INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 12, 683–704 (1991)

SIMULATION OF TWO- AND THREE-DIMENSIONAL INTERNAL SUBSONIC FLOWS USING A FINITE ELEMENT METHOD

MARC BUFFAT

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

SUMMARY

In this paper a finite element method is presented to predict internal subsonic flows. Using a low-Mach-number approximation, the pressure is decomposed into a mean thermodynamic contribution and a dynamic fluctuation to deal with the complex role of the pressure in internal aerodynamics. A semi-implicit time integration and a finite element method with a moving mesh are described to take into account complex geometries and moving boundaries. An Uzawa algorithm accelerated by a preconditioned residual method is introduced to solve the coupled non-symmetric linear system for the velocity components and the pressure. An efficient conjugate gradient method combined with an incomplete LU preconditioning is used to solve the non-symmetric linear systems arising from the discretization. The implementation of the numerical scheme on parallel supercomputers is also discussed. Efficient algorithms for the finite element assembly phase and for the solution of linear systems are described which take advantage of the parallel architecture of the new generation of supercomputers. With this technique a global speed-up of 10 is achieved on a supercomputer with eight processors. To illustrate the capabilities of the numerical method, 2D and 3D simulations of flows in the combustion chamber of a reciprocating engine and around the combustor dome of a gas turbine engine are presented.

KEY WORDS Finite element Three-dimensional flows Parallelization Internal subsonic flows Conjugate gradient ILU preconditioning Uzawa algorithm Semi-implicit scheme Combustion engine Moving mesh

1. INTRODUCTION

Subsonic internal flows are a subject of considerable importance in a wide variety of machinery components, particularly in combustion engines. In many situations of practical relevance the geometrical configuration is complicated, the movement of the fluid is complex since important recirculation zones exist, and the flow field is compressible because of variations of density with temperature (as in a gas turbine engine) or pressure (as in a reciprocating combustion engine). In order to predict such complex flows, a numerical technique based on the finite element method (FEM) has been proposed and is now applied successfully in our laboratory to compute a wide variety of such flows.[1,2]

In the past decade, numerous finite element computer fluid dynamics (CFD) codes have arisen for solving the Navier–Stokes equations. Many of these schemes can be classified into two main categories depending on their fields of application (see Reference 3 and references therein). The first category solves the incompressible Navier–Stokes equations and is mainly used by mechanical engineers to predict hydrodynamic flows and shallow water or pollutant dispersion

0271–2091/91/070683–22$11.00
© 1991 by John Wiley & Sons, Ltd.
Received September 1989
Revised June 1990

problems. The main difficulties in computing these incompressible flows are (i) the treatment of the pressure through the continuity constraint with the so-called 'inf-sup' condition[4] and (ii) the turbulence modelling. The second category includes the codes developed by aeronautic engineers to compute flows around an aircraft. For such flows the viscous effects are important only near the walls and in the wake. Elsewhere the flow can be considered as inviscid. Thus many compressible codes are based on Euler solvers with special treatment to capture the shock waves.

The present numerical method tries to fill a part of the gap between the incompressible approach and the high-speed compressible approach. Our main motivation is the treatment of the pressure. Most of the compressible codes calculate the pressure as a dependent thermodynamic variable through the equation of state and use the same spatial approximation for all the variables. However, near the wall, where the local Mach number is low, the flow is nearly incompressible and it is well known that for the incompressible Navier–Stokes equations the pressure and velocity cannot be approximated independently.[4] Thus oscillations may appear in the numerical solution which are generally damped by the numerical viscosity of the scheme.[5] In internal aerodynamics the regions of low velocity, e.g. the recirculation zones, are important and must be well predicted in order to calculate the pressure head losses accurately. In such regions the pressure is related to the velocity through the continuity equation as in an incompressible flow; therefore the numerical algorithms developed for the incompressible Navier–Stokes equations are more appropriate and are used by some authors[5, 6] to predict compressible flows.

In Section 2 we introduce the low-Mach-number approximation[7] to cope with this complex role of the pressure.

In our applications we deal with a wide range of domains, from simple academic cases to complicated industrial problems, with fixed or moving boundaries, and we study both stationary and non-stationary flows. In Section 3 we present a numerical method to simulate this class of problems efficiently. The major capabilities of this scheme are:

(a) a semi-implicit time discretization to allow a wide range of time stepping (Section 3.1)
(b) a mixed Galerkin formulation leading to a saddle-point problem[8] for the treatment of the pressure and a finite element formulation on a moving mesh that can handle complex geometries with a moving frame of reference (Section 3.2)
(c) an efficient implementation of the numerical algorithm on vector and parallel supercomputers (Section 3.3).

An other important capability of this scheme is the easy introduction of additional conservation equations related to physical models (turbulence, combustion, etc.).

Illustrative numerical results of the flow inside combustion engines are then presented in Section 4.

2. GENERAL GOVERNING EQUATIONS

We consider the flow of a Newtonian fluid inside a domain Ω limited by a boundary Γ. Given a reference velocity, temperature and pressure U_0, T_0 and P_0 respectively, and a reference length L_0, we define the following non-dimensional quantities:

for the space components $\mathbf{x}$ and the time t,

$$\mathbf{x}=\frac{\mathbf{x}^*}{L_0}, \qquad t=\frac{U_0 t^*}{L_0}; \tag{1}$$

for the velocity $\mathbf{u}$, the temperature T, the density ρ and the pressure P,

$$\mathbf{u}=\frac{\mathbf{u}^*}{U_0}, \qquad T=\frac{T^*}{T_0}, \qquad \rho=\frac{\rho^*}{\rho_0(P_0,T_0)}, \qquad P=\frac{P^*}{P_0}; \tag{2}$$

for the physical characteristics of the fluid, i.e. the kinematic viscosity μ, the diffusivity λ and the specific heat at constant pressure, C_p,

$$\mu=\frac{\mu^*}{\mu(T_0)}, \qquad \lambda=\frac{\lambda^*}{\lambda(T_0)}, \qquad C_p=\frac{C_p^*}{C_p(T_0)}. \tag{3}$$

In the above definitions, star superscripts denote the dimensional quantities while zero subscripts denote the dimensional values at the reference point U_0, T_0 and P_0.

The independent dimensionless parameters appearing in the problem are the ratio of specific heats, γ, the Prandtl number Pr, the Reynolds number Re and the reference Mach number Ma, given by

$$\gamma=\frac{C_p(T_0)}{C_v(T_0)}, \qquad Pr=\frac{\mu(T_0)C_p(T_0)}{\lambda(T_0)}, \qquad Re=\frac{\rho(P_0,T_0)U_0L_0}{\mu(T_0)}, \qquad Ma=\frac{U_0}{c_0}. \tag{4}$$

where c_0 is the speed of sound at the reference temperature.

We decompose the total pressure $P^*(x,t)$ as the sum of a spatially uniform pressure $\bar{p}^*(t)$, which takes into account the change in the static pressure with time (i.e. the compressible part of the pressure), and a fluctuation $\tilde{\Pi}^*(x,t)$, which takes into account the dynamic effects (i.e. the incompressible part of the pressure):

$$P^*(x,t)=\bar{p}^*(t)+\tilde{\Pi}^*(x,t) \quad \text{with} \quad \int_\Omega \tilde{\Pi}^*(x,t)\,dx=0. \tag{5}$$

The corresponding non-dimensional quantities are defined by

$$\bar{p}(t)=\frac{\bar{p}^*(t)}{P_0}, \qquad \tilde{\Pi}(x,t)=\frac{\tilde{\Pi}^*(x,t)}{\rho(P_0,T_0)U_0^2}, \tag{6}$$

and relation (5) becomes

$$P(x,t)=\bar{p}(t)+\gamma Ma^2\tilde{\Pi}(x,t) \quad \text{with} \quad \int_\Omega \tilde{\Pi}(x,t)\,dx=0. \tag{7}$$

The governing equations are the conservation of mass, momentum and energy and the non-dimensional form of these equations is given by

$$\begin{aligned}
&\frac{1}{\rho}\frac{D\rho}{Dt}=-\operatorname{div}\mathbf{u},\\
&\rho\frac{D\mathbf{u}}{Dt}=-\overrightarrow{\operatorname{grad}}\,\Pi+\frac{1}{Re}\overrightarrow{\operatorname{div}}\,\mu(\vec{\nabla}\cdot\mathbf{u}+\vec{\nabla}^T\cdot\mathbf{u}),\\
&\rho\frac{D(C_pT)}{Dt}=\frac{1}{RePr}\operatorname{div}(\lambda\vec{\nabla}T)+\frac{\gamma-1}{\gamma}\frac{\partial\bar{p}}{\partial t}.
\end{aligned} \tag{8}$$

The equation of state is

$$\rho=\frac{\bar{p}}{T} \quad \text{with} \quad P=\bar{p}+\gamma Ma^2\tilde{\Pi} \quad \text{and} \quad \Pi=\tilde{\Pi}+\tfrac{2}{3}\mu\operatorname{div}\mathbf{u}. \tag{9}$$

In the above equations we have introduced the small-Mach-number approximation, which was first used by Chenoweth and Paolucci[7] to solve natural convection problems without the Boussinesq approximation. Thus in the energy equation (8) and in the equation of state (9) we have neglected the contribution of the dynamic pressure $\tilde{\Pi}$, which is proportional to the square of the Mach number Ma. This hypothesis is valid only for subsonic flows ($Ma<1$) and implies that the speed of the pressure disturbances is greater than the velocity, allowing the definition of a mean thermodynamic pressure $\bar{p}$ and a dynamic fluctuation $\tilde{\Pi}$ negligible in comparison.

However, the separation of the pressure into a mean compressible contribution $\bar{p}(t)$ and an incompressible fluctuation $\tilde{\Pi}(x, t)$ introduces $\bar{p}(t)$ as an additional unknown quantity. For incompressible flows ($Ma=0$) the mean thermodynamic pressure $\bar{p}(t)$ is constant. For subsonic flows ($Ma<1$) the definition of $\bar{p}(t)$ depends on the underlying physical problem. Generally, for internal flows with an open boundary we choose a constant pressure $\bar{p}(t)$ compatible with the external conditions, whereas for internal flows in a closed domain we use a global mass conservation equation, which leads to the global state equation

$$\frac{\bar{p}\bar{V}}{\bar{T}}=\text{cste}, \tag{10}$$

where $\bar{T}(t)$ denotes a mean temperature and $\bar{V}(t)$ the volume of the domain.

The principal advantage of the formulation (8), (9) is the ability to take into account incompressible flows as well as subsonic flows with an identical formulation. Consequently the same numerical algorithm can be used for both flows.

3. NUMERICAL ALGORITHM

Equations (8) and (9) are highly coupled, non-linear partial differential equations. The numerical approximation of this set of equations presents computational difficulties, mainly the coupling between the equations, the numerical treatment of the non-linearity and the determination of the reduced pressure Π. Furthermore, the numerical method must have the capability to deal with complex geometries. It should allow local grid refinement to get accurate results, with better computer efficiency than with the use of uniform meshes. Lastly, it must take into account a deformation of the computational domain. Hence we choose a semi-implicit scheme for the time integration and a finite element method with a moving mesh for the space discretization.

3.1. Time integration

Let us consider a domain $\Omega(t)$ which deforms itself with time to take into account moving boundaries. We denote $\mathbf{s}(x, t)$ the velocity (i.e. the deformation) of each point $\mathbf{x}$ belonging to Ω at time t. Let $\Phi(x, t)$ be a scalar field solution of the transport equation in $\Omega(t)$:

$$\rho\frac{\partial\Phi}{\partial t}+\rho\,\mathbf{u}\cdot\overrightarrow{\text{grad}}\;\Phi=\text{div}(\mu\,\overrightarrow{\text{grad}}\;\Phi), \tag{11}$$

where ρ and $\mathbf{u}$ verify the continuity equation and μ and ρ are strictly positive functions.

To integrate equation (11) in time, we use the following semi-implicit scheme. Let Δt be the time step; we denote

$$t^n=n\,\Delta t, \qquad \Phi^n=\Phi(x^n, t^n), \qquad \Phi^{n+1}=\Phi(x^{n+1}, t^{n+1}), \qquad \rho^n=\rho(x^n, t^n), \qquad \mathbf{u}^n=\mathbf{u}(x^n, t^n). \tag{12}$$

The discretized equation (11) is written as follows:

$$\rho^n\frac{\Phi^{n+1}-\Phi^n}{\Delta t}+\rho^n(\mathbf{u}^n-\mathbf{s})\cdot\vec{\nabla}\Phi^{n+1}=\text{div}(\mu\vec{\nabla}\Phi^{n+1}). \tag{13}$$

In this scheme we calculate the solution by following the deformation of the domain; hence Φ^{n+1} is the value of Φ at time t^{n+1} and at the point $\mathbf{x}^{n+1}$, which is the new position of the point $\mathbf{x}^n$ given by

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \mathbf{s}\,\Delta t.$$

This scheme is first-order-accurate in time and unconditionally stable. This approach is also called an arbitrary Lagrangian–Eulerian formulation.[9]

To integrate the system of partial differential equations (8), (9) in time, we use the preceding scheme, which allows the linearization of the equations. Furthermore, the energy equation (and other scalar transport equations) and the momentum equation can now be solved sequentially instead of simultaneously. Because of this segregated solution approach, the overall scheme is not unconditionally stable, but it retains good stability properties.

Let us define $\mathbf{W} = [u_1, u_2, u_3, T]^{\mathrm{T}}$, the vector of state variables. The system of equations (8), (9) can be written symbolically as

$$\rho \frac{\mathrm{D}\mathbf{W}}{\mathrm{D}t} = \operatorname{div}(\mathbb{K}(\vec{\nabla}\mathbf{W})) + \mathbb{H}(\vec{\nabla}\Pi) + \mathbb{G}(\bar{p}), \tag{14}$$

$$\operatorname{div}\mathbf{u} = -\frac{1}{\rho}\frac{\mathrm{D}\rho}{\mathrm{D}t}, \tag{15}$$

where $\mathbb{K}(\vec{\nabla}\mathbf{W})$ is the tensor of viscous fluxes, $\mathbb{H}(\vec{\nabla}\Pi)$ is the normal stress constraint in the momentum equation and $\mathbb{G}(\bar{p})$ is the pressure contribution in the energy equation.

The semi-implicit scheme is then written as follows:

$$\rho^n\left[\left(\frac{\mathbf{W}^{n+1}-\mathbf{W}^n}{\Delta t}\right) + (\mathbf{u}^n - \mathbf{s})\cdot\vec{\nabla}\mathbf{W}^{n+1}\right] = \operatorname{div}\alpha_{\mathrm{w}}\cdot\vec{\nabla}\mathbf{W}^{n+1} + \operatorname{div}\tilde{\mathbb{K}}(\vec{\nabla}\mathbf{W}^n) + \mathbb{H}(\vec{\nabla}\Pi^{n+1}) + \mathbb{G}(\bar{p}^{n+1}),$$

$$\operatorname{div}\mathbf{u}^{n+1} = -\frac{1}{\rho^n}\left[\left(\frac{\rho^{n+1}-\rho^n}{\Delta t}\right) + (\mathbf{u}^n - \mathbf{s})\cdot\vec{\nabla}\rho^{n+1}\right], \tag{16}$$

with

$$\rho^{n+1} = \bar{p}^{n+1}/T^{n+1}. \tag{17}$$

In the system (16) we have decomposed the viscous fluxes into a linear part and a residue:

$$\mathbb{K}(\vec{\nabla}\mathbf{W}) = \alpha_{\mathrm{w}}\cdot\vec{\nabla}\mathbf{W} + \tilde{\mathbb{K}}(\vec{\nabla}\mathbf{W}) \quad \text{with} \quad \alpha_{\mathrm{w}} = \left[\frac{\mu}{Re}, \frac{\mu}{Re}, \frac{\mu}{Re}, \frac{\lambda}{Re\,Pr}\right]^{\mathrm{T}}. \tag{18}$$

The computational procedure is as follows.

1. The velocity and scalar fields are known at time t^n.
2. The energy equation is solved together with an equation for the mean thermodynamic pressure (e.g. equation (10)) to obtain T^{n+1} and $\bar{p}^{n+1}$.
3. Using current values of temperature and pressure at time t^{n+1}, we obtain the density ρ^{n+1} from the state equation.
4. The reduced pressure Π^{n+1} and the velocity $\mathbf{u}^{n+1}$ are solved from the momentum and mass conservation equations by using an Uzawa algorithm which decouples the equations.

This time discretization leads to the solution of several decoupled non-symmetric linear problems for the temperature, the velocity components and the pressure.

3.2. Finite element formulation with a moving mesh

For the space discretization we use a finite element method whose great flexibility in choosing the mesh (particularly when the elements are triangles) can be exploited if the domain has an irregular boundary or if the solution is known to have sharp gradients in some part of the domain.

For incompressible flows, in order to guarantee the convergence properties, the combination of velocity and pressure interpolation requires satisfaction of the Ladyzhenskaya–Babuška–Brezzi (LBB) consistency condition.[4] This precludes in particular the use of equal-order interpolations. Thus we have chosen the P1/iso-P2 element,[10] which gives a continuous and piecewise linear interpolation for the reduced pressure Π associated with a continuous and piecewise linear interpolation for the velocity components, the temperature and the density on a grid twice as fine as the pressure grid (each element of the pressure mesh is divided into four triangles in 2D and into eight tetrahedrons in 3D). To take into account the deformation of the domain, we apply a moving finite element method[11] with imposed nodes velocities. At each node $\mathbf{x}_i$ of the finite element mesh, we define the moving velocity $\mathbf{s}_i$ which is calculated by linear interpolation from the imposed boundary deformation. The finite element interpolation of the variables is written as

$$\mathbf{W}_h(x,t)=\sum_1^N \mathbf{W}_i(t)\,N_i(x,x_i), \qquad \rho^h=\sum_1^N \rho_i(t)\,N_i(x,x_i), \qquad \Pi_h(x,t)=\sum_1^{N1} \Pi_i(t)\,N_i^1(x,x_i). \tag{19}$$

$N1$ is the number of mesh points on the pressure mesh, N is the number of mesh points on the refined mesh and $\mathbf{x}_i(t)$ is the co-ordinate of the node i with

$$\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{d}t}=\mathbf{s}_i.$$

N_i and N_i^1 are the classical P1 basis functions associated respectively with the approximate velocity space V^h and the approximate pressure space V_1^h.

Discrete approximations of equations (16) and (17) are then sought using the standard Galerkin formulation by integration over the domain Ω^{n+1} at time t^{n+1}. The discrete weak formulation is written:

find $\mathbf{W}_h\in(V^h)^4$ and $\Pi_h\in V_1^h$ such that

$$\int_\Omega\left[\left(\frac{\rho^n}{\Delta t}\mathbf{W}_h^{n+1}+\rho^n(\mathbf{u}_h^n-\mathbf{s})\cdot\vec{\nabla}\,\mathbf{W}_h^{n+1}\right)\varphi-\alpha_w\cdot\vec{\nabla}\,\mathbf{W}_h^{n+1}\cdot\vec{\nabla}\,\varphi\right]\mathrm{d}x+\int_\Omega \mathbb{H}(\vec{\nabla}\,\Pi_h^{n+1})\cdot\varphi\,\mathrm{d}x$$

$$=\int_\Omega f(\mathbf{W}_h^n)\cdot\varphi\,\mathrm{d}x \quad \forall\varphi\in(V^h)^4 \tag{20}$$

$$\int_\Omega \operatorname{div}\mathbf{u}_h^{n+1}\cdot\psi\,\mathrm{d}x=\int_\Omega g(\rho_h^{n+1},\mathbf{u}_h^n)\cdot\psi\,\mathrm{d}x \quad \forall\psi\in V_1^h$$

We obtain the following matrix equations for the nodal values of the velocity components u_i^{n+1}, the temperature T^{n+1} and the pressure Π^{n+1}:

$$\begin{bmatrix} \mathbb{A}_{11} & 0 & 0 & 0 & \mathbb{B}_1 \\ 0 & \mathbb{A}_{22} & 0 & 0 & \mathbb{B}_2 \\ 0 & 0 & \mathbb{A}_{33} & 0 & \mathbb{B}_3 \\ 0 & 0 & 0 & \mathbb{A}_{44} & 0 \\ \mathbb{C}_1 & \mathbb{C}_2 & \mathbb{C}_3 & 0 & 0 \end{bmatrix}\begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ T^{n+1} \\ \Pi^{n+1} \end{bmatrix}=\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ G \end{bmatrix}, \tag{21}$$

which is written symbolically as

$$\mathbb{A}_{ii} \cdot W_i + \mathbb{B}_i \cdot \Pi = F_i, \qquad \mathbb{C}_i \cdot W_i = G.$$

The matrices $\mathbb{A}_{ii}$ are non-symmetric and time-dependent, so they must be recalculated at each time step. To solve the linear system of coupled equations (21), we use an iterative Uzawa algorithm. The mass conservation equation is interpreted as a linear constraint applied to the velocity $\mathbf{u}$ and the reduced pressure Π as the Lagrange multiplier of this constraint. Whereas many numerical schemes are available to solve this linear problem under linear constraint,[8] the iterative Uzawa algorithm is undoubtedly the simplest approach. It may be interpreted as an iterative solver of the following linear system for the pressure deduced from (21):

$$\mathbf{A}\Pi = \mathbf{B} \quad \text{with} \quad \mathbf{A} = C_i \cdot A_{ii}^{-1} \cdot B_i \quad \text{and} \quad \mathbf{B} = C_i \cdot A_{ii}^{-1} \cdot F_i - G. \tag{22}$$

To accelerate the speed of convergence of the classical Uzawa algorithm, we use a minimal residual method with preconditioning.[12] Whereas the matrix $\mathbf{A}$ is non-symmetric, its symmetrical part is definite positive and thus the minimal residual algorithm converges with a convergence rate proportional to cond($\mathbb{A}$) (the condition number of $\mathbf{A}$).[13]

The pressure algorithm is described below.

Initialization (23)

1. Π^0 given.
2. Compute each velocity component W_i^0 $(i = 1, 3)$ from

$$\mathbb{A}_{ii} \cdot W_i^0 = -\mathbb{B}_i \cdot \Pi^0 + F_i.$$

3. Compute the residual of the continuity equation

$$r^0 = \mathbb{C}_i \cdot W_i^0 - G.$$

4. Compute the descent direction q^0 for Π and Z_i^0 for W_i:

$$\mathbb{S} \cdot q^0 = r^0,$$

$$\mathbb{A}_{ii} \cdot Z_i^0 = -\mathbb{B}_i \cdot q^0.$$

Step $k+1$

1. Calculate the optimal descent parameter ρ^k such that the norm of r^k is minimal:

$$\rho^k = -\frac{\langle r^k, \mathbb{C}_i \cdot Z_i^k \rangle}{\langle \mathbb{C}_i \cdot Z_i^k, \mathbb{C}_i \cdot Z_i^k \rangle}.$$

2. Then obtain the update values for Π, W and r:

$$\Pi^{k+1} = \Pi^k + \rho^k \cdot q^k,$$

$$W_i^{k+1} = W_i^k + \rho^k \cdot Z_i^k,$$

$$r^{k+1} = r^k + \rho^k \cdot \mathbb{C}_i \cdot Z_i^k;$$

and calculate the gradient directions g^{k+1} and Y_i^{k+1} for Π and W_i:

$$\mathbb{S} \cdot g^{k+1} = r^{k+1},$$

$$\mathbb{A}_{ii} \cdot Y_i^{k+1} = -\mathbb{B}_i \cdot g^{k+1}.$$

3. Calculate the conjugate parameter γ^{k+1} such that $\langle \mathbb{C}_i \cdot Z_i^k, \mathbb{C}_i \cdot Z_i^{k+1} \rangle = 0$,

$$\gamma^{k+1} = -\frac{\langle \mathbb{C}_i \cdot Y_i^{k+1}, \mathbb{C}_i \cdot Z_i^k \rangle}{\langle \mathbb{C}_i \cdot Z_i^k, \mathbb{C}_i \cdot Z_i^k \rangle};$$

and then the new descent directions

$$q^{k+1} = g^{k+1} + \gamma^{k+1} \cdot q^k,$$
$$Z_i^{k+1} = Y_i^{k+1} + \gamma^{k+1} \cdot Z_i^k.$$

4. If $\| r^{k+1} \| > \varepsilon$, then iterate.

In the above algorithm the matrix preconditioner $\mathbb{S}$ is an extension of the preconditioner proposed by Cahouet and Chabard[14] and is defined as follows:

$$\mathbb{S}^{-1} = \left(\frac{Re}{\mu} \mathrm{Id} \right)^{-1} - \left(\mathrm{div} \frac{\Delta t}{\rho} \nabla \right)^{-1}. \tag{24}$$

With this preconditioner the convergence rate of the minimal residual algorithm is then proportional to cond $(\mathbb{S}^{-1} \cdot \mathbb{A})$.

Figure 1 shows a typical convergence rate of the conjugate minimal residual method with preconditioning for the test case of the lid-driven cavity ($N = 289$, $Re = 100$, $\Delta t = 1{\cdot}0$). This algorithm is about twice as fast as the minimal residual method without the conjugate relation ($\gamma = 0$) and about 20 times faster than the classical Uzawa algorithm.

In this scheme each iteration is computationally quite inexpensive since it requires the solution of three linear systems of order N (one for each component of the velocity) and two systems of order $N1$ (for the pressure preconditioning) instead of the original linear system of order $3(N + N1)$.

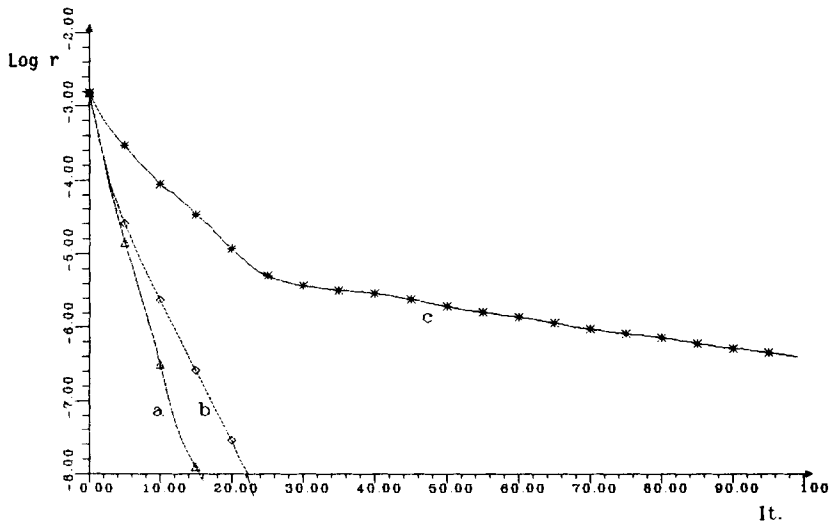

Figure 1. Logarithm of the residual r^k versus the number of iterations, k, for different pressure algorithms: (a) minimal residual method with preconditioning; (b) gradient method with preconditioning ($\gamma = 0$); (c) Uzawa algorithm without preconditioning

3.3. Solution of the non-symmetric linear system

For large problems, especially in 3D, the computational bottleneck of the code is the solution of the linear systems. Thus it is important to have an efficient non-symmetric sparse matrix solver. For the size of problems that we want to solve (several tens of thousands of nodes for a typical 3D problem), direct solvers would not be viable because the factorization process would be costly in CPU time and the factorized matrix would require very large storage. Among the iterative methods, the conjugate gradient (CG) method[15] and the multigrid (MG) methods[16] have been drawing increased attention as powerful iterative solvers for large symmetric sparse systems of linear equations. In recent years several generalizations of the CG method have been proposed for solving non-symmetric linear systems. Even if none of the generalizations emerges as a clear winner, numerical experiments[17] suggest that the conjugate gradient square (CGS)[18] and the generalized minimum residual (GMRES)[19] methods yield the best performances. We have chosen the CGS algorithm that was introduced recently by Sonneveld *et al.*[18] It is derived from the bi-conjugate gradient (BCG) method.[20] To solve the non-symmetric linear system of order N,

$$\mathbb{A} \cdot X_1 = B_1, \tag{25}$$

the BCG algorithm uses a CG method applied to the symmetric (but not definite positive) matrix system

$$\begin{bmatrix} 0 & \mathbb{A} \\ \mathbb{A}^T & 0 \end{bmatrix} \begin{bmatrix} X_2 \\ X_1 \end{bmatrix} = \begin{bmatrix} B_1 \\ B_1 \end{bmatrix}. \tag{26}$$

The BCG residue R_i for the ith iteration can be expressed as a polynomial $\Phi_i(\mathbb{A})$ of degree i in the matrix A applied to the original residue R_0:

$$R_i = \Phi_i(\mathbb{A}) \cdot R_0. \tag{27}$$

The BCG method can be accelerated appreciably (roughly by a factor of two) by using the CGS algorithm for which the residue is $\Phi_i^2(\mathbb{A}) \cdot R_0$ instead of $\Phi_i(\mathbb{A}) \cdot R_0$. Furthermore, it eliminates the need to work with $\mathbb{A}^T$. The CGS and the BCG algorithms converge in at most N iterations if they do not break down. As far as we know, there is no general theory to establish *a priori* if this condition is satisfied. However, if $\mathbb{A}$ is symmetric, the BCG method is equivalent to the CG method, which never breaks down. To improve the convergence rate of the CGS method, we use a preconditioner $\mathbb{S}$ and apply the CGS algorithm to the following problem (instead of (25)):

$$\mathbb{S}^{-1} \cdot \mathbb{A} \cdot X_1 = \mathbb{S}^{-1} \cdot B_1. \tag{28}$$

For a given residue reduction ε, the required number of iterations, n, of a preconditioned CG method is given classically[13] by

$$n \geq \tfrac{1}{2} |\ln(\varepsilon/2)| \operatorname{cond}(\mathbb{S}^{-1} \cdot \mathbb{A})^{1/2}. \tag{29}$$

An efficient preconditioner $\mathbb{S}$ is then a matrix close to $\mathbb{A}$, i.e. such that $\operatorname{cond}(\mathbb{S}^{-1} \cdot \mathbb{A})$ is close to unity.

The simplest preconditioner is the diagonal scaling where

$$\mathbb{S} = \operatorname{diag}(\mathbb{A}). \tag{30}$$

Another popular preconditioner is the polynomial preconditioning, where $\mathbb{S}^{-1}$ is a polynomial in $\mathbb{A}$. This technique involves only matrix–vector products and can be efficiently implemented on vector and parallel supercomputers.[21]

However, the most efficient class of preconditioners is the incomplete LU decomposition (ILU) or the modified incomplete LU decomposition (MILU) (see Reference 22 for a comparison of the different versions) where the matrix $\mathbb{S}$ is of the form

$$\mathbb{S}=\mathbb{L}\cdot\mathbb{D}\cdot\mathbb{U}. \tag{31}$$

$\mathbb{L}$, $\mathbb{D}$ and $\mathbb{U}$ are lower triangular, diagonal and upper triangular matrices respectively. They have the same sparsity as the original matrix $\mathbb{A}$ and are calculated from an incomplete Crout factorization of $\mathbb{A}$, restricted to the non-zero elements of $\mathbb{A}$. There is no general theory available concerning the influence of the ILU preconditioning on the condition number. However, for some discretizations of second-order elliptic problems one finds[18]

$$\text{cond}(\mathbb{S}^{-1}\cdot\mathbb{A})=\theta(h^{-1})=\theta(N^{1/2}). \tag{32}$$

According to (29), the required number of iterations is $\theta(N^{1/4})$ and the computational work per iteration is $\theta(N)$ (two sparse matrix–vector products $\mathbb{A}\cdot X$ and two solutions of sparse triangular systems $\mathbb{S}^{-1}\cdot X$). This yields a computational cost of $\theta(N^{5/4})$. This result seems to hold approximately for the CGS algorithm with ILU preconditioning and one finds that the required number of iterations increases slowly as the grid is refined. The convergence is even somewhat better for convection–diffusion problems.[23] Figure 2 shows the convergence rate of the CGS algorithm with ILU preconditioning applied to the solution of a convection–diffusion equation ($Re=100$, $\Delta t=0{\cdot}1$, $N=968$). For this test case the CGS + ILU algorithm is twice as fast as the BCG + ILU algorithm and seven times faster than the CGS algorithm with diagonal scaling.

3.4. Implementation on supercomputers

In the past decade an estimated improvement of five orders of magnitude in the cost of CFD simulations has been achieved[19] and is attributed both to the advent of high-performance vector-processing supercomputers and to algorithmic improvements which take advantage of the new architectural features of these supercomputers. Because further development of the technology of

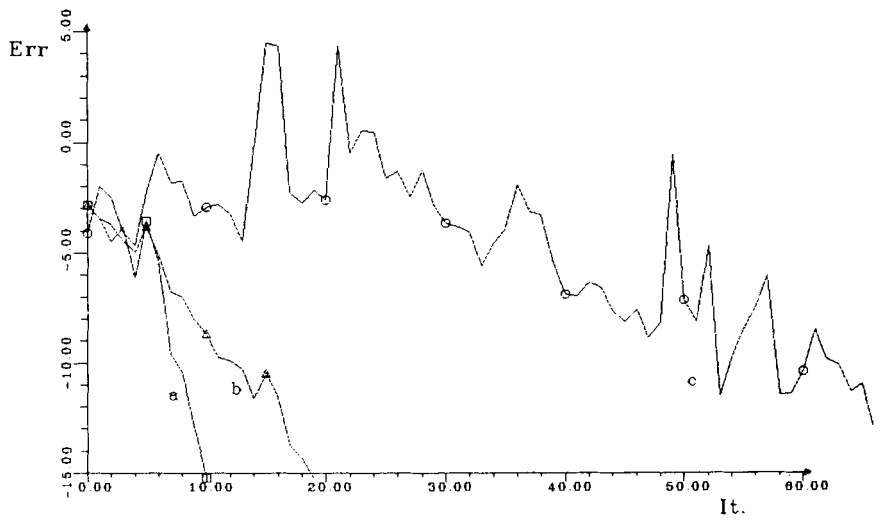

Figure 2. Logarithm of the residual R_i versus the number of iterations, i, for different iterative solvers: (a) CGS with ILU preconditioning; (b) BCG with ILU preconditioning; (c) CGS with diagonal scaling

vector computing seems unlikely to provide a large increase in performance, big gains in computational cost will come in the future from parallel processing or from a combination of parallel and vector processing. A survey of recent progress in architectures and algorithms for parallel scientific computing is found in Reference 24. The need to extract high performance levels from parallel processors causes the numerical methods and the architectures to become more and more interdependent. Algorithms must be developed to minimize the computational work and yet take advantage of these new architectures.

Although explicit methods map well onto parallel machines and are easier to implement than implicit methods because they use only local information and do not involve coupling between the locations during the computational step, they have the drawback of poor convergence rates. On the other hand, implicit methods lead to better convergence rates owing to the coupling of the locations during the computational step, but their mapping onto parallel machines is more difficult.

The present semi-implicit numerical method is implemented[23] on a shared-memory parallel computer (Alliant FX/80) with up to eight vector processors (ACEs). The architecture of this machine allows the concurrency to be carried out at the loop level (microtasking) and the maximum efficiency is obtained with nested loops, where the inner loop is vectorized and the outer loop runs in concurrency. When implementing the implicit FEM on a parallel supercomputer, the two crucial points are: (i) the computation and assembly of the matrices and right-hand sides; (ii) the solution of the large sparse non-symmetric linear systems.

3.4.1. Parallelization of the assembly phase. The computations of elementary matrices and right-hand sides (RHS) on each element of an FEM mesh are independent, but their accumulation at the nodes induces a dependence between adjacent elements. To achieve the parallelization of the assembly phase, the elements are separated into groups by using a graph-colouring algorithm so that two elements of the same group have no common nodes. This technique was used by Angrand and Erhel[25] to vectorize FEM codes. In our application the assembly is performed in parallel on the coarse mesh, yielding larger granularity of the parallel tasks and nearly optimal speed-up on an FX/80 (see Table I).

3.4.2. Parallelization of the resolution phase. When implementing the preconditioned CGS algorithm on a parallel supercomputer, the potential bottlenecks are in the sparse matrix–vector product and in the solution of the sparse triangular systems.

To optimize the matrix–vector product $Y = \mathbb{A} \cdot X$, we opt for a general sparse matrix data structure which contains only the non-zero elements of the matrix, stored row-wise. With this storage scheme each component Y_i of the resulting vector Y can be computed in parallel as the dot product of the ith row of A with the vector X. This structure allows operations on vectors of nearly constant length (the mean number of neighbours of a node) and performs well on multivector processors such as the Alliant FX/80 (see Table II).

Table I. Percentage of CPU time and speed-up on an FX/80 with four ACEs for the different parts of the solution of the energy equation (3D test case, $N = 16473$)

	Matrix	RHS	Linear system	ILU	Total
CPU time	65%	4%	10%	21%	100%
Speed-up	3·67	3·97	3·30	2·71	3·50

Table II. Performance of the dot product (based on peak performance), of the matrix–vector product and of the solution of triangular systems on an FX/80 with one and four ACEs. The timing is given in microseconds per node with four ACEs

	1 ACE (Mflops)	4 ACEs (Mflops)	Speed-up	CPU/node (μs)
Dot product				
$X(i)\cdot Y(i)$	20·0	80·0	4·0	0·025
$X(\text{num}(i))\cdot Y(i)$	5·0	20·0	4·0	0·1
Sparse matrix				
$Y=\mathbb{A}\cdot X$	2·2	8·2	3·7	2·5
$\mathbb{S}\cdot X=B$	1·0	3·6	3·6	7·6

Table III. Global speed-up on an FX/80 with eight ACEs, (3D test case, $N=9145$)

	Scalar	1 ACE	2 ACEs	4 ACEs	8 ACEs
Speed-up	1·0	2·2	4·2	7·0	10·6

The solution of the linear system $\mathbb{S}\cdot X=B$, where $\mathbb{S}$ is the product of a lower triangular, a diagonal and an upper triangular matrix, suffers from a severe drawback which inhibits easy parallelization. It requires the solution of sparse triangular systems $\mathbb{L}\cdot X=B$ where the ith unknown X_i is computed sequentially using a formula which implies generally a first-order recurrence:

$$X_i=\frac{1}{\mathbb{L}_{ii}}\left(B_i-\sum_{i<j,\,\mathbb{L}_{ij}\neq 0}\mathbb{L}_{ij}\cdot X_j\right). \tag{33}$$

To overcome this difficulty, we have implemented a level-scheduling algorithm.[19] The idea of level scheduling is to look at the adjacency graph of the sparse matrix and to determine groups of equations that can be solved simultaneously. The nodes are grouped into levels such that the nodes in the same level have their predecessors only in the preceding levels. Thus all the unknowns X_i in the same level can be computed in parallel using relation (33). Level scheduling is a very efficient technique for the parallelization of the solution of triangular systems and the same speed-up as for the matrix–vector product can be achieved (see Table II). However, the vectorization does poorly because the vector length is not constant and varies from unity to the maximum number of elements in the rows of $\mathbb{L}$, so that the computational cost is about three times the cost of the matrix–vector product (Table II).

Finally, we wish to give some global performance of the code concerning the speed-up, the CPU time and the storage requirement. Speed-up results for a 3D test problem are displayed in Table III. A global speed-up of 10·6 (compared with scalar mode) is achieved on an FX/80 with eight vector processors (ACEs). The CPU time per time step, per variable, per node ranges from 0·3 to 3 ms for typical 3D runs on an FX/80 with four ACEs. The memory requirement is linear with respect to the number of nodes and the code requires 25 words of storage per variable, per node.

4. NUMERICAL RESULTS

In this section the pressure treatment and the moving finite element formulation are first validated against an analytical solution. Then 2D and 3D simulations of flows in a combustion engine and around the combustor dome of a gas turbine engine are presented to illustrate the capabilities of the numerical scheme.

4.1. One-dimensional compression

To validate the pressure treatment and the moving finite element technique, we solve the following one-dimensional academic problem using the 2D version of our code.

We consider the 1D compression of a Newtonian fluid in a chamber of length $l(t)$, with a moving piston having a velocity $V_p(t)$. We apply Dirichlet boundary conditions for the velocity and adiabatic boundary conditions for the temperature (see Figure 3).

The 1D analytic solution can be written as

$$\begin{aligned} u_1(x,t) &= V_p(t)\frac{x}{l(t)}, \\ T(t) &= l(t), \\ \Pi(x,t) &= -\frac{\dot{V}_p}{2}\frac{x^2}{l^2}, \\ \bar{p}(t) &= l^{-\gamma}, \\ \rho(t) &= l^{-1}. \end{aligned} \tag{34}$$

Using a mesh with 21×5 nodes, we have computed an approximate solution with $V_p(t) = -0{\cdot}5\sin(t)$ and a time step equal to 0·02. In Figure 4 we have plotted the calculated velocity profiles along the central line at different times, which coincide with the analytical formula (34). In the same figure are drawn the reduced pressure profiles, which are parabolic like the exact solution (34). The calculated reduced pressures on the piston differ by less than 10% from the exact values.

4.2. Flow in a combustion chamber

To illustrate the capabilities of the scheme to calculate flows in complex geometries with moving boundaries, we have applied our numerical code to the simulation of the flow field within the cylinder of a reciprocating engine during the admission and compression strokes.

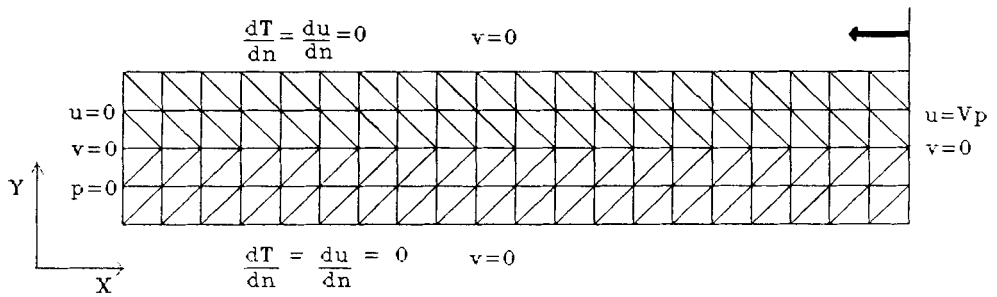

Figure 3. FEM mesh (21×5 nodes) and boundary conditions; 1D compression problem

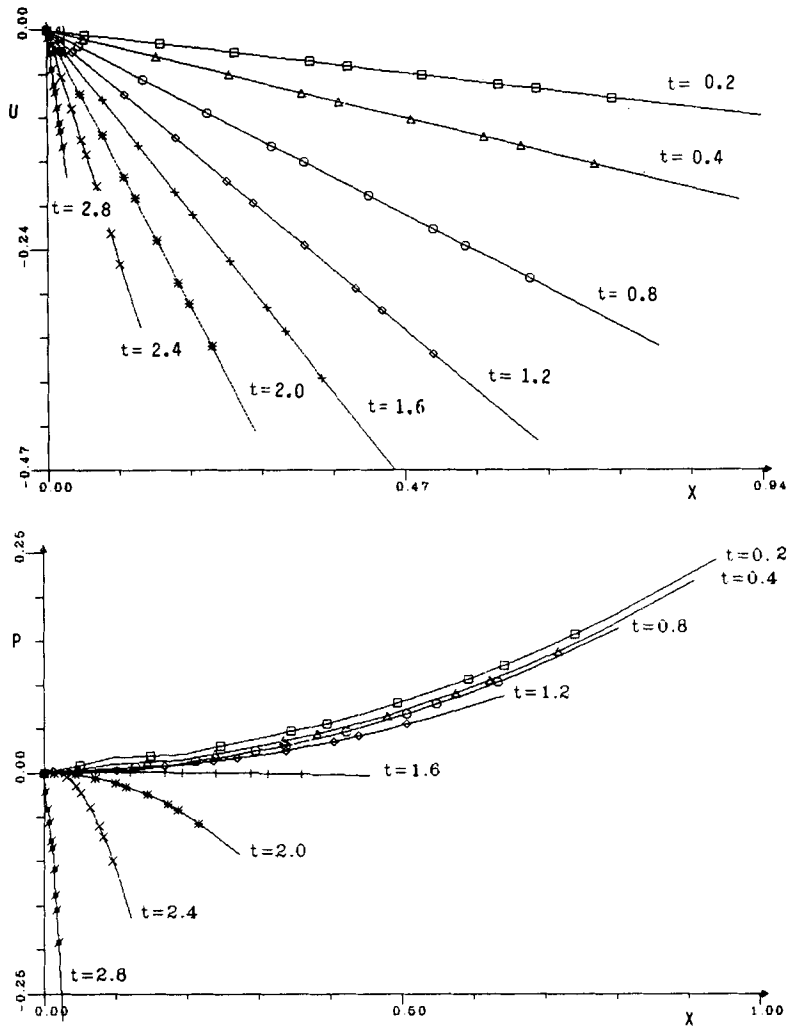

Figure 4. 1D compression solution at different times: top, velocity profiles; bottom, pressure profiles.

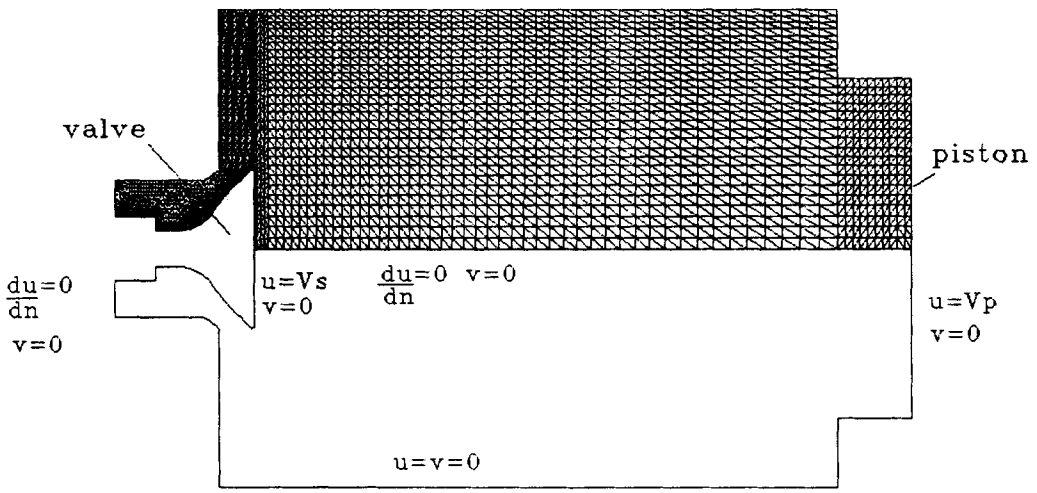

Figure 5. FEM mesh (2148 nodes) and boundary conditions; 2D combustion chamber

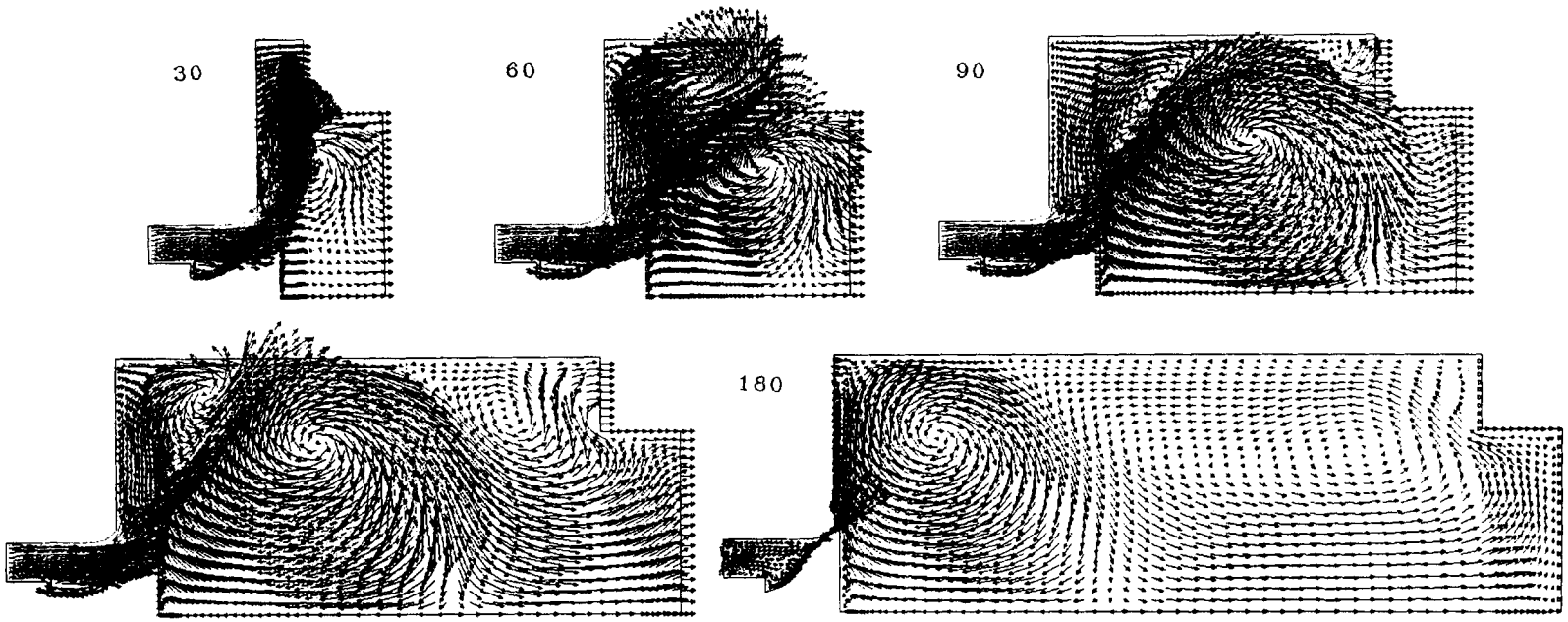

Figure 6. Velocity vector fields at different crank angles during admission ($Re = 500$)

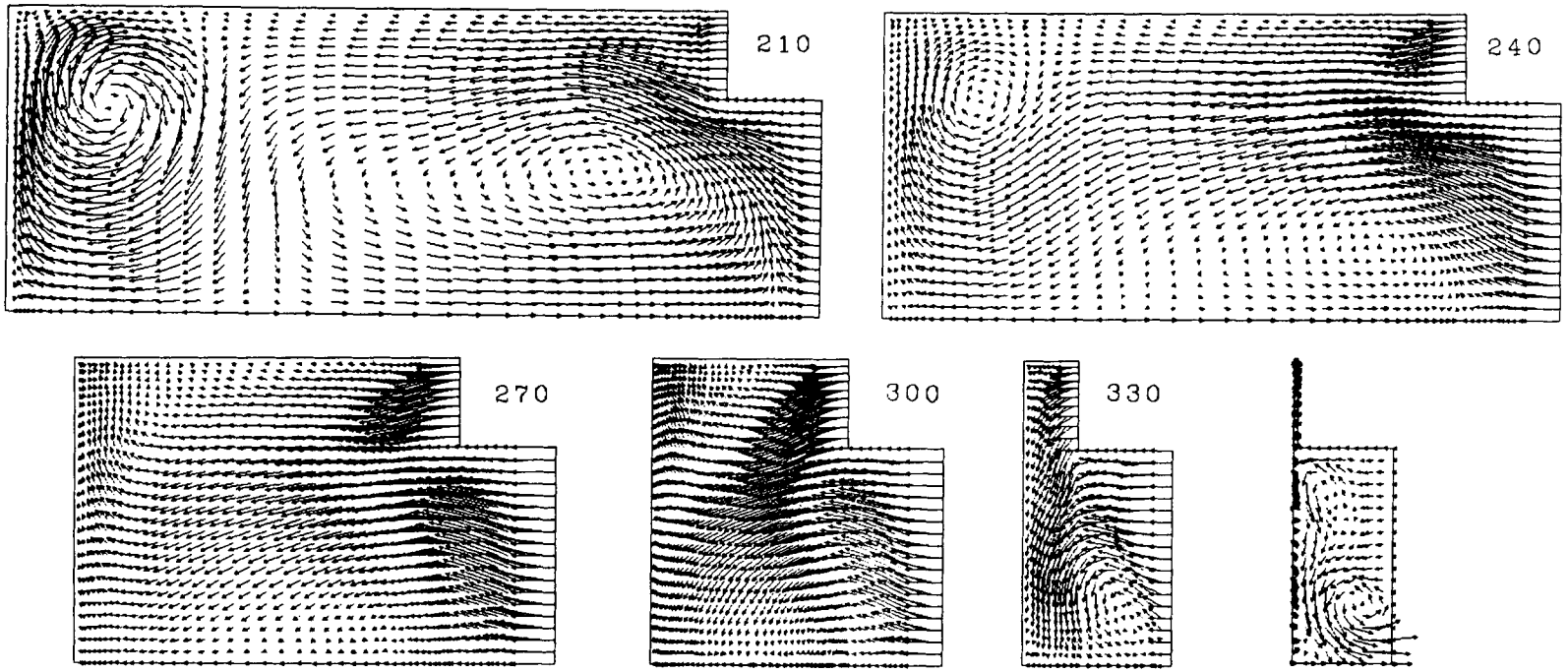

Figure 7. Velocity vector fields at different crank angles during compression ($Re = 500$)

4.2.1. 2D configuration. The finite element mesh used for the simulation of the admission stroke is drawn in Figure 5. At the beginning of the computation the valve is open with a valve lift equal to 1 mm (the radius of the chamber is 60 mm).

During the admission stroke the maximum valve lift reaches 12 mm for an engine speed of 250 rev min^{-1}. During the compression stroke the valve remains closed and the compression ratio is equal to 10 at the end of the stroke. The calculation begins with a velocity field initialized to zero and with uniform temperature and density fields. Neumann boundary conditions are used as inflow conditions and a no-slip velocity condition is imposed at the wall. For the temperature, adiabatic conditions are prescribed. Figures 6 and 7 show the velocity fields at different crank angles during the admission and compression strokes. The flow field is quite complex, with the formation of very large vortices during the induction which disappear during the compression. At the end of the compression it is interesting to notice the squish phenomenon due to the presence of a bowl in the centre of the piston.

For this configuration, simulations using the same numerical scheme with turbulence modelling have been done. The numerical results and the comparisons with experimental data are reported in Reference 26.

4.2.2. 3D configuration. We then applied the 3D version of our code to simulate flow with swirl in the same port–cylinder assembly, but without a piston and with a fixed valve. The mesh used for the formulation is drawn in Figure 8.

Figure 9 depicts the velocity field in different planes. While the swirl component is strong in the entrance and in the jet near the valve, it is rapidly damped in the chamber because of the viscosity effects. At the outlet the velocity profile becomes parabolic without swirl. In the symmetry plane the structure of the flow compares with the results of the 2D simulation in the middle of the induction phase (Figure 6). This 3D simulation requires 100 time steps to get a stationary solution starting from rest. The number of iterations for the pressure algorithm ranges from 14 at the beginning of the simulation to one for the last time steps. The number of CGS iterations varies

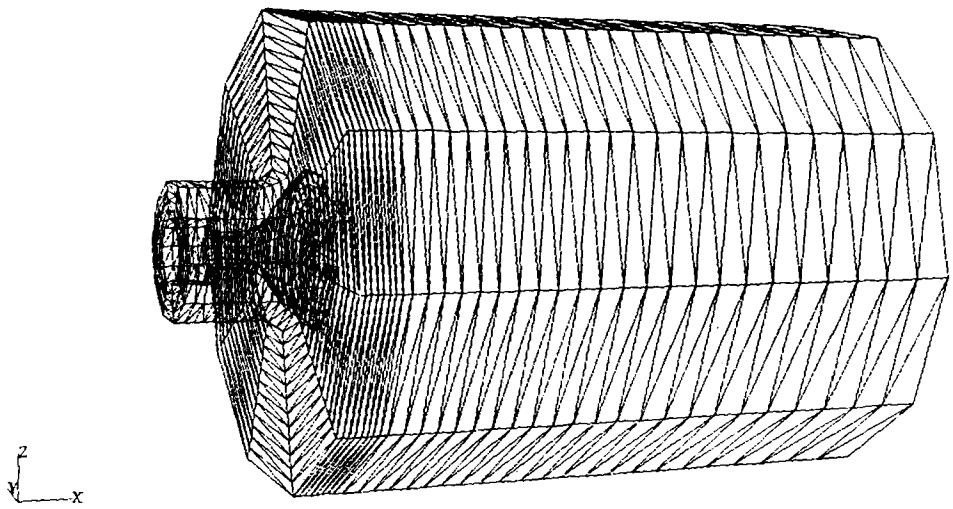

Figure 8. Perspective view of the 3D FEM mesh (41 481 nodes); 3D combustion chamber

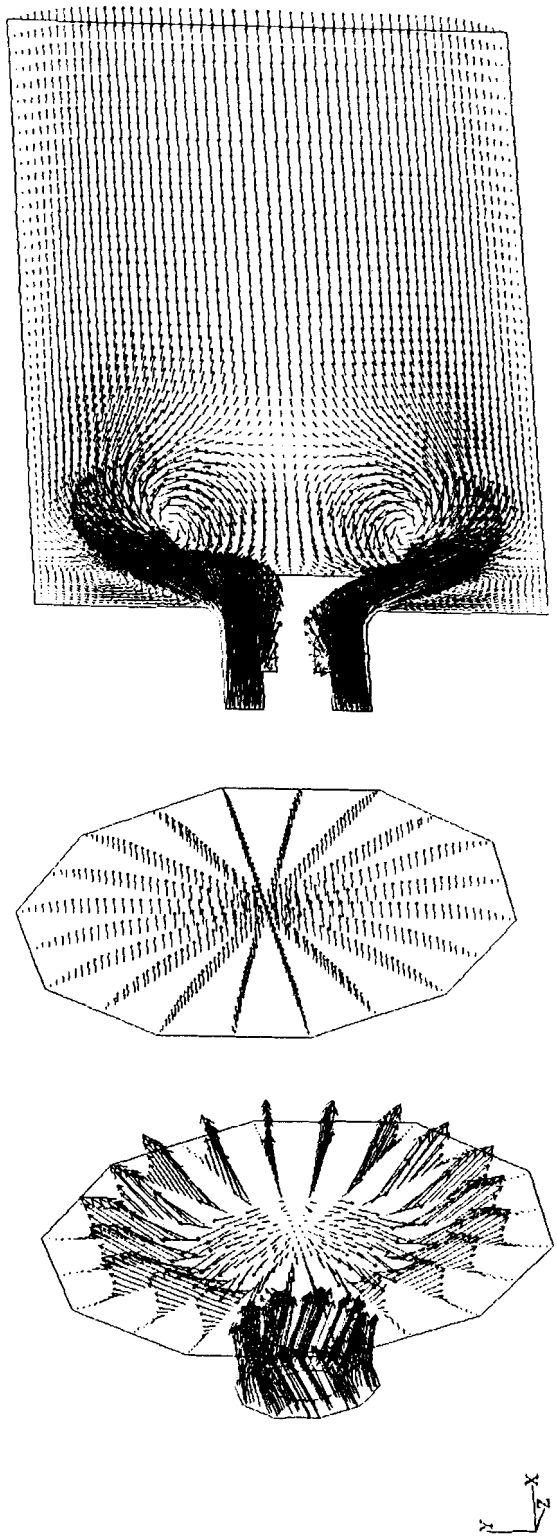

Figure 9. Perspective view of the 3D velocity vector field ($Re = 200$): left, in the axial planes $x = -0{\cdot}4$, $x = 0{\cdot}2$, $x = 2{\cdot}0$; right, in the symmetry plane $z = 0$

from four to zero. The total CPU time of this run is 217 min on an FX/80 with four ACEs and the code requires $5{\cdot}4 \times 10^6$ words of storage.

4.3. Flow around a combustor dome

The last application is the simulation of the flow around a simplified combustion chamber of a gas turbine engine. This type of application is supported by the gas turbine engine manufacturer SNECMA and we refer to Reference 2 for industrial applications with turbulence modelling.

4.3.1. 2D configuration. The geometry used for the 2D simulation is an idealized axisymmetric annular combustor. The FEM mesh has 1183 nodes and the boundary conditions used are given in Figure 10.

The velocity field calculated for a Reynolds number of 100 is drawn in Figure 11. The flow field is quite complex, with recirculation zones and multiple outlet sections.

4.3.2. 3D configuration. We then applied the 3D version of our code to simulate the flow in an annular combustor with 11 fuel nozzles. The 3D domain used for the simulation is drawn in Figure 12. We have limited the geometry to 1/11th of the total chamber by taking into account the symmetries. Dirichlet boundary conditions are applied on the inlet section and on each outlet section, and periodic boundary conditions are imposed on the two lateral planes.

The velocity vector field calculated for a Reynolds number of 100 is shown in Figure 13. This 3D simulation requires 100 time steps to obtain a stationary solution. The number of iterations for the pressure algorithm ranges from 17 at the beginning of the simulation to two for the last time steps. The number of CGS iterations varies from three to zero. The total CPU time of this run is 152 min on an FX/80 with four ACEs and the code requires $2{\cdot}8 \times 10^6$ words of storage.

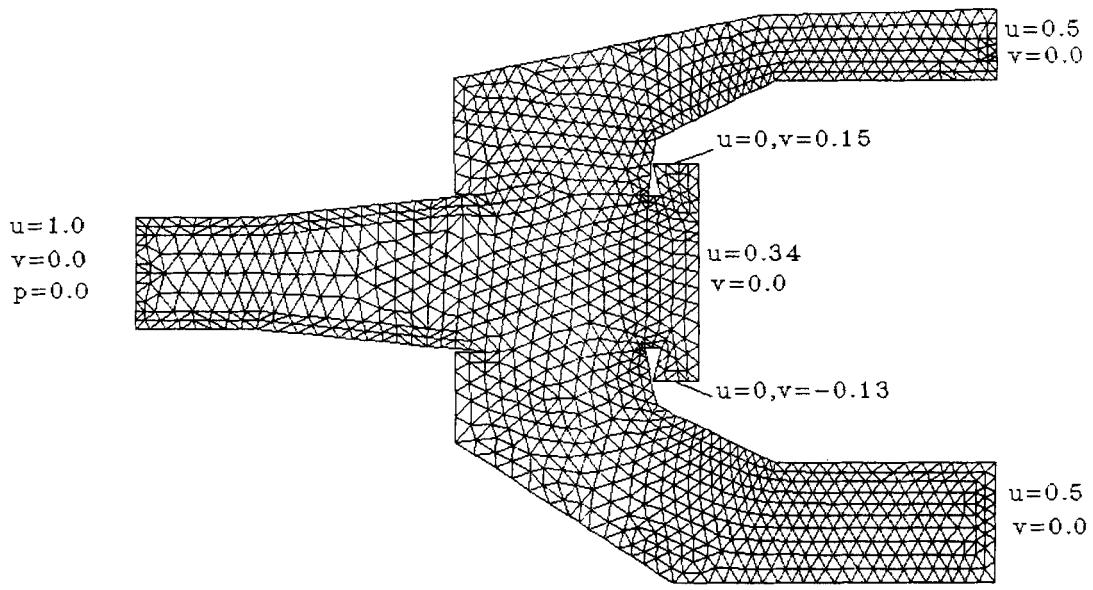

Figure 10. FEM mesh (1183 nodes) and boundary conditions; 2D domain around a combustor dome

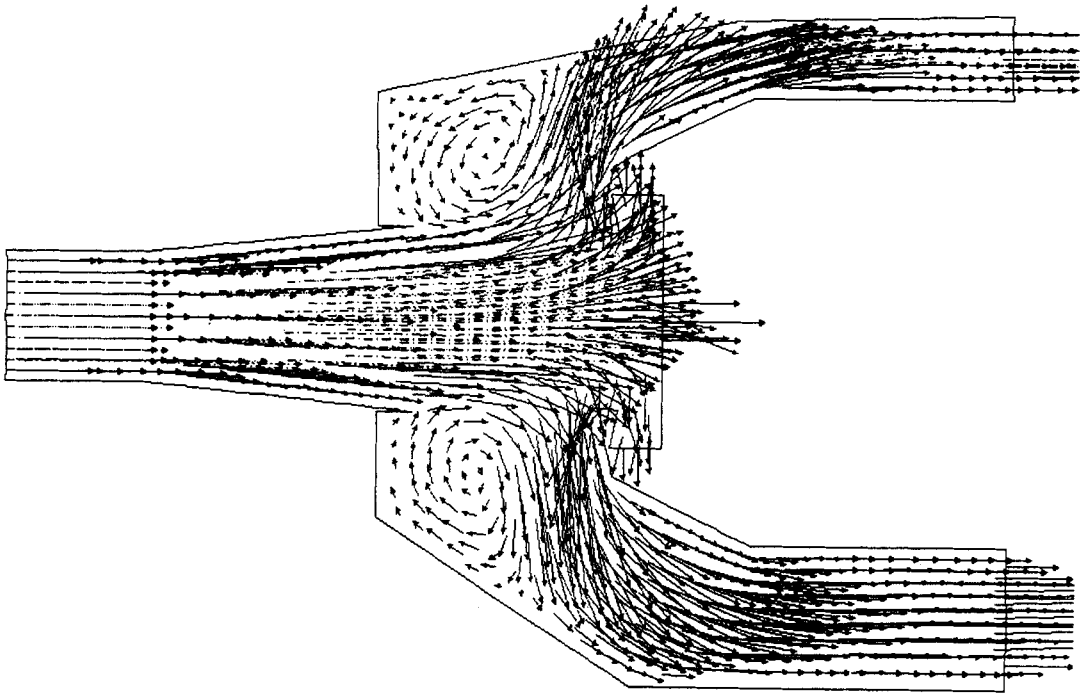

Figure 11. Velocity vector field around a combustor dome ($Re = 100$)

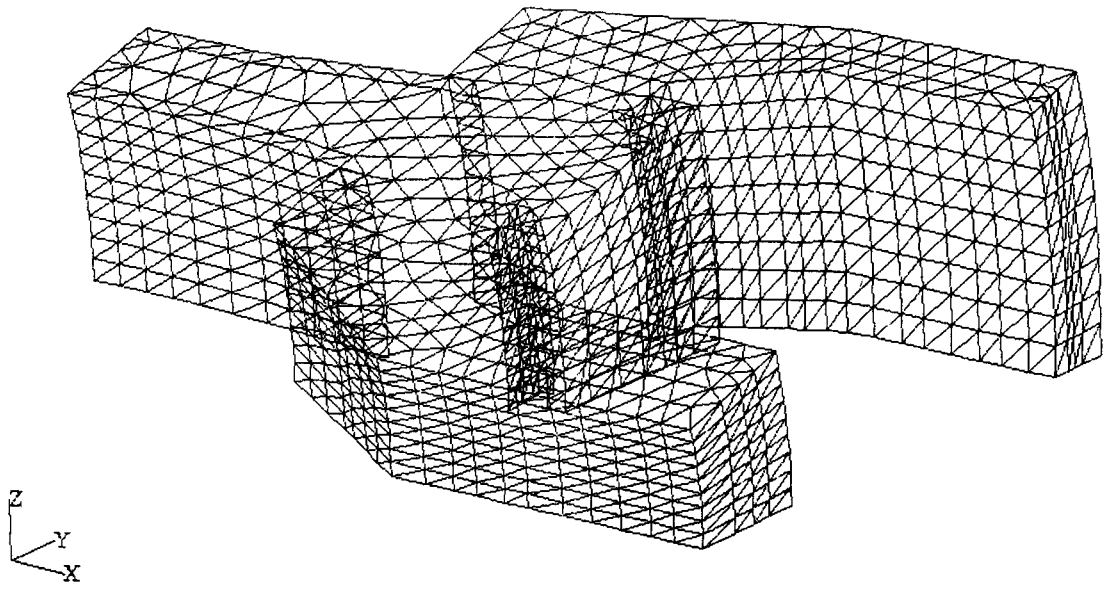

Figure 12. Perspective view of the 3D FEM mesh (22 515 nodes); 3D domain around a combustor dome

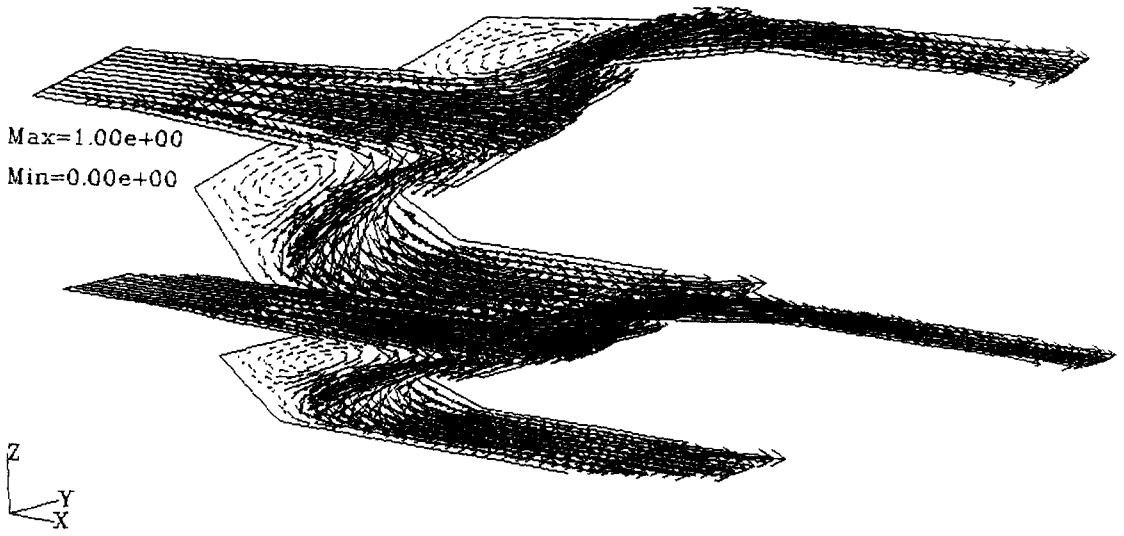

Figure 13. Perspective view of the 3D velocity field ($Re = 100$) around a combustor dome in the two symmetry planes

5. CONCLUSIONS

A numerical method based on a low-Mach-number approximation of the Navier–Stokes equations has been presented to solve subsonic flows in combustion engines. The numerical algorithm is based on a semi-implicit time discretization. A finite element approximation with a moving mesh has been introduced to deal with complex geometries and moving boundaries. A preconditioned Uzawa algorithm has been described for solving the coupled non-symmetric linear system on velocity and pressure. For the solution of the non-symmetric linear systems arising from the discretization, an efficient CGS algorithm with ILU preconditioning has been used. Efficient algorithms, such as the graph-colouring and level-scheduling algorithms, have been introduced to map the numerical method onto the new generation of parallel supercomputers. Using this technique, a global speed-up of 10 has been achieved on an Alliant FX/80 with eight ACEs. The presented numerical results have shown the ability of the method to simulate efficiently internal subsonic flows in 2D or 3D configurations. This numerical method has been successfully applied to predict the convective instabilities in liquid metals[27] and the turbulent flow inside combustion engines[1, 26] and gas turbine engines.[2] Further developments are nevertheless underway, particularly the introduction of a mixed finite volume/finite element formulation[28] as a robust upwind discretization of the convective terms at high Reynolds number, and the introduction of a second-order time integration via a predictor–corrector scheme.

ACKNOWLEDGEMENT

The author wishes to thank Professor D. Jeandel for his encouragement and is indebted to Mr. D. Henry and Mr. Y. Mao for performing the 2D calculations in the combustion chamber. The 3D calculations have been done on the Alliant computer of the Lyon Pilot Center PEPIT.

REFERENCES

1. G. Brun, M. Buffat and D. Jeandel, 'Flows characteristics in a port cylinder assembly predicted by a finite element method with turbulence modeling', *Proc. III Int. Symp. on Refined Flow Modeling and Turbulence Measurements*, Tokyo, 1988.
2. G. Brun, M. Buffat, D. Jeandel, J. L. Schultz and M. Desaulty, 'Finite element simulation of compressible turbulent flows. Validation and application to internal aerodynamic in gas-turbine engines', in T. J. Chung and G. R. Karr (eds), *Proc. VII Int. Conf. on FEM in Flow Problems*, Huntsville, AL, 1989, Huntsville Press, Huntsville, AL, 1989, pp. 1592–1597.
3. O. Pironneau, *Méthodes des Eléments Finis pour les Fluides*, Masson, Paris, 1988.
4. V. Girault and P. A. Raviart, *Finite Element Methods for the Navier–Stokes Equations, Springer Series SCM, Vol. 5*, Springer, Berlin, 1986.
5. M. O. Bristeau, R. Glowinski and J. Periaux, 'Numerical methods for the Navier–Stokes equations. Applications to the simulation of compressible and incompressible viscous flows', *Computer Physics Report, 6*, North-Holland, Amsterdam, 1987, pp. 73–187.
6. Y. M. Kim, J. L. Sohn and T. J. Chung, 'Calculations of the flow properties of a confined diffusion flame', in T. J. Chung and G. R. Karr (eds), *Proc. VII Int. Conf. on FEM in Flow Problems*, Huntsville, AL, 1989, Huntsville Press, Huntsville, AL, 1989, pp. 304–309.
7. D. R. Chenoweth and T. Paolucci, 'Natural convection in an enclosed vertical air layer with large horizontal temperature differences', *J. Fluid Mech.*, **169**, 173–210 (1986).
8. M. Fortin and R. Glowinski, *Méthodes de Lagrangien Augmentee*, Masson, Paris, 1982.
9. J. Donea, 'An arbitrary Lagrangian–Eulerian finite element method for transient dynamic fluid–structure interactions', *Comput. Methods Appl. Mech. Eng.*, **33**, 689–723 (1982).
10. M. Bercovier and O. Pironneau, 'Error estimates for finite element method solution of the Stokes problem in the primitive variables', *Numer. Math.*, **33**, 211–224 (1979).
11. R. J. Gelinas, S. K. Doss and K. Miller, 'The moving finite element method: application to general PDEs with multiple large gradients', *J. Comput. Phys.*, **40**, 202–249 (1981).
12. D. Jeandel, M. Buffat and P. Carriere, 'Numerical methods for 2D and 3D heat transfer calculations', *Introduction to Numerical Solution of Industrial Flows, VKI Lecture Notes Series, 86–07*, 1986.
13. P. Lascaux and R. Theodor, *Analyse Numérique Matricielle Appliquée à l'Art de l'Ingénieur, Vol. 2*, Masson, Paris, 1987.
14. J. Cahouet and J.-P. Chabard, 'Some fast 3D finite element solvers for generalized Stokes problem', *Int. j. numer. methods fluids*, **8**, 869–895 (1988).
15. O. Axelsson and V. A. Barker, *Finite Element Solutions of Boundary Value Problems*, Academic Press, New York, 1984.
16. W. Hackbush and U. Trottenberg, *'Multigrid Methods, Lecture Notes in Mathematics, Vol. 960*, Springer, Berlin, 1982.
17. G. Radicati, Y. Robert and S. Succi, 'Iterative algorithms for the solution of non-symmetric systems in the modelling of weak plasma turbulence', *J. Comput. Phys.*, **80**, 489–497 (1989).
18. P. Sonneveld, P. Wesseling and P. M. Zeeuw, 'Multigrid and conjugate gradient methods as convergence acceleration techniques', in D. J. Paddon and M. Holstein (eds), *Multigrid Methods*, Bristol, 1983, Oxford University Press, Oxford, 1985, pp. 117–167.
19. Y. Saad and M. H. Schultz, 'GMRES: a generalized minimal residual algorithm for solving non symmetric linear systems', *SIAM J. Sci. Stat. Comput.*, **7**, 856–869 (1986).
20. P. Joly, 'Méthodes de gradients conjugués', *Publication du Laboratoire d'Analyse Numerique, No. 84016*, Université Pierre et Marie Curie (Paris VI), 1984.
21. P. Carriere and D. Jeandel, 'A 3D finite element method for the simulation of thermoconvective flows and its performances on a vector parallel computer', to appear in *Int. j. numer. methods fluids.*
22. H. P. Langtangen, 'Conjugate gradient methods and ILU preconditioning of non-symmetric matrix systems with arbitrary sparsity patterns', *Int. j. numer. methods fluids*, **9**, 213–233 (1989).
23. M. Buffat, 'Code NADIA: méthode numérique, vectorisation et parallelisation sur Alliant FX/80', *Laboratoire de Mécanique des Fluides, ECL, Internal Report*, 1989.
24. G. Johnson, 'Parallel processing for fluid dynamics applications', in T. J. Chung and G. R. Karr (eds), *Proc. VII Int. Conf. on FEM in Flow Problems*, Huntsville, AL, 1989, Huntsville Press, Huntsville, AL, 1989, pp. 1474–1479.
25. F. Angrand and J. Erhel, 'Vectorized FEM codes for compressible flows', *Proc. VI Int. Conf. on FEM in Flow Problems*, Antibes, 1986, edt by INRIA, 1986, pp. 307–311.
26. M. Buffat, H. Haminh, D. Henry, D. Jeandel, A. Kourta, Y. Mao and Vandromme, 'Modélisation et simulation de l'aérodynamique interne des moteurs', *Colloq. Bilan PRDTTT*, Paris, 23–26 January 1989.
27. D. Henry and M. Buffat, 'Two and three dimensional study of convection in low Prandtl number fluids', Numerical Simulation of Oscillatory Convection in Low Prandtl Number Fluids, *Notes on Numerical Fluid Mechanics*, **27**, 182–188 (1990).
28. A. Dervieux, 'Steady Euler simulations using unstructured meshes', *VKI Lecture Notes Series, 84–04*, 1985.