Simulation of a Compressible Flow by the Finite Element Method Using a General Parallel Computing Approach

André Chambarel[1] and Hervé Bolvin[1]

[1] Complex Hydrodynamics Laboratory, Faculté des Sciences, 33 rue Louis Pasteur
F-84000 Avignon, France
andre.chambarel@univ-avignon.fr

Abstract. We have developed a coherent set of techniques for parallel computing. We have used the Finite Element Method associated with the C++ Object-Oriented Programming with only one database. A technique of data selection is used in the determination of the data dedicated to each processor. This method is performed by SIMD technology associated with MPI capabilities. This parallel computing is applied to very large CPU cost problems particularly the unsteady problems or steady problems using iterative methods. Different results in Computational Fluid Dynamics are presented.

1 Introduction

In Computational Fluid Dynamics (CFD) the transient flows generally request a very large memory and CPU time [1]. Generally we obtain this results in a high cost calculus because of the step-by-step process. In this paper we will present a parallel computing method for CFD problems by a Finite Element approach [2]. We propose a coherent method for easy implementation including the following key words:

- Finite Element Method with C++ Object-Oriented Programming code,
- Selection data technique, matrix-free technique and iterative method.

We have developed an easy method for parallel computing which seems to be a natural way of performing intensive computation. Our purpose is to carry out parallel algorithms without modifying the object structure of the solvers, and the data structure [3]. To answer this requirement, we use a selected data method resulting in suitable load balancing with the choice of lists of elements. This technique is independent from the geometry, and can be applied to general cases. This new concept is a natural way for the standardization of parallel codes. In fact, parallelization is here applied to the resolution of a large sized differential system by a semi-implicit algorithm associated with a matrix-free technique.

Among different hardware concepts the SIMD — Single Instruction Multiple Data — architecture has proved to be the most promising for parallel computers. This technology is used for high performance computing especially when problems such as solving large sets of differential equations are dealt with [4]. A SIMD parallel computer consists of a set of processors connected with a fast communication network. Each processor performs the same program with different data. In our work the different data are obtained from a single file and each processor selects its dedicated data. For parallel programming we use the MPI — Message Passing Interface — library.

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2329, pp. 920–929, 2002.

2 Mathematical Model

In CFD all types of cases of transient compressible flows and oscillating flows can be found. The latter case is obtained by increasing the velocity or the Reynolds number. Consequently the nozzle and its jet are often studied separately. The aim of this paper is to propose a numerical study of the transient flow in a convergent nozzle and its jet, incorporating in particular the unstable solutions.

The Navier-Stokes equations are the mathematical model. The molecular Reynolds number is approximately 10^6. The Reynolds number value Re is based on the diameter of the nozzle outlet and the sound velocity. We have here chosen a Reynolds number of 100, coherent with a zero equation turbulent model [5].
Using the usual notation, the dimensionless Navier-Stokes equations are as follows:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho . u_j\right) = 0$$

$$\rho . \left(\frac{\partial u_i}{\partial t} + u_j . \frac{\partial u_i}{\partial x_j} \right) = -\frac{1}{\gamma} . \frac{\partial}{\partial x_i}\left(\rho . T\right) + \frac{\partial}{\partial x_j}\left[\frac{1}{\mathrm{Re}} . \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3.\mathrm{Re}} . \delta_{ij} . \left(\frac{\partial u_k}{\partial x_k} \right) \right]$$

$$\rho . \left(\frac{\partial T}{\partial t} + u_j . \frac{\partial T}{\partial x_j} \right) = -(\gamma - 1) . \rho . T . \frac{\partial u_j}{\partial x_j} + \frac{\partial}{\partial x_j} \left(\frac{\gamma}{Re . \mathrm{Pr}} . \frac{\partial T}{\partial x_j} \right) + F(u_i)$$

$$with \quad F(u_i) = \gamma . (\gamma - 1) . \left[\frac{1}{\mathrm{Re}} . \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)^2 - \frac{2}{3.\mathrm{Re}} . \left(\frac{\partial u_j}{\partial x_j} \right)^2 \right] \tag{1}$$

The domain of integration (Ω) is presented in figure 1. The initial values are the normal thermodynamical conditions and the gas is motionless [5]. The boundary conditions are the following:

- at the inlet of the nozzle, we simulate the start of a turbo-engine so the time-dependent pressure is as follows:

$$0 \le t \le t_0 : p = p_0 + p_{\max} . \frac{t}{t_0} \quad and \quad t \ge t_0 : p = p_0 + p_{\max}$$

- in the nozzle, a wall condition for the velocity and adiabatic conditions for the temperature are imposed,
- in the free space boundary, an outflow condition is used.

With the finite element formulation the following matricial form is obtained:

$$\sum_{i=1}^{ne} < \delta U_i > . \left([m_i] . \frac{\partial}{\partial t} \{U_i\} - \{f_i\} + [k_i] . \{U_i\} \right) = 0 \tag{2}$$

After the assemblage process, we built the following differential system:

$$[M].\frac{d}{dt}\{U\} = \{F\} - [K].\{U\} \tag{3}$$

We built a grid with 10392 nodes and 20346 elements. The differential system above is composed of approximately 40,000 equations.

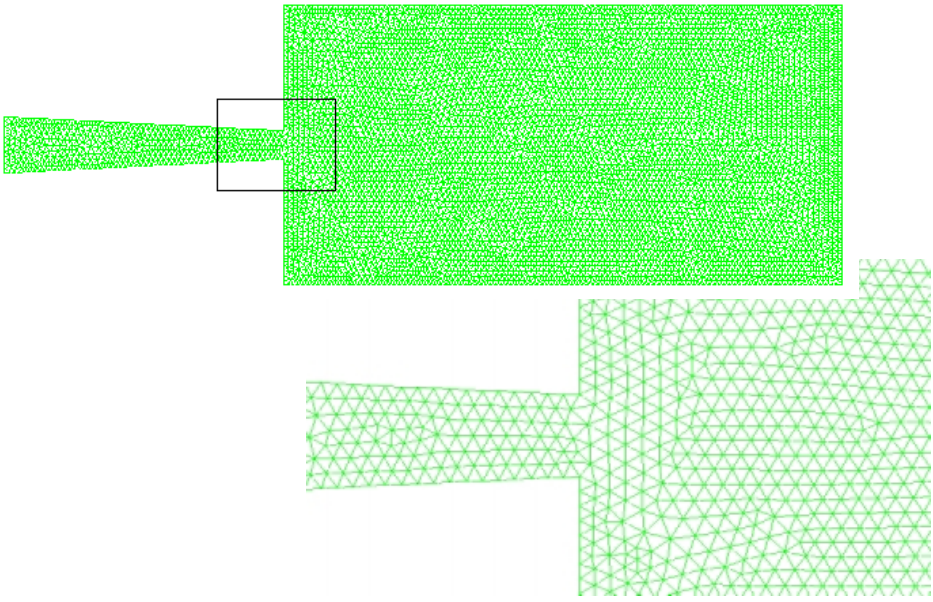

Fig. 1. Finite element grid.

3 Structure of Code

Figure 2 shows the general structure of the compact code. It is organized in three classes corresponding to the functional blocks of the Finite Element Method's different stages. With these classes we built three objects that are connected by a single heritage. So the transmission of the parameters between the objects is defined by a list technique.

We use efficient C++ Object-Oriented Programming for the Finite Element code called FAFEMO (Fast Adaptive Finite Element Modular Object) [4]. In practice, three objects compose a solver and they are separated in three source files. When we built a solver, we merge the three source files of each object. This process is performed by a software called FEOM (Finite Element Object Manager). This technology allows the implementation of very low sized solvers. In our examples their sizes are about 31 Kb — 900 C++ lines — Each solver is dedicated to a problem and can be considered as an element of an algebraic structure. In fact the set of solvers is organized into a valued graph [3].

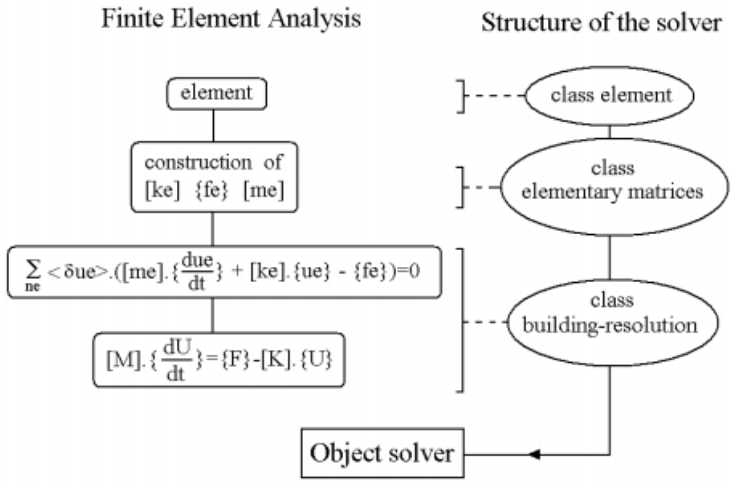

Fig. 2. Object structure of a standard solver.

4 Method of Parallel Computing

4.1 Principle of Parallelization

The principal CPU cost corresponds to the elementary matrices computing and secondarily to the time step updating. In the case of an unsteady problem, the analytical discretization of the problem with the Finite Element Method is given by the following scalar product [6]:

$$\sum_{NE} \langle \delta ue \rangle.([me]\left\{\frac{due}{dt}\right\} + [ke]\{ue\} - \{fe\}) = 0 \text{ with } NE = 1 .. ne \tag{4}$$

If p is the number of processors, we select a list of elements Nk by an expert system called AMS (Automatic Multigrid System) described above:

$$\bigcup_{k=1}^{p} Nk = NE \quad \text{and} \quad Ni \cap Nj = 0 \quad \text{for } i \neq j \tag{5}$$

Each elementary matrix can be assembled into a global matrix by classical Finite Element process [6]. Each processor computes its part of the global matrices:

$$\sum_{k=1}^{p} [Mk] = [M] \; mass\ matrix \qquad \sum_{k=1}^{p} \{\Psi k\} = \{\Psi\} \; residuum \tag{6}$$

In this case the Bernstein conditions are verified [1]. So we have a correct load balancing if the lists size of elements is similar for each processor. The communications between the processors exist only at the end of the time step. Each

processor builds his part of the differential system and algorithm below allows the updating of solution {U}. A semi-implicit algorithm is presented [4] [7] :

$$
\begin{array}{l}
t_n = 0 \\
\textit{while} \quad (t_n \le t_{\max}) \\
\quad \left\{ \begin{array}{l} \textit{for } j = 1 \textit{ to } p \quad \left\{\Delta U_n^i\right\}_j = \Delta t_n . \left[M_n^i\right]^{-1} . \left\{\Psi_j \left(U_n + \alpha . \Delta U_n^{i-1}, t_n + \alpha . \Delta t_n\right)\right\} \\ \qquad\qquad i = 1, 2, \ldots \quad \textit{until} \quad \left\|\Delta U_n^i - \Delta U_n^{i-1}\right\| \le \textit{tolerence} \end{array} \right\} \\
\quad \left\{U_{n+1}\right\} = \left\{U_n\right\} + \left\{\Delta U_n\right\} \\
\quad t_{n+1} = t_n + \Delta t_n \\
\textit{end} \quad \textit{while}
\end{array}
$$

4.2 Technique of Parallelization.

Each solver is endowed with a capability called AMS (Automatic Multigrid System). It is an expert system with several possibilities. The applications of this capability are very large, i.e. multiprocessor computing (in this paper), wave front, multi-domain calculus, multi-grid simulation, ...
According to the problem, the AMS expert system can choose different analytical or geometrical components (Fig. 3).
In the case of parallel computing the AMS expert system chooses here the elements dedicated to each processor for the sharing of the scalar product (2) .

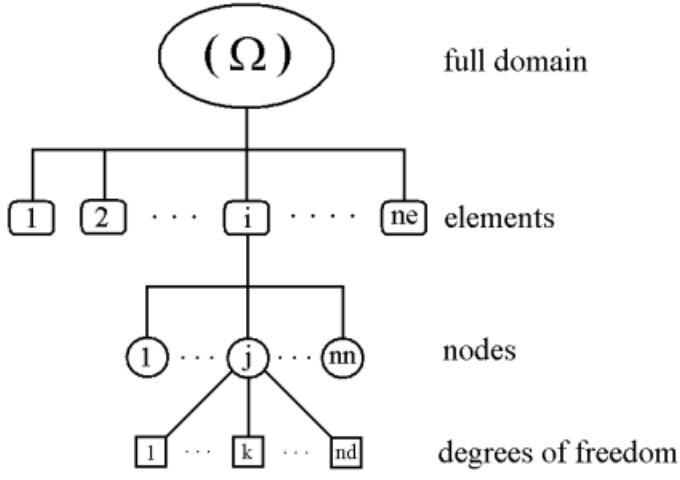

Fig. 3. Taxinomy of the Finite Element parameters.

In fact we can summarize the principal stages of parallelization:

- A first stage consists in creating an expert system for the selection of the data for each processor.

- In the second stage each processor calculates its elementary matrix without communication.
- In the third stage each secondary processor sends its assembled elementary matrices to the principal processor, so that the solution may be updated.

5 Application

Nevertheless, no communications are required between the processors. Each of them performs a completely independent computation for each iteration. This is particularly well adapted to the object structure of the solver. The SIMD architecture is used for the parallel computing management [8]. The AMS capabilities select the data for each processor. The corresponding software is developed with the MPI-C++ library.

Table 1. Parallel computing efficiency.

Equations	CPU time	Nb. of processors	Speed Up (%)
40,000	22 h 28 mn	1	---
40,000	12 h 20	2	87 %

Table 1 presents the parallel efficiency for a CFD problem [9]. We notice that the parallel solver is almost the same as the one used in a sequential process. These examples are performed on a 2 processor-PC. The different examples of parallel computing are applied to unsteady CFD problems. Different cases of compressible flow are presented in figures 4, 5 and 6. Figure 5a presents a qualitative comparison between the experiment and the numerical simulation [10] .

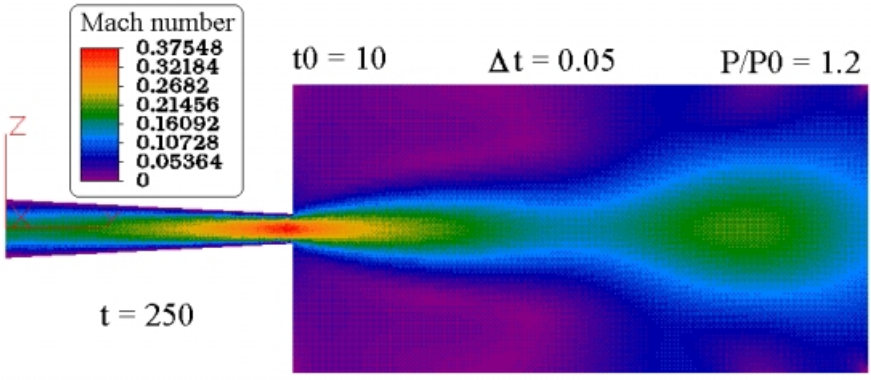

Fig. 4a. Transient compressible flow.

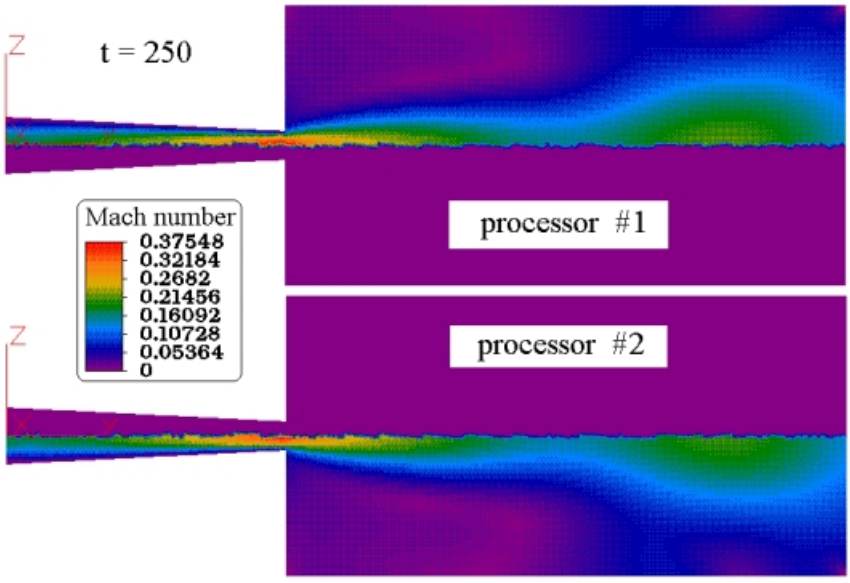

Fig. 4b. Computing with two processors.

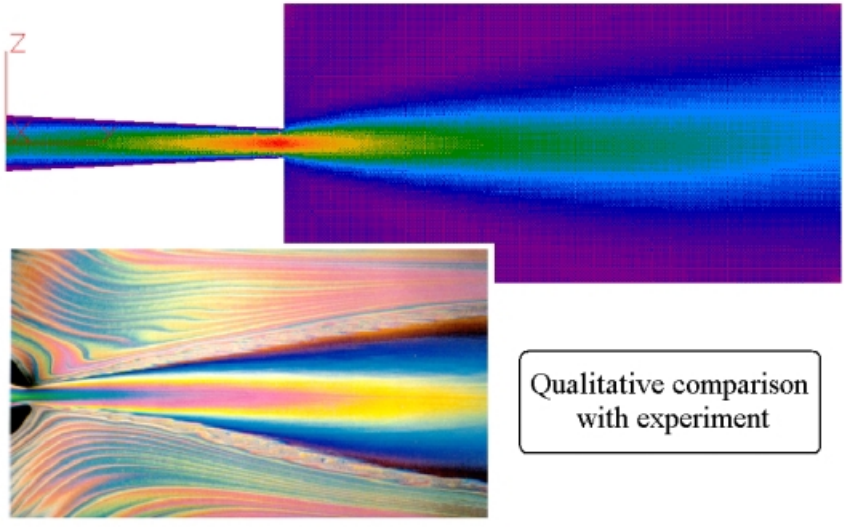

Fig. 5a. Permanent jet.

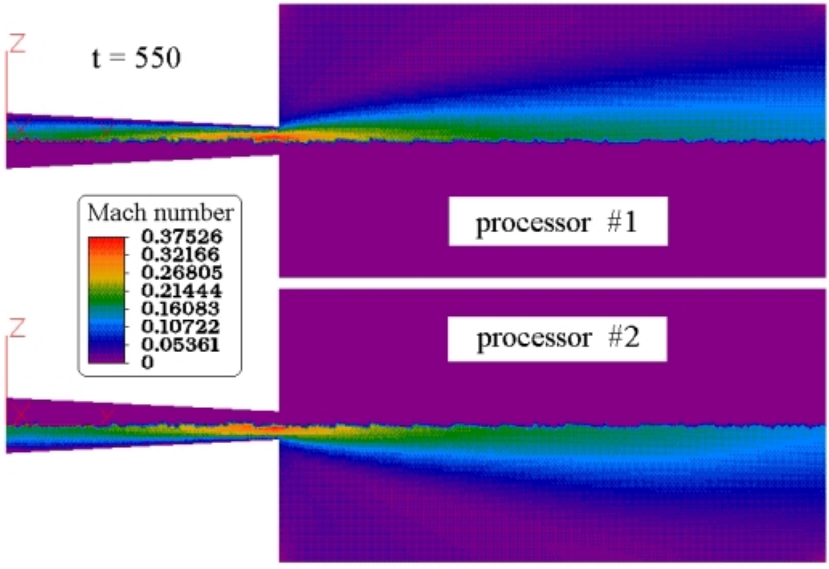

Fig. 5b. Computing with two processors.

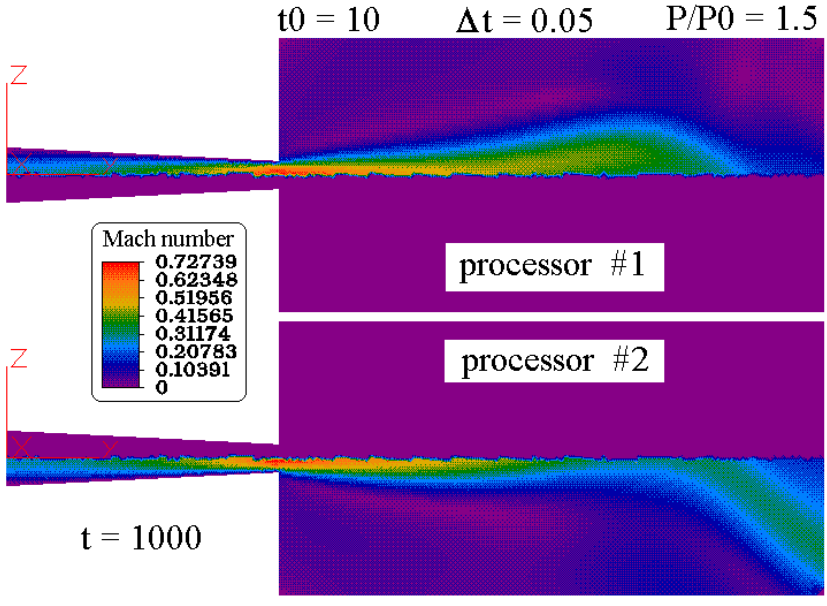

Fig. 6a. Two processor computing of an oscillating jet.

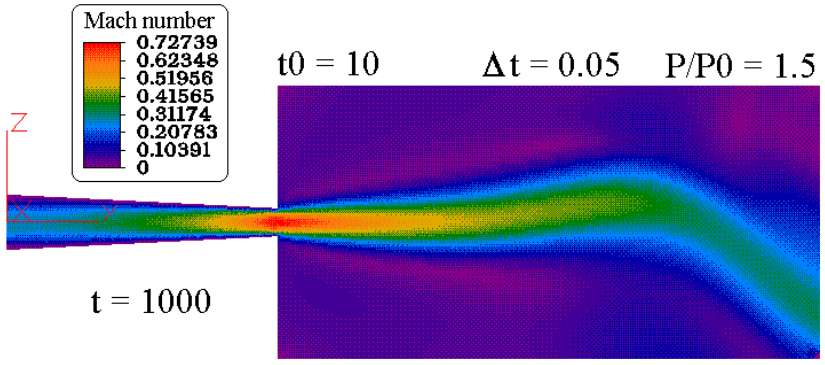

Fig. 6b. Oscillating jet.

If the pressure ratio is greater than a limit value (1.28 here) we obtain an oscillating jet. Figure 6 shows an asymmetrical position of the jet during the oscillation.

6 Conclusion

An easy method of parallel computing for engineering problems is proposed. It consists in using a coherent set of techniques including :

- The Finite Element Method,
- C++ Object-Oriented Programming by FAFEMO software,
- A selection data technique by AMS expert system,
- Matrix-free algorithms.

In this context the implementation of the low sized solvers concerned is very easy. The SIMD architecture associated with the MPI-C++ library is used. So we have an efficient method for the parallelization of differential systems coming from the Finite Element Method. The performances are interesting [11]. We particularly notice the low sized memory and the good load balancing. Different examples in Computational Fluid Dynamics are presented.

References

1. Yeckel, A., Smith, J.W., Derby, J.J.: Parallel finite element calculation of flow in a three dimensional lid-driven cavity using the CM-5 and T3D. Int. J. Num. Methods in Fluids, Vol. 24 (1997) 1449-1461.

2. Chambarel, A., Bolvin, H.: Application of the parallel computing technology to a wave front model using the Finite Element method. Lecture Notes in Computer Science, Vol. 2127, Springer-Verlag (2001) 421-427.

3. Chambarel, A., Onuphre, E.: Finite Element software based on Object Programming. International Conference of the twelfth I.A.S.T.E.D., Annecy France, May 18-20, 1994.

4. Chambarel, A., Ferry, E.: Finite Element formulation for Maxwell's equations with space dependent electric properties. Revue européenne des Eléments Finis, Vol. 9, n° 8 (2000) 941-967.

5. Gutmark, E., Schadow, K.C., Bicker, C.J.: Near acoustic field and shock structure of rectangular supersonic jet. A.I.A.A. Journal, Vol. 28 (1990) 1163-1170.

6. Dhatt, G., Touzot, G.: Une présentation de la méthode des éléments finis. Editions Maloine S.A., Paris (1981).

7. Gresho, P.M.: On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. Int. J. Numer. Meth.Fluids, Vol. 11 (1990) 621-659.

8. Hempel, R., Calkin R., Hess, R., Joppich, W., Keller, U., Koike, N., Oosterlee, C.W., Ritzdorf, H., Washio, T., Wypior, P., Ziegler, W.: Real applications on the new parallel system NEC Cenju-3. Parallel Computing, Vol. 22 (1996) 131-148.

9. Chambarel, A., Fougère, D.: A general parallel computing approach using the Finite Element method and the object-oriented programming by selected data technique. 6th International Conference, PACT 2001, Novosibirsk, Russia, September 3-7, 2001.

10. Gharib, M., Derango, P.:Flow studies of a two-dimensional flow. Physics of fluids, Vol. 31, n°9 (1998) 2389-2394.

11. Laevsky, Y.M., Banushkina, P.V., Litvinenko, S.A., Zotkevich, A.A.: Parallel algorithms for non-stationary problems: survey of new generation of explicit schemes. Lecture Notes in Computer Science, Vol. 2127, Springer-Verlag (2001) 442-446.