Journal of Computational Physics 228 (2009) 7596–7613

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A Fictitious Domain, parallel numerical method for rigid particulate flows

Jordi Blasco [a,*,1], M. Carmen Calzada [b,2], Mercedes Marín [b,2]

[a] Departament de Matemàtica Aplicada I, Univ. Politècnica de Catalunya, Campus Sud, Edifici H, Avgda. Diagonal 647, 08028, Barcelona, Spain
[b] Departamento de Informática y Análisis Numérico, Univ. de Córdoba, Campus de Rabanales, Ed. C2-3, E-14071, Córdoba, Spain

ARTICLE INFO

Article history:
Received 28 October 2008
Received in revised form 6 July 2009
Accepted 16 July 2009
Available online 25 July 2009

Keywords:
Parallel computation
Particulate flows
Navier–Stokes equations
SDI algorithm
Fictitious Domain method

ABSTRACT

In this paper, we develop a Fictitious Domain, parallel numerical method for the Direct Numerical Simulation of the flow of rigid particles in an incompressible viscous Newtonian fluid. A Simultaneous Directions Implicit algorithm is employed which gives the model a high level of parallelization. The projection of the fluid velocity onto rigid motion on the particles is based on a fast computational technique which relies on the conservation of linear and angular momenta. Numerical results are presented which confirm the ability of the proposed method to simulate the sedimentation of one and many particles; the parallel efficiency of the algorithm is also assessed.

1. Introduction

The Direct Numerical Simulation (DNS) of the motion of rigid particles in an incompressible fluid has been an active area of research in the last decade and many different numerical schemes have been proposed for that purpose. The main difficulties of this problem are the treatment of the time-varying fluid domain, the enforcement of rigidity on the particles, the advection of the particles by the fluid and the computation of the reciprocal forces that the particles and the fluid exert on each other; this difficulties are added to those already encountered in the solution of transient incompressible flow problems for a single-phase fluid. Solid–liquid two phase flows, however, are of practical importance in different fields such as chemistry, food industries and geophysical flows, among others.

The earliest numerical approaches to this problem were commonly based on unstructured body-fitted meshes and a remeshing technique to discretize the actual fluid domain in each time step. A stabilized space-time Finite Element formulation was used in [25] in this sense, and an Arbitrary–Lagrangian–Eulerian (ALE) method was employed in [22–24,28] to move the mesh according to the particle motion.

But the most widespread approach, which was introduced by Glowinsky and his coworkers (see [20,21,29,30,32] and the references therein), is based on the concept of Fictitious Domain (FD). The basic idea consists in assuming that the whole domain containing both the fluid and the particles is filled with fluid in order to solve the hydrodynamical problem; moreover, rigidity has to be dynamically enforced on the particle domain. The great advantage of FD methods is that a single mesh can be used, which avoids the need for remeshing and mesh projections. Besides, since the embedding computational

* Corresponding author.
E-mail addresses: jorge.blasco@upc.edu (J. Blasco), ma1canam@uco.es (M.C. Calzada), ma1mabem@uco.es (M. Marín).
URL: http://www.ma1.upc.edu/~blasco (J. Blasco).

[1] This author's work was partially supported by the Spanish MEC under Projects MTM2005-07660-C02-01 and MTM2006-07932.
[2] These authors' work was partially supported by the Spanish MEC under Project MTM2006-07932.

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.07.010

domain usually has a simple geometry, structured meshes can be used, which is an advantage for parallel computations. Other FD methods have also been proposed in [5,7,8,33,37].

The way rigidity is enforced on the particles on FD methods affects directly their efficiency and several variants have been developed. The Distributed Lagrange Multiplier (DLM) approach (see [20,21]) is again the most popular technique. In this scheme, a Lagrange multiplier associated to the rigid solid condition for the velocity on the particles is incorporated into the weak formulation of the problem, just as the pressure arises in incompressible flow problems to enforce incompressibility. A FD/DLM method was also considered in [29], but this time it was associated to the vanishing of the deformation on the particles, which implies rigidity; also, a formulation using a global Lagrange multiplier with support in all the domain was presented in [7]. Other FD/DLM methods can be found in [8,37]. The DLM approach, however, requires a separate mesh for the particles to compute the Lagrange multiplier, and mesh projections are needed; moreover, the computational burden of the velocity projection onto rigid motion can rise significantly for large numbers of particles.

A fast computational technique for the enforcement of rigidity on the particles in FD methods was developed by Patankar and Sharma in [30,32]. The idea is based on the conservation of linear and angular momenta in the particle motion; the translational velocities of the particles are efficiently computed by simple integrals on the particles, and the fluid velocity is explicitly projected onto rigid motion. The simplicity of this approach makes it adequate for handling large numbers of particles. A similar technique, which is also non-Lagrange multiplier based, was considered in [33].

A novel approach for the imposition of rigidity on solid objects immmersed in fluids was introduced by Wan and Turek in [34,35] in the context of a multigrid finite element method. Their technique is based on the concept of Fictitious Boundary (FB), which basically imposes the rigid motion of the particles explicitly as if it were a Dirichlet boundary condition. In [36] these authors incorporated an ALE technique into their model to allow for mesh motion. More recently, other numerical methods have been proposed for particulate flows, such as Ref. [31], where the fluid velocity is matched to a spectral solution of Stokes flow around the particles; Ref. [5], which uses a level-set method to track the fluid-particle interface and a penalty formulation for rigidity; Ref. [6], which is based on a moving Lagrangian interface method and imposes rigidity by taking a very large value of viscosity on the particles and treating them as a very viscous fluid; and the immersed boundary method (see [26,27], for instance), in which bodies in the flow field are considered as a kind of momentum forcing in the fluid flow equations, rather than real bodies, which allows to employ orthogonal grids even on irregular geometries.

Our aim in this paper is to develop an efficient numerical model for the Direct Numerical Simulation of solid–liquid two phase flows. The method we propose is based on a Fictitious Domain formulation and uses the fast computational technique of Patankar and Sharma ([30,32]) for the enforcement of rigidity on the particles, relying on the conservation of linear and angular momenta. As in most other numerical models for particulate flows, a fractional step method is employed for the time advancement, with a first substep for the fluid motion and a second substep for the rigid body projection of the intermediate velocity.

Parallel computation is nowadays a useful tool for improving the efficiency of numerical algorithms. The parallel Simultaneous Directions Implicit method (SDI) was developed and analyzed in [14–18] for the numerical solution of elliptic and parabolic problems and then extended to the incompressible Navier–Stokes equations. A high level of parallelization is achieved with this method since it allows to reduce the problem to the solution of independent 1D differential equations on each of the *integrable segments* of the mesh. We use the SDI parallel algorithm in our particulate flow model for the solution of the fluid flow phase. As will be shown, the efficiency of the SDI algorithm is not diminished by the need to account for the presence of the particles, although the computation of their motion is essentially sequential.

The outline of the paper is the following: Section 2 introduces the mathematical model problem. Section 3 describes the Fictitious Domain parallel numerical method and in Section 4 numerical results are presented, where the sedimentation of one and many particles is simulated; a parallel efficiency analysis is also reported. Finally, some conclusions are drawn.

2. Governing equations

In this Section we describe the equations which govern the motion of rigid particles immersed in a two-dimensional viscous incompressible fluid. The formulation presented here can be easily extended to three-dimensional problems.

2.1. Fluid motion

Let $\Omega \subset \mathbb{R}^2$ be a polygonal domain, the boundary of which we call Γ, occupied by a viscous incompressible fluid (which we assume to be Newtonian and homogeneous) and by N_p rigid particles. We call $P_i(t) \subset \Omega$ the region occupied by particle $i (i = 1, \ldots, N_p)$ at time $t \in [0, T]$ (with $T > 0$ given), and let $P(t) = \cup_{i=1,\ldots,N_p} P_i(t)$. We assume that the initial position of the particles $P(0) \subset \Omega$ is known, and that the initial velocity field of the fluid $\mathbf{u}_0$ is given on $P(0)$. The motion of the fluid is governed by the unsteady, incompressible Navier–Stokes equations on the fluid domain in each time, together with the appropriate boundary and initial conditions:

$$\rho_f\left(\frac{\partial \mathbf{u}}{\partial t}+(\mathbf{u}\cdot\nabla)\mathbf{u}\right)-\mu\Delta\mathbf{u}+\nabla\mathcal{P}=\rho_f\mathbf{g}\quad \text{in } \Omega\setminus\overline{P(t)},\ t\in(0,T), \tag{1}$$

$$\nabla\cdot\mathbf{u}=0\quad \text{in } \Omega\setminus\overline{P(t)},\ t\in(0,T), \tag{2}$$

$$\mathbf{u}=\mathbf{0}\quad \text{on } \partial\Omega,\ t\in(0,T), \tag{3}$$

$$\mathbf{u}=\mathbf{u}_i\quad \text{on } \partial P_i(t),\ t\in(0,T), \tag{4}$$

$$\boldsymbol{\sigma}\cdot\mathbf{n}=\mathbf{t}_i\quad \text{on } \partial P_i(t),\ t\in(0,T), \tag{5}$$

$$\mathbf{u}=\mathbf{u}_0\quad \text{in } \Omega\setminus\overline{P(0)},\ t=0. \tag{6}$$

Here, $\rho_f>0$ is the fluid density, $\mathbf{u}$ the fluid velocity, $\mathcal{P}$ the fluid pressure, $\mu>0$ the fluid dynamic viscosity (we call $p=\mathcal{P}/\rho_f$ the kinematic pressure and $\nu=\mu/\rho_f$ the kinematic viscosity), $\mathbf{g}=(0,-g)$ the gravitational acceleration, $\mathbf{u}_i$ is the velocity of particle i at its surface, $\boldsymbol{\sigma}=-\mathcal{P}\mathbf{I}+\mu(\nabla\mathbf{u}+\nabla\mathbf{u}^t)$ is the stress tensor, $\mathbf{n}$ the (fluid) outward normal at the particle surface and $\mathbf{t}_i$ the traction. We have considered the homogeneous Dirichlet condition (3) on the external boundary of Ω for simplicity, while at the fluid-particle interface $\partial P_i(t)$, (4 and 5) are the kinematic and dynamic conditions, respectively. The initial velocity field $\mathbf{u}_0$ in (6) is assumed to satisfy $\nabla\cdot\mathbf{u}_0=0$ in $\Omega\setminus\overline{P(0)}$.

2.2. Particle motion

The motion of the particles, which are assumed to be homogeneous rigid solids, is governed by the Newton and Euler equations. For $i=1,\ldots,N_p$, let $\mathbf{X}_i(t)\in\mathbb{R}^2$ be the position of the center of mass of particle i at time t, $\mathbf{U}_i(t)\in\mathbb{R}^2$ its translational velocity, $\theta_i(t)$ the angular position of the particle with respect to a fixed axis and $\omega_i(t)$ its angular velocity. Writing $\boldsymbol{\omega}_i=(0,0,\omega_i)$, the following set of ordinary differential equations is satisfied:

$$\frac{d\mathbf{X}_i}{dt}=\mathbf{U}_i, \tag{7}$$

$$M_i\frac{d\mathbf{U}_i}{dt}=\mathbf{F}_i^h+\mathbf{F}_i^p+\mathbf{F}_i^w+(\rho_{s,i}-\rho_f)A_i\mathbf{g}, \tag{8}$$

$$\frac{d\theta_i}{dt}=\omega_i, \tag{9}$$

$$\frac{d}{dt}(I_i\boldsymbol{\omega}_i)=\mathbf{T}_i. \tag{10}$$

Here, M_i is the mass of particle i, $\rho_{s,i}$ the density, A_i the area and I_i the moment of inertia tensor. $\mathbf{F}_i^p$ and $\mathbf{F}_i^w$ are the repulsion forces acting on particle i due to the other particles and to the walls of Ω, respectively; expressions for these forces will be given in the next Subsection. Moreover, $\mathbf{F}_i^h$ and $\mathbf{T}_i$ are the hydrodynamical force and torque about the center of mass:

$$\mathbf{F}_i^h=-\int_{\partial P_i}\boldsymbol{\sigma}\cdot\mathbf{n}\,d\,\partial P_i,$$

$$\mathbf{T}_i=-\int_{\partial P_i}\mathbf{r}\times(\boldsymbol{\sigma}\cdot\mathbf{n})\,d\,\partial P_i,$$

where $\mathbf{r}$ is the position vector from the center of mass of the particle (2-dimensional vectors are extended by 0 to $\mathbb{R}^3$). We will assume in what follows that the particles are circular, and we call R_i the radius of particle i.

The set of Eqs. (7)–(10) is satisfied by the motion of the particles. We will explain in Section 3 in what way these equations are used in the proposed algorithm to account for that motion.

When needed, the hydrodynamical forces and torques can be computed using a step function on each particle, as in [36], which avoids the need to compute surface integrals by expressing them as volume integrals. Thus, for $i=1,\ldots,N_p$ and $\mathbf{X}\in\Omega$ let:

$$\alpha_i(\mathbf{X})=\begin{cases}1, \mathbf{X}\in P_i\\ 0, \mathbf{X}\in\Omega\setminus P_i\end{cases}$$

The gradient of such function is zero everywhere except at the surface of particle i, where it equals the unit normal vector to the particle surface, that is, $\nabla\alpha_i=\mathbf{n}_i$. The hydrodynamical forces and torques acting on particle i can then be computed by:

$$\mathbf{F}_i^h=-\int_\Omega\boldsymbol{\sigma}\cdot\nabla\alpha_i\,d\,\partial P_i,$$

$$\mathbf{T}_i=-\int_\Omega\mathbf{r}\times(\boldsymbol{\sigma}\cdot\nabla\alpha_i)\,d\,\partial P_i,$$

Since $\nabla\alpha_i=0$ everywhere except on ∂P_i, these integrals need only be computed in a narrow band around the particle surface.

2.3. Collision model

A collision model is needed to close the equation system and to prevent particles from interpenetrating each other and the walls. If overlapping occurs, conflicting rigid body motion constraints from different particles would arise at some velocity nodes. Based on physical models, Glowinski et al. ([21]) proposed repulsive force models in which an artificial short-range force is introduced to keep particles apart from each other and the walls. The range of the repulsive force is recommended to be taken of the order of the mesh size. Modified repulsive force models permitting particle ovelapping were proposed in [35]; however, we have adopted the original formulation of [21] and have observed numerically that using the optimal values of the force parameters recommended in that reference (and in particular, a sufficiently small value of the stiffness parameters ϵ_p and ϵ_w, see below) produces a large enough repulsive force to avoid overlapping. The repulsive force model between particles that we consider is, thus:

$$\mathbf{F}_i^p = \sum_{j=1, j\neq i}^{N_p} \mathbf{F}_{i,j}^p,$$

$$\mathbf{F}_{i,j}^p = \begin{cases} \mathbf{0}, & d_{i,j} > R_i + R_j + \delta \\ \frac{\pi R_i^2 \rho_{s,i} g}{\epsilon_p} \left(\frac{\mathbf{X}_i - \mathbf{X}_j}{d_{i,j}}\right) \left(\frac{R_i + R_j + \delta - d_{i,j}}{\delta}\right)^2, & d_{i,j} \leqslant R_i + R_j + \delta, \end{cases}$$

where $d_{i,j}$ is the distance between the centers of mass of particles i and j and the range δ and the stiffness ϵ_p are algorithmic parameters. The repulsive force with the walls is calculated, likewise, as:

$$\mathbf{F}_i^w = \sum_{j=1}^{N_w} \mathbf{F}_{i,j}^w,$$

$$\mathbf{F}_{i,j}^w = \begin{cases} \mathbf{0}, & d'_{i,j} > 2R_i + \delta \\ \frac{\pi R_i^2 \rho_{s,i} g}{\epsilon_w} \left(\frac{\mathbf{X}_i - \mathbf{X}'_{i,j}}{d'_{i,j}}\right) \left(\frac{2R_i + \delta - d'_{i,j}}{\delta}\right)^2, & d'_{i,j} \leqslant 2R_i + \delta, \end{cases}$$

where N_w is the number of sides of Γ, $d'_{i,j}$ the distance between the center of mass of particle i and that of a fictitious particle with radius R_i and tangent to side j and $\mathbf{X}'_{i,j}$ the position of the center of mass of that fictitious particle.

Finally, the initial positions and translational velocities of the centers of mass of the particles and their angular velocities are assumed to be given:

$$\mathbf{X}_i(0) = \mathbf{X}_i^0, \quad \mathbf{U}_i(0) = \mathbf{U}_i^0, \quad \omega_i(0) = \omega_i^0.$$

3. Numerical approximation

3.1. Fictitious Domain, fractional-step method

We describe in this section the numerical algorithm for the time integration of the flow problem just stated. A Fictitious Domain approach is employed here (see [5,7,8,20,21,29,30,32,33,37] and the references therein), in which the fluid flow is calculated assuming it fills the whole domain Ω, and the resulting velocity field is enforced to conform to rigid body motion on the particles. The solution of the flow equations (on a fixed mesh) is achieved by means of a parallel fractional-step method introduced and analyzed in [1,2]. On the other hand, the projection of the intermediate velocity field onto the rigid body motion is usually based on Distributed Lagrange Multipliers (DLM) associated with the rigidity restriction (see [7,8,20,21,29,37]), which are somehow incorporated into the weak form of the problem. However, in [30,32] Patankar and Sharma developed a fast computational technique to enforce the rigid body motion on the particles without using the DLM method. This technique, which we employ here, is based on the conservation of the linear and angular momenta on the particles, and it reduces the projection onto rigid motion to the computation of two integrals on each particle.

The fluid phase solution and the rigidity constrain are fully decoupled in this formulation. Thus, an increase in the number of particles does not result in an increase in the dimension of the discrete problem to be solved in the fluid phase, unlike in the DLM method where rigidity constrains are embedded in the augmented variational formulation of the fluid problem. Although the computation of collision forces between particles is of quadratic order in the number of particles, since the fluid step is the most time consuming part of the algorithm, the computational time is expected not to grow too much with an increase in the number of particles using this formulation.

The Fictitious Domain, fractional-step method that we use is described next. Let $\mathbf{X} \in \mathbb{R}^{2N_p}$, $\mathbf{U} \in \mathbb{R}^{2N_p}$ and $\boldsymbol{\omega} \in \mathbb{R}^{N_p}$ denote, respectively, the vectors containing the positions, translational and angular velocities of all particles, which will be written with superscripts corresponding to time levels $t_n = n\Delta t (n = 0, \ldots, [T/\Delta t] - 1)$, where $\Delta t > 0$ is a given time step size. Let $\mathbf{X}^0 = (\mathbf{X}_i^0)_{i=1,\ldots,N_p}$ and $\mathbf{U}^0 = (\mathbf{U}_i^0)_{i=1,\ldots,N_p}$. We compute an approximation P^0 of $P(0)$ and set $\mathbf{u}^0 = \mathbf{u}_0$ in $\Omega \setminus P^0$ and $\mathbf{u}^0 = \mathbf{U}_i^0 + \boldsymbol{\omega}_i^0 \times \mathbf{r}$ on each particle P_i^0. The time advancement from t_n to t_{n+1} is performed by the following three-step algorithm:

3.1.1. Step 1: Particle motion

Particles are moved at the start of each time step by means of the following explicit update algorithm, which is similar to that considered in [29,30,32]: let $\mathbf{X}^{n+1,0} = \mathbf{X}^n$ and for $k = 1, \ldots, K$ (K being the number of substeps) let

$$\overset{*}{\mathbf{X}}{}^{n+1,k} = \mathbf{X}^{n+1,k-1} + \mathbf{U}^n \frac{\Delta t}{K},$$
$$\mathbf{X}^{n+1,k} = \overset{*}{\mathbf{X}}{}^{n+1,k} + \left(\mathbf{F}^h + \frac{\mathbf{F}(\overset{*}{\mathbf{X}}{}^{n+1,k}) + \mathbf{F}(\mathbf{X}^{n+1,k-1})}{2} + \mathbf{G} \right) \frac{(\Delta t)^2}{2MK^2}.$$

Here, $\mathbf{F} = \mathbf{F}^p + \mathbf{F}^w$ is the sum of the repulsive forces between particles and with the walls of Ω, $\mathbf{F}^h$ accounts for the hydrodynamical forces on all particles and $\mathbf{G}$ for the gravitational acceleration. Finally, take $\mathbf{X}^{n+1} = \mathbf{X}^{n+1,K}$. Once $\mathbf{X}^{n+1}$ is known, a new approximation P^{n+1} of $P(t_{n+1})$ is calculated.

As can be observed, each iteration of this algorithm for the particle motion can be understood as a time-discretization of the combined Eqs. (7 and 8).

Remark 3.1. In the case of non-circular particles, Eqs. (9 and 10) would also be considered in this phase to track the angular positions of the particles. For two-dimensional particles, the new angular position θ_i^{n+1} of particle i could be found, for instance, as:

$$\theta_i^{n+1} = \theta_i^n + \Delta t\, \omega_i^n + \frac{(\Delta t)^2}{2} (T_i^z)^n$$

where $(T_i^z)^n$ represents the third component of the torque vector over particle i evaluated at $t = t_n$. The hydrodynamical force F_i^h and torque T_i and the collision forces F_i^p and F_i^w may be computed, for instance, by representing the boundary of particle i in polar coordinates with origin at the particle center of mass as a finite number of radii $\{R_i^l\}_{l=1,\ldots,L}$ associated to equally spaced angles $\phi_l = \frac{2\pi l}{L} (l = 1, \ldots, L)$. This extension is now under development.

3.1.2. Step 2: Fluid motion

The fluid motion is computed using a Fictitious Domain method. We assume in this step that the particles are also filled with fluid which has a density equal to that of the solid. The flow is computed on the whole domain on a fixed mesh, thus avoiding the need for remeshing in each time step. An intermediate velocity field $\tilde{\mathbf{u}}$ and a pressure field $\tilde{p}$ are calculated which satisfy:

$$\frac{\rho}{\rho_f} \left(\frac{\partial \tilde{\mathbf{u}}}{\partial t} + (\tilde{\mathbf{u}} \cdot \nabla) \tilde{\mathbf{u}} \right) - \nu \Delta \tilde{\mathbf{u}} + \nabla \tilde{p} = \frac{\rho}{\rho_f} \mathbf{g} \quad \text{in } \Omega \times (t_n, t_{n+1}),$$
$$\nabla \cdot \tilde{\mathbf{u}} = 0 \quad \text{in } \Omega \times (t_n, t_{n+1}),$$
$$\tilde{\mathbf{u}} = \mathbf{0} \quad \text{on } \Gamma \times (t_n, t_{n+1}),$$
$$\tilde{\mathbf{u}} = \mathbf{u}^n \quad \text{in } \Omega \times \{t_n\},$$

where $\rho = \rho_f (1 - H^{n+1}) + \rho_s H^{n+1}$, with $H^{n+1} = 1$ in $P^{n+1}, H^{n+1} = 0$ in $\Omega \setminus P^{n+1}$ and $\rho_s(t) = \rho_{s,i}$ in $P_i(t)$ for $i = 1, \ldots, N_p$.

Among the variety of numerical methods available nowadays to solve the Navier–Stokes problem, we have considered fractional-step methods in which the time advancement from t_n to t_{n+1} is decomposed into a number of substeps (see [3,4,11] and the references therein). This strategy allows to separate the different operators appearing in the equations and thus the different difficulties of the problem. More precisely, we use a parallel fractional-step method introduced in [1,2] in which two intermediate velocities are computed independently of each other associated to the effects of convection and incompressibility, respectively, both of which are affected by diffusion. The two velocities can therefore be calculated simultaneously, which provides a first level of parallelization to the resulting algorithm. Thus, a velocity $\mathbf{u}^{n+a}$ is obtained by solving the (nonlinear) Burger's problem:

$$\frac{\rho}{\rho_f} \left[\frac{\mathbf{u}^{n+a} - \mathbf{u}^n}{a \Delta t} + \frac{2\theta}{a} (\mathbf{u}^{n+a} \cdot \nabla) \mathbf{u}^{n+a} \right] - \nu \sigma \Delta \mathbf{u}^{n+a} = \frac{\rho}{\rho_f} \mathbf{g} + \nu (1 - \sigma) \Delta \mathbf{u}^n - \nabla p^n, \tag{11}$$
$$\mathbf{u}^{n+a} = \mathbf{0} \text{ on } \partial\Omega, \tag{12}$$

while another velocity $\mathbf{u}^{n+b}$ and a pressure p^{n+b} are obtained by solving the generalized Stokes problem:

$$\frac{\rho}{\rho_f} \frac{\mathbf{u}^{n+b} - \mathbf{u}^n}{b \Delta t} - \nu (1 - \sigma) \Delta \mathbf{u}^{n+b} + \nabla p^{n+b} = \frac{\rho}{\rho_f} \mathbf{g} + \nu \sigma \Delta \mathbf{u}^n - \frac{\rho}{\rho_f} \frac{2(1 - \theta)}{b} (\mathbf{u}^n \cdot \nabla) \mathbf{u}^n, \tag{13}$$
$$\nabla \cdot \mathbf{u}^{n+b} = 0, \tag{14}$$
$$\mathbf{u}^{n+b} = \mathbf{0} \text{ on } \partial\Omega. \tag{15}$$

The different algorithmic parameters appearing in these equations must satisfy $a, b > 0$ with $a + b = 2, \sigma \in (1/2, 1]$ and $\theta \in [0, 1]$ (see [1,2]). The two intermediate solutions are coordinated as $\tilde{\mathbf{u}}^{n+1} = (1/2)(\mathbf{u}^{n+a} + \mathbf{u}^{n+b})$. Finally, we take $p^{n+1} = p^{n+b}$ in Ω and $\mathbf{u}^{n+1} = \tilde{\mathbf{u}}^{n+1}$ in the fluid domain $\Omega \setminus P^{n+1}$.

3.1.3. Step 3: Projection onto rigid motion

The third and last step of the algorithm can be seen as a projection of the intermediate velocity field $\tilde{\mathbf{u}}^{n+1}$ onto the space of rigid body motions over the particles. The most common method to enforce the rigid body motion when using Fictitious Domain formulations, which was developed by Glowinski and his coworkers, consists in using distributed Lagrange multipliers (DLM) associated to the rigidity condition $\mathbf{u} = \mathbf{U}_i + \omega_i \times \mathbf{r}$ on each particle (see [7,8,20,21,29,37]). A different approach, based on the fictitious boundary method, was introduced by Wan and Turek in [34–36]. Furthermore, in [6] rigidity was enforced by using a very large value of the viscosity coefficient on the particles when solving the flow equations. In [30,32], however, Patankar and Sharma introduced a fast computational technique to compute the projection onto rigid motion which consists in imposing the following additional condition: in the projection step, the total linear and angular momenta should be conserved on each particle. Thus, the new translational and angular velocities of the particles can be computed, for $i = 1, \ldots, N_p$, as:

$$\mathbf{U}_i^{n+1} = \frac{\rho_{s,i}}{M_i} \int_{P_i^{n+1}} \tilde{\mathbf{u}}^{n+1}\, d\Omega,$$

$$\omega_i^{n+1} = \left(\int_{P_i^{n+1}} \mathbf{r} \times \tilde{\mathbf{u}}^{n+1}\, d\Omega \right) \Big/ \left(\int_{P_i^{n+1}} |\mathbf{r}|^2\, d\Omega \right).$$

and the end-of-step velocities are then obtained as:

$$\mathbf{u}^{n+1} = \mathbf{U}_i^{n+1} + \omega_i^{n+1} \times \mathbf{r} \quad \text{in } P_i^{n+1}.$$

The projection is thus reduced to the computation of these two integrals on each particle, which can be done fast using a nodal quadrature rule. As can be observed, in this Fictitious Domain formulation mutual forces cancel each other and there is no need to compute them. Moreover, according to [32] the problem is thus formulated in terms of the fluid equation and a rigid motion constrain in the particle domain, so that the particle equations of motion (7)–(10) are not explicitly used in this step of the algorithm. As we have seen before, Eqs. (7) and (8) are used, however, in the explicit update method of Step 1 to find the new particle positions. We use this fast computational technique to enforce the rigid body motion on the particles in our scheme.

Remark 3.2. The fluid-particle scheme that we have just described can be readily applied to more complex flow situations such as turbulent flow simulations, where instead of the molecular viscosity μ in Eq. (1) turbulent eddie viscosities μ_e are considered, which are provided by an adequate turbulence closure model. For that purpose, the fluid flow Eqs. (11) and (13) should be modified accordingly, but the particle equations would be unaltered. The coefficient β in the Helmholz's equations of the form of (16), needed to be solved with this scheme, may indeed depend on the viscosity; however, the coefficient α in (16) depends on the density ratio between the solid and the fluid in the regions occupied by the particles. Thus, the system matrices for the local (1D) problems (18) and (20) have to be recomputed every time step in the fluid-particle solver even in the laminar regime. Adding a turbulence model, therefore, would not require additional computational effort as far as the fluid-particle solver is concerned.

3.2. Simultaneous directions implicit (SDI) parallel algorithm

We now describe the spatial approximation of problems (11)–(15). For that purpose, we consider the parallel Simultaneous Directions Implicit algorithm (SDI), which was developed and analysed in [14–18]. This scheme reduces the solution of those problems to a number of one-dimensional independent Dirichlet problems for second-order differential equations, each one on each of the *integrable segments* of the mesh (see [17,18]); these problems can be solved in parallel, which gives the scheme a high level of scalability due to the large number of independent integrable segments that can eventually be used in a mesh.

In order to apply the SDI method to the present problem, the nonlinear Burger's Eqs. (11) and (12) are solved iteratively by a fixed point method and the generalized Stokes problem (13)–(15) is also solved iteratively by a conjugate gradient method (see [15] for the details). This allows to express each iteration as a Dirichlet problem for a Helmholz's equation such as: find $v \in H^1(\Omega)$ such that

$$\alpha v - \beta \Delta v = f \text{ in } \Omega \tag{16}$$

$$v = \bar{v} \text{ on } \partial\Omega \tag{17}$$

for some given constants $\alpha, \beta > 0$. In the SDI algorithm, problem (16, 17) is solved iteratively starting from some initial approximation v^0; then, in each iteration j, two independent approximations $v^{j+1,1}$ and $v^{j+1,2}$, one for each spatial direction, are computed simultaneously by solving the problems:

$$\left(I + \tau \left(\frac{\alpha}{2} I - \beta \frac{\partial^2}{\partial x_1^2} \right) \right) v^{j+1,1} = \left(I - \tau \left(\frac{\alpha}{2} I - \beta \frac{\partial^2}{\partial x_2^2} \right) \right) v^j + \tau f \text{ in } \Omega \tag{18}$$

$$v^{j+1,1} = \bar{v} \text{ on } \partial\Omega \tag{19}$$

and

$$\left(I+\tau\left(\frac{\alpha}{2}I-\beta\frac{\partial^2}{\partial x_2^2}\right)\right)v^{j+1,2}=\left(I-\tau\left(\frac{\alpha}{2}I-\beta\frac{\partial^2}{\partial x_1^2}\right)\right)v^{j}+\tau f \text{ in } \Omega \tag{20}$$

$$v^{j+1,2}=\bar{v} \text{ on } \partial\Omega \tag{21}$$

where $\tau > 0$ is the *evolution parameter*. Both solutions are coordinated at the end of the iteration by taking:

$$v^{j+1} = (\gamma/2)\,(v^{j+1,1} + v^{j+1,2}) + (1-\gamma)\,v^{j}$$

where $\gamma \in (0,1)$ is the *coordination parameter*. In order to solve these 1D problems, the domain Ω is discretized using a Cartesian grid. The solution of the independent problems (18)–(21) can in turn be solved independently on each of the *integrable segments* of the mesh, which are either horizontal or vertical. The solution of the problem on each segment is obtained by a second-order centered finite difference scheme, which leads to a small tridiagonal linear system on each segment that can be solved very fast.

The SDI algorithm has potentially a high level of parallelization since each integrable segment, both the horizontal and vertical ones, can eventually be solved on a different processor, if so many of them were available. This parallelization is independent of the space dimension or the number of implicit Cartesian directions of the problem. Another important

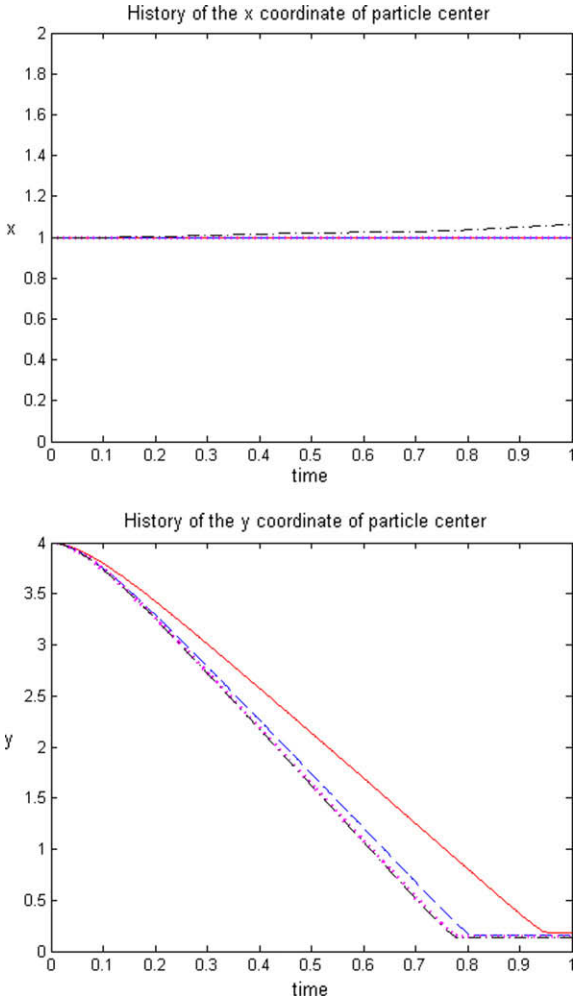

Fig. 1. Sedimentation of a circular particle: time history of the x (top) and y (bottom) coordinates of the position of the particle center. $\Delta t = 0.001.h = 1/25$ (solid, red line), 1/50 (dashed, blue line), 1/100 (dotted, magenta line) and 1/150 (dash-dotted, black line). (For interpretation of the references in colour in this figure legend, the reader is referred to the web version of this article.)

advantage of the SDI algorithm is that, although Cartesian meshes have to be used, they can be adapted to irregular domains since integrable segments are completely independent from one another, and the discretization within each segment can have variable size so as to fit the domain boundary (see [18] for some examples both in 2D and 3D). In order to illustrate this fact, a test case is presented in the next Section in which the domain is not a rectangle. Moreover, another positive fact is the possibility of combining SDI methods with multigrid techniques for the solution of Poisson problems, in view of the smoother properties of the former (see [17]). In fact, it was shown in that reference that the smoothing factor of the SDI algorithm improves that of standard smoothers such as Jacobi or Gauss-Seidel relaxation; furthermore, in [13] numerical examples were given of the solution of Poisson problems using SDI together with multigrid techniques, which led to a substantial reduction of execution times. We plan to incorporate these multigrid techniques to our fluid-particle solver in order to reduce the computational times.

4. Numerical results

In this Section we present some numerical results obtained with the Fictitious Domain parallel algorithm described in the previous Section on different problems related to the sedimentation of particles.

4.1. Sedimentation of a circular particle

We first present the results of three test cases related to the sedimentation of a unique circular particle.

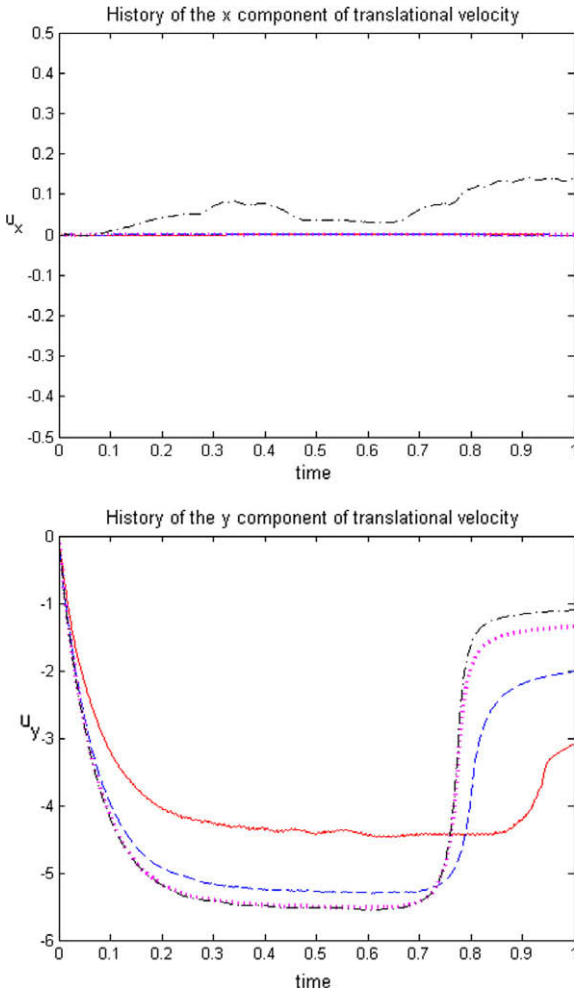

Fig. 2. Sedimentation of a circular particle: time history of the horizontal (top) and vertical (bottom) components of the translational velocity of the particle center. $\Delta t = 0.001.h = 1/25$ (solid, red line), 1/50 (dashed, blue line), 1/100 (dotted, magenta line) and 1/150 (dash-dotted, black line). (For interpretation of the references in colour in this figure legend, the reader is referred to the web version of this article.)

4.1.1. Grid and time-step convergence analysis

As a first test case for the proposed method, we consider the problem of the sedimentation of a rigid circular disk due to gravity in a bounded cavity filled with an incompressible, viscous Newtonian fluid, when the particle is dropped from the centerline of the cavity. Simulating the impact of the disk with the bottom boundary is part of the computational experiment. This flow problem has been studied before in [5,7,19,21,31,32,34,35], for instance.

The computational domain is a rectangular channel of dimensions 2×6. The particle radius is $R = 0.125$ and the particle center is initially situated at the point (1,4); both the fluid and the particle are initially at rest. The fluid viscosity is $\mu = 0.1$ and its density is $\rho = 1$, while the particle density is $\rho_s = 1.25$; gravity acceleration is $g = 980$. These parameter values are the same as those taken in one of the cases studied in references [19,34,35], with which we compare our results.

The method we propose can be efficiently implemented by using a parallel computation scheme based on a shared memory multiprocessor architecture. In this case, it has been implemented in a Sun X4600 server with 16 cores, using the OpenMP parallel computing model. All 16 cores have been used in the simulations performed for this test case.

We first present the results of a grid convergence analysis. Four different meshes were considered with mesh sizes of 1/25, 1/50, 1/100 and 1/150, respectively. A time step size of $\Delta t = 0.001$ was used in all cases, and the solution was computed until a final time of $T = 1$. The values used for the parameters of the repulsive forces were those recommended in [20], namely, $\delta = 2h, \epsilon_p = h^2$ and $\epsilon_w = h^2/2$, and 10 substeps were employed for the update of the position of the particle.

Fig. 1 shows the temporal evolution of the x and y coordinates of the center of mass of the particle. As can be observed, the y-positions obtained for $h = 1/100$ and $h = 1/150$ are practically identical, so that the solution can be considered to have converged for these meshes. However, a small symmetry braking takes place for the finest mesh; this is in agreement with the results of [21], where a small symmetry braking was observed for these values of viscosity and particle density with meshes of size $h = 1/192$ and smaller (symmetry braking increases for larger values of the particle density). On the other hand, symmetric solutions were reported in [35] for these values of the physical parameters with meshes of the order of 1/100, which is also in agreement with our results.

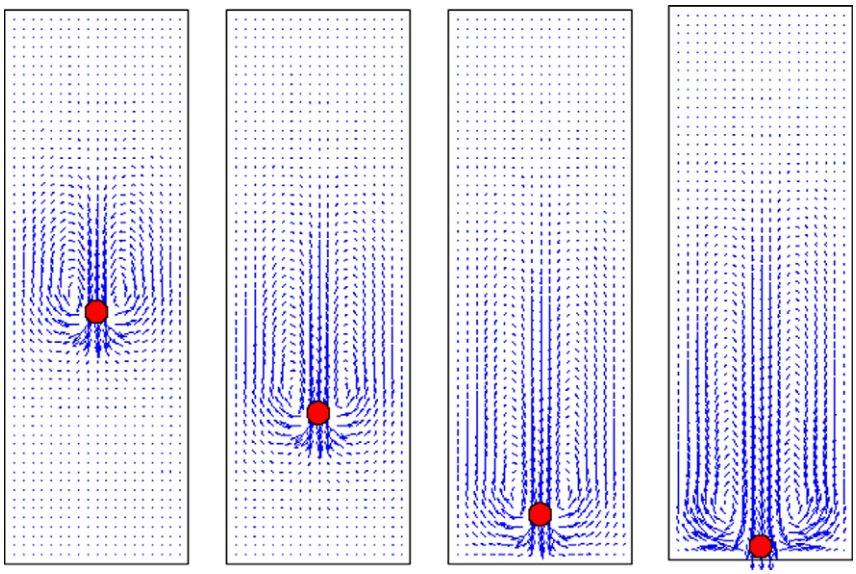

Fig. 3. Sedimentation of a circular particle: velocity field of the fluid and position of the particle at $t = 0.3$, 0.5, 0.7 and 0.9 obtained with $h = 1/100$ and $\Delta t = 0.001$.

Table 1
Sedimentation of a circular particle: number of nodal points, average computational time per time step (in seconds), terminal settling velocity and maximum Reynolds number for each mesh. $\Delta t = 0.001$.

h	Num. nodes	CPU/step	V	Re
1/25	7701	0.67	4.474	13.98
1/50	30401	5.95	5.302	16.57
1/100	120801	61.17	5.515	17.23
1/150	271201	250.74	5.550	17.34

Fig. 2 plots the evolution of the two components of the translational velocity of the center of mass of the particle; the small symmetry braking for $h = 1/150$ can be also observed there. Although the particle had sedimented at $T = 1$ in all cases (see Fig. 1), a longer simulation time would be needed in order for the particle velocity to become zero.

Fig. 3 shows the velocity field of the fluid and the position of the particle at different times: $t = 0.3, 0.5, 0.7$ and 0.9, obtained with $h = 1/100$. Symmetry is clearly maintained in these results.

As a more quantitative measure of the quality of the results obtained and the solution procedure, Table 1 gives the number of nodal points, the average computational time per time step (in seconds), the terminal velocity and the maximum particle Reynolds number for each mesh. The terminal settling velocity V, defined as the maximum of the particle translational velocity $\|\mathbf{U}(t)\|$, is one of the main quantities of interest in this problem. The maximum particle Reynolds number is then $\mathrm{Re}_p = \frac{VD\rho_s}{\mu}$, where D is the particle diameter; this definition was also used by Glowinski in [19] and by Wan and Turek in [34,35], where values for Re_p were given for this same test problem.

Glowinski obtained a value of $\mathrm{Re}_p = 17.31$ using a mesh of size $h = 1/256$ and a time step size of 7.510^{-4}, while the result of Wan and Turek in [34,35] was $\mathrm{Re}_p = 17.15$ with $h = 1/96$. Our results give $\mathrm{Re}_p = 17.34$ for $h = 1/150$ and $\Delta t = 10^{-3}$, which is remarkable close to the value given by Glowinski in [19] (with a relative difference of only 0.17%).

Furthermore, we next present the results of a convergence analysis in the time step size. This time we fixed a mesh of size $h = 1/100$ and considered four different time step sizes: $\Delta t = 0.008, 0.004, 0.002$ and 0.001. Fig. 4 shows the evolution of the y component of the position of the particle center and the y component of its translational velocity (the solution was symmetric about the cavity centerline in all cases). An erratic behavior is observed after sedimentation for the largest time step size, which is near the stability limit for this mesh size.

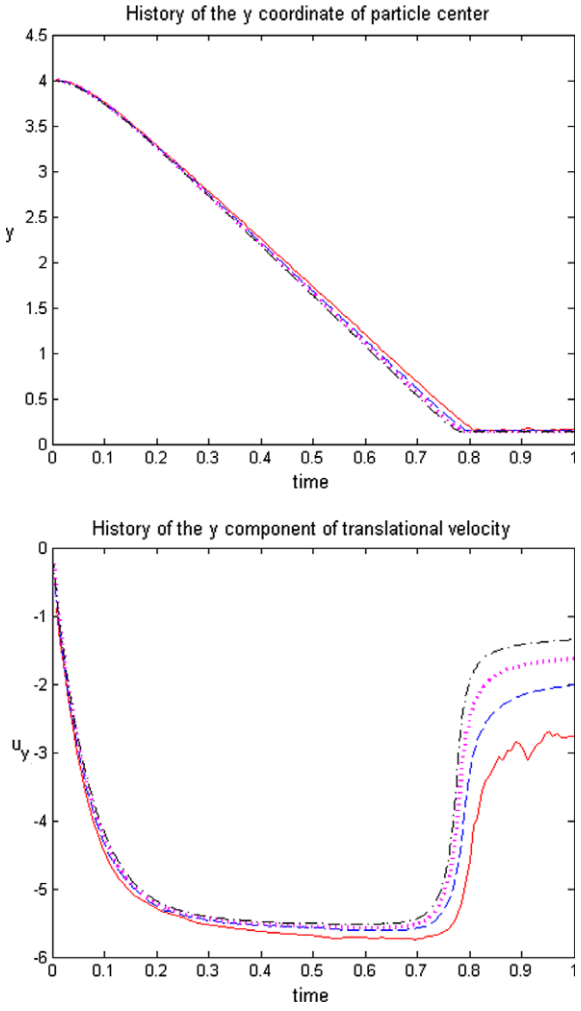

Fig. 4. Sedimentation of a circular particle: time history of the y coordinate of the position of the particle center (top) and the y component of its translational velocity (bottom). $h = 1/100$. $\Delta t = 0.008$ (solid, red line), 0.004 (dashed, blue line), 0.002 (dotted, magenta line) and 0.001 (dash-dotted, black line). (For interpretation of the references in colour in this figure legend, the reader is referred to the web version of this article.)

The computational time per time step, the terminal velocity and the maximum Reynolds number obtained with each time step are presented in Table 2.

4.1.2. Parallel efficiency study

This time, the algorithm has been implemented in a Fujitsu PRIMEPOWER 650 server with 6 processors, using the OpenMP parallel computing model. In order to measure the performance of the parallel algorithm, we introduce two parameters: the *speedup*, S_{NP} defined by

$$S_{NP} = \frac{\text{Resolution time with 1processor}}{\text{Resolution time with NP processors}},$$

and the *efficiency*

$$\eta = \frac{S_{NP}}{NP},$$

that represents the speedup for each processor.

Table 2
Sedimentation of a circular particle: average computational time per time step (in seconds), terminal settling velocity and maximum Reynolds number for each time step size. $h = 1/100$.

Δt	CPU/step	V	Re
0.008	107.03	5.742	17.94
0.004	53.03	5.613	17.54
0.002	57.74	5.573	17.41
0.001	61.17	5.515	17.23

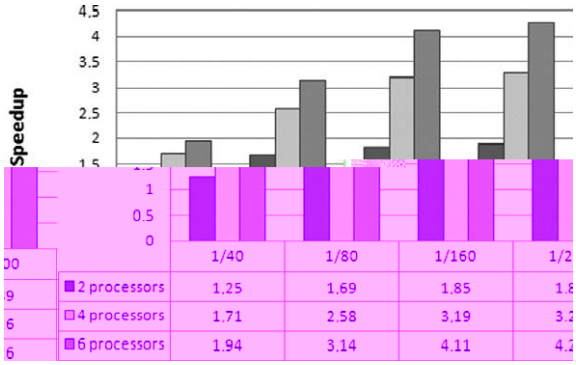

Fig. 5. Parallel execution: speedup for different meshes and number of processors.

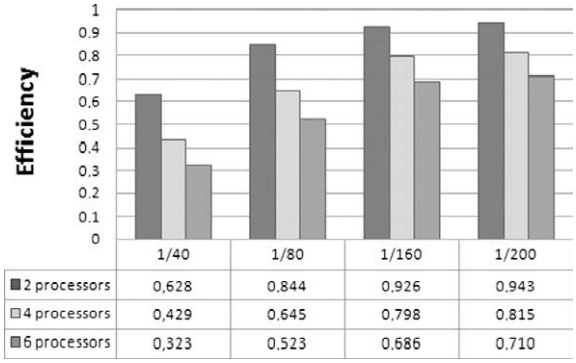

Fig. 6. Parallel execution: efficiency for different meshes and number of processors.

The computational domain this time is a rectangular channel of dimensions 2×8. Once again, the sedimentation of a single particle of radius $R = 0.3125$ is considered and the particle center is initially situated at the point (1,7); both the fluid and the particle are initially at rest. The fluid viscosity is $\mu = 0.025$ and its density is $\rho = 1$, while the particle density is $\rho_s = 1.5$.

Four uniform meshes were used to perform the calculations, with sizes $h = 1/40, 1/80, 1/160$ and $1/200$, respectively. The time step size was $\Delta t = 0.01$ in all four cases; 50 time steps were performed in each case. We used the same parameter values for the repulsive forces and the same number of substeps for the update of the position of the particle as in the previous example.

We have obtained results for different meshes and for 2, 4 and 6 processors. In Figs. 5 and 6, the corresponding speedup and efficiency are shown. These parallel performance results are comparable to those obtained in [18] for the SDI algorithm in the case of a linear elliptic problem; this shows that the incorporation of the particle motion into the algorithm does not degrade the efficiency of the SDI method.

4.1.3. Wall effects

In this third example the particle is initially located off the centerline of the cavity in order to study wall effects. This problem has been studied numerically in [9,10], among others. As in those references, a circular particle of diameter d is released in a rectangular channel of width $L = 4d$ and height $10L$ from a point $(x_0, 0)$ (the y coordinate increases downward). Both the particle and the fluid are initially at rest. We have taken $d = 0.25$ and $x_0 = 0.2$.

It is well known that in the absence of inertia the particle would drift straight down with no lateral motion. But as inertial forces increase, the lateral motion becomes more and more important. Different flow regimes occur as the Reynolds number

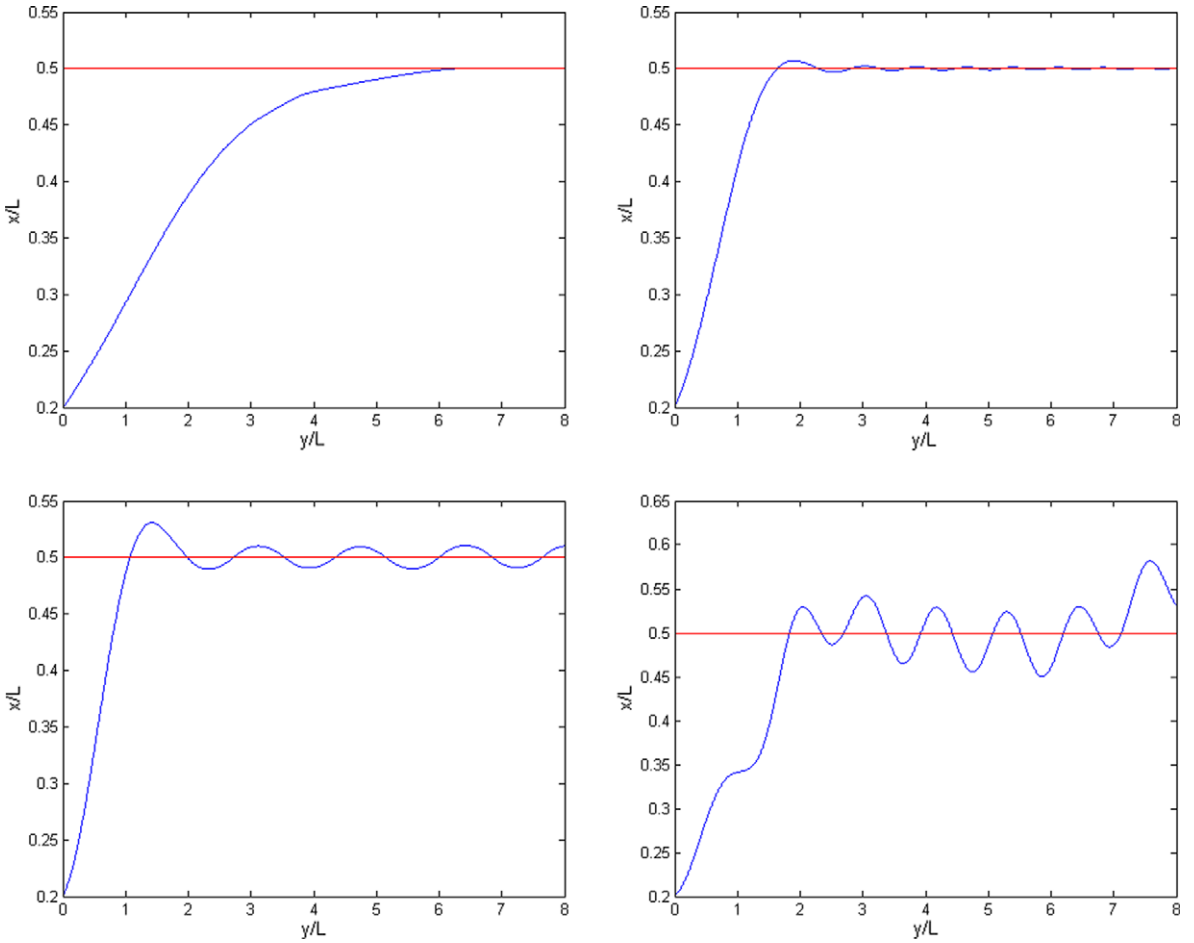

Fig. 7. Sedimentation of a non-centered circular particle: particle trajectory for Re = 0.85 (top left), 2.71 (top right), 7.80 (bottom left) and 260.81 (bottom right).

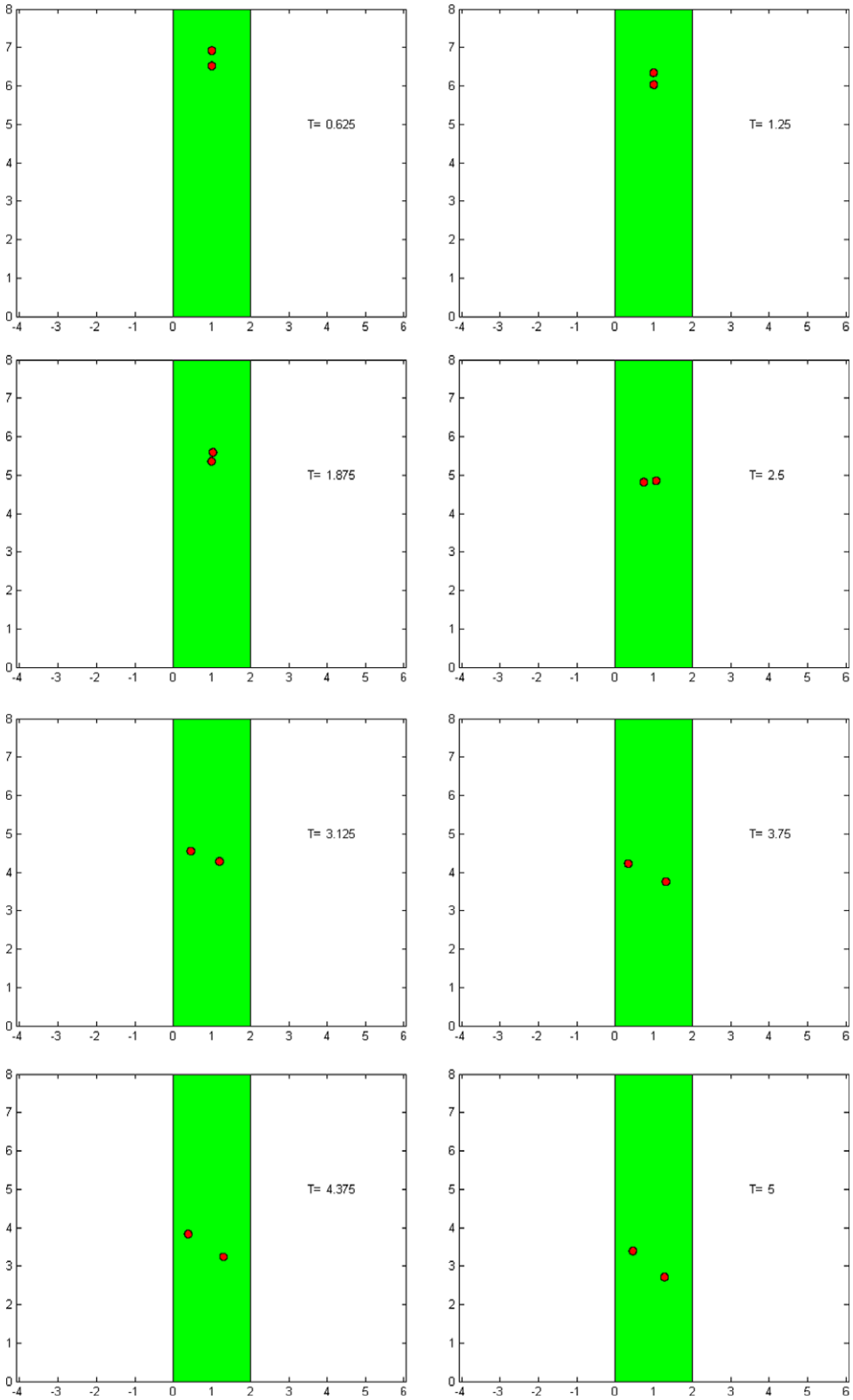

Fig. 8. Sedimentation of two circular particles: position of the particles at different times.

$\mathrm{Re} = \frac{Vd}{\nu}$ (V is the settling velocity of the particle) is increased. Five different flow regimes were clearly identified in [10], and Reynolds number intervals were proposed for each of them, namely:

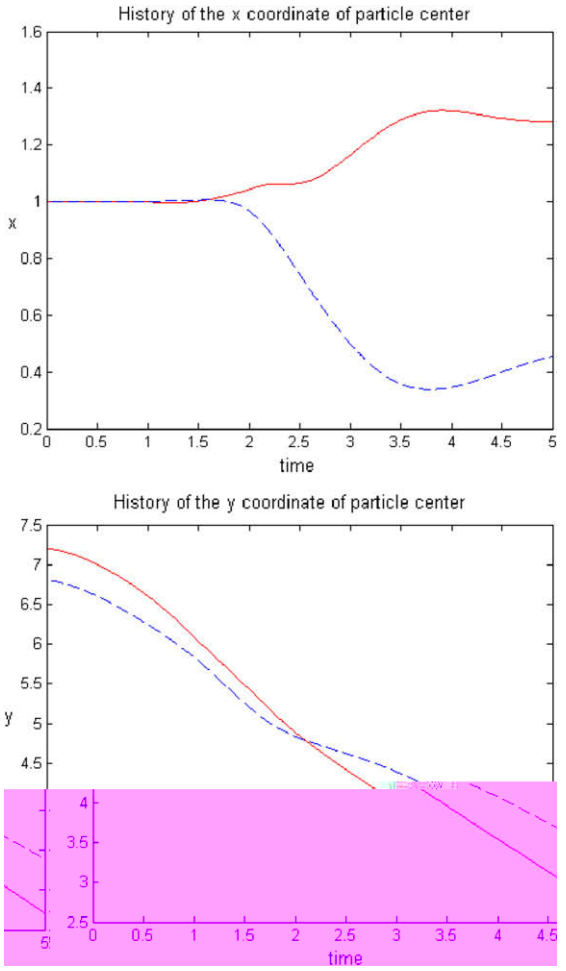

Fig. 9. Sedimentation of two circular particles: time history of the horizontal and vertical components of the positions of particles P1 (dashed, blue line) and P2 (solid, red line). (For interpretation of the references in colour in this figure legend, the reader is referred to the web version of this article.)

0.1 < Re < 2 : steady equilibrium (the cavity centerline) with a monotonic approach.
3 <Re <Re_c : steady equilibrium (the cavity centerline) with a transient overshoot. Re_c is some critical value which is not specified.
Re_c < Re <60 : weak oscillatory motion.
60 <Re <300 : strong oscillatory motion.
300 <Re : irregular oscillatory motion.

We solved this problem on a mesh of size $h = 1/40$, which was chosen so that there are at least 10 nodal points inside the particle, as indicated in [9]. Plotted in Fig. 7 are the particle trajectories obtained for four different Reynolds numbers which correspond to four different values of the fluid viscosity (the results for $Re = 260.81$ were obtained with a mesh of size $h = 1/100$). Each Reynolds number lies within one of the first four intervals given in [10]. As can be clearly observed in Fig. 7, the different flow regimes established in [10] are reproduced by this results. These results also compare well with those of Reference [9].

4.2. Sedimentation of two circular particles

As a further check of the performance of the proposed algorithm, we considered the problem of simulating the motion and interaction of two rigid circular disks sedimenting in a vertical channel filled with an incompressible viscous Newtonian fluid. It is well known that when two particles are dropped close to each other they experience the phenomenon of *drafting, kissing and tumbling* (see [12]). We expect the simulation to reproduce this phenomenon.

This particulate flow problem has been solved numerically before in [21,29,30,34,36], among others. We considered the same data as in References [29,30,34] in order to compare our results with them. The domain is a rectangular cavity of width

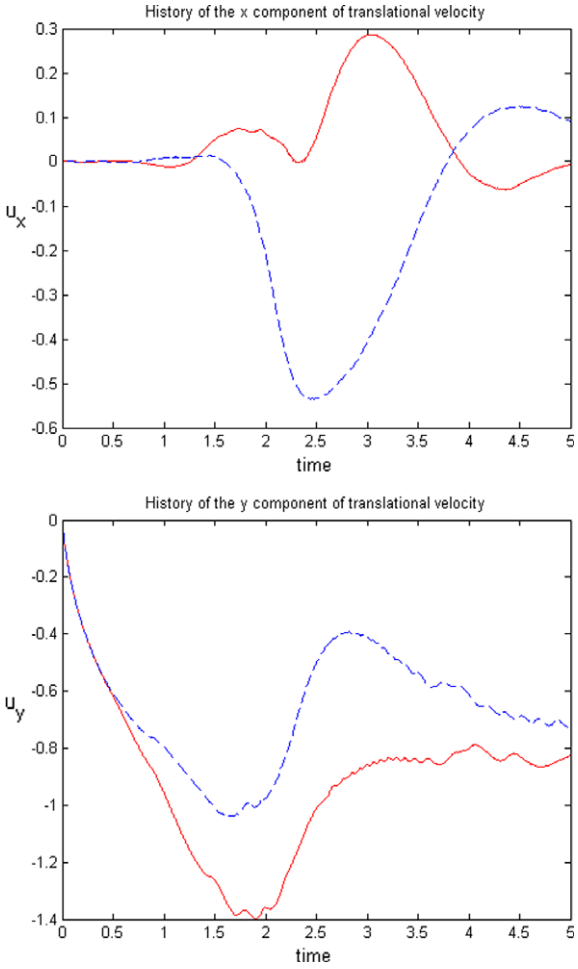

Fig. 10. Sedimentation of two circular particles: time history of the horizontal and vertical components of the velocities of particles P1 (dashed, blue line) and P2 (solid, red line). (For interpretation of the references in colour in this figure legend, the reader is referred to the web version of this article.)

2 and height 8; the radii of the particles are 0.1, the fluid and particle densities are $\rho_f = 1$ and $\rho_s = 1.01$, respectively, and the fluid viscosity is $\mu = 0.01$. The acceleration of gravity is $g = 980$ and acts in the negative y direction. The fluid and the particles are initially at rest, and the particles are dropped at the centerline of the channel at heights $y_1 = 6.8$ (particle P1) and $y_2 = 7.2$ (particle P2).

We performed the simulations on a mesh of size $h = 1/50$ using a time step size of $\Delta t = 0.0025$. Fig. 8 plots the positions of the two particles at times $t = 0.625, 1.25, 1.875, 2.5, 3.125, 3.75, 4.375$ and 5. The phenomenon of *drafting, kissing and tumbling* can be clearly observed in these results, which agree well with those of the references mentioned above. In fact, tumbling makes particle P1 (the one which was dropped lowest) deviate towards the left of the channel and particle P2 towards the right, something that can also be observed in the results of [30,34].

Figs. 9 and 10 show the time history of the two components of the positions and velocities of the two particles, respectively. These results agree well qualitatively with those of References [21,29,30,34,36].

4.3. Sedimentation of 512 circular particles in a non-rectangular domain

In order to increase the difficulty of the test cases for the algorithm, we considered the problem of the sedimentation of 512 particles in a non-rectangular domain which is filled with an incompressible, viscous Newtonian fluid. The computational domain is a channel with a slope in part of the bottom (see Fig. 11). This example is intended to show the capability of the SDI algorithm (as thus, of the proposed fluid-particle solver) to deal with non-rectangular geometries.

The fluid viscosity and density are $\mu = 0.025$ and $\rho = 1$, respectively. All particles have a radius of $R = 0.04$ and a density of $\rho_s = 2$, and they are initially at rest and uniformly distributed in the upper part of the cavity, arranged in 32 rows and 16 columns.

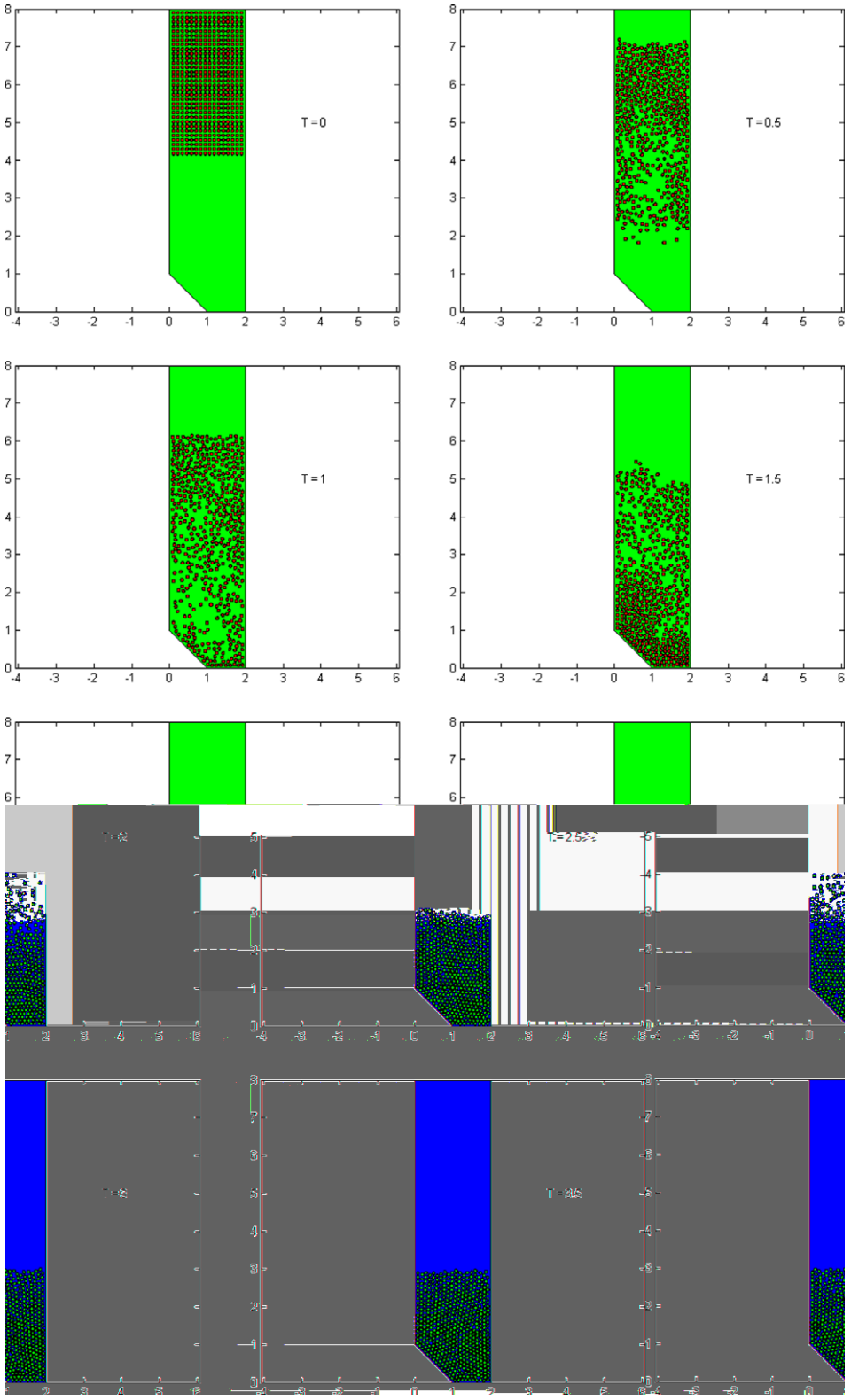

Fig. 11. Sedimentation of 512 circular particles: position of the particles at times $t = 0, 0.5, 1, 1.5, 2, 2.5$ and 3.5 (from left to right and from top to bottom).

A uniform mesh of size $h = 1/50$ was employed to solve this problem and a time step size of $\Delta t = 0.01$ was taken; the computation was sequential. The parameters for the repulsive forces were the same as in the previous examples.

Fig. 11 shows some snapshots of the position of the particles at different times; the particles had sedimented completely by $t = 3.5$, although they were still rearranging their positions in order to minimize the potential energy. As can be observed, the repulsive forces effectively prevent particles from overlapping.

5. Conclusions

A Fictitious Domain, parallel numerical method for the Direct Numerical Simulation of the motion of rigid particles in an incompressible viscous fluid has been developed. A fast computational technique is used in this scheme to enforce rigidity on the particles, and a SDI algorithm is employed in order to achieve a high level of parallelization. Accurate results have been obtained with this method in problems of sedimentation of one and several circular particles, and the results of a parallel efficiency study are comparable to similar techniques applied to much simpler problems. Practical applications of this method to problems with larger numbers of particles are under development.

Acknowledgment

The authors wish to thank Professor Enrique Fernández Cara for his valuable comments and suggestions about this work.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.jcp.2009.07.010.

References

[1] I.I. Albarreal, M.C. Calzada, J.L. Cruz, E. Fernández-Cara, J.R. Galo, M. Marín, Convergence analysis and error estimates for a parallel algorithm for solving the Navier–Stokes equations, Numerische Mathematik 93 (2002) 201–221.
[2] I.I. Albarreal, M.C. Calzada, J.L. Cruz, E. Fernández-Cara, J.R. Galo, M. Marín, Time and space parallelization of the Navier–Stokes equations, Computational & Applied Mathematics 24 (2005) 417–438.
[3] J. Blasco, R. Codina, A. Huerta, A fractional-step method for the incompressible Navier–Stokes equations related to a predictor–multicorrector algorithm, International Journal for Numerical Methods in Fluids 28 (1998) 1391–1419.
[4] J. Blasco, R. Codina, Error estimates for an operator-splitting method for incompressible flows, Applied Numerical Mathematics 51 (2004) 1–17.
[5] M. Coquerelle, G.H. Cottet, A vortex level set method for the two-way coupling of an incompressible fluid with colliding rigid bodies, Journal of Computational Physics 227 (2008) 9121–9137.
[6] M.A. Cruchaga, C.M. Muñoz, D.J. Celentano, Simulation and experimental validation of the motion of immersed rigid bodies in viscous fluids, Computer Methods in Applied Mechanics and Engineering 197 (2008) 2823–2835.
[7] C. Dí az-Goano, P.D. Minev, K. Nandakumar, A fictitious domain/finite element method for particulate flows, Journal of Computational Physics 192 (2003) 105–123.
[8] S. Dong, D. Liu, M.R. Maxey, G.E. Karniadakis, Spectral distributed Lagrange multiplier method: algorithm and benchmark tests, Journal of Computational Physics 195 (2004) 695–717.
[9] C. Duchanoy, T.R.G. Jongen, Efficient simulation of liquid–solid flows with high solids fraction in complex geometries, Computers and Fluids 32 (2003) 1453–1471.
[10] J. Feng, H.H. Hu, D.D. Joseph, Direct simulation of initial value problems for the motion of solid bodies in a Newtonian fluid. Part 1. Sedimentation, Journal of Fluid Mechanics 261 (1994) 95–134.
[11] E. Fernández-Cara, M. Marín, The convergence of two numerical schemes for the Navier–Stokes equations, Numerische Mathematik 55 (1989) 33–60.
[12] A. Fortes, D.D. Joseph, T.S. Lundgren, Nonlinear mechanics of fluidization of beds of spherical particles, Journal of Fluid Mechanics 177 (1987) 467–483.
[13] J.R. Galo, Resolución en paralelo de las ecuaciones de Navier–Stokes 2D y 3D mediante el método de direcciones simultáneas, PhD Thesis, University of Seville, Spain, 2002.
[14] J.R. Galo, I. Albarreal, M.C. Calzada, J.L. Cruz, E. Fernández-Cara, M. Marín, Simultaneous directions parallel methods for elliptic and parabolic systems, Comptes Rendus Academic de Sciences, Paris, Series I 339 (2004) 145–150.
[15] J.R. Galo, I. Albarreal, M.C. Calzada, J.L. Cruz, E. Fernández-Cara, M. Marín, A Simultaneous directions parallel algorithm for the Navier–Stokes equations, Comptes Rendus Academic de Sciences, Paris, Series I 339 (2004) 235–241.
[16] J.R. Galo, I. Albarreal, M.C. Calzada, J.L. Cruz, E. Fernández-Cara, M. Marín, Stability and convergence of a parallel fractional step method for the solution of linear parabolic problems, AMRX Applied Mathematics Research Express 4 (2005).
[17] J.R. Galo, M.C. Calzada, J.L. Cruz, M. Marín, I. Albarreal, E. Fernández-Cara, The smoothing effect of a simultaneous directions parallel method as applied to Poisson problems, Numerical Methods for Partial Differential Equations 22 (2006) 414–434.
[18] J.R. Galo, I. Albarreal, M.C. Calzada, J.L. Cruz, E. Fernández-Cara, M. Marín, Convergence and optimization of the parallel method of simultaneous directions for the solution of elliptic problems, Journal of Computational and Applied Mathematics 222 (2008) 458–476.
[19] R. Glowinski, Numerical methods for fluids (Part 3), in: P.G. Ciarlet, J.L. Lions (Eds.), Handbook of Numerical Analysis, vol. 9, North-Holland, Amsterdam, 2003.
[20] R. Glowinski, T.W. Pan, T.I. Hesla, D.D. Joseph, A distributed Lagrange multiplier/fictitious domain method for particulate flows, International Journal of Multiphase Flow 25 (1999) 755–794.
[21] R. Glowinski, T.W. Pan, T.I. Hesla, D.D. Joseph, J. Periaux, A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: application to particulate flow, Journal of Computational Physics 169 (2001) 363–426.
[22] H.H. Hu, Direct simulations of flows of solid–liquid mixtures, International Journal of Multiphase Flow 22 (1996) 335–352.
[23] H.H. Hu, D.D. Joseph, M.J. Crochet, Direct simulation of fluid particle motions, Theoretical and Computational Fluid Dynamics 3 (1992) 285–306.
[24] H.H. Hu, N.A. Patankar, M.Y. Zhu, Direct numerical simulations of fluid–solid systems using the Arbitrary–Lagrangian–Eulerian techniques, Journal of Computational Physics 169 (2001) 427–462.
[25] A.A. Johnson, T.E. Tezduyar, Simulation of multiple spheres falling in a liquid-filled tube, Computer Methods in Applied Mechanics and Engineering 134 (1996) 351–373.
[26] D. Kim, H. Choi, Immersed boundary method for flow around an arbitrarily moving body, Journal of Computational Physics 212 (2006) 662–680.
[27] J. Kim, D. Kim, H. Choi, An immersed-boundary finite-volume method for simulations of flow in complex geometries, Journal of Computational Physics 171 (2001) 132–150.
[28] B. Maury, Direct simulations of 2D fluid-particle flows in biperiodic domains, Journal of Computational Physics 156 (1999) 325–351.

[29] N.A. Patankar, P. Singh, D.D. Joseph, R. Glowinski, T.-W. Pan, A new formulation of the distributed Lagrange multiplier/fictitious domain method for particulate flow, International Journal of Multiphase Flow 26 (2000) 1509–1524.
[30] N.A. Patankar, N. Sharma, A fast projection scheme for the direct numerical simulation of particle flows, Communications in Numerical Methods in Engineering 21 (2005) 419–432.
[31] A. Perrin, H.H. Hu, An explicit finite difference scheme with spectral boundary conditions for particle flows, Journal of Computational Physics 227 (2008) 8776–8791.
[32] N. Sharma, N.A. Patankar, A fast computation technique for the direct numerical simulation of rigid particulate flows, Journal of Computational Physics 205 (2005) 439–457.
[33] C. Veeramani, P.D. Minev, K. Nandakumar, A fictitious domain formulation for flows with rigid particles: a non-Lagrange multiplier version, Journal of Computational Physics 224 (2007) 867–879.
[34] D. Wan, S. Turek, Direct numerical simulation of particulate flow via multigrid FEM techniques and the fictitious boundary method, International Journal for Numerical Methods in Fluids 51 (2006) 531–566.
[35] D. Wan, S. Turek, An efficient multigrid-FEM method for the simulation of solid–liquid two phase flows, Journal of Computational and Applied Mathematics 203 (2007) 561–580.
[36] D. Wan, S. Turek, Fictitious boundary and moving mesh methods for the numerical simulation of rigid particulate flows, Journal of Computational Physics 222 (2007) 28–56.
[37] Z. Yu, X. Shao, A direct-forcing fictitious domain method for particulate flows, Journal of Computational Physics 227 (2007) 292–314.