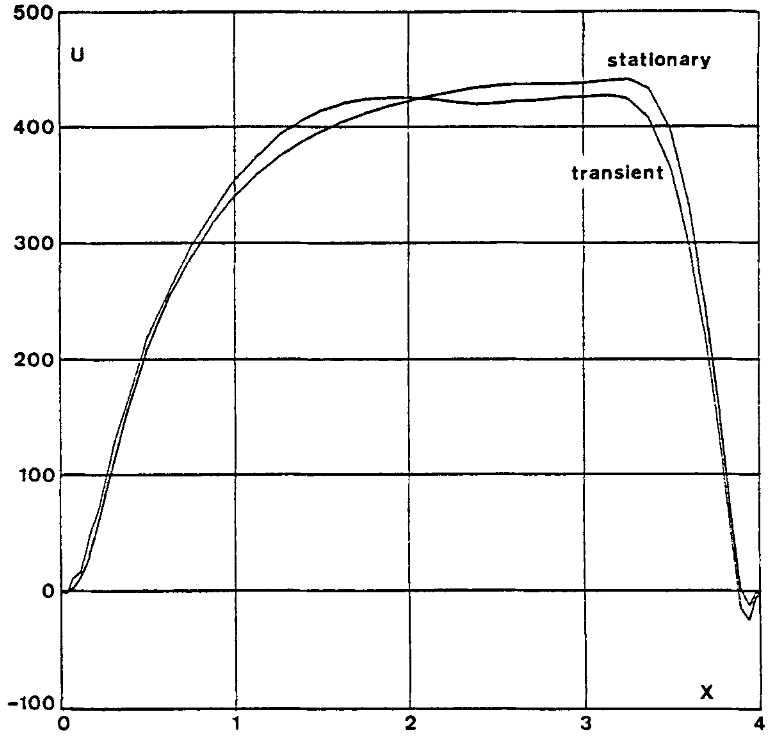

Figure 6. Velocity along the line ($z = 0, y = 1$). Stationary and transient solutions

not find a stationary solution. For that purpose we have used the mesh 3D2 and a refined mesh 3D3, shown in Figure 7, that contains $49 \times 14 \times 12$ elements, 9750 nodes and 47232 variables. The size of the smallest element is $0{\cdot}03 \times 0{\cdot}028 \times 0{\cdot}0234$. The results shown have been obtained on the refined mesh 3D3; we will also discuss the agreement between the results on different meshes. By means of numerical experimentation we found that the present explicit–implicit algorithm is stable on 3D3 as long as $\Delta t \leqslant 1{\cdot}5 \times 10^{-4}$. By trial and error we have also found that the original fully explicit algorithm would require a time step $\Delta t \leqslant 2{\cdot}5 \times 10^{-5}$, and thus our modification of the time-marching technique allowed us for the present problem to multiply the time step by a factor of 6.

On 3D3 the computation has been pursued over 5600 time steps with $\Delta t = 10^{-4}$; this time step has been chosen with the purpose of achieving a better accuracy over a large number of steps. In a real experiment with gallium arsenide and a crucible depth of 2·5 cm, the ratio between real and non-dimensional time would be 1250, and the real time covered by the simulation would be 700 s. As initial conditions we selected a vanishing velocity field and a non-dimensional temperature field given by $T = 1 - x/4$.

On both meshes at $Gr = 7 \times 10^5$ the solutions do not tend to a stationary flow. The graph of the kinetic energy as a function of time shown in Figure 8 (3D3) shows large and undampened oscillations. While the computation has been carried on over a large time interval covering many oscillations of the kinetic energy, we have not been able to identify a periodic behaviour of the kinetic energy. This result disagrees with the two-dimensional flow observations, where a true periodic behaviour sets in after a few oscillations of the kinetic energy.[6] In order to verify the accuracy of our three-dimensional code for low Prandtl number flows, we have reproduced some two-dimensional situations studied in Reference 6 and have found identical results (i.e., they

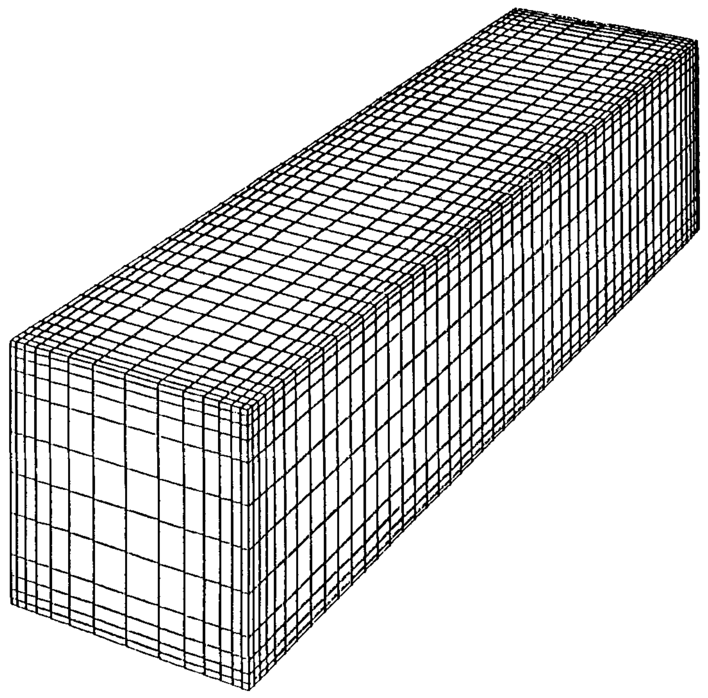
Figure 7. Perspective view of the three-dimensional finite element mesh 3D3

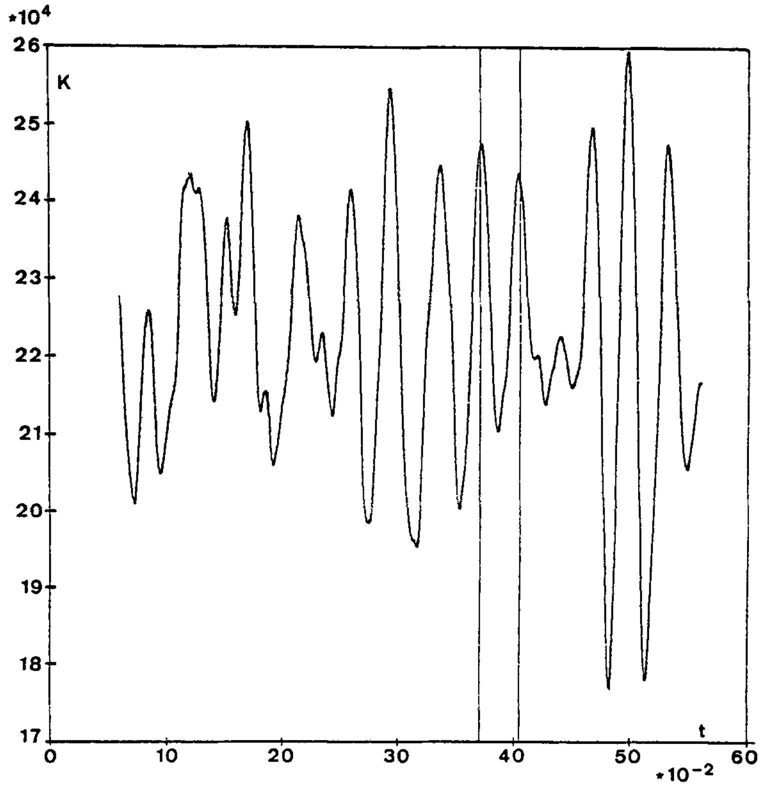

Figure 8. Kinetic energy versus non-dimensional time; $Gr = 7 \times 10^5$

exhibit the same periodicity). The lack of true periodicity of the three-dimensional motion has been observed on both meshes. The oscillatory motions were very similar, and the flows on the different meshes remained close over two or three oscillations of the kinetic energy. When pursued beyond two or three oscillations of the kinetic energy, the two solutions differ somewhat in view of some small differences in the main frequencies. The typical flow pattern will be described in 3D3, and the spectra obtained with the two meshes will then be compared.

The three-dimensional convection pattern is much more complex than the two-dimensional one. To illustrate this we have chosen a time interval $0.370 \leqslant t \leqslant 0{\cdot}405$ identified in Figure 8 by two vertical reference markers. An enlarged view of the kinetic energy over this interval is given in Figure 9; the three-dimensional solution will now be described at eight discrete times t_1 to t_8 separated from each other by 50 equal time steps. Figure 10 shows the velocity vectors in the plane of symmetry. One observes the motion of the eddy located in the upper right corner at time t_1, at mid-height at t_5 and in the upper right corner again at t_8. One should not conclude that the interval $[t_1, t_8]$ is a period of the flow, because the convection pattern does not show the same periodicity in other regions of the flow domain. Figure 11 shows the velocity field on the free surface. A small eddy appears and then disappears near the plane of symmetry. The convection pattern is even more complicated inside the domain: Figure 12 shows the projection of the velocity vectors on the horizontal plane at mid-height. While several convection cells successively appear and disappear, one cannot detect the periodic character of the flow.

Figure 13 and 14 show the isotherms in the plane of symmetry and in the horizontal plane at mid-height respectively. Figure 14 shows that the temperature field depends strongly upon z and

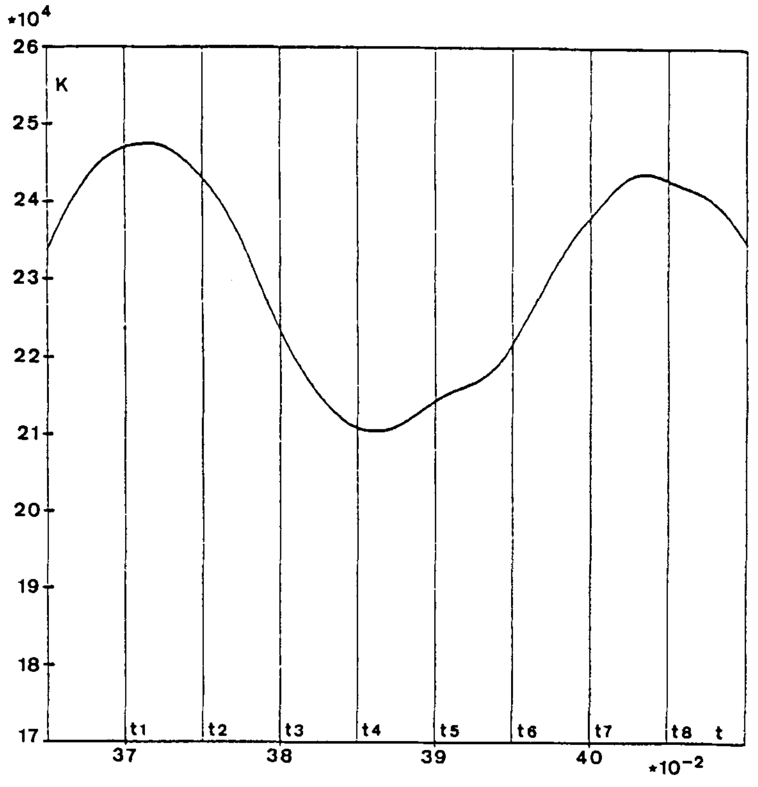

Figure 9. Detail of Figure 8

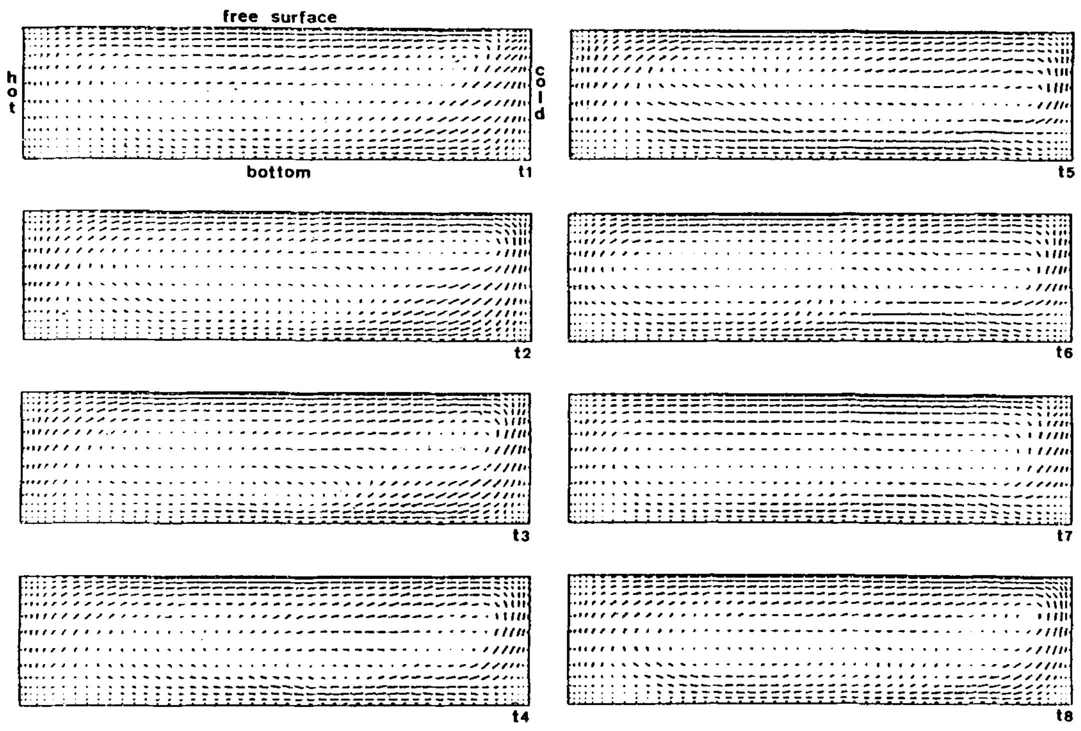

Figure 10. Velocity vectors in the plane of symmetry

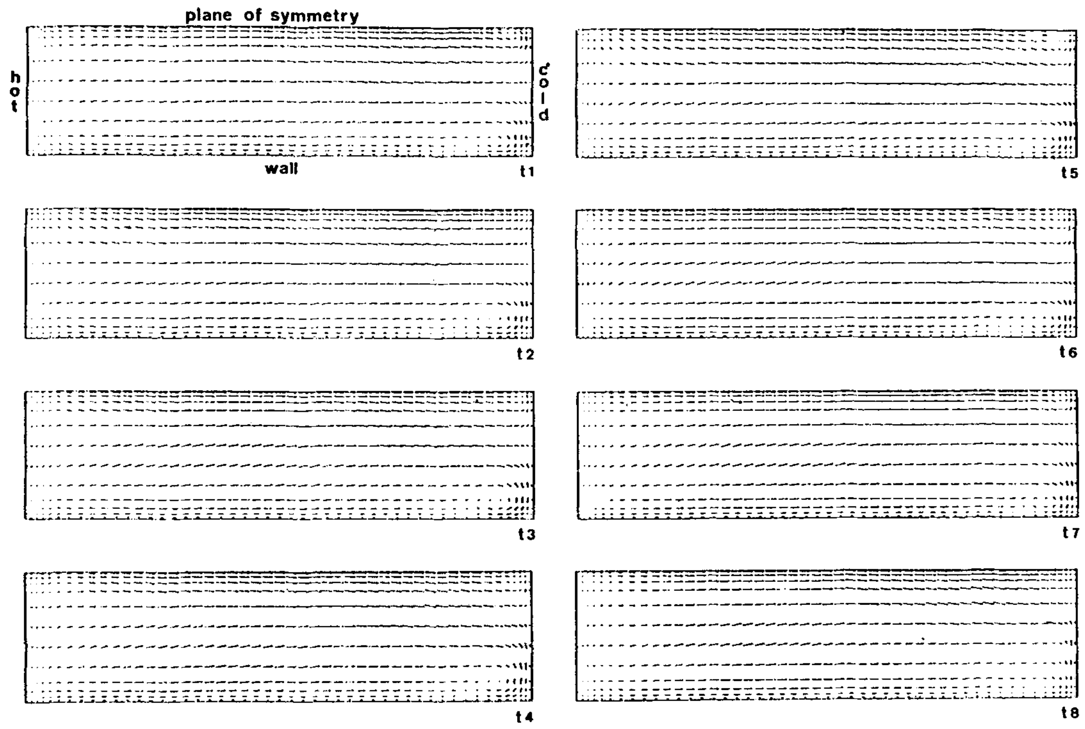

Figure 11. Velocity vectors on the free surface

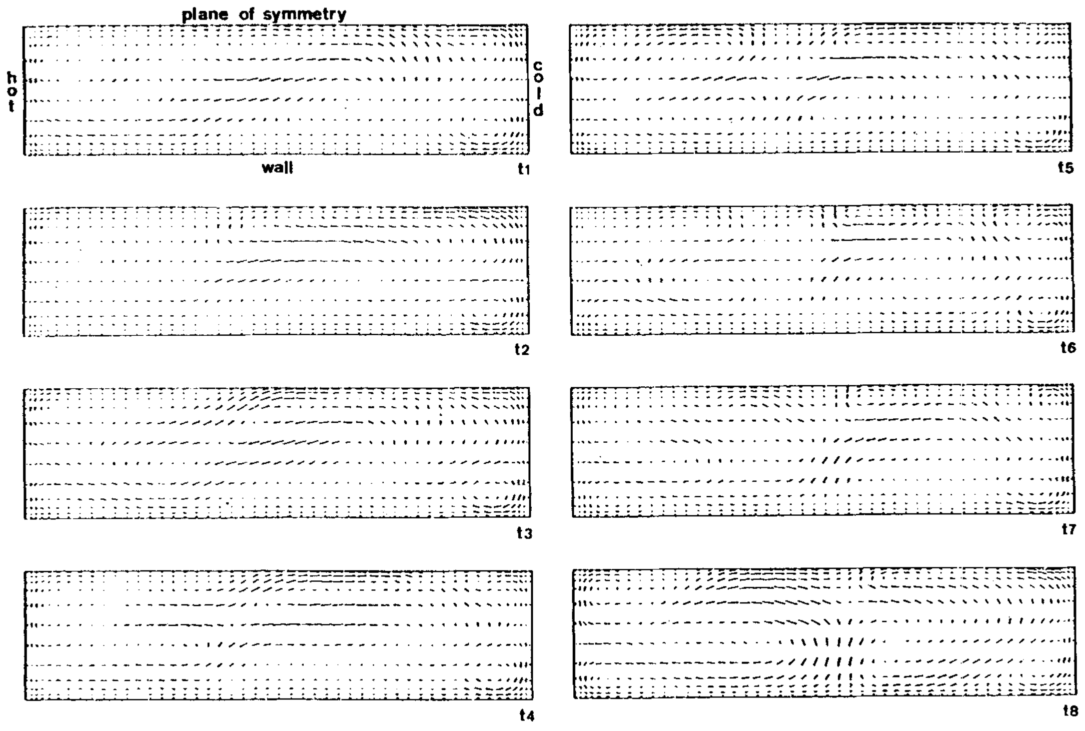

Figure 12. Projection of the velocity vectors on the $y = 0{\cdot}5$ horizontal plane

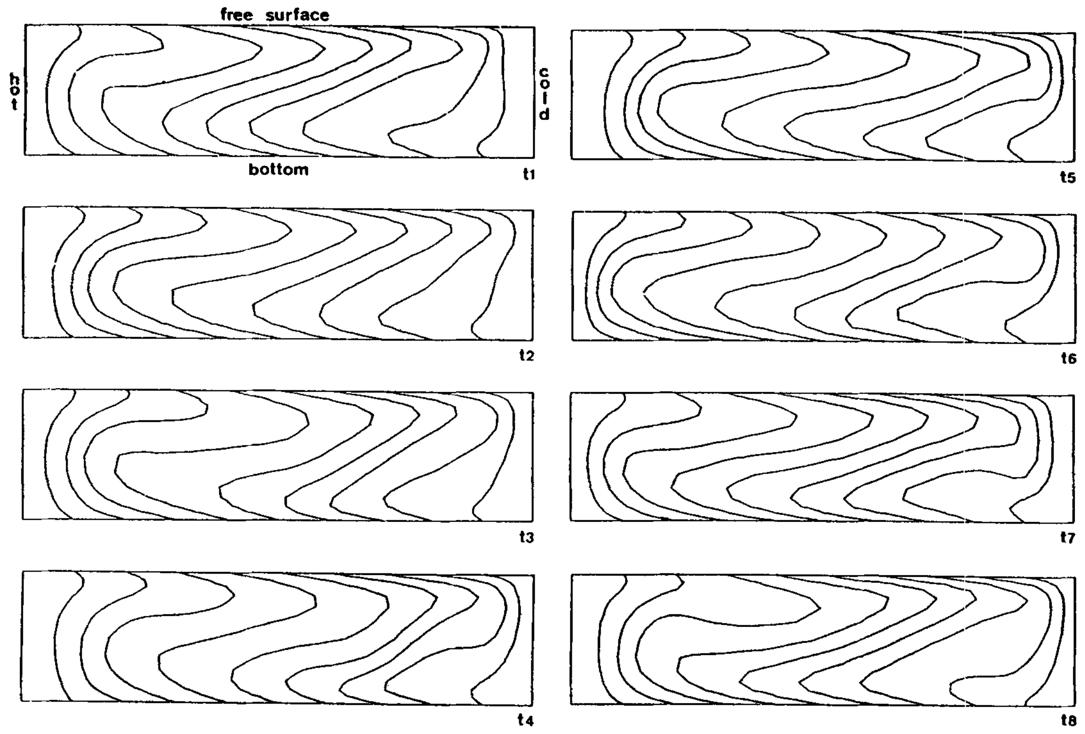

Figure 13. Isotherms in the plane of symmetry

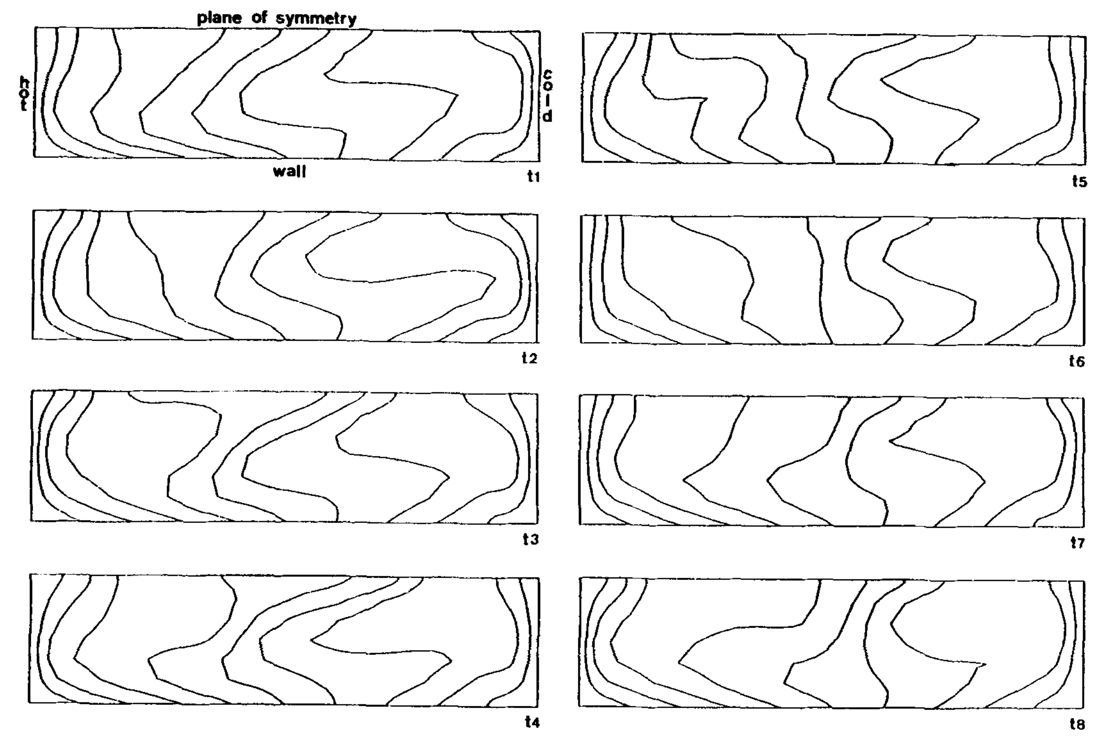

Figure 14. Isotherms in the $y = 0{\cdot}5$ horizontal plane

that the amplitude of the oscillations is important in the core of the domain. Quantitative information about the velocity field is given in Figures 15 and 16, where we plot the (non-dimensional) velocity u versus x along the intersection of the plane of symmetry and of the free surface.

With a view to detecting the periodicity of the flow on 3D2 and 3D3, we have performed a Fourier analysis of the kinetic energy. Figure 17(a) shows the Fourier transform of the 3D2 signal versus the non-dimensional frequency f_n. One detects a dominating frequency $f_1 = 167{\cdot}6$, which in the real experiment would correspond to a period of 46.9 Figure 17(b) shows the Fourier transform of the 3D3 signal. Here one finds two dominating frequencies, $f_1 = 150{\cdot}8$ and $f_2 = 188{\cdot}5$, which in the real experiment would correspond to periods of 52.1 and 41.7.

Finally we wish to give some technical details about the performance of the code with respect to the run at $Gr = 7 \times 10^5$. The relative convergence criteria for the ICCG algorithm were $\varepsilon = 5 \times 10^{-5}$ for the '**A** system' and $\varepsilon = 10^{-4}$ for the '**B** system' introduced in the third section. The velocity field has been projected on a solenoidal subspace every 20 time steps. In solving the consistent Poisson equation we opted for these conservative values of ε and n_c because we wanted to avoid an artificial excitation of oscillating modes.

On a CRAY-1S computer the CPU time was 4 s per time step, without any I/O cost. This CPU time was composed as follows:

1. 1·0 s for the computation of the right-hand sides, the multiplication by $\mathbf{C}^T$, the time-marching algorithm, etc.
2. 1·75 s for solving the consistent Poisson equation.

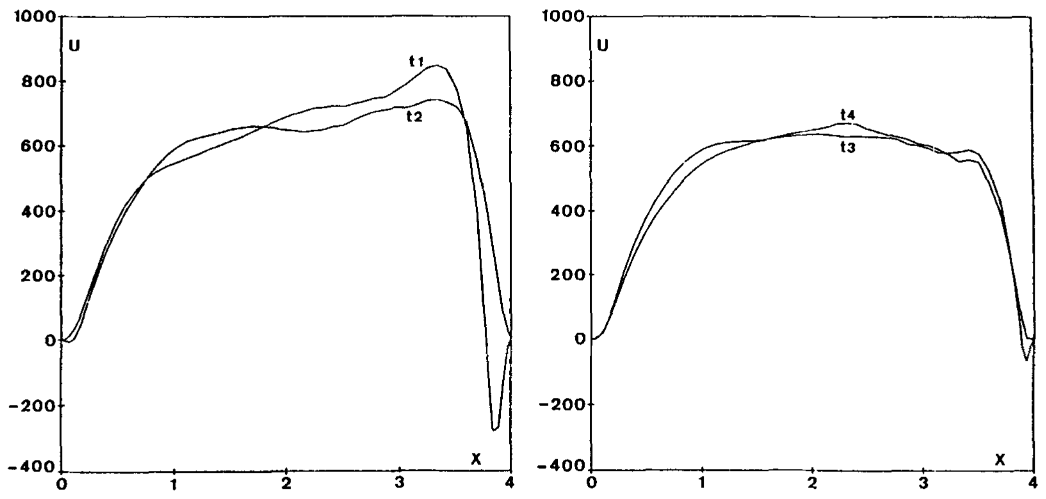

Figure 15. Velocity along the line ($z = 0, y = 1$)

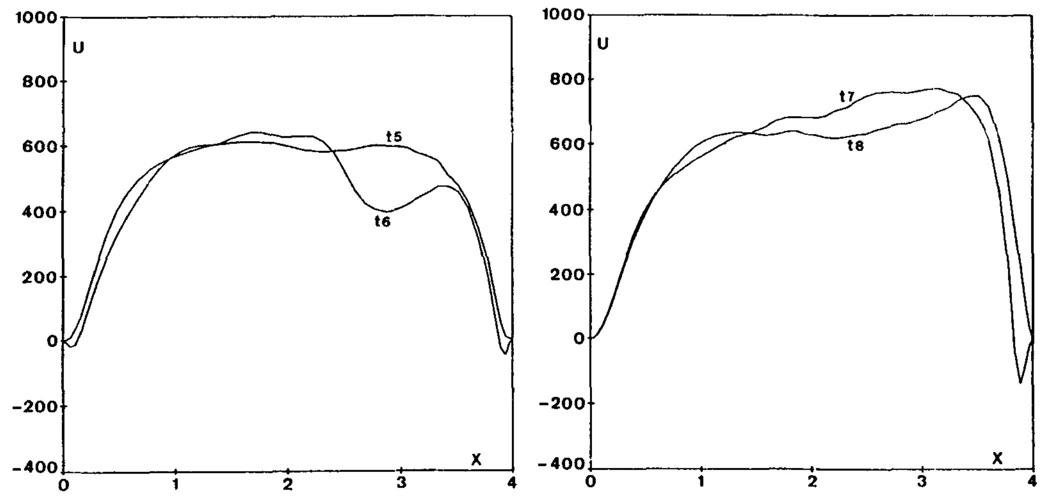

Figure 16. Velocity along the line ($z = 0, y = 1$)

3. 1·0 s (i.e., 20 s every 20 time steps) for the projection of the velocity field.
4. 0·26 s for solving the implicit system for the temperature.

The performance of the ICCG solver, compared with simple diagonal scaling, is summarized in Table II. Both algorithms have been vectorized. One observes that the truncated ICCG(1, 1) scheme is faster than diagonal scaling when the number of iterations is large (when ε is small or when the initial guess lies far from the solution), while the diagonal scaling is cost-efficient for solving the implicit system of the energy equation.

The code required 870000 words of storage: 125000 words were used for storing **A**, 149000 for **B** and 49000 for auxiliary tables of the ICCG algorithm; i.e., a total of 323000 words for the **A** and **B** systems. This number is far less than the 3.5 million words that a direct Choleski solver would require.

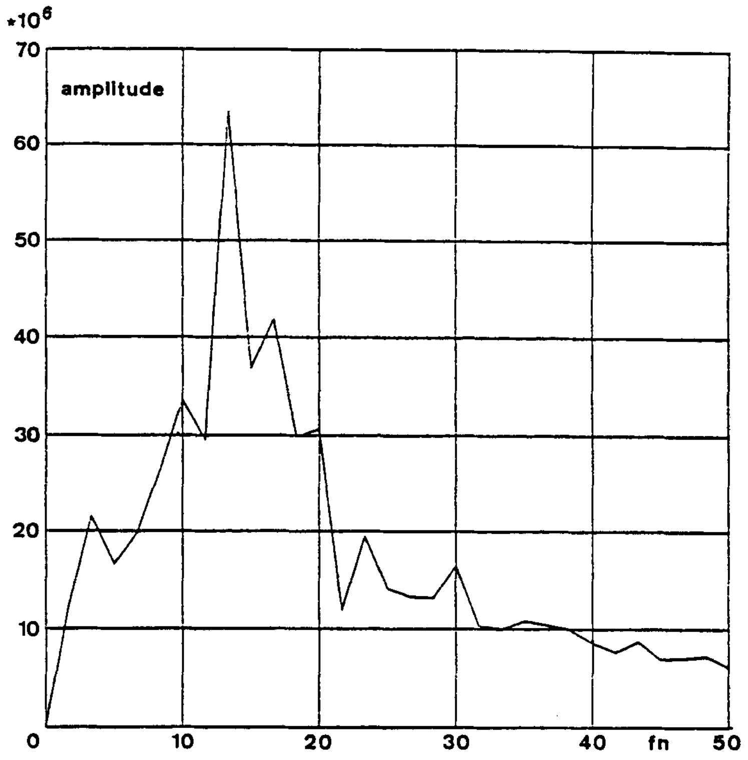

Figure 17(a). Fourier transform of the curve 'kinetic energy versus time' for the 3D2 mesh

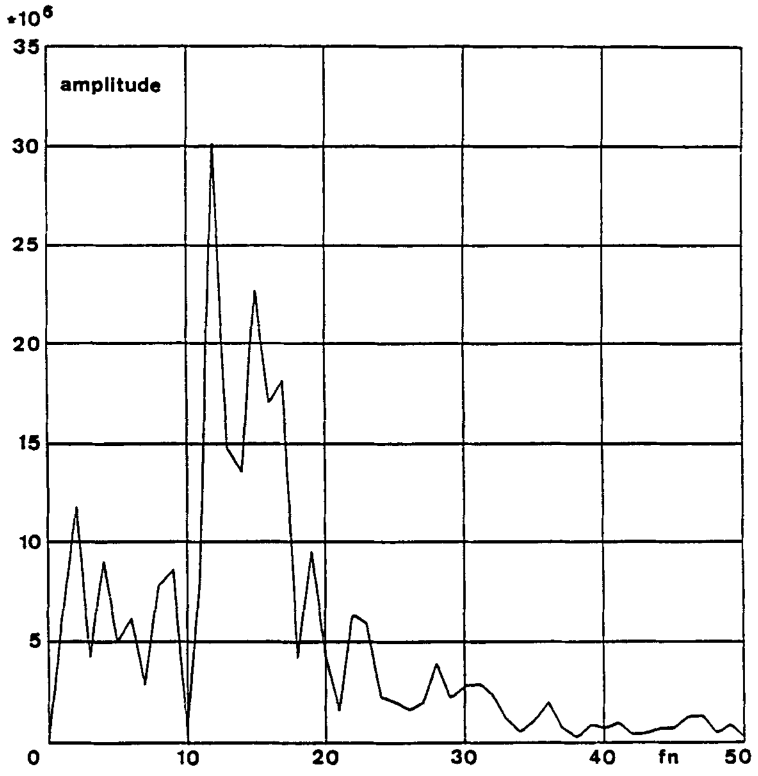

Figure 17(b). Fourier transform of the curve 'kinetic energy versus time' for the 3D3 mesh

Table II.

System	ICCG(1, 1), truncated		Diagonal scaling	
	Iterations	CPU time (s)	Iterations	CPU time (s)
Cons. Poisson ($\varepsilon = 5 \times 10^{-5}$)	32	1·75	120	1·97
Projection of velocity ($\varepsilon = 5 \times 10^{-5}$)	365	20	1550	25
Temperature ($\varepsilon = 10^{-4}$)	3·5	1·0	4·5	0·38

CONCLUSIONS

We have shown that the ICCG algorithm is a valuable tool for solving the symmetric and positive-definite systems encountered in the numerical solution of the transient three-dimensional Boussinesq equations. One of the main advantages of the algorithm is to reduce the memory requirements, while the CPU time remains reasonable as compared with the forward reduction/back substitution technique. In fact the reduction of the memory space has been so significant that we were able to introduce a second large system in the original algorithm suggested by Gresho *et al.*[1] Indeed the advection–diffusion equation for the temperature has been integrated in time by an explicit–implicit method which is quite advantageous for low Prandtl number flows. In our present example at $Pr = 0.069$ and with 47000 variables the time step of the fully explicit algorithm has been multiplied by 6, while the increase in CPU time per time step was only marginal. The ICCG algorithm is also expected to be efficient in the case of **A** and **B** matrices which do not remain constant in time.

We have found that the flow taking place in a horizontal Bridgman furnace bifurcates from a stationary state to an oscillatory one at a critical value of the Grashof number. Despite the fact that some features of the flow exhibit a periodic behaviour, we have not been able to identify a strictly periodic three-dimensional flow.

A serious limitation of the example given in the last section is the assumed symmetry of the flow. A recent investigation of the same problem allowing for a non-symmetric solution has revealed that the symmetric one switches over to a non-symmetric oscillatory flow.

ACKNOWLEDGEMENT

The authors wish to acknowledge a scholarship from the Belgian ‘Fonds National de la Recherche Scientifique’.

REFERENCES

1. P. M. Gresho, S. T. Chan, R. L. Lee and C. D. Upson, ‘A modified finite element method for solving time-dependent incompressible Navier–Stokes equations’, *Int. j. numer. methods fluids*, **4**, 557–598 (1984).
2. P. M. Gresho and S. T. Chan, ‘A new semi-implicit method for solving the time-dependent conservation equations for incompressible flow’, in *Numerical Methods in Laminar and Turbulent Flow*, Pineridge Press, Swansea, 1985, pp. 3–21.
3. P. M. Gresho, ‘Time integration and conjugate gradient methods for the incompressible Navier–Stokes equations’, in *Finite Elements on Water Resources*, Springer, New York, 1986, pp. 3–27.
4. H. A. Van der Vorst, ‘A vectorizable variant of some ICCG methods’, *SIAM J. Sci. Stat. Comput.*, **3**, 350–356 (1982).

5. S. Dupont, J. M. Marchal, M. J. Crochet and F. T. Geyling, 'Numerical simulation of the horizontal Bridgman growth. Part II: three-dimensional flow', *Int. j. numer. methods fluids*, **7**, 49–67 (1987).
6. M. J. Crochet, F. T. Geyling and J. J. Van Schaftingen, 'Numerical simulation of the horizontal Bridgman growth. Part I: two-dimensional flow', *Int. j. numer. methods fluids*, **7**, 29–47 (1987).
7. O. Axelson and V. A. Barker, *Finite Element Solution of Boundary Value Problems*, Academic Press, New York, 1984.
8. 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.* **44**, 134–155 (1981).
9. P. F. Dubois, A. Greenbaum and G. H. Rodrigue, 'Approximating the inverse of a matrix for use in iterative algorithms on vector processors', *Computing*, **22**, 257–268 (1979).