
ELSEVIER

Parallel Computing 27 (2001) 363–389

PARALLEL COMPUTING

www.elsevier.com/locate/parco

A parallel solver for unsteady incompressible 3D Navier–Stokes equations

M. Garbey [a,*], Yu.V. Vassilevski [b]

[a] *University Claude Bernard Lyon-1, CDCSP, ISTIL-Bâtiment 101, 43 Bd du 11 Novembre 1918, 69622 Villeurbanne Cedex, Lyon, France*

[b] *Institute of Numerical Mathematics, Russian Academy of Sciences, 8 ul. Gubkina, 117333 Moscow, Russian Federation*

Received 25 February 2000; received in revised form 22 June 2000

Abstract

We describe some algorithms and software components that allow us to solve on parallel computers classical test cases for unsteady incompressible 3D Navier–Stokes equations. Our main focus is the design of robust and efficient parallel solvers for systems with singularly perturbed convection–reaction–diffusion and Laplace operators, which are important constituents of the Navier–Stokes solvers. The performance of the solvers on two parallel computers is examined.

Keywords: Navier–Stokes equations; Overlapping domain decomposition; Fictitious domain method; Finite differences

1. Introduction

In this paper, we describe some algorithms and software components that allow us to solve on parallel computers classical test cases for unsteady incompressible 3D Navier–Stokes equations.

The choice of a methodology in scientific computing is always a compromise between different criteria such as numerical efficiency and machine efficiency of the software on a given parallel architecture, simplicity and flexibility of the method,

* Corresponding author. Tel.: +33-472-44-80-55; fax: +33-472-43-11-45.

E-mail addresses: garbey@cdcsp.univ-lyon1.fr (M. Garbey), vasilevs@dodo.inm.ras.ru (Y.V. Vassilevski).

0167-8191/01/$ - see front matter
PII: S0167-8191(00)00067-3

robustness and peak performance of the code. This choice depends strongly on the application itself and varies from the academic point of view to the industrial point of view depending on economical or business constraints.

We have restricted ourselves to consider convection dominant flows with a leading direction of propagation of the flow, such as one may encounter for internal flow in pipes. We decided to stay away from adaptive mesh generation and to use simple boundary fitted Cartesian grids. In this framework, our main focus is the design of robust and efficient parallel solvers for linear operators of the following two types

$$-\epsilon\Delta + \delta\mathbf{u}\cdot\nabla + Id \tag{1}$$

and

$$-\Delta. \tag{2}$$

These two operators appear naturally when one discretizes in time the unsteady incompressible Navier–Stokes equation; ϵ (resp. δ) is then a small parameter proportional to the time step and the inverse of the Reynolds number (resp. proportional to the time step). These operators are also quasi-universal since they appear in many other mathematical models like diffusion and/or convection effect simulation. These operators are consequently essential components of computational physics, chemistry or electromagnetism.

We observe that (1) and (2) have essentially different properties in the way that perturbation of boundary conditions propagates, see [5] and references therein. In domain decomposition methodology [9], these features are essential to the design of efficient parallel solvers, since interprocessor communications are bound to be related to the propagation of numerical errors at the artificial interfaces. In addition to this, the overall efficiency of the algorithm is strongly dependent on the parallel architecture: in this paper we restrict ourselves to affordable parallel computers with few dozen of processors or clusters of servers.

The plan of this paper is as follows. In Section 2, we describe the problem formulation that leads to the operators (1) and (2). In Section 3, we construct the discretized problems. In Section 4, we describe our solvers. Section 5 is devoted to the numerical experiment and efficiency of the code in the *steady* flow case. In Section 6 we study the efficiency of the code in the *unsteady* case and give additional details on the numerical validation of the method.

An essential numerical part of the work is the usage of a parallel fast direct solver (courtesy of T. Rossi and J. Toivanen, University of Jyvaškylä, Finland) developed and delivered in the framework of Russian–Finnish academy cooperation. The authors are very thankful to Damien Tromeur-Dervout (CDCSP, University of Lyon 1, France) for fruitful discussions.

2. Problem formulation

Let $\Omega \in \mathbb{R}^3$ be a domain with a piece-wise smooth boundary $\partial\Omega$. The domain is occupied by a fluid with a kinematic viscosity v and a density ρ. We denote by $\mathbf{u}(x,t)$

the velocity with components (u_1, u_2, u_3) and by $p(x,t) = P(x,t)/\rho$ the normalized pressure of the fluid. The flow of incompressible fluid with prescribed values of the velocity on $\partial\Omega$ obeys the Navier–Stokes equations:

$$\frac{\partial \mathbf{u}}{\partial t} - \nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = f \text{ in } \Omega, \tag{3}$$

$$\begin{aligned} &\mathbf{u} = \mathbf{g} \text{ on } \partial\Omega, \\ &\operatorname{div}\mathbf{u} = 0 \text{ in } \Omega. \end{aligned} \tag{4}$$

An important parameter for flow similarity is the Reynolds number $Re = \bar{U}\bar{D}/\nu$, where $\bar{U}$ and $\bar{D}$ are characteristic velocity and length, respectively.

The time discretization is performed in the framework of the projection algorithm:

- *step* 1. prediction of the velocity $\hat{\mathbf{u}}^{k+1}$ by solving

$$\begin{aligned} &\frac{\hat{\mathbf{u}}^{k+1} - \mathbf{u}^k}{\Delta t} - \nu\Delta\frac{\hat{\mathbf{u}}^{k+1} + \mathbf{u}^k}{2} + \left(\left(\frac{3}{2}\mathbf{u}^k - \frac{1}{2}\mathbf{u}^{k-1}\right)\cdot\nabla\right)\frac{\hat{\mathbf{u}}^{k+1} + \mathbf{u}^k}{2} \\ &\quad = \mathbf{f}^{k+1/2} - \nabla p^k \text{ in } \Omega, \end{aligned} \tag{5}$$

$$\hat{\mathbf{u}}^{k+1} = \mathbf{g} \text{ on } \partial\Omega; \tag{6}$$

- *step* 2. projection of the predicted velocity to the space of divergence free functions

$$-\operatorname{div}\nabla\delta p = -\frac{1}{\Delta t}\operatorname{div}\hat{\mathbf{u}}^{k+1}, \tag{7}$$

$$\mathbf{u}^{k+1} = \hat{\mathbf{u}}^{k+1} - \Delta t\nabla\delta p, \quad p^{k+1} = p^k + \delta p. \tag{8}$$

Two important remarks are pertinent here. Eq. (5) is the Crank–Nicolson scheme with the second-order extrapolation of the convection field. Thus, we may expect the second-order of accuracy in time. Second, we pose no boundary condition for pressure correction. As a matter of fact, the Navier–Stokes equations may be considered as a boundary value problem for velocity (3) with Lagrange multiplier p which is introduced in order to compensate the additional constraint (4). From this point of view, no boundary condition for pressure is needed. The number of degrees of freedom for the discrete pressure variable should be then equal to the number of discrete constraint Eq. (4). Eq. (7) is to be considered as a constituent of the projection operator (7) and (8); on discrete level, it may appear from algebraic arguments rather than from an approximation of a boundary value problem. Following this strategy, we approximate (5) as an elliptic problem for $\hat{\mathbf{u}}^{k+1}$ with the right-hand side $\mathbf{f}^{k+1/2} - \nabla p^k$ plus terms depending on $\mathbf{u}^k$,

$$-\frac{\nu}{2}\Delta\hat{\mathbf{u}}^{k+1}+\left(\left(\frac{3}{2}\mathbf{u}^{k}-\frac{1}{2}\mathbf{u}^{k-1}\right)\cdot\nabla\right)\frac{\hat{\mathbf{u}}^{k+1}}{2}+\frac{\hat{\mathbf{u}}^{k+1}}{\Delta t}=\mathrm{rhs}^{k,k+1/2}\ \text{in}\ \Omega,$$
$$\hat{\mathbf{u}}^{k+1}=\mathbf{g}\ \text{on}\ \partial\Omega. \tag{9}$$

In contrast, Eq. (7) is not *approximated* as a b.v.p.; its discrete counterpart is generated from algebraic considerations.

We are now going to discuss the space discretization of the problem.

3. Construction of the discretized problems

Let Ω_i^h be a rectangular grid associated with a velocity component u_i, $i=1,2,3$ (see Fig. 1; for the sake of simplicity, all figures below present the 2D case).

The Laplace operator is approximated by the standard finite difference scheme resulting in the 7-point stencil, see Fig. 2. The convective term is approximated either by the upwind finite differences, or by the central finite differences, or their linear combination. To define the second order approximation of Eq. (5), we proceed as follows [13]. The set of grid nodes of Ω_i^h is split into nodes within Ω (inner nodes), and the others (external nodes). The inner nodes are split into two subsets denoted by Type 1, Type 2 in Fig. 3. All the stencil neighbors of the nodes of Type 1 are inner nodes, while any node of Type 2 has at least one external neighbor. The discrete counterpart of Eq. (5) corresponding to the node of Type 1 is nothing but the most straightforward and simplest finite difference approximation; for the Laplace operator, we have

$$-\Delta u\approx\frac{(u_r-u_0)/h_r-(u_0-u_l)/h_l}{(h_r+h_l)/2}+\frac{(u_u-u_0)/h_u-(u_0-u_d)/h_d}{(h_u+h_d)/2}+\frac{(u_f-u_0)/h_f-(u_0-u_b)/h_b}{(h_f+h_b)/2}.$$

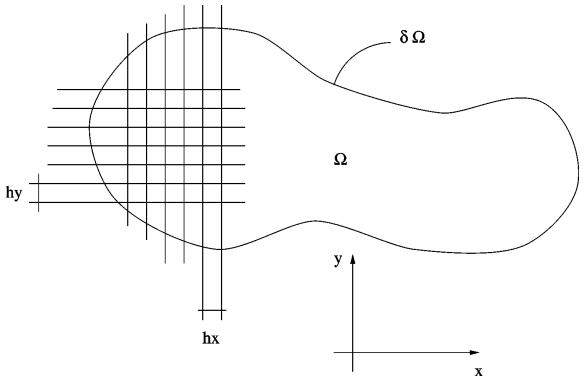

Fig. 1. The grid Ω_i^h for velocity component u_i.

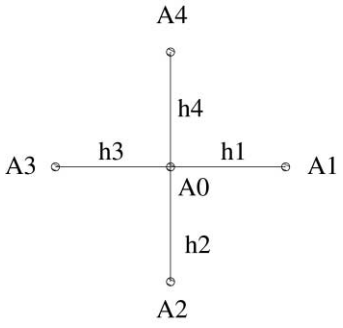

Fig. 2. Finite difference stencil for the Laplace operator.

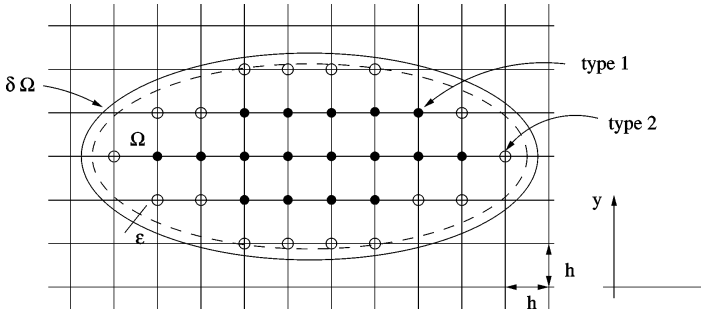

Fig. 3. Inner nodes of the grid Ω_i^h.

For the convective term, we will take central finite differences

$$(\mathbf{b}\nabla)u \approx b_1 \frac{u_r - u_l}{h_r + h_l} + b_2 \frac{u_u - u_d}{h_u + h_d} + b_3 \frac{u_f - u_b}{h_f + h_b}$$

or one-sided finite differences

$$\begin{aligned}(\mathbf{b}\nabla)u \approx {} & \max(b_1, 0) \frac{u_0 - u_l}{h_l} + \min(b_1, 0) \frac{u_r - u_0}{h_r} + \max(b_2, 0) \frac{u_0 - u_d}{h_d} \\ & + \min(b_2, 0) \frac{u_u - u_0}{h_u} + \max(b_3, 0) \frac{u_0 - u_b}{h_b} + \min(b_3, 0) \frac{u_f - u_0}{h_f}\end{aligned}$$

depending on the cell Reynolds number.

The grid equation at a node of Type 2 may not be written as easy as above, since there exists at least one grid node from the stencil where the grid function is not defined (see Fig. 4). However, this external node may be replaced by its projection onto $\partial\Omega$ (point A_3 in Fig. 4). The value of grid function is known at A_3, since $\mathbf{u}(A_3) = \mathbf{g}(A_3)$. This operation deforms the stencil, but the second-order accuracy may be still proven for the Poisson problem [13]. From implementation point of

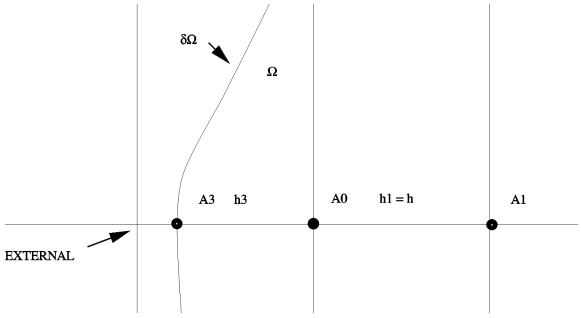

Fig. 4. Deformed stencil for the node of Type 2.

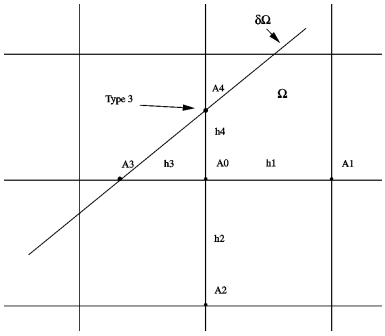

Fig. 5. Virtual nodes of Type 3.

view, we define the nodes of Type 3 which are not the grid nodes but are intersections of the grid lines and $\partial\Omega$ (see Fig. 5). The nodes of Type 3 contribute to stencils for inner nodes of Type 2, but they are not the nodes where Eq. (5) has to be approximated. For the sake of stability of approximation, Type 3 is reassigned also to those inner nodes, whose distance to $\partial\Omega$ is small with respect to local mesh sizes. In other words, the nodes from a narrow close-to-boundary strip are eliminated from the set of inner nodes (see Fig. 3).

Given arbitrary rectangular grids Ω_i^h, $i = 1, 2, 3$, each component of the velocity u_i is approximated on its grid Ω_i^h independently of other components. The second-order accuracy both in time and space is provided for the problem (5). However, the need in approximation of the constraint condition (4) and pressure variable p poses severe restrictions on freedom for choosing the grids Ω_i^h, $i = 1, 2, 3$. One of the earliest and the most elegant solutions is the use of staggered (or MAC) grids. These grids are constructed as follows [2,8]. In domain Ω we define a basic rectangular grid Ω_0^h. The grids Ω_i^h, $i = 1, 2, 3$, are built on the basis of Ω_0^h. The nodes of Ω_1^h are located at the

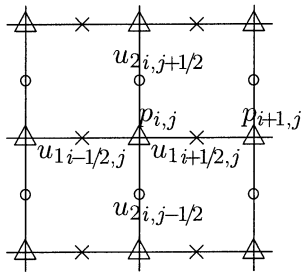

Fig. 6. Staggered grids.

midpoints of horizontal (i.e., parallel to the x-axis) edges of the basic grid Ω_0^h (they are shown by $\times$ in Fig. 6); the grid Ω_2^h has its nodes at the midpoints of vertical (i.e., parallel to the y-axis) edges of Ω_0^h ($\circ$ in Fig. 6). The grid Ω_3^h is constructed analogously, by shifting Ω_0^h along the z-axis.

For regular grids the pressure derivative is naturally approximated by the central differences:

$$\left[\frac{\partial p}{\partial x}\right]_{i+1/2,j} \approx \frac{p_{i+1,j} - p_{ij}}{h_0}.$$

The divergence of the vector field $\mathbf{u}$ is approximated at the nodes of Ω_0^h by the central differences as well:

$$[\operatorname{div}\mathbf{u}]_{i,j} \approx \frac{u_{1_{i+1/2,j}} - u_{1_{i-1/2,j}}}{h_1} + \frac{u_{2_{i,j+1/2}} - u_{2_{i,j-1/2}}}{h_2} + \cdots \tag{10}$$

The above stencils both for the pressure gradient and the velocity divergence provide the second-order accuracy of approximation even on non-uniform meshes with smooth mesh size function [13]. Unfortunately, this holds true only for the nodes apart from the boundary $\partial\Omega$. At close-to-boundary nodes (Type 2) we proceed as follows.

The grid Ω_0^h associated with the pressure and the divergence serves as the basis for generation of the grids Ω_i^h, $i = 1, 2, 3$. The latters are used to approximate (5) in the domain Ω. Those nodes of Ω_i^h, $i = 1, 2, 3$, which contribute a degree-of-freedom to (5), are identified by their location with respect to $\partial\Omega$. The nodes contributing degrees-of-freedom to the pressure/divergence space are identified alternatively. A pressure node is said to be a contributor if at least one of the terms of the sum (10) contains a value associated with a node of Type 1 or Type 2. For example, the node $\{i, j\}$ in Fig. 7 is the contributor, since the node $\{i + 1/2, j\}$ is of Type 2 for the component u_1. Such a definition enables us to specify degrees-of-freedom for the pressure *automatically*. The approximation of the pressure derivative at node $\{i + 1/2, j\}$ maintains the second-order of accuracy. However, the divergence approximation at node $\{i, j\}$ faces severe problems, since the values of the velocity components $u_{1_{i-1/2,j}}$, $u_{2_{i,j+1/2}}$, $u_{2_{i,j-1/2}}$ *are not defined*. The remedy is to use instead of

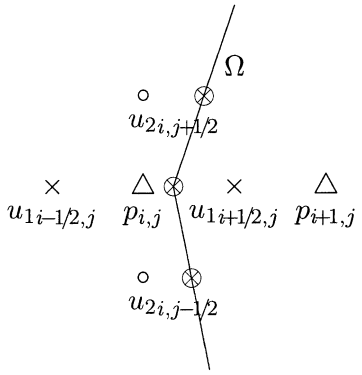

Fig. 7. Contributor-nodes for the pressure and the divergence stencil.

them the boundary values at points of $\partial\Omega$ which are the projections of $\{i-1/2,j\}$, $\{i,j+1/2\}$, $\{i,j-1/2\}$ (shown by $\otimes$ in Fig. 7). The substitution deforms the stencil for the divergence yielding no more than the first-order of approximation. Moreover, due to different stencils, the pressure gradient matrix is not equal to the transposed divergence matrix. The "unsymmetry" might be unsatisfactory. So we can preserve the "symmetry" in approximation of divergence and gradient operators as follows. We define the pressure gradient submatrix associated with any node of Type 2 to be the transposed submatrix of the divergence operator. The pressure gradient is not approximated with the second-order of accuracy anymore. Moreover, its stencil looks as if we "staggered" the pressure node into three different midpoints of the divergence stencil edges. We apply both options for the pressure gradient discretization distinguishing them as symmetric and unsymmetric divergence/gradient pairs.

Finite difference approximations on staggered grids are proven to be stable. It means that the convergence of the scheme for the velocity causes the convergence for the pressure. The fundamental LBB condition

$$\sup_{\mathbf{u}\perp\operatorname{Ker}\Delta}\frac{(p,\operatorname{div}\mathbf{u})}{\|u\|_{H^1}}\geqslant C_0\|p\|_{L_2}\quad\forall p\perp\operatorname{Ker}\nabla$$

with constant C_0 independent of the number of degrees-of-freedom, was proven in [6] for a rectangular domain, and analyzed in [1] for a domain with curved boundary and logically rectangular mesh.

Remark. Though the above scheme is very simple and thus attractive, it has severe drawbacks. First, the stability arguments may require the upwind approximation of the convective term. The above simple form of the upwinding leads to the first-order accuracy. More complicated upwind approximations preserve the second-order of spatial accuracy [14]; we do not discuss them since for the Reynolds numbers and meshes in our numerical experiments it is sufficient to apply the central finite differences except in the region far away from the obstacle where the space mesh step is

large. Second, the discrete divergence approximates the differential one with the first-order in the close-to-boundary nodes. To cope with the problem, we might modify locally the discrete scheme either by using one or two layers of finite elements, or by applying the gridless method, or by fitting grids to the boundary. However, as we show in experiments, this weakness has a little impact on the computation of lift and drag which are the most important practical values for most simulations.

4. Description of the linear solvers

Taking the advantage of the technique discussed above we write the discrete counterpart of the projection algorithm (5)–(8). We denote by Δ_i, $(\mathbf{b}\nabla_i)$ the finite difference Laplace and convection operators on the grid Ω_i^h, $i = 1, 2, 3$. By div_0 and ∇_0 we denote the discrete divergence operator and the pressure gradient operators, respectively. Due to the choice of the discrete approximation, the matrix $-\mathrm{div}_0\nabla_0$ is a square matrix of order equal to the number of degrees-of-freedom in the discrete pressure/divergence space. We avoid the problem of boundary conditions for pressure in (7) by generating the problem (7) *algebraically*. Let $\prod_i \mathbf{u}$ stand for the second-order interpolation of the staggered velocity components on the grids Ω_i^h, $i = 1, 2, 3$. Then the discrete projection algorithm reads as follows.

Predict the velocity component $\hat{u}_i^{k+1}$ by solving the momentum equation

$$\frac{\hat{u}_i^{k+1} - u_i^k}{\Delta t} - \nu\Delta\frac{\hat{u}_i^{k+1} + u_i^k}{2} + \left(\prod_i\left(\frac{3}{2}\mathbf{u}^k - \frac{1}{2}\mathbf{u}^{k-1}\right)\cdot\nabla_i\right)\frac{\hat{u}_i^{k+1} + u_i^k}{2} = f_i^{k+1/2} - (\nabla_0 p^k)_i \text{ in } \Omega, \tag{11}$$

$$\hat{u}_i^{k+1} = g_i \text{ on } \partial\Omega, \quad i = 1, 2, 3.$$

Find the pressure correction and project the predicted velocity

$$-\mathrm{div}_0\nabla_0\delta p = -\frac{1}{\Delta t}\mathrm{div}_0\,\hat{\mathbf{u}}^{k+1}, \quad \mathbf{u}^{k+1} = \hat{\mathbf{u}}^{k+1} - \Delta t\nabla_0\delta p, \quad p^{k+1} = p^k + \delta p. \tag{12}$$

4.1. Parallel Schwarz method

Problem (11) is a discrete convection–diffusion–reaction problem. This is a two-parameter singular perturbation problem with respect to the small diffusion term (ν) and the time step (Δt). A parallel solver based on a two-level Schwarz method was presented in [3,5]. At the outer level, the original domain is split into overlapping crosswind strips (slices) Ω_m, see Fig. 8.

The classical Schwarz method in the downwind direction is the outer part of the algorithm. Even with minimal overlaps very few outer iterations are needed to reach a prescribed accuracy. The fast convergence is a consequence of the very fast upwind decay of numerical errors introduced at artificial interfaces parallel to the z-directions. It is natural to use a Schwarz method for the inner iterations. We split each

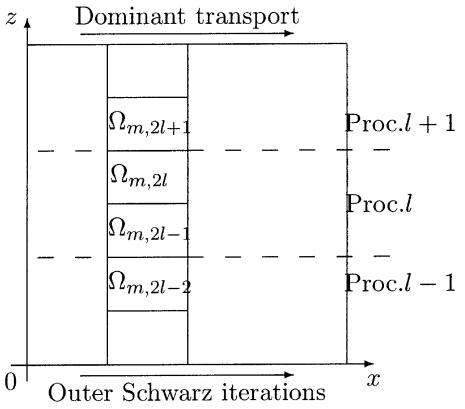

Fig. 8. Data allocation for a velocity component cluster.

strip-subdomain Ω_m into overlapping boxes $\Omega_{m,l}$ and define the inner iterations using a version of the multiplicative Schwarz method. The novel feature of the version of the Schwarz method selected for the inner iterations is that the choice of overlapping domain decomposition is based on the fast crosswind decay phenomenon [5]. At the level of the inner iterations, we have natural parallelism, since each of the box-subdomains can be treated by its own processor. It is very important that the number of iterations between the box-subdomains with odd and even indices may be very small (1 or 2) and small overlaps practically do not affect the fast convergence of the algorithm. In the current implementation the uniform domain decomposition is logically two-dimensional, the overlaps between subdomains are small (two mesh steps). As it was shown in [3], no adaptive decomposition is needed due to the presence of the large factor $1/\Delta t$ in the reaction term.

4.2. Parallel generalized conjugate residual method with a fictitious domain preconditioner

Though the matrix of the pressure correction problem (12) is singular, the system is consistent due to algebraic generation of the matrix $\mathrm{div}_0\nabla_0$. We apply the preconditioned generalized conjugate residual (GCR) method [12] to the system (12). Let matrix B be a non-singular preconditioner for matrix $-\mathrm{div}_0\nabla_0$. Then a system $-\mathrm{div}_0\nabla_0 x = y$ may be transformed to a system $Ax = b$ with the matrix $A = -B^{-1}\mathrm{div}_0\nabla_0$ and the right-hand side $b = B^{-1}y$. The GCR method is as follows.

1. Compute $r_0 = b - Ax_0$. Set $p_0 = r_0$.
2. For $j = 0, 1, 2, \ldots$, until convergence Do:
3. $\alpha_i = \frac{(r_j, Ap_j)}{(Ap_j, Ap_j)}$.
4. $x_{j+1} = x_j + \alpha_j p_j$.
5. $r_{j+1} = r_j - \alpha_j A p_j$.
6. Compute $\beta_{ij} = -\frac{(Ar_{j+1}, Ap_i)}{(Ap_i, Ap_i)}$ for $i = 0, 1, \ldots, j$.

7. $p_{j+1} = r_{j+1} + \sum_{j=0}^{j} \beta_{ij} p_i$.
8. EndDo

The preconditioner is based on the fictitious domain method [7]. Let the number of contributor nodes be n_1, and the number of the remaining nodes in Ω_0^h be n_2, and the contributor nodes are numerated first. We define a preconditioner $B^{-1} = TC^{-1}T^{\mathrm{T}}$, where $T = [I \quad O]$, $I \in \mathbb{R}^{n_1 \times n_1}$, $T \in \mathbb{R}^{n_1 \times (n_1+n_2)}$, $C \in \mathbb{R}^{(n_1+n_2)\times(n_1+n_2)}$. Matrix C is an easily invertible separable discrete operator:

$$C = (M_1 \otimes M_2 \otimes M_3)^{-1}(A_1 \otimes M_2 \otimes M_3 + M_1 \otimes A_2 \otimes M_3 + M_1 \otimes M_2 \otimes A_3).$$

Matrices A_i, M_i are the one-dimensional stiffness and lumped mass matrices (with Neumann boundary conditions) associated with the axes traces of the grid Ω_0^h. It may be shown that the matrix C is spectrally equivalent to a finite difference counterpart of the Laplace operator with homogeneous Neumann boundary conditions on Ω_0^h. The lines of the matrix $-\mathrm{div}_0\nabla_0$ do not differ from the lines of finite difference Laplace operator in all the contributor-nodes except the close-to-boundary nodes. The lines of $-\mathrm{div}_0\nabla_0$ associated with the close-to-boundary nodes may be thought as some discretizations of inhomogeneous Neumann boundary condition. Thus, according to the theory of the fictitious domain method [7], we may expect some kind of spectral equivalence between matrices B and $-\mathrm{div}_0\nabla_0$. Since matrix $\mathrm{div}_0\nabla_0$ is not symmetric, the spectral equivalence has to be understood as existence of such a fixed domain in the complex plane separated from the origin, that all the eigenvalues of the matrix $A = -B^{-1}\mathrm{div}_0\nabla_0$ belong to it. As a result, the GCR method will converge with a rate independent of the order of matrix $-\mathrm{div}_0\nabla_0$. Linear systems with matrix C are assumed to be solved by a parallel fast direct solver for discrete separable operators [10,11].

In order to minimize the mean number of GCR iterations to be performed at each time step, we accumulate the Krylov space in terms of vectors p_i's and Ap_i's. Before applying GCR to the system with a new right-hand side b, we find a good initial guess by projecting $x_0 = 0$ onto accumulated Krylov space $\{p_i\}_{i=1}^{k}$ [16]:

$$x_0 := \sum_{i=1}^{k} \frac{(b, Ap_i)}{(Ap_i, Ap_i)} p_i.$$

In fact, we invert the matrix of the system projected onto the Krylov space. This operation costs k scalar products and requires no matrix–vector or preconditioner–vector multiplications. As the flow is fully developed, this projection saves a lot of computation. The reason is that almost all information about the solution for the new time step may be extracted from the accumulated Krylov space. The projection does not affect the convergence rate. It does generate a very good initial guess to start with. In the case of essentially unsteady flows such a good initial guess may not be obtained by extrapolations from the previous time steps: the error of initial guess due to extrapolations is quite large and spans many vectors (if not all) from the future Krylov space. Hence, in view of absence of effective projection, the iterative reduc-

tion of the error will be dictated only by the properties of the preconditioner in this case.

4.3. Computer realization

An important feature of parallel implementation is a clusterization of the solver. Each velocity component satisfies the momentum equation (11) independently of other components. Therefore, it is natural to parallelize the solution of (11) as follows. We split the set of available processors into three velocity component clusters. Each cluster solves the momentum equation for respective velocity component: the grid Ω_i^h is generated, the subdomain matrices are formed and factorized, right-hand sides of the linear systems are generated. The intercluster communication is needed only for the transport vector $\prod_i(\frac{3}{2}\mathbf{u}^k - \frac{1}{2}\mathbf{u}^{k-1})$ interpolation.

The pressure correction (12) has to be done after the momentum equation (11) solution. Hence, the pressure cluster may be formed on the basis of the velocity component clusters. Ideally, the pressure cluster should be the union of all the velocity clusters. Then the load balancing would be very good since no processor would stay idle. However, the current version of the parallel fast direct solver poses severe restrictions on the number of processors in the pressure cluster (2^n). As we will see later on, in practice the pressure cluster does not coincide with a velocity cluster. In order to pass the right-hand side of (12) to the pressure cluster and return the respective solution to the velocity clusters, we do apply intercluster communications as well.

Now we discuss briefly the parallel implementation of the Schwarz algorithm. Although it is possible to pipeline the Schwarz fractional steps associated with the global downwind propagation, we do not consider this opportunity since we deal with an algorithm which requires very few iterates and consequently makes the pipelining inefficient. In spite of the sequential realization of the downwind computations, we can parallelize the algorithm when solving the crosswind subproblems. More precisely, the l-processor treats subdomains $\Omega_{m,2l-1}$, $\Omega_{m,2l}$, and exchanges with the processors $l-1$, $l+1$, by the solution traces on $\partial\Omega_{m,l}$, $s = 2l-2,\ 2l+1$ (see Fig. 8).

Thus, given NPr_i processors in a velocity component cluster, the l-processor deals with data associated with the downwind strip $\bigcup_m (\Omega_{m,2l-1} \cup \Omega_{m,2l})$, $l = 1, \ldots, NPr_i$. We solve the subdomain problems exactly, taking advantage of a factorization technique. We motivate factorizing the respective matrices by the nature of the domain decomposition algorithm: first, we factorize matrices once only at each time step, second, the algorithm is well suited to a parallel computer which enables to reduce considerably both the dimensions of the subproblems and the natural bandwidth of the matrices, third, with factorization we avoid the problem of an optimal stop criterion for any iterative solver. Since we solve the grid subproblems in $\Omega_{k,s}$ by a factorization technique, for the same grid and the same overlap the increase of NPr_i would decrease the arithmetical complexity of the global Schwarz iteration. At the same time, with NPr_i growing the communications time does not vanish due to the latency time.

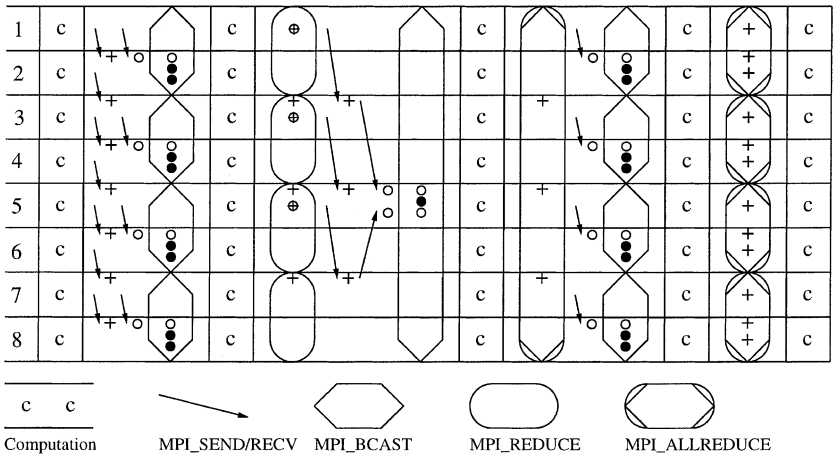

Fig. 9. A sample of communication pattern in the case of eight processors in the pressure cluster.

Data in the pressure cluster are also split in the downwind strips. The number of the "pressure" strips is equal to the number of processors in the pressure cluster. The number of nodes in any cross-section of the pressure strip is about the same for all processors from the cluster. Thus, a reasonable load balance has to be expected. The communications within the pressure cluster are not trivial (see Fig. 9 for the case of eight processors). It is pertinent to note that communications occur not only between neighbor-processors. It makes the pressure correction solver more heavy from the point of view of communications. The latter observation will be clearly illustrated by the parallel scalability measurements given in the next sections.

5. Numerical experiments and efficiency of the code in the steady case

We are going to report on numerical results and parallel efficiency of our method for a given steady flow configuration first.

5.1. Description of the test case for steady flow

An incompressible Newtonian fluid is considered. The kinematic viscosity is v and the density is ρ. The flow around a cylinder with circular cross-section has to be simulated [15]. The problem configuration and boundary conditions are illustrated in Fig. 10. The outflow condition can be selected by the user. The inflow condition is

$$U(0,y,z) = 16yz(H-y)(H-z)/H^4, \quad V = W = 0.$$

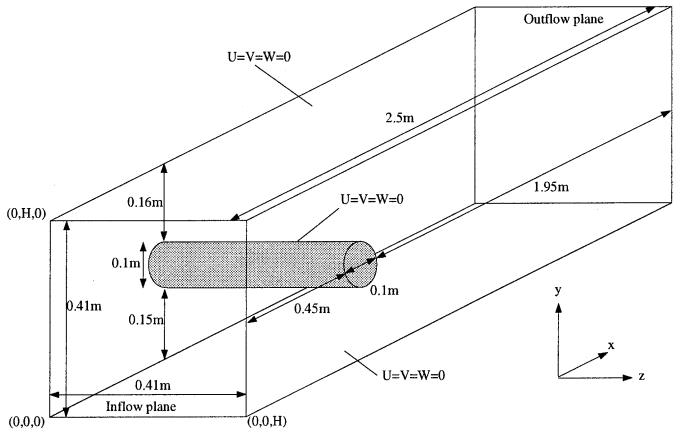

Fig. 10. Configuration and boundary conditions for flow around a cylinder with circular cross-section.

Some definitions are introduced to specify the values which have to be computed. The height and width of the channel is $H = 0.41$ m, and the side length is 2.5 m and diameter of the cylinder is $D = 0.1$ m. The characteristic velocity is $\bar{U}(t) = 4U(0, H/2, H/2, t)/9$, and the Reynolds number is defined by $Re = \bar{U}D/v$. The drag and lift forces are

$$F_D = \int_S \left(\rho v \frac{\partial u_t}{\partial \mathbf{n}} n_y - P n_x \right) \mathrm{d}S, \quad F_L = -\int_S \left(\rho v \frac{\partial u_t}{\partial \mathbf{n}} n_x + P n_y \right) \mathrm{d}S$$

with the following notations: surface of cylinder S, normal vector $\mathbf{n}$ on S with x-component n_x and y-component n_y, tangential velocity u_t on S and tangent vector $\mathbf{t} = (n_y, -n_x, 0)$. The drag and lift coefficients are

$$c_D = \frac{2F_D}{\rho \bar{U}^2 DH}, \quad c_L = \frac{2F_L}{\rho \bar{U}^2 DH}.$$

The case of $Re = 20$ ($v = 1/450$) corresponds to the steady case solution. The time step is $\Delta t = 0.01$, the number of time steps is 200. The Cartesian grids are refined near the cylinder location, as it is shown in Fig. 11.

In Table 1, we present the computed drag and lift coefficients on different grids for two types of the pressure gradient discretization. The central finite differences can still be used to discretize the convective term because the Reynolds number is not very large.

The data in Table 1 show that:

- The drag and lift coefficients are in reasonable agreement with referenced values [15];
- The symmetric choice of divergence/pressure gradient discretization is a bit better then unsymmetric one;
- The twofold coarsening of the grid in z-direction does not affect the drag and lift coefficients.

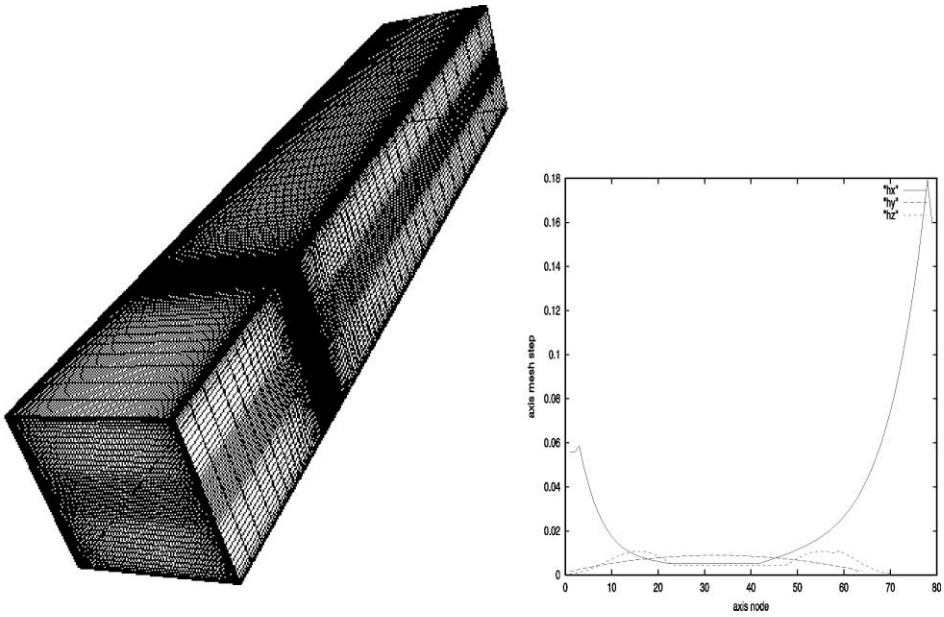

Fig. 11. Example of a Cartesian grid used in computations.

In the sequel, we present time measurements only for the symmetric pair of divergence/pressure gradient. In Figs. 12–14 we display the spatial distribution of the velocity modulo, pressure, and x-component of the velocity, respectively.

All numerical results reported in Section 5 have been obtained using a DEC TruCluster: this machine has six 4100 alpha servers (hypernodes) connected through a memory channel. Each alpha server has four Dec alpha processors running at 400 MHz, with 4 MB of cache memory and about 800 MB of shared memory. The maximum bandwidth of the memory channel is about 800 Mb/s.

5.2. Arithmetical scalability

The stop criteria for the momentum equation and the pressure correction solvers are chosen to be 10^{-7} and 10^{-5} for Euclidean norms of residuals. For the pressure correction it means relative reduction of the residual norm by a factor of 10^{-10}–10^{-7} depending on the time step (with zero initial guess). The projection onto the Krylov space produces an initial guess with residual norm as much as the norm of the right-hand side times 10^{-7}–10^{-5}! The number of GCR iterations needed to satisfy the stop criterion drops from 10–15 to 2–5 in case of fully developed flow. The values for stop criteria were chosen rather artificially. The criteria may be relaxed if arguments of stability were taken into account. Such a relaxation may cause better balance between the velocity and the pressure solvers. In Table 2 we

Table 1
Steady case: the drag and lift coefficients

Div/pres.grad	Coef.\grid	$40 \times 25 \times 72$	$80 \times 25 \times 72$	$40 \times 50 \times 72$	$80 \times 50 \times 36$	$80 \times 50 \times 72$	Ref. coef.
Symmetric	c_D	6.05	5.9	6.15	6.06	6.04	6.05–6.25
	c_L	0.013	0.01	0.0093	0.014	0.013	0.008–0.01
Unsymmetric	c_D	6.16	5.93	6.18	6.06	6.04	6.05–6.25
	c_L	0.015	0.012	0.0098	0.016	0.016	0.008–0.01

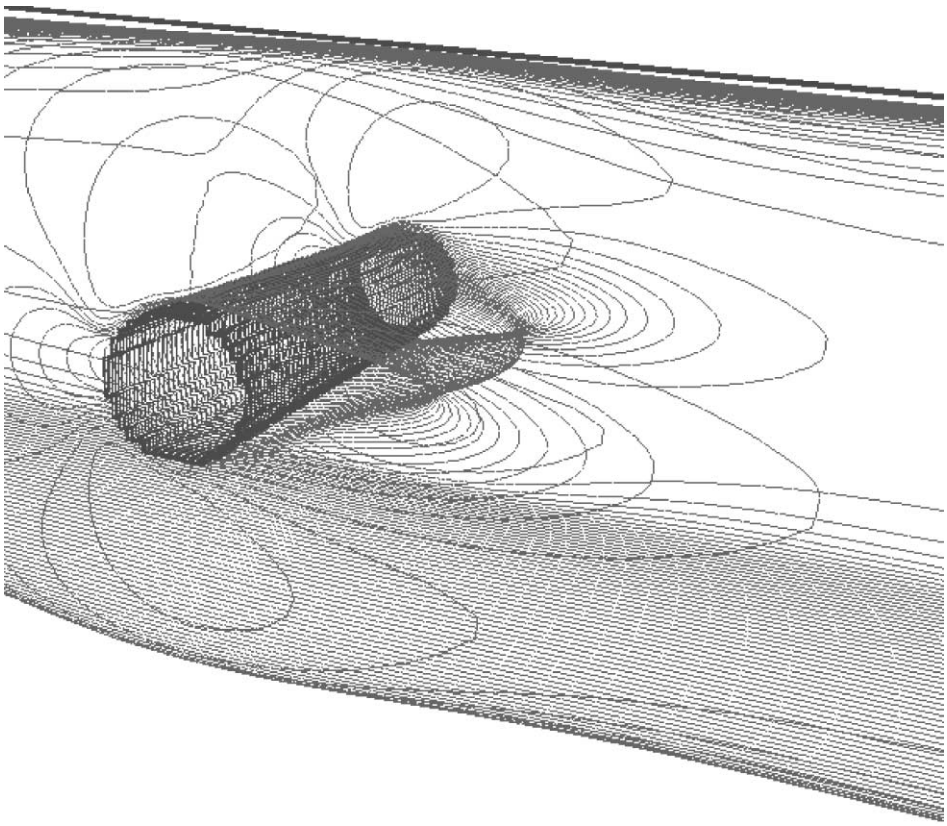

Fig. 12. Test case 1: the absolute value of the velocity.

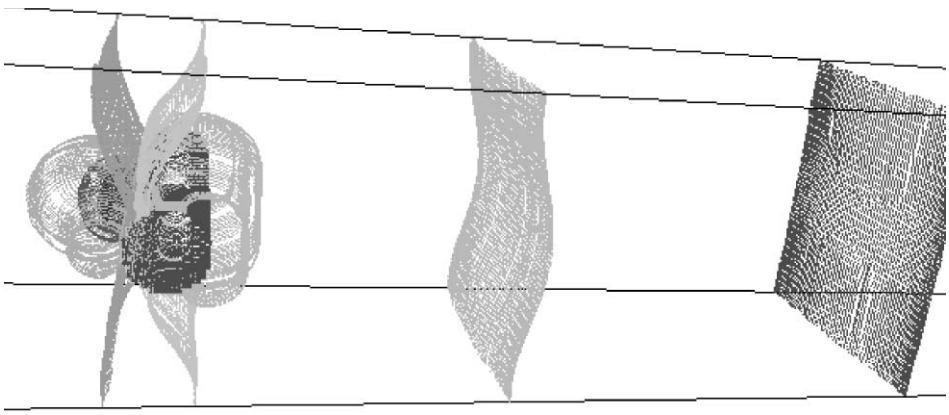

Fig. 13. Test case 1: the (normalized) pressure.

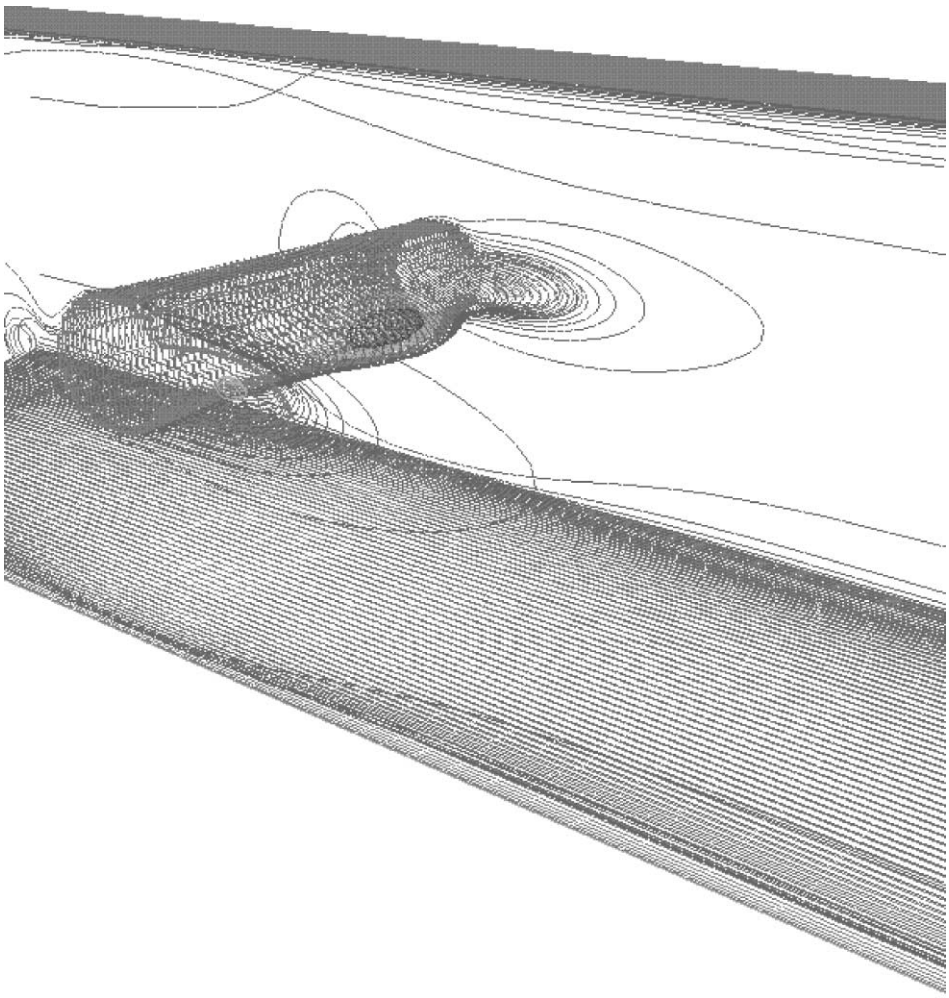

Fig. 14. Test case 1: u_1.

Table 2
Execution at 18 processors, number of time steps is 200

Grid	$80 \times 50 \times 72$	$40 \times 50 \times 72$	$80 \times 25 \times 72$	$40 \times 25 \times 72$
Exec. time (s)	6653	4080	5050	1439
Velocity solver				
Factor. time + Schwarz time	$442 + 953$	$210 + 425$	$207 + 480$	$98 + 227$
Total no. Schwarz iter.	1155	1099	1124	1091
Pressure solver				
GCR time	3619	2579	3495	550
Total no. GCR iter.	1630	2177	3140	2908

Table 3
Arithmetical scalability in terms of averaged execution time

Grid	$80 \times 50 \times 72$	$40 \times 50 \times 72$	$80 \times 25 \times 72$	$40 \times 25 \times 72$
Velocity solver				
Factor time at each time step	2.2	1.05	1.03	0.49
One Schwarz iter. time	0.82	0.39	0.42	0.21
Pressure solver				
One GCR iter. time	2.2	1.18	1.11	0.19

show time measurements obtained with 18 DEC alpha processors of our DEC TruCluster with different grids.

As it is seen, the total number of Schwarz iterations is insensitive to the grid which is essentially anisotropic. The total number of GCR iterations is affected by the grid. It is surprising that the smaller is the grid, the more iterations should be done. The reason is that small grids have the highest anisotropy.

Although we may judge the balance of the arithmetical load and overall elapsed time from the data presented in Table 2, precise conclusions may be derived from Table 3. Here we show the factorization time for the velocity solver at each time step, the average durations of one Schwarz and one GCR iterations. These data are independent of the number of iterations and may be used for the arithmetical scalability presentation.

We see that any averaged execution time is proportional to the number of nodes along x- and y-axes. The only exception is the duration of GCR iteration on small grid (last column). Such discrepancy is probably caused by the cache memory effect.

5.3. Parallel scalability

In order to measure a parallel scalability, we fix the arithmetical load per processor and increase the number of processors both in the velocity component and the pressure clusters. In Table 4 the total timing is shown. The following notations are

Table 4
Execution at NPr processors, number of time steps is 200

Grid	$80 \times 50 \times 20$	$80 \times 50 \times 36$	$80 \times 50 \times 72$
$NPr/NPr_i/NPr_0$	6/2/2	9/3/4	18/6/8
Exec. time	1485	2438	6653
Velocity solver			
Factor. time + Schwarz time	$325 + 418$	$442 + 556$	$442 + 952$
Total no. Schwarz iter.	808	824	1155
Pressure solver			
GCR time	315	680	3619
Total no. GCR iter.	998	1081	1630

used here. NPr stands for the total number of processors, NPr_i the number of processors in each velocity component cluster, and NPr_0 is the number of processors in the pressure cluster.

Similarly to the arithmetical scalability measurements, the variable number of iterations does not allow one to judge the actual parallel scalability. To this end, we consider the averaged time of factorizations at each time step, as well as averaged time of one Schwarz and one GCR iteration.

A parallel algorithm is scalable, if execution time (per iteration) does not increase as the number of processors grows. As it is seen from Table 5, it is not the case for both linear solvers. However, if we take into account that the actual computational load on the first grid per processor from velocity component clusters is 20% less than the loads on other grids, we get a satisfactory parallel scalability: 1.6 becomes 1.92, 0.52 becomes 0.63. The reason for bad parallel scalability for the pressure solver is not completely clear. We suggest three complementary reasons as follows:

- the overall arithmetical complexity of the fast direct method is $N \log N$ (N is the number of unknowns), and $\log N$ comes from the number of unknowns along the z-axis (which is split among processors);
- higher number of processors in the pressure cluster results in higher level of exchanges between processors (see Fig. 9), and the weight of communications increases;
- the memory channel that we use does not have enough hardware resources and MPI-exchange within a hypernode is faster than between hypernodes. On computers with more homogeneous physical links the pressure solver exhibits the good parallel scalability [10,11].

We have focussed our attention on the last issue by measuring intercluster communications. Though the main computations are localized within the clusters, the implementation requires intercluster communication. It is necessary for the interpolation of the velocity on different grids and gathering right-hand sides/broadcasting solutions of the pressure correction problem. The intercluster communication sends or receives global data associated with some downwind strips. It occurs three times per each time step. In Table 6 we present characteristic time for intercluster communication compared to overall execution time.

We are now going to extend our numerical experiments to unsteady flow.

Table 5
Parallel scalability in terms of averaged execution time

Grid	$80 \times 50 \times 20$	$80 \times 50 \times 36$	$80 \times 50 \times 72$
$NPr/NPr_i/NPr_0$	6/2/2	9/3/4	18/6/8
Velocity solver			
Factor. time at each time step	1.6	2.2	2.2
One Schwarz iter. time	0.52	0.67	0.82
Pressure solver			
One GCR iter. time	0.32	0.63	2.2

Table 6
Intercluster communications

Grid	$80 \times 50 \times 20$	$80 \times 50 \times 36$	$80 \times 50 \times 72$
$NPr/NPr_i/NPr_0$	6/2/2	9/3/4	18/6/8
Exec. time	1485	2438	6653
Intercluster comm.	45	87	295
No. of intercluster communications	600	600	600

6. Validation and efficiency of the code in the unsteady case

6.1. Description of the test cases for unsteady flows

We consider the same test case as above but with a Reynolds number $Re = 100$ in order to have the flow unsteady. The flow exhibits then periodic vortex separations. This test case will be named **3D-2Z** as in [15]. We will also consider the test case so-called **3D-2Q** for which the cylinder cross-section is a square and can be matched exactly or not with the Cartesian grid. The domain and type of boundary conditions are then no different from those in the steady case considered in Section 5. The following quantities are to be computed: maximum drag and lift coefficients and Strouhal number $St = Df/\bar{U}$. In this expression f is the frequency of separation of vortices. Computations were performed with different time steps on the grids $80 \times 70 \times 64$ (resp. $80 \times 60 \times 64$) with 24 processors of our DEC TruCluster (resp. a Cray T3E).

Our numerical grid is similar to the steady case described above, except that the aspect ratio of space step is as large as 60. The central finite difference approximation for the convective term appears then to be unstable on this grid, and the upwind finite differences are not accurate enough. We use then a linear combination of the central and upwind finite differences weighted properly depending on the cell Reynolds number. The weights are chosen in such a way that the scheme in the subdomain $\Omega \cap \{(x,y,z) \,|\, 0.1 < x < 1.5\}$ is pure central finite difference, and in the remaining part of Ω the upwind difference is substituted linearly for the central difference.

6.2. Validation of the method

In Table 7 we present the computed maximal drag and lift coefficients and Strouhal number. The time interval is $[0; 8]$.

Table 7
Unsteady case: the drag and lift coefficients and Strouhal number for the grid $80 \times 70 \times 64$

Δt	Computed			Referenced		
	$c_{D_{max}}$	$c_{L_{max}}$	St	$c_{D_{max}}$	$c_{L_{max}}$	St
0.01	3.23	−0.008	0.35	3.29–3.31	−0.011 to −0.008	0.29–0.35

The result of Table 7 is certainly not enough to validate our methodology, since the effect of the time stepping, the grid discretization and the robustness of the iterative solver should be investigated (see Tables 8–11).

Let us consider first the effect of time stepping.

Table 8
The effect of the shape of the domain. First-order of accuracy, the time step is 0.008, tolerance for velocity is 10^{-7}, tolerance for pressure is 10^{-4}

Cross-section of the obstacle	n_p	n_U
Circular	6	4
Square	3	4

Table 9
The effect of the time step[a]

Time step	n_p	n_U
0.02	10	8
0.01	7	5
0.008	6	4

[a] Circular cross-section of the cylinder, first-order of accuracy, tolerance for velocity is 10^{-7}, tolerance for pressure is 10^{-4}.

Table 10
The effect of the order of the spatial accuracy[a]

Order of accuracy	n_p	n_U
First	3	4
Second	3	4

[a] Square cross-section of the cylinder, the time step is 0.008, tolerance for velocity is 10^{-7}, tolerance for pressure is 10^{-4}.

Table 11
The effect of the tolerances for the linear solvers[a]

Absolute tolerance		n_p	n_U
Pressure	Velocity		
10^{-4}	10^{-7}	7	5
2×10^{-4}	2×10^{-4}	6	3

[a] Circular cross-section of the cylinder, first-order of accuracy, the time step is 0.01.

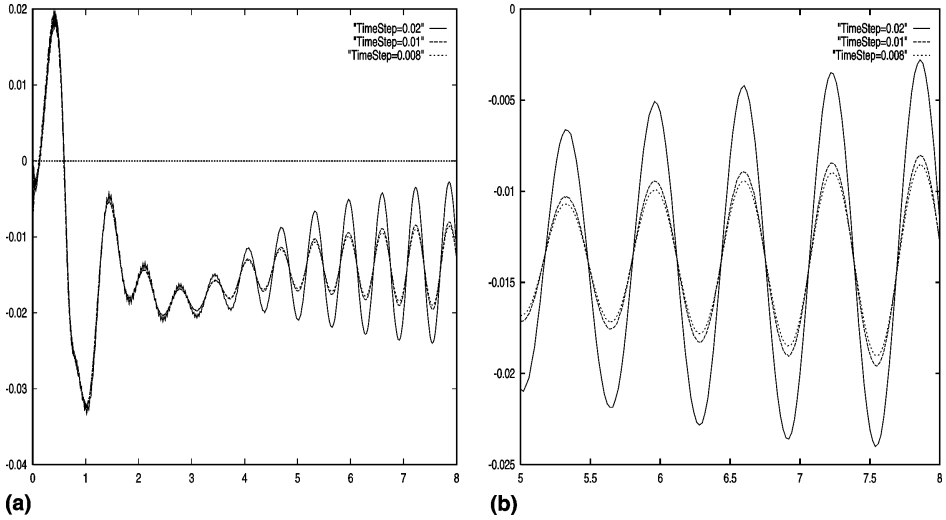

Fig. 15. The lift coefficient computed with different time steps (left) and a zoom of the plot (right).

6.2.1. Effect of the time stepping (3D-2Z)

In Fig. 15 we present the lift coefficient computed with different time steps.

The smaller time step gives the closer maximal value of the lift coefficient to the referenced value for the fully developed flow (−0.011:−0.008) [15,17]. However, the frequency of separation (and Strouhal number) are the same.

Let us notice however that we have restricted ourselves to constant time stepping. As a matter of fact, the adaptive time stepping for the projection technique is not very well known [17].

We proceed to the effect of the space grid.

6.2.2. Effect of the curved boundary (3D-2Q)

In order to check the sensitivity of the computation to the discretization of the curved boundary condition, we consider the cylinder with a square cross-section instead of a circular cross-section. We can then compare the results when the cross-section boundary points match the space grid, with the case where it does not. Though the reference solution for 3D-2Q is not known [15], we can compare the solutions due to different orders of approximations for the pressure gradient and the velocity divergence. In the first case the approximation is analogous to the case 3D-2Z with curved boundary approximations. In the second case, we shift the nodes of the grid in such a way that both the divergence and pressure operators are approximated with the second order in close-to-boundary cells. In Fig. 16 we show the lift coefficient in both cases.

The frequency of separation (and Strouhal number) are the same. Since no reference solution is known, we cannot conclude that the second order is better. At

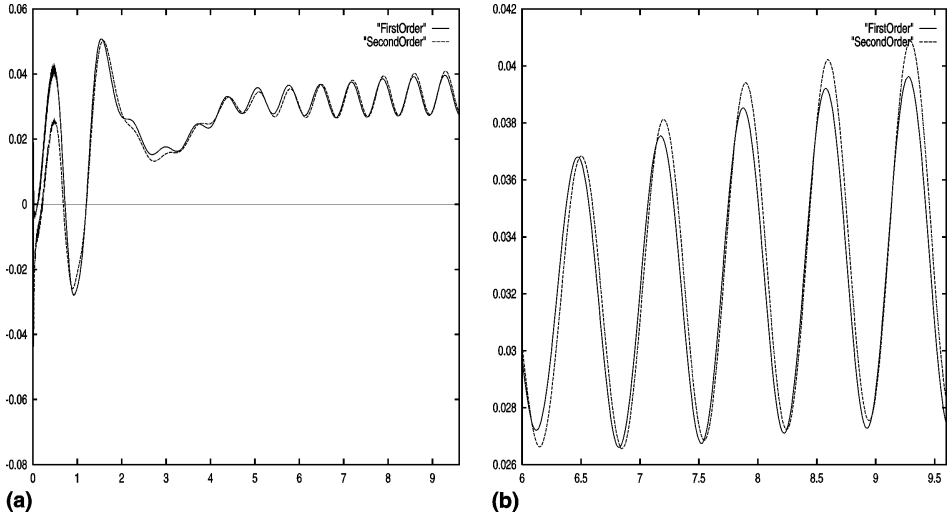

Fig. 16. The lift coefficient computed with different order of spatial accuracy (left) and a zoom of the plot (right). The time step is 0.008.

least, the discrepancy is rather small. This "advocates" our approach. Next, we are going to test the robustness of our iterative solvers with respect to various parameters.

6.2.3. Convergence of iterative methods

In Tables 11–13 below we consider the effect of the domain shape, time step, order of accuracy of the numerical approximation and tolerances for the stop criterion in

Table 12
Execution time for unsteady case[a]

Computer	DEC TruCluster	Cray T3E
Grid	$80 \times 70 \times 64$	$80 \times 60 \times 64$
$NPr/NPr_i/NPr_0$	24/8/8	24/8/16
Exec. time	34611	12305
Velocity solver		
Factor. time + Schwarz time	$1306 + 3898$	$1854 + 4462$
Total no. Schwarz iter.	4043	4049
Pressure solver		
GCR time	21150	3090
Total no. GCR iter.	4912	3882

[a] Total number of nodes is 373815 ($80 \times 70 \times 64$), and 321165 ($80 \times 60 \times 64$), $\Delta t = 0.01$, the number of time steps is 800.

Table 13
Convergence and execution time on 24 processors of Cray T3E[a]

Solver	Velocity	Pressure	Velocity	Pressure
Stop criterion, $\|.\|_{L_2}$	10^{-7}	10^{-4}	2×10^{-4}	2×10^{-4}
Total number iterations	4049	3882	2498	3439
Exec. time of iterations	4462	3090	2757	2737

[a] Total number of nodes is 321165, the number of time steps is 800.

iterative solvers on the convergence of linear solvers. By n_p and n_U we denote the mean number of iterations for the pressure and velocity solvers, per time step, respectively.

These results exemplify the robustness of our iterative solvers. Let us compare now the elapse time to run our code with two different parallel computers.

6.3. Comparison of the performances with two different parallel architectures

In Table 12 we exhibit time measurements on DEC TruCluster and Cray T3E-750 for 800 time steps with $\Delta t = 0.01$. The DEC alpha processors of the Cray computer have 375 MHz clock, the bandwidth between two neighbor processors is 480 MB/s and the latency is of the order of one microsecond. The stop criterion for pressure correction is relaxed to 10^{-4} (absolute value of residual norm). As it can be seen from Table 12, there is a dramatic difference of performance for the pressure solver between the CrayT3E (24 processors) and the DEC TruCluster (18 processors) elapse time. In the mean time the velocity solver requires slightly more time on the Cray T3E than on the DEC TruCluster. This last result is coherent with the ratio of clock speed of alpha processors on both computers. In addition, for the velocity solver, vector data exchange takes place only between "neighboring" processors whereas for the pressure solver vector data exchange occurs between processors which are "neighbors" on different levels of the processor network, see Fig. 9. In case of large amount of data, the intercluster communications on the DEC TruCluster induced by the pressure solver slow down the computation. In particular, the pressure solver on 16 processors runs *slower* than on eight processors, though the velocity solver shows good parallel scalability (see Table 5). Alternatively, both solvers are scalable on Cray T3E. Our main conclusion is that we do need a different two-level parallel algorithm specific to the multicluster architecture for the pressure solver that is less sensitive to the elapse time and bandwidth of the inter-cluster communication network [4].

In order to illustrate the effect of stop criteria on the computational time, we present in Table 13 the execution time and total number of iterations depending on the value of the tolerance for the residual norms.

7. Conclusion

In this paper, we have focussed our attention on fast and robust parallel solvers for the Laplace operator and singularly perturbed convection–diffusion–reaction operator that appeared to be essential components of the incompressible Navier–Stokes solver. We have restricted ourselves to simple stretched 3D Cartesian grids and validated our methods with the computation of lift and drags for internal flow past a cylinder.

Two different parallel techniques were used for solving the momentum equation and the pressure correction. The two-level Schwarz method based on overlapping domain decomposition was applied to the momentum equations and the fictitious domain method with parallel separable preconditioner (divide and conquer direct method) was used as a pressure solver. Both of them reveal high parallel efficiency on parallel computer with uniform architecture as the Cray T3E. On non-uniform parallel architecture as multiclusters, we have shown that the solver for the momentum equation is still scalable. However, the pressure solver can suffer from intercluster communications, when one uses a divide and conquer direct solver for the preconditioner. For heterogeneous architectures, the solvers are to be adjustable to the hierarchy of intercluster links [4].

References

[1] E. Dyakonov, Asymptotic Minimization of Computational Work, Nauka, Moscow, 1989.
[2] C. Fletcher, Computational Techniques for Fluid Dynamics, II, Springer, Berlin, 1988.
[3] M. Garbey, Yu.A. Kuznetsov, Yu.V. Vassilevski, Parallel Schwarz method for a convection–diffusion problem. SIAM J. Sci. Comp. 22 (3) (2000) 891–916.
[4] M. Garbey, D. Tromeur-Dervout, Two level domain decomposition for multiclusters, in: Proceedings of the 12th Int. Conf. on Domain Decomposition Methods, Chiba, Japan, Oct. 1999.
[5] M. Garbey, Yu.A. Kuznetsov, Yu.V. Vassilevski, Parallel Schwarz algorithm for a problem with a singular-perturbed convection-diffusion operator, Preprint CDCSP-97-08, Université Claude Bernard Lyon 1, 1997; SIAM J. Sci. Comp. 2000, to appear.
[6] G. Kobel'kov, V. Valedinskiy, On the inequality $\|p\|_{L_2} \leqslant c\|\nabla p\|_{W_2^{-1}}$ and its finite dimensional image, Soviet J. Numer. Anal. Math. Modelling 1 (1986) 189–200.
[7] G. Marchuk, Y. Kuznetsov, A. Matsokin, Fictitious domain and domain decomposition methods, Soviet J. Numer. Anal. Math. Modelling 1 (1986) 3–35.
[8] R. Peyret, T.D. Taylor, Comput. Methods Fluid Flow, Springer, Berlin, 1985.
[9] Proceedings of Domain Decomposition Conferences, http://www.ddm.org.
[10] T. Rossi, J. Toivanen, A parallel fast direct solver for the discrete solution of separable elliptic equations, in: Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing, SIAM, Philadelphia, 1997.
[11] T. Rossi, J. Toivanen, A parallel fast direct solver for the discrete solver for block tridiagonal systems with separable matrices of arbitrary dimension, SIAM J. Sci. Comput. 20 (1999) 1778–1793.
[12] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing Co, Boston, 1996.
[13] A. Samarski, V. Andréev, Méthodes aux Différences pour Équations Elliptiques, MIR, Moscow, 1978.
[14] A. Samarskii, E. Nikolaev, Numerical Methods for Grid Equations, Birkhäuser, Basel, 1989.
[15] M. Schefer, S.Turek, Benchmark computations of laminar flow around a cylinder, in: E.H. Hirschel (Ed.), Flow Simulation with High-Performance Computers II, Notes on Numerical Fluid Mechanics, vol. 52, Vieweg, Braunschweig, 1996, pp. 547–566.

[16] D. Tromeur-Dervout, Résolution des Equations de Navier–Stokes en Formulation Vitesse Tourbillon sur Systèmes multiprocesseurs à Mémoire Distribuée, Thesis, University of Paris VI, 1993.
[17] S. Turek, Efficient Solvers for Incompressible Flow Problems: an Algorithm Approach in View of Computational Aspects, Springer, Berlin, 1999.