INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2004; **45**:1123–1136 (DOI: 10.1002/fld.732)

Parallel-in-time simulation of the unsteady Navier–Stokes equations for incompressible flow

J. M. F. Trindade[1,‡] and J. C. F. Pereira[2,*,†]

[1]*Departamento de Máquinas Marítimas, Escola Náutica Infante D. Henrique, Portugal*
[2]*Department of Mechanical Engineering, Instituto Superior Técnico, Universidade Técnica de Lisboa, LASEF, Av. Rovisco Pais, 1049-001 Lisboa, Portugal*

SUMMARY

In this paper, we present an application of a parallel-in-time algorithm for the solution of the unsteady Navier–Stokes model equations that are of parabolic–elliptic type. This method is based on the alternated use of a coarse global sequential solver and a fine local parallel one. A standard finite volume/finite differences first-order approach is used for discretization of the unsteady two-dimensional Navier–Stokes equations. The Taylor vortex decay problem and the confined flow around a square cylinder were selected as unsteady flow examples to illustrate and analyse the properties of the parallel-in-time method through numerical experiments. The influence of several parameters on the computing time required to perform a parallel-in-time calculation on a PC cluster was verified. Among them we have analysed the influence of the number of processors, the number of iterations for convergence, the resolution of the spatial domain and the influence of the time-step sizes ratio between the coarse and fine grids. Significant computer time saving was achieved when compared with the single processor computing time, particularly when the spatial dimension of the problem is low and the temporal scale is large.

KEY WORDS: parallel-in-time; parareal; Navier–Stokes; finite volume

1. INTRODUCTION

Unsteady fluid flow phenomena are important for a wide range of engineering problems and tools such as numerical simulation plays a vital role in providing solutions. For some applications, such as feedback control processes, it would be beneficial to obtain solutions faster then real time. However, even without real time applications in mind, reducing the

*Correspondence to: J. C. F. Pereira, Department of Mechanical Engineering, Instituto Superior Técnico, Universidade Técnica de Lisboa, LASEF, Av. Rovisco Pais, 1049-001, Lisboa, Portugal.
†E-mail: jose@navier.ist.utl.pt
‡E-mail: jorgetrindade@enautica.pt

Published online 14 May 2004
Received 11 July 2003
Revised 7 March 2004

computational cost to solve unsteady flow problems is always beneficial as this makes it possible to study increasingly larger and more complex problems.

Using multiple processors in parallel computing is the most effective way to significantly speed-up the solution. A low cost PC cluster can be assembled when supercomputers are unavailable. The PC cluster can be considered as a message-passing architecture parallel machine that provides communications between processors as explicit I/O operations, with the limitation that these communications are much slower than the processor. The usual approach for setting up parallel CFD calculations is to divide up the domain between processors using the spatial domain decomposition [1, 2]. However, it is well known that domain decomposition techniques are not efficient when the spatial dimension of the problem is small and a large number of processors are used. For such problems the communication costs are very significant because data at sub-domain boundaries needs to be exchanged between processors, frequently several times, at each time iteration. Although some overlapping between computation and communication is possible, parallel efficiency and speed-up are drastically reduced.

The number of nodes of parallel computers will naturally increase in the future having as limit the largest number of linked computers which is ultimately the World Wide Web. For such scenario, a problem considered large nowadays may become a small one if a very large number of computer nodes is available. For unsteady problems, one possible way to fully exploit the large number of nodes available in the future is the time domain decomposition. Recently, new algorithms have been proposed to parallelize the temporal evolution of a parabolic system of equations. These new parallel methods are called *parareal* because the main goal on their initial development was the real time solution of a problem using a parallel structure [3]. The basics of the technique was devised in order to be potentially applied to the solution of the Navier–Stokes equations [4]. Some modifications of the original *parareal* algorithm have been introduced to obtain better stability and performance, see Reference [5]. An application of this algorithm to carry out molecular-dynamics simulations was recently presented [6].

The method is based on the alternated use of coarse global sequential solvers with fine local parallel ones. Calculation starts with a sequential solution along the time domain of the problem on a coarse time-grid and is followed by an iterative procedure using the coarse time-grid and a finer one. The temporal evolution on the finer time-grid is calculated in parallel. This iterative procedure provides successive corrections for the problem solution.

Some problems may emerge from the use of two temporal grids in this predictor–corrector fashion when solving the Navier–Stokes equations. Major concerns are related with the stability and accuracy of the method. Another issue that requires some attention is the optimal solution of the time integration method used on each time-grid.

The objective of present work is the application of the modified algorithm [5] to the solution of the unsteady Navier–Stokes equations and to evaluate its properties through numerical experiments on a PC cluster.

The remainder of this paper is organized as follows. Section 2 presents the parallel-in-time numerical method. Section 3 is devoted to the presentation of the numerical experiments comprising the temporal evolution of the so-called Taylor vortex and the confined flow around a square cylinder. The paper ends with a summary of the main conclusions highlighting the potential of the method.

2. NUMERICAL METHOD

For an incompressible Newtonian fluid and unsteady flow, the governing continuity and Navier–Stokes equations are integrated in each finite control volume and after application of Gauss theorem read as

$$\int_S \mathbf{v} \cdot \mathbf{n}\, \mathrm{d}S = 0 \tag{1}$$

$$\frac{\partial}{\partial t}\int_\Omega u_i\, \mathrm{d}\Omega + \int_S u_i \mathbf{v} \cdot \mathbf{n}\, \mathrm{d}S = \int_S \gamma\, \mathrm{grad}\, u_i \cdot \mathbf{n}\, \mathrm{d}S - \frac{1}{\rho}\int_S p\mathbf{i} \cdot \mathbf{n}\, \mathrm{d}S \tag{2}$$

where Ω is the volume and S is the surface of an arbitrary control volume, $\mathbf{n}$ is the unit vector normal to S and directed outwards, $\mathbf{v}$ is the velocity vector, u_i are the cartesian velocity components, ρ is the density, γ is the viscosity and p is the pressure.

Equations (1) and (2) are discretized with second-order central differences on a finite volume staggered uniform mesh together with the Euler implicit temporal discretization scheme. After discretization Equation (2) reads

$$\frac{u^{n+1} - u^n}{\Delta t}\Omega + \sum_{i \in S} C_i^{n+1} - \sum_{i \in S} D_i^{n+1} = -G^n \tag{3}$$

where C_i^{n+1} and D_i^{n+1} stands for the convective and diffusive fluxes evaluated at the time level $n+1$ and G represents the pressure source term. The SIMPLE [7] method is used to correct the velocity and pressure fields during each time iteration.

The fully explicit, first-order temporal discretization, counterpart of equation (3) is obtained by the evaluation of all the fluxes at time level n. For the explicit formulation, the stability constraints dictate that the Courant number, $Cu = u\delta t/\Delta x$, and the diffusive parameter, $r = \gamma \delta t(1/\Delta x^2 + 1/\Delta y^2)$, are required to be less than 1 and 0.5, respectively. In addition, the Peclet number should be less than 2. At each time step a projection method is required to enforce a divergence free velocity field. A Poisson equation,

$$\mathrm{div}(\mathrm{grad}\, \phi) = \frac{1}{\Delta t}\, \mathrm{div}(\mathbf{v}) \tag{4}$$

is solved and ϕ is used to update the velocity and pressure fields.

The standard domain decomposition method to solve the implicit, or explicit, form of Equation (3) decomposes the solution domain into smaller regions, each one assigned to a processor, on which the governing equations are solved in parallel. The interaction between the sub-domains occurs only at boundaries.

The new parallel-in-time algorithm, [5], is based on the iterative use of coarse global sequential solvers (or integrators) with fine local parallel ones, allowing the time domain decomposition and the propagation of solution jumps on the coarse time-grid. The time interval $[0, T]$ of the problem under consideration is decomposed into a sequence of P (number of processors) sub-domains of size $\Delta t = T/P$, which will be called coarse time-grid. The integrator on the coarse time-grid employs an interval Δt and the integrator on the finer time-grid uses a smaller time-step size given by $\delta t = T/M$, for some integer M. Denoting by u_0 the initial velocity field and by $(u_1, \ldots, u_P)$ the successive velocity fields, at $T^p = p\Delta t$, where p

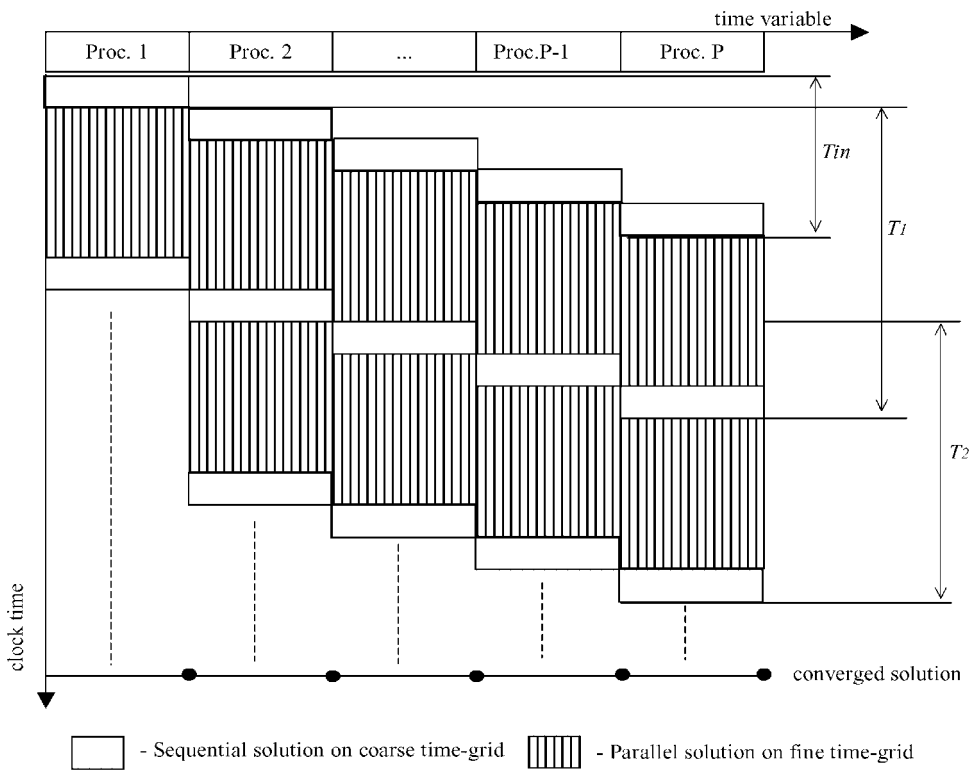


Figure 1. Parallel-in-time algorithm diagram.

is the processor number, the solution proceeds as follows:

(i) *Initialization*: A coarse time-grid solution is obtained sequentially. Each processor solves the spatial field for a single time-step, Δt,

$$\begin{aligned} u_p^0 &= G_{\Delta t}(u_{p-1}^0) \\ u_0^0 &= u_0 \end{aligned} \tag{5}$$

for all processors, $p = 1, 2, \ldots, P$.
The operator $G_{\Delta t}$ corresponds to the implicit solution of the momentum equations during a time step Δt.

The diagram in Figure 1 describes the parallel-in-time methodology. The time variable is represented in abscissas and in ordinates the computing time. Firstly, the number of processors divides the time domain dictating the coarse time-grid resolution. When a small number of processors are used, in the present case up to 16 processors, the integration method, using the coarse time-grid, should be implicit to verify stability constraints. The Peclet number must not be too large in regions of strong velocity gradient to avoid oscillatory solutions. So, Equation (5) stands for the implicit solution of the momentum equations over a single time-grid interval (T/P) requiring the

computing time T_{in}. This sequential solution corresponds to a coarse grain solution that requires correction, provided by the iterative scheme that follows.

(ii) *Iterative procedure*: The initial solution obtained previously is used to start an iterative procedure. The time-step in the finer time-grid, δt, can be small enough to select an explicit method for the solution of the momentum equations (3)

$$\begin{aligned} y_p^k &= F_{\delta t}(u_{p-1}^{k-1}) \\ u_0^k &= u_0 \end{aligned} \tag{6}$$

for $1 \leqslant p \leqslant P$ and $k \geqslant 1$. The operator $F_{\delta t}$ denotes the operator corresponding to the fully explicit solution of the momentum equations together with the Poisson equation. Equation (6) translates the parallel solution of the momentum equations on the finer time-grid, in which each processor calculates the flow solution for M/P time-steps of size δt. The integrator scheme corresponds to a fully explicit temporal discretization of the momentum equations due to the small time-step size.

This step corresponds in Figure 1 to the parallel solution in P processors in which each processor solves M/P time increments from T^{p-1} to T^p.

Completed the parallel solution on the finer time-grid, the solution jumps at each T^p are calculated by each processor, according to the difference between the new solution calculated on the finer time-grid and the solution on the coarse time-grid at the previous iteration

$$S_p^k = y_p^k - u_p^{k-1} \tag{7}$$

Finally, a new sequential solution is calculated. For $1 \leqslant p \leqslant P$ a solution is predicted using the coarse time-grid solver

$$\tilde{u}_p^k = G_{\Delta t}(u_{p-1}^k) \tag{8}$$

and corrected by the solution jumps, as introduced in Reference [5]

$$u_p^k = \tilde{u}_p^k + \sum_{l=1}^{k} S_p^l \tag{9}$$

Figure 1 displays two iterations of the cycle. The computing time required to perform the first iteration is denoted by T_1.

One should note that after the first iteration the solution at time $t = T^1$ corresponds to the final solution at this time level. More generally, at iteration k the solution at time $t = T^k$ does not need further corrections because the final solution is found. Fortunately, one only needs to perform few iterations until convergence be reached. In other words, the solution jumps are very small after few iterations.

The combination of an implicit procedure on the coarse evolution integrator, that uses a large time step, and an explicit integrator on the finer time step allows to verify the CFL criterion on a reasonable long time prediction with a very small number of processors. Other options should be considered if more processors are available.

3. NUMERICAL EXPERIMENTS

3.1. Taylor vortex

The two-dimensional Taylor vortex-decay problem [8] was used to evaluate the numerical time-parallel technique applied to the Navier–Stokes and continuity equations (1) and (2), respectively. The analytical solution is given by

$$u(x, y, t) = -\cos(x)\sin(y)\mathrm{e}^{-\sigma t} \tag{10}$$

$$v(x, y, t) = \sin(x)\cos(y)\mathrm{e}^{-\sigma t} \tag{11}$$

$$p(x, y, t) = -\tfrac{1}{4}(\cos(2x) + \sin(2y))\mathrm{e}^{-2\sigma t} \tag{12}$$

for $0 \leqslant x, y \leqslant \pi$, and $\sigma = 2/Re$. Computations were performed with $Re = 100$ and the time domain considered up to $T = 40$ s. The spatial meshes considered comprise 32×32 and 64×64 nodes. The Bi-CGSTAB [9] solver is used to solve the Poisson and the implicit momentum equations.

The numerical experiment was conducted on a PC cluster with 16 nodes, each one with one Pentium IV 2.4 GHz processor and 256 Mb Ram. An ethernet switch, 100 Mbps, is used for the node connection. The time domain was decomposed into P, number of processors, sub-domains (up to 16) yielding $\Delta t = 40/P$s and the finer time step was set equal to 4×10^{-3}s.

The solution and the computing time obtained with the parallel-in-time method depend on several parameters. The selection of the numerical methods for both temporal grids is discussed later. The computing time required to perform a parallel-in-time calculation depends on

(i) The number of iterations performed in the algorithm;
(ii) The spatial resolution;
(iii) The ratio of coarse to fine time-step sizes $\Delta t/\delta t$;

and for each one of the above items depends obviously on the number of processors.

The dependence of the computing time and the solution accuracy on items (i), (ii) and (iii) is presented in the following paragraphs.

3.1.1. Influence of the number of iterations. The parallel-in-time method involves two integrators, for the coarse and fine time-grids, that are used in an iterative fashion. Consequently, the computing time of the parallel-in-time solution is dependent on the number of iterations. Figure 2(a) shows the computing time versus the number of processors for different number of iterations considered. The spatial mesh comprises 32×32 nodes. For this test case the number of iterations is prescribed (2, 3 or 4) and Figure 2(a) shows that the computing time obviously decreases with the increase in the number of processors and increases with the increase in the number of iterations. Figure 2(a) shows also the computing time for a non-parallel calculation performed with a fully explicit method using a time-step equal to the finer time-step of the parallel-in-time calculation. The computing time required for the serial calculation is equal to 106 s. The computing time of the parallel-in-time calculation was equal to 41 s on 16 processors and four iterations. The speed-up is rather low, and equal to 4.1, using 16 processors and two iterations.

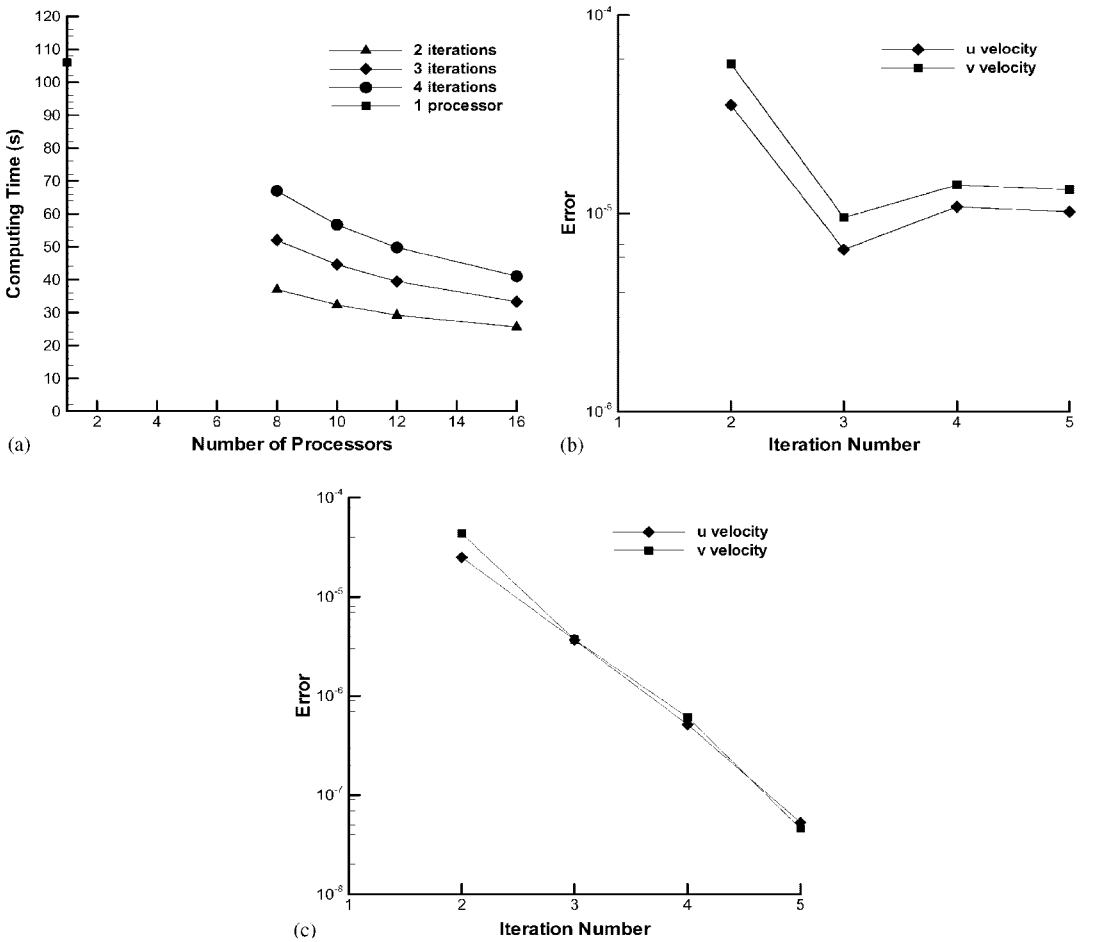


Figure 2. (a) Parallel-in-time computing time dependency on the number of iterations performed (32×32 nodes; $\delta t = 4 \times 10^{-3}$ s). (b) Parallel-in-time solution L_1 norm error dependency on the number of iterations performed (32×32 nodes; $\delta t = 4 \times 10^{-3}$ s). (c) Maximum deviation between parallel-in-time and serial solutions dependency on the number of iterations performed (32×32 nodes; $\delta t = 4 \times 10^{-3}$ s).

Concerning the accuracy of the method, Figure 2(b) shows the L_1 error norm of the calculated solution at $t = 40$s as a function of the number of iterations prescribed. A very small error in the velocity field, approximately equal to 1×10^{-5}, occurs. This error may be considered very small because the maximum value of the velocity components is approximately equal to 0.45. Figure 2(c) shows that the maximum deviation between the parallel-in-time and serial solutions decreases with the increase in the number of iterations. For the present case the use of four iterations induces a maximum deviation smaller than 1×10^{-6} and consequently, the parallel-in-time and serial solutions are virtually identical.

3.1.2. Influence of the spatial resolution. The temporal solution of any flow problem requires for each time step the solution of the dependent variables in the discrete spatial domain.

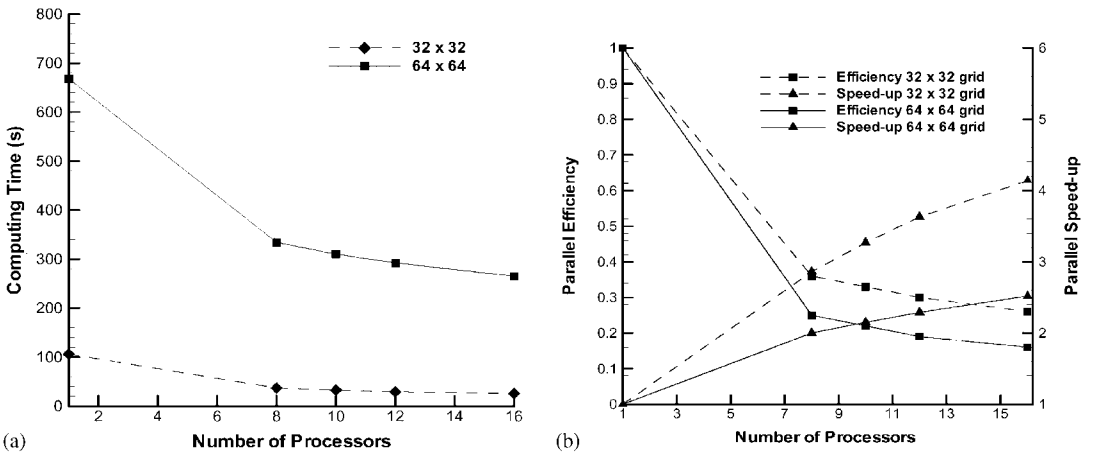


Figure 3. (a) Computing time dependency on space resolution (2 iterations; $\delta t = 4 \times 10^{-3}$ s). (b) Speed-up and parallel efficiency dependency on space resolution (2 iterations; $\delta t = 4 \times 10^{-3}$ s).

The use of an implicit method in the coarse time-grid penalizes the computing time when the number of nodes increases in the spatial mesh. Figure 3(a) shows the computing time obtained with the present parallel-in-time method for two spatial meshes with 32×32 and 64×64 nodes. The serial calculation computing time is also shown in the figure. For both spatial discretizations considered, a substantial computing time saving is verified when using the parallel-in-time method. This computing time saving increases with the increase in the spatial dimension of the problem. Speed-up and parallel efficiency for both spatial meshes considered are represented in Figure 3(b) showing that higher speed-up is achieved on coarser spatial meshes.

3.1.3. Influence of the ratio of coarse to fine time-step sizes. The computing time of the parallel-in-time calculations can be splitted into two parts. One is the time required for the sequential procedures of the algorithm and the other is used in parallel calculations. The computing time saving depends on the ratio between parallel and sequential computational efforts carried out during the iteration process. The selection of δt should correspond to the desired time resolution in a serial computation.

Three time-grid increments were considered on the fine time-grid, $\delta t_1 = 4 \times 10^{-2}$ s, $\delta t_2 = 4 \times 10^{-3}$ s and $\delta t_3 = 4 \times 10^{-4}$ s, to which correspond $M = 10^3$, 10^4 and 10^5, respectively. Figure 4 shows the computing time as a function of the number of processors for the three finer time-grids considered. The computing time saving increases with the increase of the time scales ratio (M/P).

3.1.4. Comparison of spatial domain decomposition and parallel-in-time results. The parallel-in-time method was compared with the standard spatial domain decomposition method. Calculations of the Taylor problem, on a 64×64 nodes mesh, were performed with spatial domain decomposition method using up to sixteen processors. A fully explicit procedure was used for the temporal evolution of the momentum equations. An explicit outer iteration coupling was

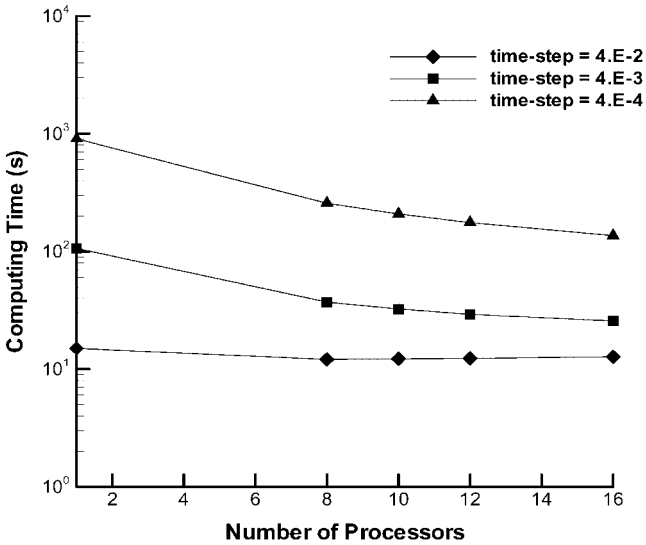


Figure 4. Computing time dependency on the size of the finer time-grid increment (2 iterations on a 32×32 nodes mesh).

used during the solution of the Poisson equation. After each outer iteration pressure and velocities on partition boundaries are exchanged between processors until convergence is reached. The parallel efficiency rapidly decreases with the increase in the number of processors due to the high ratio between the communication and computation efforts as shown in Figure 5(a).

The definition of parallel efficiency and speed-up for parallel-in-time results should correspond to the classical one developed for space domain decomposition calculations. In the present parallel-in-time method the iterative use of two integrators, for the coarser and finer time-grids, causes a speed-up and efficiency decrease. However, large computer time reduction can still be achieved when comparing with a single processor calculation. Figure 5(b) shows the parallel efficiency of the parallel-in-time calculations. The low efficiency of the spatial domain decomposition method is due to the low dimension of the problem, while the low efficiency of the parallel-in-time calculations is inherent to the present method for the reasons explained above. Nevertheless, the efficiencies ratio, between parallel-in-time and spatial domain decomposition methods, is important for the user since it will help to select the domain, spatial or temporal, that should be parallelized. Figure 5(c) shows the parallel efficiencies ratio between parallel-in-time and spatial domain decomposition methods. Figure 5(c) shows that when the number of processors increases the parallel-in-time method efficiency is higher than the parallel domain decomposition efficiency. This result was expected since the domain decomposition parallel efficiency decreases with the increase in the number of processors.

3.2. Vortex shedding flow

The numerical simulation of flow past a two-dimensional square cylinder between parallel walls for Reynolds number equal to 1000 was selected to illustrate the application of the parallel-in-time method to a more demanding unsteady flow problem. The local mesh Reynolds

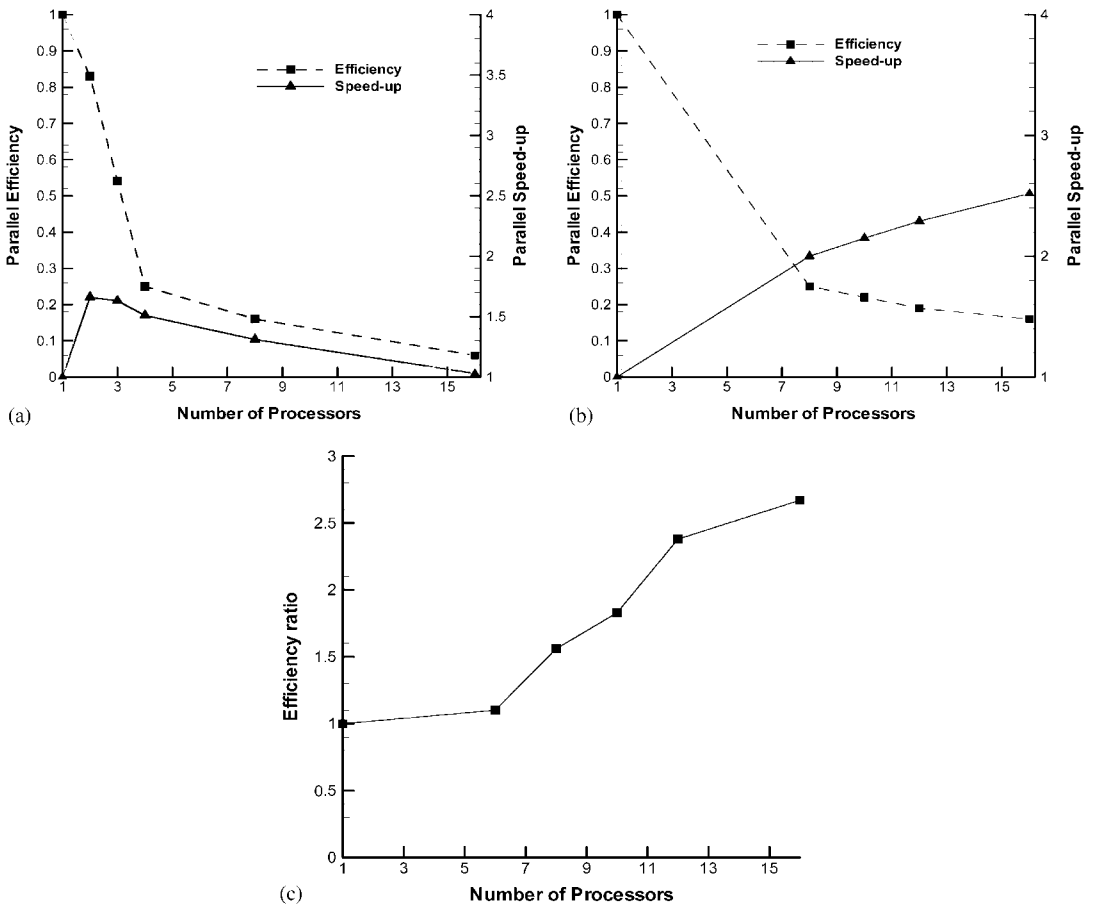


Figure 5. (a) Spatial domain decomposition parallel efficiency and speed-up on a 64×64 nodes mesh and $\delta t = 4 \times 10^{-3}$ s. (b) Parallel-in-time method parallel efficiency and speed-up on a 64×64 nodes mesh and $\delta t = 4 \times 10^{-3}$ s. (c) Parallel-in-time and domain decomposition methods parallel efficiency ratio on a 64×64 nodes mesh.

number or Peclet number is higher than 2 and consequently a deferred correction method was used for convection discretization. The deferred correction employed about 80% of central differences and the remaining 20% of upwind contribution. The main objective of the work is to employ the parallel-in-time method to unsteady flow problems. The quasi-first-order scheme used does not prevent or limit the main conclusions of this work.

The flow configuration comprises the square cylinder, of width unity, confined in a channel with $H = 6$ that is sketched in Figure 6. The blockage ratio is therefore $\frac{1}{6}$ and the channel length is equal to 24. The flow is impulsively started at $t = 0$ being a uniform flow prescribed at the inlet. At the outlet, the convective wave open boundary condition was used for velocity components. In addition, no-slip conditions were prescribed on walls. The numerical mesh

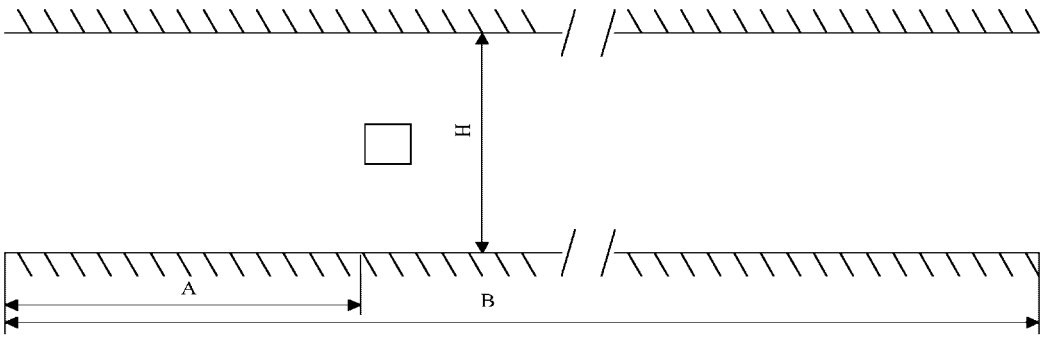


Figure 6. Flow configuration ($A = 9,\ B = 24,\ H = 6$).

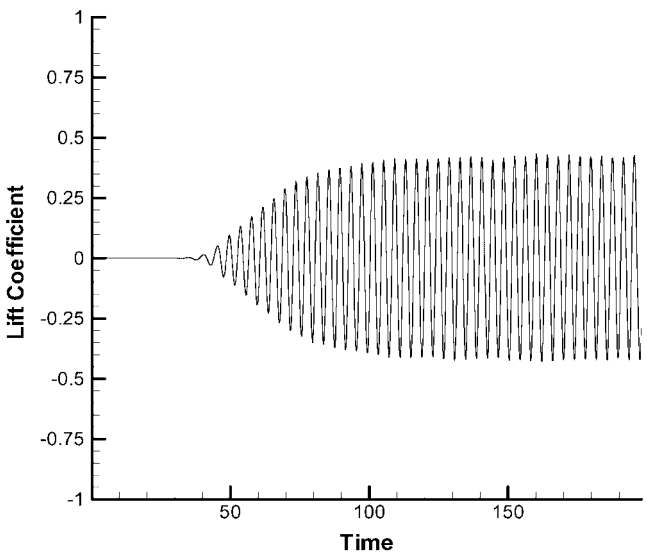


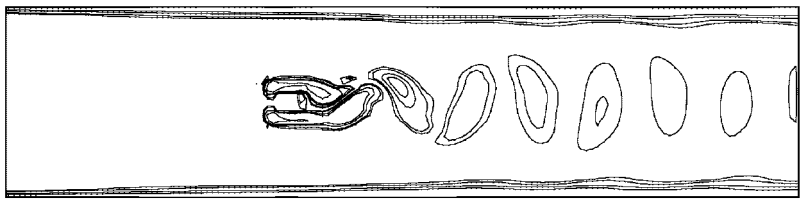
Figure 7. Lift coefficient temporal evolution.

was uniform and comprises 145×37 nodes to discretize the non-dimensionalized domain of 24×6.

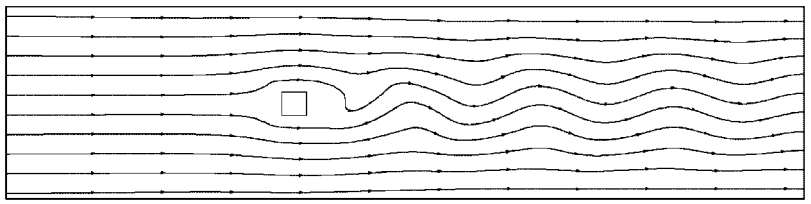
Flow around bluff bodies is characterized after a critical Reynolds number by the onset of periodic oscillations, the von Karman vortex street.

The simulation was performed on the PC cluster described above using 15 processors. Since the number of processors available is insufficient to perform an entire calculation, time blocks, corresponding each one to P processors handling $P \times \Delta t$, has to be solved sequentially.

The time-step sizes are equal to $\delta t = 3 \times 10^{-4}$ and $\Delta t = 2 \times 10^{-1}$ in the fine and coarser time-grids, respectively. Figure 7 shows the temporal evolution of the lift coefficient where the onset of breaking flow can be observed at $t \approx 40$ s after the impulsive start. The bifurcation of the solution was triggered by numerical noise without any prescribed perturbation and in a



(a)



(b)

Figure 8. Predicted vorticity iso-contours (a) and streamlines (b).

similar fashion to the pure sequential simulations. Figures 8(a) and 8(b) show predicted vorticity contours and streamlines, respectively. Prediction for the Strouhal number, $St = fD/U_0$, where f is the shedding frequency and D is the square cylinder width, is equal to 0.25.

The predictions of the flow were also obtained on a single processor with the parameters used in the finer time-grid temporal evolution of the parallel-in-time procedure. The results were virtually identical, showing that the time decomposition procedure did not deteriorate the solution. Deviation to the reference solution [10] reported value of Strouhal number equal to 0.161 is then originated by the spatial discretization and the first-order method used for the temporal evolution. One should stress that the deviation is not due to the parallel-in-time method. It is well known that first-order schemes induce too much amplitude or phase errors when simulating vortex shedding flows. However, this is not the objective of the present work. The technique may well be applied to high order discretizations.

Convergence monitoring was based on the solution jumps values at each iteration because the analytical solution is not available for this test case. The iterative procedure is performed until the average absolute value of the velocity solution jumps, Equation (7), over all the time block domain, reaches the prescribed convergence criterion of 1×10^{-3}. Fortunately, few iterations are required to reach the convergence. Hence, after the onset of the shedding flow only two iterations were needed to satisfy the criterion. The speed-up achieved with the parallel-in-time solution is equal to 2.3. When the convergence criterion is set to 5×10^{-4} three iterations are required to reach convergence and the speed-up is reduced to 1.8. A typical example of the average absolute value of the velocity solution jumps as a function of the number of iterations is shown in Figure 9.

Finally one should briefly comment that this parallel-in-time method is still in the beginning of development or application. The numerical methods used in the coarse and finer time-grids should be selected accordingly with the problem time domain length and the number of

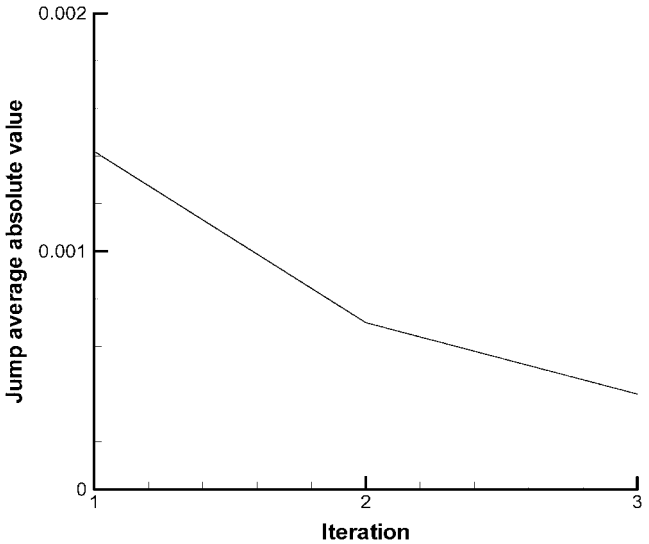


Figure 9. Evolution of the average absolute value of the velocity solution jumps at $t = 90$ s.

processors available. The implementation of hybrid spatial and temporal methodology is also possible. Both strategies will be available in future.

4. CONCLUSIONS

A parallel-in-time method, based on the temporal domain decomposition, was applied for the solution of the incompressible unsteady parabolic Navier–Stokes equations. The two-dimensional Taylor vortex-decay problem with $Re = 100$ was selected to conduct a sensitivity analysis on some of the parallel-in-time method influencing parameters. The following conclusions were derived:

(i) The parallel-in-time solution requires less computational time than the single processor solution. Speed-up depends on several parameters that were investigated.
(ii) The parallel-in-time computing time decreases with the number of processors and increases with the number of iterations required for convergence of the iterative process, (between the sequential coarse time-grid and the finer parallel time-grid solutions).
(iii) The parallel efficiency of the parallel-in-time method increases with the decrease of the spatial dimension of the problem.
(iv) The parallel efficiency of the present method increases when the computational effort ratio, (between fine and coarse time-grid integrators), increases.

The simulation of a confined flow around a square cylinder for $Re = 1000$, shows that the parallel-in-time method can solve realistic unsteady flow problems. For the present case, since only 15 processors were available, time blocks were used and only a small number of iterations were required to reach the convergence criterion.

Parallelization-in-time is a very promising technique, in particular when spatial dimension of the problem is low and the temporal scale is large. Nevertheless, some issues, like stability, or optimal temporal methods for coarse and fine time-grids, or application to equations of hyperbolic nature, need further research.

REFERENCES

1. Quarteroni A, Valli A. *Domain Decomposition Methods for Partial Differential Equations*. Oxford University Press: Oxford, 1999.
2. Dongarra J, Foster I, Fox G, Kennedy K, White A, Torczon L, Gropp W. *The Source Book of Parallel Computing*. Elsevier Science: USA, 2003.
3. Lions J, Maday Y, Turinici G. Resolution d'EDP par un schema en temps 'pararéel'. *Comptes Rendus des Seances de l Academie des Sciences*, *Paris*, *Séries I, Mathematique* 2001; **332**:661–668.
4. Maday Y. Parareal in time simulation for the Navier–Stokes equations. In Sequeira A, Albuquerque C (eds), *Book of Abstracts of the Third International Conference of Applied Mathematics for Industrial Flow Problems*, Lisboa, Portugal, 2002.
5. Bal G, Maday Y. A 'parareal' time discretization for non-linear PDE's with application to the pricing of an American put. In Pavarino LF, Toselli A (eds), *Proceedings of the Workshop on Domain Decomposition*, Zurich, Switzerland, Lecture Notes in Computer Science and Engineering Series, vol. 23. Springer: Berlin, 2002.
6. Baffico L, Bernard S, Maday Y, Turinici G, Zerah G. Parallel-in-time molecular-dynamics simulations. *Physical Review* 2002; **E66**:057701.
7. Patankar S. *Numerical Heat Transfer and Fluid Flow*. Hemisphere Publishing Corporation: Washington, DC, 1980.
8. Taylor G. On the decay of vortices in a viscous fluid. *Philosophical Magazine* 1923; **46**:671–675.
9. van der Vorst H. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* 1992; **13**(2):631–644.
10. Davies R, Moore E, Purtell L. A numerical-experimental study of confined flow around rectangular cylinders. *Physics of Fluids* 1984; **27**:46–59.

