Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 222 (2007) 28–56
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Fictitious boundary and moving mesh methods for the numerical simulation of rigid particulate flows

Decheng Wan [a,*], Stefan Turek [b]

[a] School of Naval Architecture, Ocean and Civil Engineering, Shanghai Jiao Tong University, Huashan Road 1954, 200030 Shanghai, China
[b] Institute of Applied Mathematics LS III, University of Dortmund, Vogelpothsweg 87, 44227 Dortmund, Germany

Received 30 December 2005; received in revised form 1 June 2006; accepted 2 June 2006
Available online 1 November 2006

Abstract

In this paper, we investigate the numerical simulation of particulate flows using a new moving mesh method combined with the multigrid fictitious boundary method (FBM) [S. Turek, D.C. Wan, L.S. Rivkind, The fictitious boundary method for the implicit treatment of Dirichlet boundary conditions with applications to incompressible flow simulations. Challenges in Scientific Computing, Lecture Notes in Computational Science and Engineering, vol. 35, Springer, Berlin, 2003, pp. 37–68; D.C. Wan, S. Turek, L.S. Rivkind, An efficient multigrid FEM solution technique for incompressible flow with moving rigid bodies. Numerical Mathematics and Advanced Applications, ENUMATH 2003, Springer, Berlin, 2004, pp. 844–853; D.C. Wan, S. Turek, Direct numerical simulation of particulate flow via multigrid FEM techniques and the fictitious boundary method, Int. J. Numer. Method Fluids 51 (2006) 531–566]. With this approach, the mesh is dynamically relocated through a (linear) partial differential equation to capture the surface of the moving particles with a relatively small number of grid points. The complete system is realized by solving the mesh movement and the partial differential equations of the flow problem alternately via an operator-splitting approach. The flow is computed by a special ALE formulation with a multigrid finite element solver, and the solid particles are allowed to move freely through the computational mesh which is adaptively aligned by the moving mesh method in every time step. One important aspect is that the data structure of the undeformed initial mesh, in many cases a tensor-product mesh or a semi-structured grid consisting of many tensor-product meshes, is preserved, while only the spacing between the grid points is adapted in each time step so that the high efficiency of structured meshes can be exploited. Numerical results demonstrate that the interaction between the fluid and the particles can be accurately and efficiently handled by the presented method. It is also shown that the presented method significantly improves the accuracy of the previous multigrid FBM to simulate particulate flows with many moving rigid particles.
© 2006 Elsevier Inc. All rights reserved.

Keywords: Particulate flows; Multigrid; FEM; Fictitious boundary; Moving mesh; ALE

* Corresponding author. Fax: +86 21 62933156.
E-mail address: dcwan@sjtu.edu.cn (D. Wan).

0021-9991/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2006.06.002

1. Introduction

The numerical simulation of particulate flows or the motion of small rigid particles in a viscous liquid is one of the main focuses of engineering research and still a challenging task in many applications. Depending on the area of application, these types of problems arise frequently in numerous settings, such as sedimenting and fluidized suspensions, lubricated transport, hydraulic fracturing of reservoirs, slurries, understanding solid–liquid interaction, etc.

Several numerical simulation techniques for the particulate flows have been developed over the past decade. In these methods, the fluid flow is governed by the continuity and momentum equations, while the particles are governed by the equation of motion for a rigid body. The flow field around each individual particle is resolved, the hydrodynamic force between the particle and the fluid is obtained from the solutions. Hu, Joseph and coworkers [1,2], Galdi and Heuveline [3] as well as Maury [4] developed a finite element method based on unstructured grids to simulate the motion of a large number of rigid objects in Newtonian and viscoelastic fluids. This approach is based on an arbitrary Lagrangian–Eulerian (ALE) technique. Both the fluid and solid equations of motion are incorporated into a single coupled variational equation. The hydrodynamic forces and torques acting on the particles are eliminated in the formulation. The nodes on the particle surface move with the particle, while the nodes in the interior of the fluid are computed using Laplace's equation to guarantee a smoothly varying distribution of the nodes. At each time step, a new mesh is generated when the old one becomes too distorted, and the flow field is projected onto the new mesh. In this scheme, the positions of the particles and grid nodes are updated explicitly, while the velocities of the fluid and the solid particles are determined implicitly. In the case of 2D, the remeshing of the body-fitted meshes can be done by available grid generation software, but in the more interesting case of a full 3D simulation, the problem of efficient body-fitted grid generation is not yet solved in a satisfying manner yet.

In a series of papers by Glowinski, Joseph and coauthors [5–8], they proposed a distributed Lagrange multiplier (DLM)/fictitious domain method for the direct numerical simulation of large number of rigid particles in fluids. In the DLM method, the entire fluid-particle domain is assumed to be a fluid and the particle domain is constrained to move with the rigid motion. The fluid-particle motion is treated implicitly using a combined weak formulation in which the mutual forces cancel. This formulation permits the use of a fixed structured grid thus eliminating the need for remeshing the domain. In [9–12], we presented a similar, but different multigrid fictitious boundary method (FBM) for the detailed simulation of particulate flow. The method is based on a fixed (unstructured) FEM background grid. The motion of the solid particles is modeled by the Newton–Euler equations. Based on the boundary conditions applied at the interface between the particles and the fluid which can be seen as an additional constraint to the governing Navier–Stokes equations, the fluid domain can be extended into the whole domain which covers both fluid and particle parts. The FBM starts with a coarse mesh which may contain already many of the geometrical fine-scale details, and employs a (rough) boundary parametrization which sufficiently describes all large-scale structures with regard to the (geometric) boundary conditions. Then, all fine-scale features are treated as interior objects such that the corresponding components in all matrices and vectors are unknown degrees of freedom which are implicitly incorporated into all iterative solution steps.

An advantage of these fictitious domain methods over the generalized standard Galerkin finite element method is that they allow a fixed grid to be used, eliminating the need for remeshing, and they can be handled independently from the flow features. Much progress has been made for adopting the fictitious domain, respectively, boundary methods to simulate particulate flows, yet the quest for more accurate and efficient methods remains active, particularly for many particles of different shape and size: An underlying problem when adopting the fictitious domain methods is that the boundary approximation is of low accuracy only. Particularly in 3D, the ability of the fictitious domain methods to deal accurately with the interaction between fluid and rigid particles is greatly limited unless very fine meshes are used. One remedy could be to preserve the mesh topology, for instance as generalized tensor-product or blockstructured meshes, while a local alignment of the positions of the grid points with the physical boundary of the solid is achieved by a moving mesh process such that the boundary approximation error can be significantly decreased.

Many have come to recognize mesh adaption as an effective tool for simulating sharp fronts or moving interfaces and reducing numerical dispersion and oscillation. It has been demonstrated that significant improvements in accuracy and efficiency can be gained by adapting the mesh nodes so that they remain concentrated in regions of sharp fronts or interfaces. There are many existing mesh adaptive methods to achieve this purpose. Generally speaking, mesh adaptivity is usually applied in the form of local mesh refinement or through a continuous mesh mapping. In locally adaptive mesh refinement methods [13], an adaptive mesh is obtained by adding or removing points to achieve a desired level of accuracy. This allows a systematic error analysis. However, local refinement methods require more complicated data structures, compared to simple tensor-product meshes, and fairly technical methods to communicate information among different levels of hierarchical refinements. In the mapping approach [14,15], the mesh points are moved continuously in the whole domain to concentrate in regions where the solution has the largest variations or moving interfaces locate. These solution-adaptive or geometry-adaptive meshes can be used to compute accurately the sharp variation or the moving interface problems. They also have the additional advantage of allowing the use of standard solvers since all computations are performed in the same logical domain using a uniform mesh.

Over the past decade, several mesh adaptive techniques have been developed, namely the so-called *h*-, *p*- and *r*-methods. The first two do static remeshing; here, the *h*-method does automatic refinement or coarsening of the spatial mesh based on a posteriori error estimates or error indicators and the *p*-method takes higher or lower order approximations locally as needed. In contrast, the *r*-method (also known as moving mesh method) relocates grid points in a mesh having a fixed number of nodes in such a way that the nodes remain concentrated in regions of rapid variation of the solution or corresponding moving interfaces. The *r*-method is often a dynamic method which means that it uses time stepping or pseudo-time stepping approaches to construct the desired transformation. The *r*-method or moving mesh method differs from the *h*-, and *p*-methods in that the former keeps the same number of mesh points throughout the entire solution process, while the later have to treat hanging node problems. Thus, the size of computation and the data structures are fixed, which enables the *r*-method much easier to be incorporated into most CFD codes without the need for changing the system matrix structures and adding special interpolation procedures. The *r*-method has received more attention due to some new developments which clearly demonstrate its potential for problems such as those having moving interfaces [16–20]. Nevertheless, it is clear that the preferred method of choice in future might be of *r*–*h*–*p*-type, that means combining all these adaptive techniques.

The primary objective of this paper is to combine the multigrid fictitious boundary method (FBM) [9–11] with a prototypical version of a new moving mesh method described in [20] for the simulation of particulate flow and to analyse the accuracy of the proposed combining method, comparing its results with the previous pure multigrid fictitious boundary method without mesh adaptation. However, due to the preliminary character of these studies, we do not concentrate on efficiency aspects so far. As we have shown in [11], the use of the multigrid FBM does not require to change the mesh during the simulations, although the rigid particles vary their positions. The advantage is that no expensive remeshing has to be performed while a fixed mesh can be used such that in combination with appropriate data structures and fast CFD solvers very high efficiency rates can be reached. However, the accuracy for capturing the surfaces of solid particles is only of first order which might lead to accuracy problems. For a better approximation of the particle surfaces, we adopt a deformed grid, created from an arbitrary block-structured mesh, in which the topology is preserved: only the grid spacing is changed such that the grid points are concentrated near the surface of the rigid particles. Here, the solution of additional linear Poisson problems in every time step is required for generating the deformed grid, which means that the additional work is significantly less than the main fluid-particle part.

The paper is organized as follows. In Section 2, the physical models together with collision models for particulate flows are described. The moving mesh method and examples are presented in Section 3. The detailed numerical schemes and algorithms including the multigrid FBM, ALE formulation, time and space discretizations, as well as the complete numerical algorithm are given in Section 4. Prototypical numerical experiments are implemented and computational results will be presented in Section 5. Improvements of accuracy over the previous pure multigrid FBM will be emphasized. The concluding remarks will be given in Section 6.

2. Descriptions of the physical models

2.1. Governing equations

In our numerical studies of particle motion in a fluid, we will assume that the fluids are immiscible and Newtonian. The particles are assumed to be rigid. Let us consider the unsteady flow of N particles with mass M_i ($i = 1, \ldots, N$) in a fluid with density ρ_f and viscosity ν. Denote $\Omega_f(t)$ as the domain occupied by the fluid at time t, and $\Omega_i(t)$ as the domain occupied by the ith particle. So, the motion of an incompressible fluid is governed by the following Navier–Stokes equations in $\Omega_f(t)$,

$$\rho_f\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) - \nabla\cdot\sigma = 0, \qquad \nabla\cdot\mathbf{u} = 0 \quad \forall t \in (0, T), \tag{1}$$

where σ is the total stress tensor in the fluid phase defined as

$$\sigma = -p\mathbf{I} + \mu_f[\nabla\mathbf{u} + (\nabla\mathbf{u})^{\mathrm{T}}]. \tag{2}$$

where $\mathbf{I}$ is the identity tensor, $\mu_f = \rho_f \cdot \nu$, p is the pressure and $\mathbf{u}$ is the fluid velocity. Let $\Omega_T = \Omega_f(t) \cup \{\Omega_i(t)\}_{i=1}^N$ be the entire computational domain which shall be independent of t. Dirichlet- and Neumann-type boundary conditions can be imposed on the outer boundary $\Gamma = \partial\Omega_f(t)$. Since $\Omega_f = \Omega_f(t)$ and $\Omega_i = \Omega_i(t)$ are always depending on t, we drop t in all following notations.

The equations that govern the motion of each particle are the following Newton–Euler equations, i.e., the translational velocities $\mathbf{U}_i$ and angular velocities ω_i of the ith particle satisfy

$$M_i\frac{\mathrm{d}\mathbf{U}_i}{\mathrm{d}t} = (\Delta M_i)\mathbf{g} + \mathbf{F}_i + \mathbf{F}_i', \quad \mathbf{I}_i\frac{\mathrm{d}\omega_i}{\mathrm{d}t} + \omega_i \times (\mathbf{I}_i\omega_i) = T_i, \tag{3}$$

where M_i is the mass of the ith particle; $\mathbf{I}_i$ is the moment of the inertia tensor; ΔM_i is the mass difference between the mass M_i and the mass of the fluid occupying the same volume; $\mathbf{g}$ is the gravity vector; $\mathbf{F}_i'$ are collision forces acting on the ith particle due to other particles which come close to each other. We assume that the particles are smooth without tangential forces of collisions acting on them; the details of the collision model will be discussed in the following subsection. $\mathbf{F}_i$ and T_i are the resultants of the hydrodynamic forces and the torque about the center of mass acting on the ith particle which are calculated by

$$\mathbf{F}_i = (-1)\int_{\partial\Omega_i} \sigma\cdot\mathbf{n}\,\mathrm{d}\Gamma_i, \quad T_i = (-1)\int_{\partial\Omega_i} (\mathbf{X} - \mathbf{X}_i)\times(\sigma\cdot\mathbf{n})\,\mathrm{d}\Gamma_i, \tag{4}$$

where σ is the total stress tensor in the fluid phase defined by Eq. (2), $\mathbf{X}_i$ is the position of the mass center of the ith particle, $\partial\Omega_i$ is the boundary of the ith particle, $\mathbf{n}$ is the unit normal vector on the boundary $\partial\Omega_i$ pointing outward to the flow region. The position $\mathbf{X}_i$ of the ith particle and its angle θ_i are obtained by integration of the kinematic equations

$$\frac{\mathrm{d}\mathbf{X}_i}{\mathrm{d}t} = \mathbf{U}_i, \quad \frac{\mathrm{d}\theta_i}{\mathrm{d}t} = \omega_i. \tag{5}$$

No-slip boundary conditions are applied at the interface $\partial\Omega_i$ between the ith particle and the fluid, i.e., for any $\mathbf{X} \in \overline{\Omega}_i$, the velocity $\mathbf{u}(\mathbf{X})$ is defined by

$$\mathbf{u}(\mathbf{X}) = \mathbf{U}_i + \omega_i \times (\mathbf{X} - \mathbf{X}_i). \tag{6}$$

2.2. Collision models

For handling more than one particle, a collision model is needed to prevent the particles from interpenetrating each other. Glowinski, Joseph and coauthors [7,8] proposed repulsive force models in which an artificial short-range repulsive force between particles is introduced keeping the particle surfaces more than one element (the range of the repulsive force) apart from each other. In these models, overlapping of the regions occupied by the rigid bodies is not allowed since conflicting rigid body motion constraints from two different particles are not imposed at the same velocity nodes. However, in numerical calculations, the overlapping of particles might happen: for solving this problem, Joseph et al. [28] suggested a modified repulsive force model in which the particles are allowed to come arbitrarily close and even to overlap slightly each other. When conflicting rigid body motion constraints from two different particles are applied onto a velocity node, then the constraint from the particle that is closer to that node is used. A repulsive force is only applied when the particles overlap each other.

Following such ideas, we examine a modified collision model with a new definition of short range repulsive forces which cannot only prevent the particles from getting too close, it can also deal with the case of particles overlapping each other when numerical simulations bring the particles very close due to unavoidable numerical truncation errors. For the particle-particle collisions, the repulsive force is determined as,

$$\mathbf{F}^{\mathrm{P}}_{i,j} = \begin{cases} 0 & \text{for } d_{i,j} > R_i + R_j + \rho, \\ \frac{1}{\epsilon'_{\mathrm{P}}}(\mathbf{X}_i - \mathbf{X}_j)(R_i + R_j - d_{i,j}) & \text{for } d_{i,j} \leqslant R_i + R_j, \\ \frac{1}{\epsilon_{\mathrm{P}}}(\mathbf{X}_i - \mathbf{X}_j)(R_i + R_j + \rho - d_{i,j})^2 & \text{for } R_i + R_j \leqslant d_{i,j} \leqslant R_i + R_j + \rho, \end{cases} \tag{7}$$

where R_i and R_j are the radius of the ith and jth particle, $\mathbf{X}_i$ and $\mathbf{X}_j$ are the coordinates of their mass centers, $d_{i,j} = |\mathbf{X}_i - \mathbf{X}_j|$ is the distance between their mass centers, ρ is the range of the repulsive force (usually $\rho = 0.5$–$2.5\Delta h$, Δh is the local mesh size), ϵ_{P} and ϵ'_{P} are small positive stiffness parameters for particle-particle collisions. If the fluid is sufficiently viscous, and $\rho \simeq \Delta h$ as well as ρ_i/ρ_{f} are of order 1 (ρ_i is the density of the ith particle, ρ_{f} is the fluid density), then we can take $\epsilon_{\mathrm{P}} \simeq (\Delta h)^2$ and $\epsilon'_{\mathrm{P}} \simeq \Delta h$ in the calculations. For the particle–wall collisions, the corresponding repulsive force reads,

$$\mathbf{F}^{\mathrm{W}}_{i} = \begin{cases} 0 & \text{for } d'_i > 2R_i + \rho, \\ \frac{1}{\epsilon'_{\mathrm{W}}}(\mathbf{X}_i - \mathbf{X}'_i)(2R_i - d'_i) & \text{for } d'_i \leqslant 2R_i, \\ \frac{1}{\epsilon_{\mathrm{W}}}(\mathbf{X}_i - \mathbf{X}'_i)(2R_i + \rho - d'_i)^2 & \text{for } 2R_i \leqslant d'_i \leqslant 2R_i + \rho, \end{cases} \tag{8}$$

where $\mathbf{X}'_i$ is the coordinate vector of the center of the nearest imaginary particle P'_i located on the boundary wall Γ w.r.t. the ith particle, $d'_i = |\mathbf{X}_i - \mathbf{X}'_i|$ is the distance between the mass centers of the ith particle and the center of the imaginary particle P'_i, ϵ_{W} is a small positive stiffness parameter for particle–wall collisions, usually $\epsilon_{\mathrm{W}} = \epsilon_{\mathrm{P}}/2$ and $\epsilon'_{\mathrm{W}} = \epsilon'_{\mathrm{P}}/2$ in our calculations. Then, the total repulsive forces (i.e. collision forces) exerted on the ith particle by the other particles and the walls can be expressed as follows:

$$\mathbf{F}'_i = \sum_{j=1, j\neq i}^{N} \mathbf{F}^{\mathrm{P}}_{i,j} + \mathbf{F}^{\mathrm{W}}_{i}. \tag{9}$$

3. Moving mesh method

In this section, we briefly describe the moving mesh method which will be adopted and coupled with our multigrid fictitious boundary method (FBM) [9–11] to solve numerically the particulate flow equations. The details of the moving mesh method can be found in [20].

The moving mesh problem can be treated to constructing a transformation φ, $x = \varphi(\xi)$ from the computational space (with coordinate ξ) to the physical space (with coordinate x). There are several types of moving mesh methods, generally computing x by minimizing a variational form or computing the mesh velocity $v = x_t$ using a Lagrangian-like formulation. The moving mesh method we will employ belongs to the velocity-type class, which is based on Liao's work [16–19] and Moser's work [21]. This method has several advantages: only linear Poisson problems on fixed meshes are to be solved, monitor functions can be obtained directly from error distributions or distance functions, mesh tangling can be prevented, and the data structure from the starting mesh is preserved.

Suppose $g(x)$ (area function) to be the area distribution on the undeformed mesh, while $f(x)$ (monitor function) in contrast describes the absolute mesh size distribution of the target grid, which is independent of the starting grid and chosen according to the need of the physical problem. Then, the transformation φ can be computed via the following four steps:

1. Compute the scaling factors c_f and c_g for the given monitor function $f(x) > 0$ and the area function g using

   $$c_f \int_\Omega \frac{1}{f(x)}\,dx = c_g \int_\Omega \frac{1}{g(x)}\,dx = |\Omega|, \tag{10}$$

   where $\Omega \subset \mathbb{R}^2$ is a computational domain, and $f(x) \approx$ local mesh area. Let $\tilde{f}$ and $\tilde{g}$ denote the reciprocals of the scaled functions f and g, i.e.,

   $$\tilde{f} = \frac{c_f}{f}, \quad \tilde{g} = \frac{c_g}{g}. \tag{11}$$

2. Compute a grid-velocity vector field $v : \Omega \to \mathbb{R}^n$ by solving the Poisson problem

   $$-\mathrm{div}(v(x)) = \tilde{f}(x) - \tilde{g}(x), \; x \in \Omega \quad \text{and} \quad v(x) \cdot \mathrm{n} = 0, \; x \in \partial\Omega, \tag{12}$$

   where n is the outer normal vector of the domain boundary $\partial\Omega$ which may consist of several boundary components.

3. For each grid point x, solve the following ODE system:

   $$\frac{\partial \varphi(x,t)}{\partial t} = \eta(\varphi(x,t),t), \quad 0 \leqslant t \leqslant 1, \; \varphi(x,0) = x \tag{13}$$

   with

   $$\eta(y,s) := \frac{v(y)}{s\tilde{f}(y) + (1-s)\tilde{g}(y)}, \quad y \in \Omega, \; s \in [0,1]. \tag{14}$$

4. Get the new grid points via

   $$\varphi(x) := \varphi(x,1). \tag{15}$$

Here, we give two examples for the generation of deformed grids using the described moving mesh method. Fig. 1 shows the case of two objects with one rectangle and one ellipse inside a square domain. The starting mesh presented in Fig. 1(a) is equidistant, and we want to generate a deformed mesh which can align the grid lines near both the surface of the ellipse and the rectangle. We choose the monitor function $f(x)$ as a function of Δd, here Δd is the minimum distance of grid points to the both surfaces of the two solid bodies of the ellipse and the rectangle. Fig. 1(b) shows the generated deformed mesh. We can see that the grid lines are concentrated around the surfaces of the two solid bodies.

Fig. 2 presents another case with two ellipses in a long channel. The starting mesh and positions of the two ellipses are given in Fig. 2(a). If we choose the monitor function $f(x)$ as in the above case, i.e. to be a function

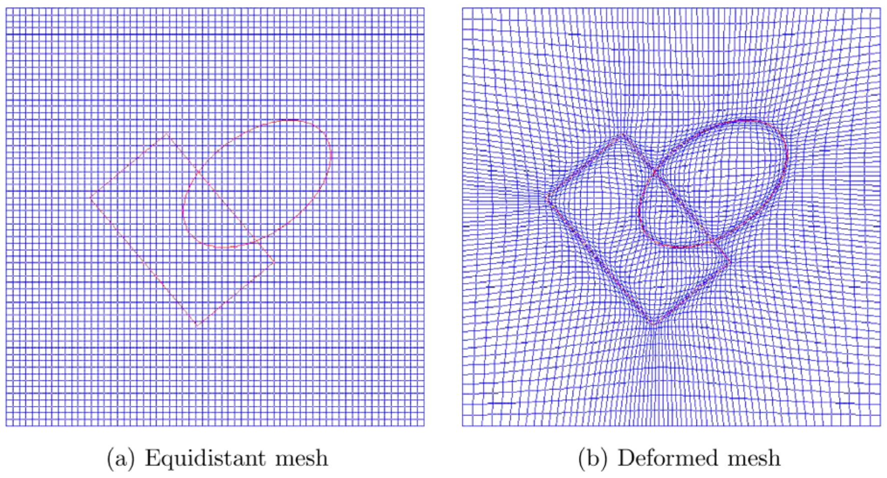

(a) Equidistant mesh (b) Deformed mesh

Fig. 1. Example of a deformed mesh generated from an equidistant mesh.

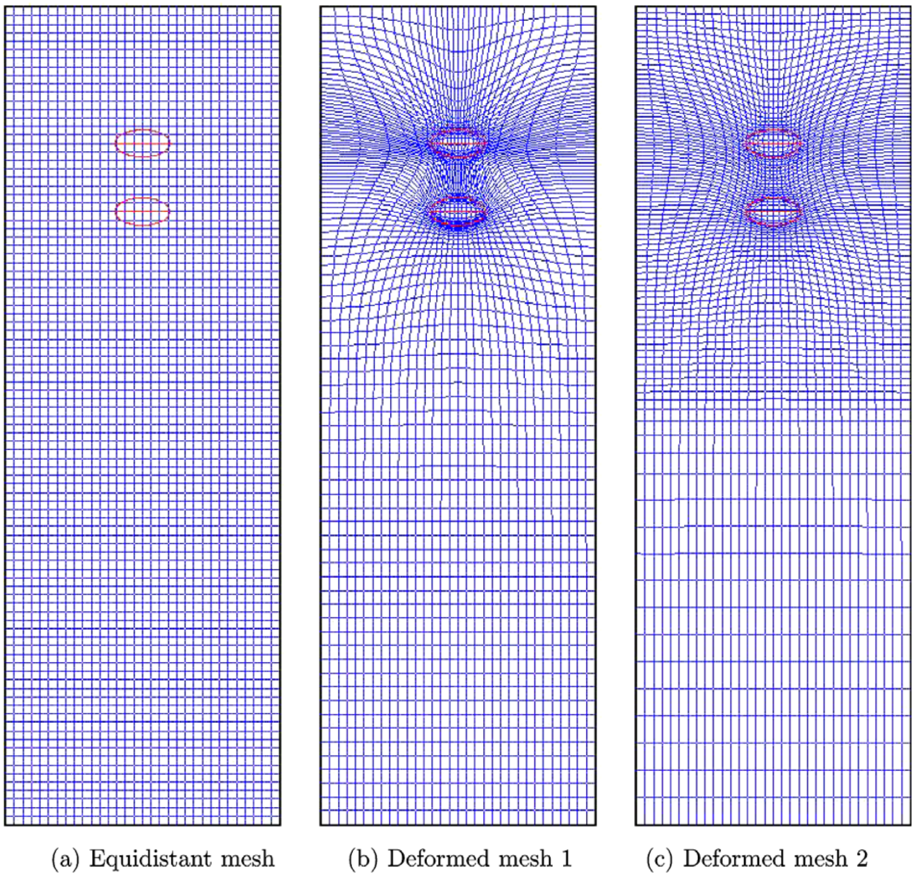


Fig. 2. Second example of deformed meshes generated from an equidistant mesh.

of Δd, with Δd being the minimum distance of grid points to both surfaces of the two solid ellipses, then the generated deformed mesh is shown in Fig. 2(b): we can see that there are still too many grid lines remaining in the lower part of the channel; however, there is no solid body. Moreover, the grid points in the gap between the two ellipses are not distributed very well. Next, we try to use another monitor function $f(x)$, a digit filter function $\Delta d_1 \times \Delta d_2$, here Δd_1 and Δd_2 are the distances of the grid points to the surface of both ellipses; then, a much better deformed mesh can be generated. We can see in Fig. 2(c) that the grid lines are more concentrated around the surfaces of both ellipses and in the region of the gap between the two ellipses, and there are less grid lines staying in the down part of the channel, too. It is clearly shown that the quality of the grid distribution is depending on the choice of the monitor function which is subject of further research.

4. Numerical method

4.1. Multigrid FEM fictitious boundary method

The details of the multigrid FEM fictitious boundary method have been presented in [9–11]. For illustration, a brief description is given below.

The fictitious boundary method (FBM) is based on a multigrid FEM background grid which covers the whole computational domain Ω_T and can be chosen independently from the particles of arbitrary shape, size and number. It starts with a coarse mesh which may already contain many of the geometrical details of Ω_i $(i = 1, \ldots, N)$, and it employs a fictitious boundary indicator (see [9]) which sufficiently describes all fine-scale structures of the particles with regard to the fluid-particle matching conditions of Eq. (6). Then, all fine-scale features of the particles are treated as interior objects such that the corresponding components in all matrices and vectors are unknown degrees of freedom which are implicitly incorporated into all iterative solution steps (see [10]). Hence, by making use of Eq. (6), we can perform calculations for the fluid in the whole domain Ω_T. The considerable advantage of the multigrid FBM is that the total mixture domain Ω_T does not have to change in time, and can be meshed only once. The domain of definition of the fluid velocity $\mathbf{u}$ is extended according to Eq. (6), which can be seen as an additional constraint to the Navier–Stokes equations (1), i.e.,

$$\begin{cases} \nabla \cdot \mathbf{u} = 0 & \text{(a)} \quad \text{for } \mathbf{X} \in \Omega_T, \\ \rho_f\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}\right) - \nabla \cdot \sigma = 0 & \text{(b)} \quad \text{for } \mathbf{X} \in \Omega_f, \\ \mathbf{u}(\mathbf{X}) = \mathbf{U}_i + \omega_i \times (\mathbf{X} - \mathbf{X}_i) & \text{(c)} \quad \text{for } \mathbf{X} \in \overline{\Omega}_i, \quad i = 1, \ldots, N. \end{cases} \tag{16}$$

For the study of interactions between the fluid and the particles, the calculation of the hydrodynamic forces acting on the moving particles is very important. From Eq. (4), we can see that the surface integrals on the wall surfaces of the particles should be conducted for the calculation of the forces $\mathbf{F}_i$ and T_i. However, in the presented multigrid FBM method, the shapes of the wall surface of the moving particles are implicitly imposed in the fluid field. If we reconstruct the shapes of the wall surface of the particles, it is not only a time consuming work, but also the accuracy is only of first order due to a piecewise constant interpolation from our indicator function. For overcoming this problem, we perform the hydrodynamic force calculations using a volume based integral formulation. To replace the surface integral in Eq. (4) we introduce a function α_i,

$$\alpha_i(\mathbf{X}) = \begin{cases} 1 & \text{for } \mathbf{X} \in \Omega_i, \\ 0 & \text{for } \mathbf{X} \in \Omega_T \setminus \Omega_i, \end{cases} \tag{17}$$

where $\mathbf{X}$ denotes the coordinates. The importance of such a definition can be seen from the fact that the gradient of α_i is zero everywhere except at the wall surface of the ith particle, and equals to the normal vector $\mathbf{n}_i$ of wall surface of the ith particle defined on the grid, i.e., $\mathbf{n}_i = \nabla\alpha_i$. Then, the hydrodynamic forces acting on the ith particle can be computed by

$$\mathbf{F}_i = -\int_{\Omega_T} \sigma \cdot \nabla\alpha_i \, \mathrm{d}\Omega, \quad T_i = -\int_{\Omega_T} (\mathbf{X} - \mathbf{X}_i) \times (\sigma \cdot \nabla\alpha_i) \, \mathrm{d}\Omega. \tag{18}$$

The integral over each element covering the whole domain Ω_T can be calculated with a standard Gaussian quadrature of sufficiently high order. Since the gradient $\nabla\alpha_i$ is non-zero only near the wall surface of the ith particle, thus the volume integrals need to be computed only in one layer of mesh cells around the ith particle, which leads to a very efficient treatment.

The algorithm of the (classical) multigrid FEM fictitious boundary method for solving the coupled system of fluid and particles can be summarized as follows:

1. Given the positions and velocities of the particles, solve the fluid equations (16)(a) and (b) in the corresponding fluid domain involving the position of the particles for the fictitious boundary conditions.
2. Calculate the corresponding hydrodynamic forces and the torque acting on the particles by using Eq. (18), and compute the collision forces by Eq. (9).
3. Solve Eq. (3) to get the translational and angular velocities of the particles, and then obtain the new positions and velocities of the particles by Eq. (5).
4. Use Eq. (16)(c) to set the new fluid domain and fictitious boundary conditions, and solve for the new velocity and pressure of the fluid phase as described in step 1.

4.2. ALE formulation of the FBM

For a better approximation of the solid surfaces, we adopt the above described moving mesh method such that we can preserve the mesh topology as generalized tensor-product or blockstructured meshes, while a local alignment with the rigid body surface is reached. The moving mesh method is sometimes referred to as quasi-Lagrangian method. When the moving mesh method is applied to the multigrid FBM, a mesh velocity $\mathbf{W}_\mathrm{m}$ in the convective term in Eq. (16)(b) should be introduced, i.e.,

$$\rho_\mathrm{f}\left[\frac{\partial\mathbf{u}}{\partial t}+(\mathbf{u}-\mathbf{W}_\mathrm{m})\cdot\nabla\mathbf{u}\right]-\nabla\cdot\sigma=0\quad\text{for }\mathbf{X}\in\Omega_\mathrm{f}\tag{19}$$

so that no interpolation routines from old to new meshes is required since the grid topology is preserved.

In the literature this is also referred to as an arbitrary Lagrangian–Eulerian (ALE) formulation. Note that the mesh velocities $\mathbf{W}_\mathrm{m}$ do not appear in the continuity equation, as a pressure-Poisson problem is solved to satisfy the continuity equation in an outer loop. Care has to be taken to satisfy the geometric conservation law (GCL), where the mesh velocity $\mathbf{W}_\mathrm{m}$ must be equal to the movement of the mesh velocity $\Delta\mathbf{x}$ during the time step. Therefore, the mesh velocities $\mathbf{W}_\mathrm{m}$ should be calculated according to the nodal movement from the previous time step by

$$\mathbf{W}_\mathrm{m}=\frac{1}{\Delta t}(\mathbf{x}^{n+1}-\mathbf{x}^{n}),\tag{20}$$

where Δt is the time step size and n denotes the time step number.

In each time step, a new deformed mesh is generated based on a starting mesh, then the system matrices are updated and the mesh velocity according to the new position of the deformed mesh nodes will be calculated. Since the moving mesh method keeps the same number of mesh points throughout the entire solution process, the size of computation and the data structure are fixed, which enables this method to be much easier to incorporate into most CFD codes without the need for changing the system matrix structures and for special interpolation procedures.

4.3. Time discretization by fractional-step-θ scheme

The fractional-step-θ scheme is a strongly A-stable time stepping approach; it possesses the full smoothing property which is important in the case of rough initial or boundary data. It also contains only very little numerical dissipation which is crucial in the computation of non-enforced temporal oscillations. A more detailed discussion of these aspects can be found in [22,24]. We first semi-discretize Eqs. (16)(a) and (19) in time by the fractional-step-θ scheme. Given $\mathbf{u}^n$ and the time step $K=t_{n+1}-t_n$, then solve for $\mathbf{u}=\mathbf{u}^{n+1}$ and $p=p^{n+1}$. In the fractional-step-θ-scheme, one macro time step $t_n\to t_{n+1}=t_n+K$ is split into three consecutive substeps with $\tilde{\theta}:=\alpha\theta K=\beta\theta' K$,

$$\begin{aligned}
&[I+\tilde{\theta}N(\mathbf{u}^{n+\theta})]\mathbf{u}^{n+\theta}+\theta K\nabla p^{n+\theta}=[I-\beta\theta KN(\mathbf{u}^{n})]\mathbf{u}^{n}\\
&\nabla\cdot\mathbf{u}^{n+\theta}=0,\\
&[I+\tilde{\theta}N(\mathbf{u}^{n+1-\theta})]\mathbf{u}^{n+1-\theta}+\theta' K\nabla p^{n+1-\theta}=[I-\alpha\theta' KN(\mathbf{u}^{n+\theta})]\mathbf{u}^{n+\theta}\\
&\nabla\cdot\mathbf{u}^{n+1-\theta}=0,\\
&[I+\tilde{\theta}N(\mathbf{u}^{n+1})]\mathbf{u}^{n+1}+\theta K\nabla p^{n+1}=[I-\beta\theta KN(\mathbf{u}^{n+1-\theta})]\mathbf{u}^{n+1-\theta}\\
&\nabla\cdot\mathbf{u}^{n+1}=0,
\end{aligned}\tag{21}$$

where $\theta=1-\frac{\sqrt{2}}{2}$, $\theta'=1-2\theta$, and $\alpha=\frac{1-2\theta}{1-\theta}$, $\beta=1-\alpha$, $N(\mathbf{v})\mathbf{u}$ is a compact form for the diffusive and convective part,

$$N(\mathbf{v})\mathbf{u}:=-\nu\nabla\cdot[\nabla\mathbf{u}+(\nabla\mathbf{u})^\mathrm{T}]+(\mathbf{v}-\mathbf{W}_\mathrm{m})\cdot\nabla\mathbf{u}.\tag{22}$$

Therefore, from Eq. (21) in each time step, we have to solve nonlinear problems of the following type:

$$[I+\theta_1 KN(\mathbf{u})]\mathbf{u}+\theta_2 K\nabla p=\mathbf{f},\qquad \mathbf{f}:=[I-\theta_3 KN(\mathbf{u}^n)]\mathbf{u}^n,\quad \nabla\cdot\mathbf{u}=0.\tag{23}$$

For Eq. (16)(c), we simply take an explicit expression like,

$$\mathbf{u}^{n+1}=\mathbf{U}_i^n+\omega_i^n\times(\mathbf{X}^n-\mathbf{X}_i^n).\tag{24}$$

4.4. Space discretization by finite element method

If we define a pair $\{\mathbf{u},p\} \in H := \mathbf{H}_0^1(\Omega) \times L := L_0^2(\Omega)$, and bilinear forms $a(\mathbf{u},\mathbf{v}) := (\nabla\mathbf{u}, \nabla\mathbf{v})$ and $b(p,\mathbf{v}) := -(p, \nabla\cdot\mathbf{v})$, a weak formulation of Eq. (23) reads as follows:

$$\begin{cases} (\mathbf{u},\mathbf{v}) + \theta_1 K[a(\mathbf{u},\mathbf{v}) + n(\mathbf{u},\mathbf{u},\mathbf{v})] + \theta_2 K b(p,\mathbf{v}) = (\mathbf{f},\mathbf{v}) & \forall \mathbf{v} \in H, \\ b(q,\mathbf{u}) = 0 & \forall q \in L. \end{cases} \tag{25}$$

Here, $L_0^2(\Omega)$ and $\mathbf{H}_0^1(\Omega)$ are the usual Lebesgue and Sobolev spaces, $n(\mathbf{u},\mathbf{u},\mathbf{v})$ is a trilinear form defined by

$$n(\mathbf{u},\mathbf{v},\mathbf{w}) := \int_\Omega [u_i - (w_\mathrm{m})_i]\left(\frac{\partial v_j}{\partial x_i} + \frac{\partial v_i}{\partial x_j}\right) w_j \,\mathrm{d}x. \tag{26}$$

To discretize Eq. (25) in space, we introduce a quadrilateral mesh T_h for the whole computational domain Ω_T, where h is a parameter characterizing the maximum width of the elements of T_h. To obtain the fine mesh T_h from a coarse mesh T_{2h}, we simply connect opposing midpoints. In the fine grid T_h, the old midpoints of the coarse mesh T_{2h} become vertices. We choose the $\widetilde{Q}1/Q0$ element pair which uses rotated bilinear shape functions for the velocity spanned by $\langle x^2 - y^2, x, y, 1\rangle$ in 2D and piecewise constants for the pressure in cells. The nodal values are the mean values of the velocity vector over the element edges and the mean values of the pressure over the elements rendering this approach non-conforming. This $\widetilde{Q}1/Q0$ element pair has several important features. It satisfies the Babuška–Brezzi condition without any additional stabilization, and the stability constant seems to be independent of the shape and size of the element. In particular on meshes containing highly stretched and anisotropic cells, the stability and the approximation properties are always satisfied. In addition, it admits simple upwind strategies which lead to matrices with certain **M**-matrix properties [22]. If we choose finite-dimensional spaces H_h and L_h and define a pair $\{\mathbf{u}_h, p_h\} \in H_h \times L_h$, the discrete version of (25) reads,

$$\begin{cases} (\mathbf{u}_h,\mathbf{v}_h) + \theta_1 K[a_h(\mathbf{u}_h,\mathbf{v}_h) + \tilde{n}_h(\mathbf{u}_h,\mathbf{u}_h,\mathbf{v}_h)] + \theta_2 K b_h(p_h,\mathbf{v}_h) = (\mathbf{f},\mathbf{v}_h) & \forall \mathbf{v}_h \in H_h, \\ b_h(q_h,\mathbf{u}_h) = 0 & \forall q_h \in L_h, \end{cases} \tag{27}$$

where $a_h(\mathbf{u}_h,\mathbf{v}_h) := \sum_{T\in T_h} a(\mathbf{u}_h,\mathbf{v}_h)_{|T}$ and $b_h(p_h,\mathbf{v}_h) := \sum_{T\in T_h} b(p_h,\mathbf{v}_h)_{|T}$. Note $\tilde{n}_h(\mathbf{u}_h,\mathbf{u}_h,\mathbf{v}_h)$ is a new convective term which includes streamline-diffusion stabilizations defined by

$$\tilde{n}_h(\mathbf{u}_h,\mathbf{v}_h,\mathbf{w}_h) := \sum_{T\in T_h} n(\mathbf{u}_h,\mathbf{v}_h,\mathbf{w}_h)_{|T} + \sum_{T\in T_h} \delta_T(\mathbf{u}_h\cdot\nabla\mathbf{v}_h, \mathbf{u}_h\cdot\nabla\mathbf{w}_h)_{|T}, \tag{28}$$

here δ_T is a local artificial viscosity which is a function of a local Reynolds number Re_T,

$$\delta_T := \delta^* \cdot \frac{h_T}{||\mathbf{u}||_\Omega} \cdot \frac{2Re_T}{1 + Re_T}, \quad Re_T = \frac{||\mathbf{u}||_T \cdot h_T}{\nu}, \tag{29}$$

where $\|\mathbf{u}\|_\Omega$ means the maximum norm of velocity in Ω_T, $\|\mathbf{u}\|_T$ is an averaged norm of velocity over T, h_T denotes the local mesh size of T, and δ^* is an additional free parameter which can be chosen arbitrarily ($\delta^* = 0.1$ is used in our calculations, see [22]). Obviously, for small local Reynolds numbers, with $Re_T \to 0$, δ_T is decreasing such that we reach in the limit case the second order central discretization. Vice versa, for convection dominated flows with $Re_T \gg 1$, we add an diffusion term of size O(h) which is aligned to the streamline direction $\mathbf{u}_h$.

4.5. Discrete projection scheme

For solving the discrete nonlinear problems after time and space discretizations, we have to take the following points into account, i.e., treatment of the nonlinearity, treatment of the incompressibility, and complete outer control like convergence criteria for the overall outer iteration, number of splitting steps, convergence control, embedding into multigrid, etc. In general, there are (at least) two possible approaches for solving the discrete problems [24]. One is the so-called full Galerkin schemes: first, we treat the nonlinearity by an outer nonlinear iteration of fixed point- or quasi-Newton type or by linearization via extrapolation in time, and then we obtain linear subproblems (Oseen equations) which can be solved by a direct coupled or a splitting approach separately for velocity and pressure. Typical schemes are preconditioned GMRES-like or multigrid solvers based on smoothers/preconditioners like Vanka, SIMPLE or local pressure Schur complement (see [22]). The disadvantage of these approaches is the high numerical cost for small time steps which are typical for particulate flows. Another possibility are the projection-type schemes: First, we split the coupled problem and obtain definite problems in $\mathbf{u}$ (Burgers equations) as well as in p (pressure-Poisson problems). Then, we treat the nonlinear problems in $\mathbf{u}$ by an appropriate nonlinear iteration or linearization technique while optimal multigrid solvers are used for the Poisson-like problems. Classical schemes belonging to this class are the Chorin and van Kan projection schemes and the discrete projection method, all of them are well suited for dynamic configurations which require small time steps (see [25]).

In this paper, based on the latter approach combined with multigrid methods, we adopt the discrete projection method (DPM) as special variant of the more general multigrid pressure Schur complement (MPSC) schemes to solve the discrete nonlinear problems after time and space discretizations. A detailed description of DPM and MPSC schemes has been presented in [22]: we first perform as outer iteration a fixed point iteration, applied to the fully nonlinear momentum equations. Then, in the inner loop, we solve the corresponding velocity equations involving linear transport-diffusion problems. Finally, the pressure is updated via a pressure Poisson-like problem, and the corresponding velocity field is adjusted. Since every time step requires the solution of linearized Burgers equations and Poisson-like problems, an optimized multigrid approach is used. The most important components are matrix–vector multiplication, smoothing operator and grid transfer routines (prolongation and restriction) for the underlying FEM spaces which have been realized in FEATFLOW (see [22,23] for the details).

4.6. Numerical algorithm

The whole algorithm of the multigrid FEM–FBM and moving mesh methods for the numerical simulation of rigid particulate flows can be summarized as follows:

1. Given (initial) particle positions $\mathbf{X}_i$ and velocity $\mathbf{U}_i$ in the overall domain Ω_T. Next, we assume that we have finished the calculations at time t_n.
2. Generate a new deformed mesh via the four steps of Eqs. (10)–(15).
3. Compute the mesh velocity $\mathbf{W}_\mathrm{m}$ by using Eq. (20) based on the new and the previous deformed mesh.
4. Set the fictitious boundary conditions by using Eq. (16)(c) with the 'old' particle positions $\mathbf{X}_i^n$ and the velocity $\mathbf{U}_i^n$ at time t_n.
5. By using the FBM and implementing the discrete projection scheme, build the system matrix and solve the fluid ALE equations of Eq. (27) to get the fluid velocity $\mathbf{u}^{n+1}$ and the pressure p^{n+1} at time t_{n+1} on the full computational domain Ω_T.
6. Calculate the hydrodynamic forces $\mathbf{F}_i^{n+1}$ and T_i^{n+1} exerted on every particle ($i = 1,\ldots,N$) by using the volume integration formulation of Eq. (18).
7. When two particles come too close, the time step has to be reduced. Then, we adopt several substeps with $\Delta t_k = K/\Lambda$ ($k = 1,\ldots,\Lambda$, Λ is the number of substeps calculations, $K = t_{n+1} - t_n$) for calculating the collisions and updating the particle positions and velocities during the collisions. Set $\mathbf{U}_i^{n,0} := \mathbf{U}_i^n$ and $\mathbf{X}_i^{n,0} := \mathbf{X}_i^n$.
8. Determine the number of substep calculations Λ by

$$\Lambda = \begin{cases} 1 & \text{if } (d_{i,j})_{\min} \geqslant (R_i + R_j)_{\max}, \\ \mathrm{MIN}\left\{10, 1 + \mathrm{MAX}\left(\frac{|d_{i,j} - R_i - R_j|}{\varrho}\right)\right\} & \text{if } (d_{i,j})_{\min} < (R_i + R_j - \varrho)_{\max}, \end{cases} \tag{30}$$

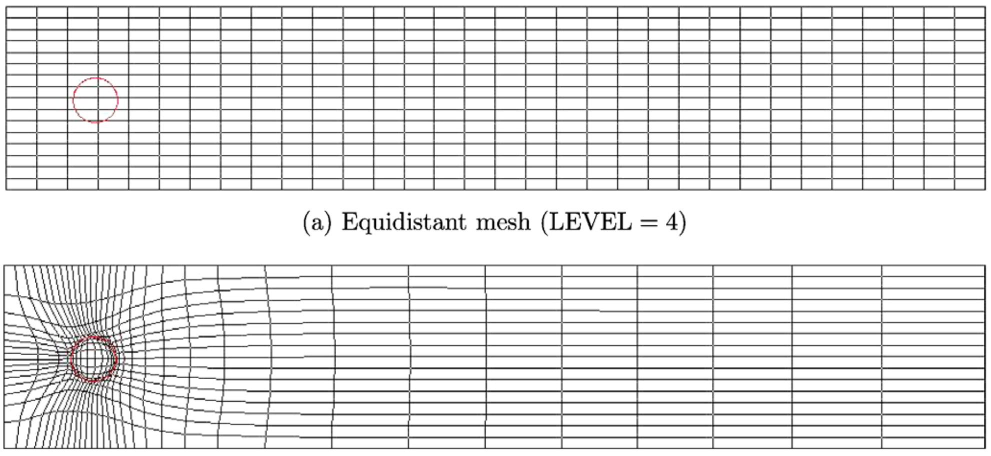
   where ϱ is the maximum penetration distance to be allowed (maximal overlapping range).
9. By using the Newton–Euler equations of Eq. (5) to calculate the motion of the solid particles, we obtain the new interim particle position $\mathbf{X}_i^{n+1/2,k}$ and velocity $\mathbf{U}_i^{n+1/2,k}$ as well as the new angular velocity ω_i^{n+1} and angle θ_i^{n+1} by

$$\mathbf{U}_i^{n+1/2,k} = \mathbf{U}_i^{n,k} + \left(\frac{\Delta M_i \mathbf{g}}{M_i} + \frac{\mathbf{F}_i^n + \mathbf{F}_i^{n+1}}{2M_i}\right)\Delta t_k, \tag{31}$$

$$\mathbf{X}_i^{n+1/2,k} = \mathbf{X}_i^{n,k} + \left(\frac{\Delta M_i \mathbf{g}}{M_i} + \frac{\mathbf{F}_i^n + \mathbf{F}_i^{n+1}}{2M_i}\right)(\Delta t_k)^2, \tag{32}$$

$$\omega_i^{n+1} = \omega_i^n + \left(\frac{T_i^n + T_i^{n+1}}{2\mathbf{I}_i}\right)K, \quad \theta_i^{n+1} = \theta_i^n + \left(\frac{\omega_i^n + \omega_i^{n+1}}{2}\right)K. \tag{33}$$

10. Use the collision model of Eqs. (7) and (8) to calculate the repulsive forces $(\mathbf{F}_i')^{n+1,k}$ with the interim particle position $\mathbf{X}_i^{n+1/2,k}$.
11. Update the particle positions and the velocity by the repulsive forces to obtain the new particle position $\mathbf{X}_i^{n+1,k}$ and the velocity $\mathbf{U}_i^{n+1,k}$ at time t_{n+1} by


(a) Equidistant mesh (LEVEL = 4)

(b) Deformed mesh (LEVEL = 4)

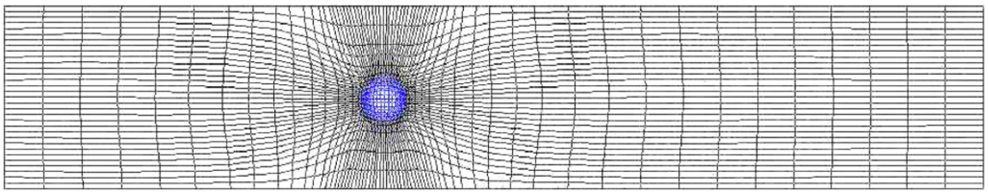
Fig. 3. Different meshes adopted for flow around a fixed circular cylinder.

Table 1
Drag and lift coefficients for flow around a fixed circular cylinder with $Re = 20$ by using equidistant meshes

LEVEL	NVT	NEL	Drag coefficient C_d	Lift coefficient C_l	VEF (%)	CPU
4	561	512	4.44688	−0.0649815	91.643	0.8
5	2145	2048	5.31808	−0.3508040	97.697	3.0
6	8385	8192	5.50358	−0.0093675	99.261	12.0
7	33,153	327,68	5.50585	0.0312388	99.842	54.6
8	131,841	131,072	5.53049	0.0239737	99.958	375.0
Reference values			$C_d = 5.5795$	$C_l = 0.010618$		

Table 2
Drag and lift coefficients for flow around a fixed circular cylinder with $Re = 20$ by using deformed meshes

LEVEL	NVT	NEL	Drag coefficient C_d	Lift coefficient C_l	VEF (%)	CPU
4	561	512	6.25486	0.0682610	99.332	1.3
5	2145	2048	5.72950	0.0297392	99.788	4.0
6	8385	8192	5.61971	0.0291702	99.946	14.0
7	33,153	32,768	5.58139	0.0118296	99.985	76.0
8	131,841	131,072	5.57706	0.0104031	99.997	402.0
Reference values			$C_d = 5.5795$	$C_l = 0.010618$		


(a) Deformed mesh ($t = 18.9$)

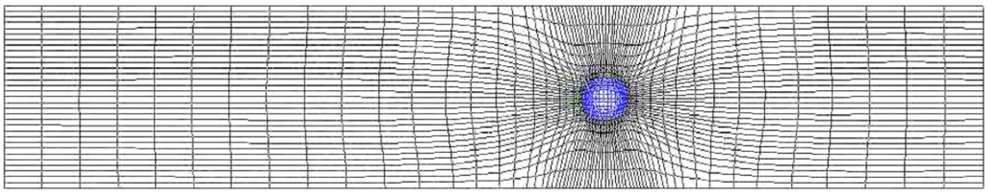
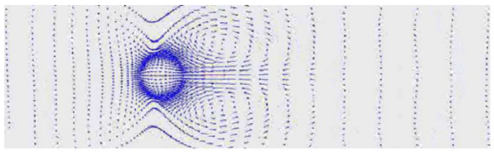
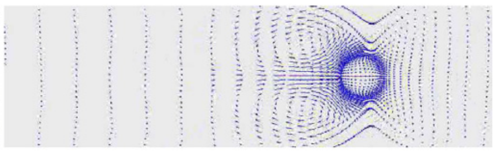
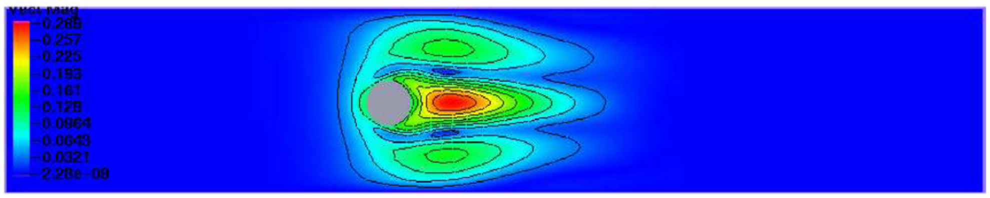

(b) Deformed mesh ($t = 21$)

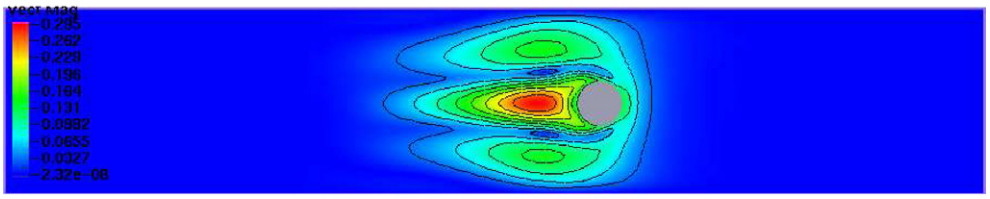
Fig. 4. Deformed meshes for one oscillating circular cylinder in a channel.


(a) Local vector field ($t = 18.9$)

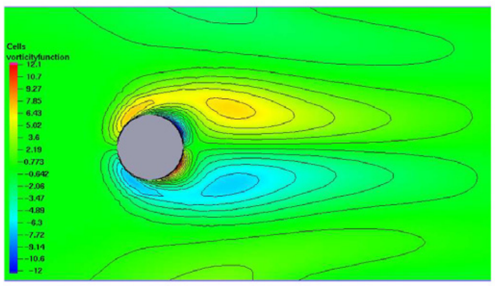

(b) Local vector field ($t = 21.0$)



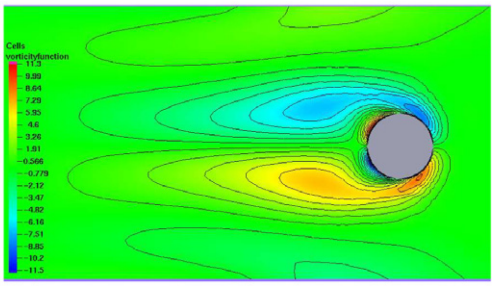
(c) Norm of velocity ($t = 18.9$)



(d) Norm of Velocity ($t = 21$)



(e) Local vorticity values ($t = 18.9$)



(f) Local vorticity values ($t = 21$)

Fig. 5. Oscillating circular cylinder in a channel.

$$\mathbf{U}_i^{n+1,k} = \mathbf{U}_i^{n+1/2,k} + \frac{(\mathbf{F}_i')^{n+1}}{M_i}\Delta t_k, \quad \mathbf{X}_i^{n+1,k} = \mathbf{X}_i^{n+1/2,k} + \frac{(\mathbf{F}_i')^{n+1}}{M_i}(\Delta t_k)^2. \tag{34}$$

12. Set $\mathbf{U}_i^{n,k+1} := \mathbf{U}_i^{n+1,k}$ and $\mathbf{X}_i^{n,k+1} := \mathbf{X}_i^{n+1,k}$ if $k < \Lambda$, and repeat steps 9–12.
13. Set $\mathbf{U}_i^{n+1} := \mathbf{U}_i^{n+1,\Lambda}$ and $\mathbf{X}_i^{n+1} := \mathbf{X}_i^{n+1,\Lambda}$.
14. Advance to the next new time step, set $t_n := t_{n+1}$ and repeat steps 2–14.

5. Numerical results

First of all, a benchmark configuration of 2D flow around a circular body in a channel is given to assess the suitability and accuracy of the hydrodynamic force calculations based on the combination of the multigrid

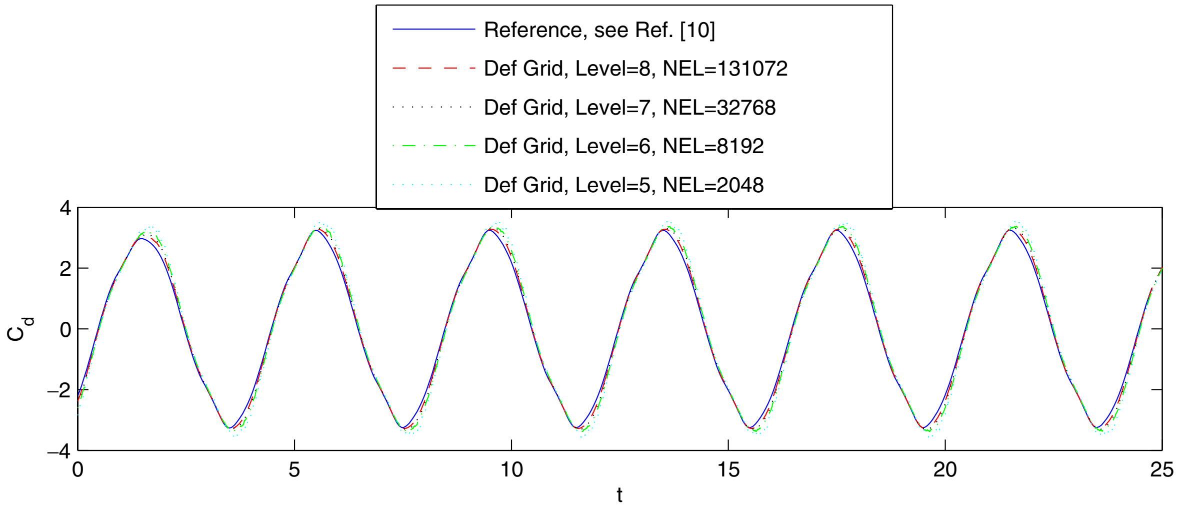


Fig. 6. Drag coefficients for one oscillating circular cylinder in a channel.

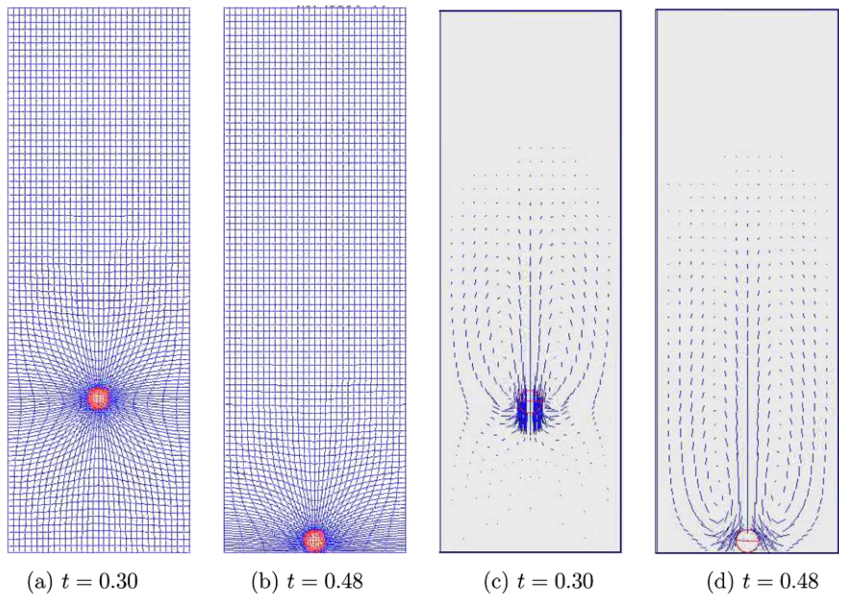


Fig. 7. One 2D circular ball falling down in a channel.

FBM and the new moving mesh technique. Then, three cases of a single moving particle in the fluid are presented to further validate the improvement of accuracy and efficiency using the presented method. Finally, the drafting, kissing and tumbling of two disks in a channel and the sedimentation of 120 circular particles in a cavity are provided to show that the presented method can be easily implemented for the simulation of particulate flows with large number of particles. Since we have used a prototypical implementation of the new mesh deformation method only, efficiency considerations are of only preliminary type while we concentrated more on the validation and accuracy aspects of the new FEM–ALE fictitious boundary method for particulate flows.

5.1. Benchmark experiment

We first consider a benchmark case of flow around a fixed circular cylinder in a channel as described in [26]. The channel height is $H = 0.41$, length $L = 2.2$, the cylinder diameter $D = 0.1$. The center point of the cylinder is located at $(0.2, 0.2)$. The Reynolds number is defined by $Re = \overline{U}D/\nu$ with the mean velocity $\overline{U} = 2U(0, H/2, t)/3$. The kinematic viscosity of the fluid is given as $\nu = \mu_f/\rho_f = 10^{-3}$ and its density $\rho_f = 1$. The inflow profiles are parabolic $U(0, Y, t) = 6.0\overline{U}Y(H - Y)/H_2$ with $\overline{U} = 0.2$ such that the resulting Reynolds number is $Re = 20$ which leads to steady-state solutions.

Fig. 3 shows the equidistant and deformed (static) meshes employed in our calculations, in which the circle shows the position of the cylinder. The shown meshes are successively refined by connecting opposite midpoints. The deformed mesh is generated from the equidistant mesh, but it has more grid nodes and elements concentrated and aligned around the surface of the cylinder. In Tables 1 and 2, the characteristics of these meshes after several global refinements are given. The meaning of “LEVEL” is the number of refinements, “NVT” the number of vertices, and “NEL” the number of elements, “VEF” the ratio of the effective cylinder area covered by the fixed mesh through the fictitious boundary method w.r.t. the real cylinder area, “CPU” the typical CPU time needed (DELL Precision Workstation 670, 2.66 GHz) for reaching steady-state.

Tables 1 and 2 present the computed results by using the equidistant meshes and deformed meshes, respectively. The corresponding reference results of drag and lift coefficients for this benchmark problem are also listed in these tables for comparison. From the tables, we can see that all results of drag and lift coefficients are converging w.r.t. mesh refinement except those for the lift coefficients when using equidistant mesh. The results for the deformed meshes are much better and more accurate than those for the equidistant meshes. For the deformed mesh cases, on LEVEL = 6 with only NEL = 8192, quite satisfying results have already been obtained. Obviously, the accuracy is improved by using the grid deformation techniques. Moreover, the effective area ratio of the cylinder captured by the deformed mesh lines has reached more than 99% in LEVEL = 4,

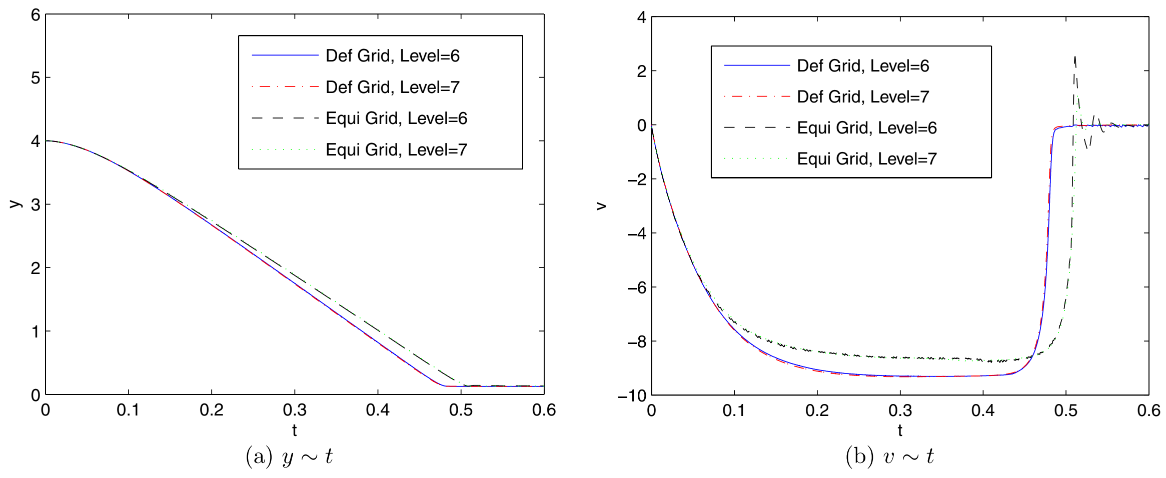


Fig. 8. Time history of y-position and v-velocity component of the center of the falling ball.

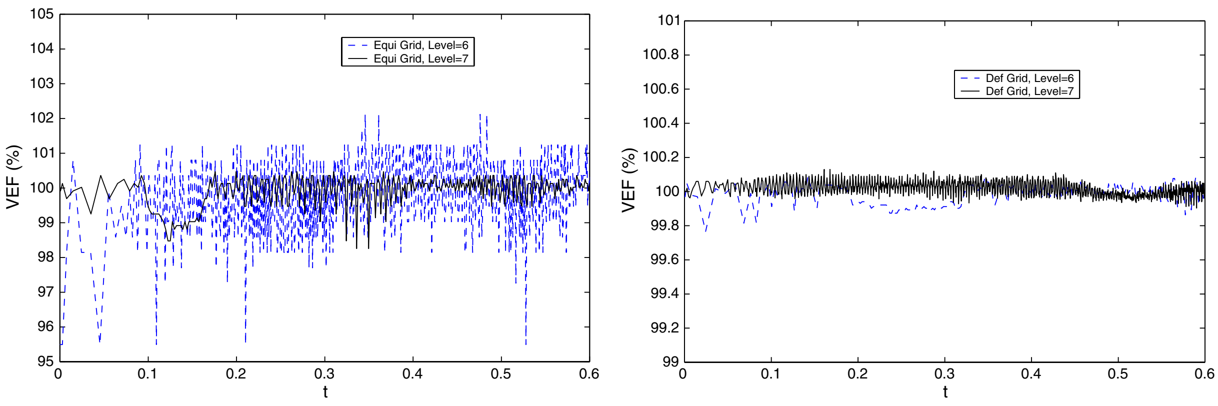
while it requires LEVEL = 6 for equidistant meshes. For CPU time, about 10% time is increased for the cases of the deformed meshes over the equidistant mesh on the same level due to additional time needed for the deformed mesh generation. However, the main CPU time is still needed for the fluid-solid part, not for the part of deformed mesh generation. Moreover, in the deformed mesh cases, we can adopt a lower level mesh which can obtain high accuracy results but only small CPU time increases compared to the case of equidistant meshes. For example, for the deformed mesh in LEVEL = 7, very accurate results of force calculations are possible and only 76 s CPU is needed, while for the equidistant mesh in LEVEL = 8, the CPU time of 375 s is required but the results of force calculations are still not so accurate. Hence, it shows that the deformed mesh can significantly improve the force calculations.

5.2. One oscillating circular cylinder in a channel

Next, an oscillating circular cylinder in a channel with a prescribed velocity is considered. The channel height is $H = 0.41$, length $L = 2.2$, the cylinder diameter $D = 0.1$. The center point of the cylinder is located initially at (1.1, 0.2).

The prescribed velocity for the cylinder is given by $u = 2\pi f A\cos(2\pi f t)$, $A = 0.25$, $f = 0.25$, $v = 0$, and no-slip velocity conditions are imposed at the two walls, inlet and outlet of the channel. The kinematic viscosity of the fluid is given by $\nu = \mu_f/\rho_f = 10^{-3}$ and its density $\rho_f = 1$. The fluid in channel is initially at rest.

The deformed meshes are generated by using the moving mesh method in every time step in order to align the mesh around the surface of the moving cylinder. Fig. 4 gives two snapshot results at $t = 18.9$ and $t = 21$ of the deformed meshes, and Fig. 5 presents the corresponding local vector field, norm of velocity and local vorticity distribution, respectively. These pictures show that the flow in the channel is disturbed by the oscillating cylinder, and the vortex is generated periodically in the wake of the cylinder. Fig. 6 illustrates the comparison



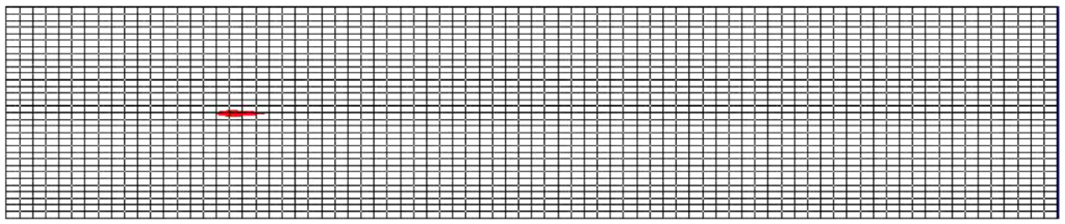
(a) Equidistant mesh (b) Deformed mesh

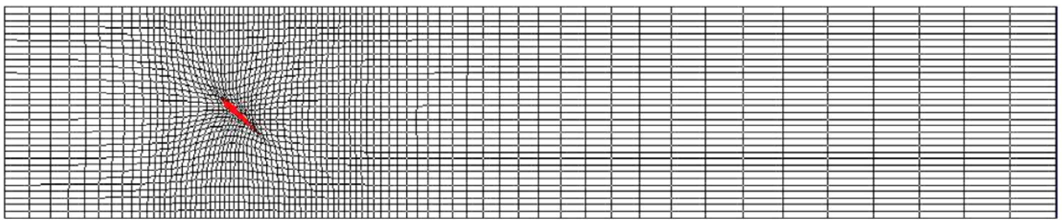
Fig. 9. Time history of the effective area ratio of the falling ball.

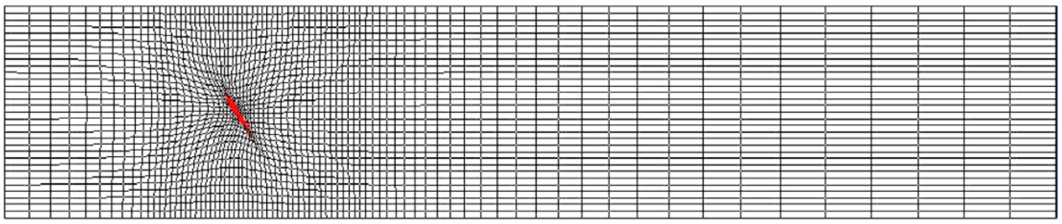
Table 3
CPU time for one time step and storage for one falling ball

LEVEL	Equidistant mesh		Deformed mesh	
	CPU time	Storage (MB)	CPU time	Storage (MB)
4	0.11	0.15	0.13	0.19
5	0.53	0.71	0.61	0.76
6	1.89	2.97	2.36	3.10
7	7.36	11.82	8.95	12.83

of the computed drag coefficient C_d by the presented method with the reference result based on the body-fitted mesh (see [10]). The results calculated from LEVEL = 5 to LEVEL = 8 and the parameters of the number of elements "NEL" for each refinement level are all shown together. From the comparisons, we can see that the presented results are identical with increasing the mesh refinement, and they agree very well with the reference results. On the deformed mesh of LEVEL = 5 with 2048 elements, very good results have already been reached.


(a) Starting mesh


(b) Deformed mesh ($t = 14.7$)


(c) Deformed mesh ($t = 16.0$)

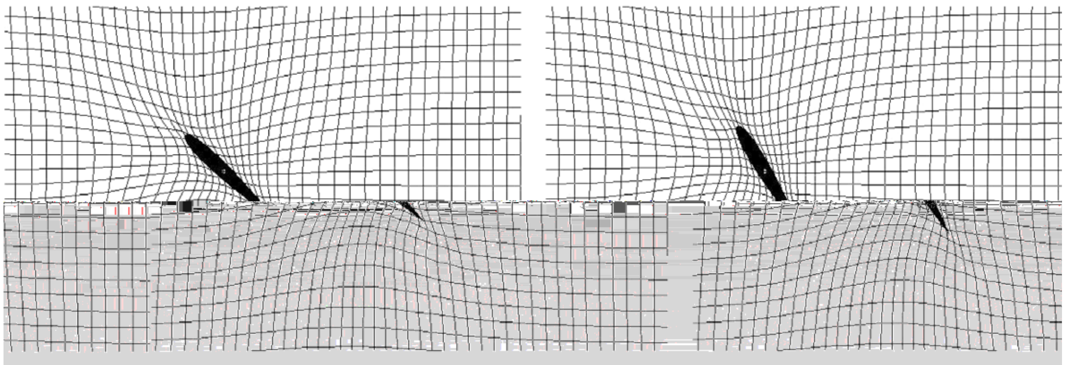
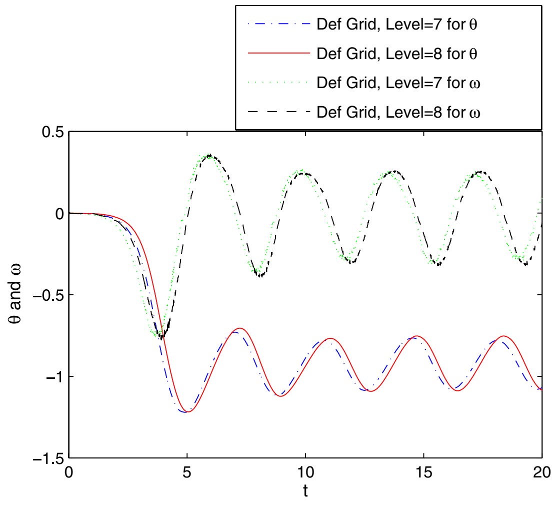

.0) (d) Zoomrd mesh ($t = 14.7$) (e) Zoomed mesh ($t = 1$

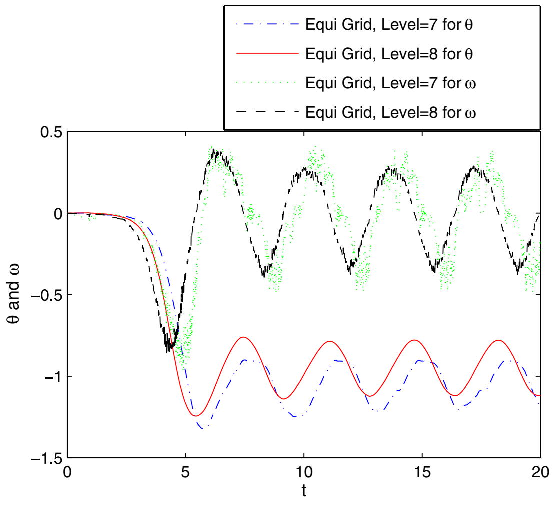
Fig. 10. Initial mesh and deformed meshes during the simulation of the induced rotation of a NACA0012 airfoil in a channel.

5.3. One 2D circular ball falling down in a channel

The computational domain is a channel of width 2 and height 6. A rigid circular ball with diameter $d = 0.25$ and density $\rho_p = 1.5$ is located at (1, 4) at time $t = 0$, and it is falling down under gravity in an incompressible fluid with density $\rho_f = 1$ and viscosity $\nu = 0.1$, with gravitational acceleration constant $g = -980$. We suppose that the ball and the fluid are initially at rest. The simulation is carried out on fixed equidistant meshes and moving deformed meshes, respectively, each of them having two different level, i.e., LEVEL = 6 with 12,545

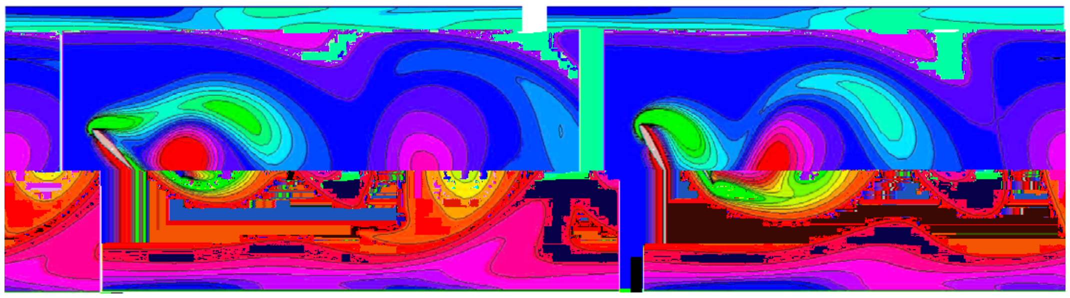


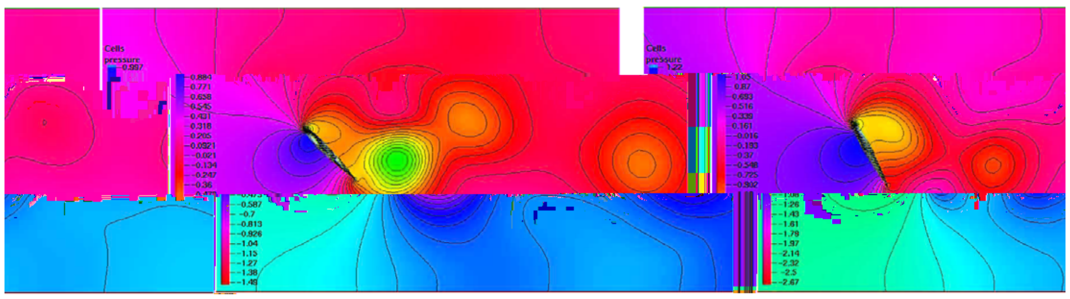
(a) Deformed meshes



(b) Equidistant meshes

Fig. 11. Time history of the angle of incidence θ and angular velocity ω for a rotating NACA0012 airfoil in a channel.


(a) Vorticity ($t = 14.7$) (b) Vorticity ($t = 16.0$)

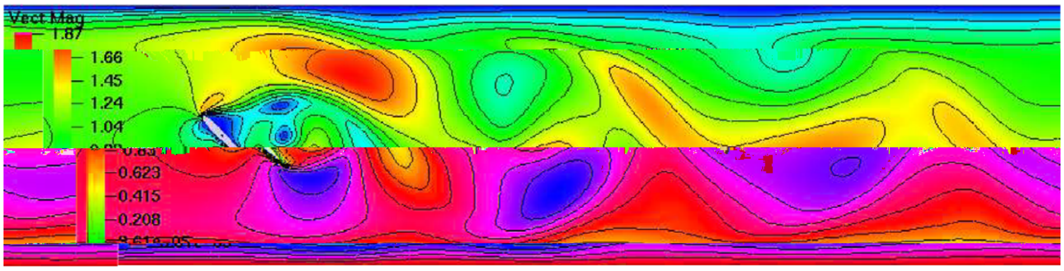


= 16.0) (c) Pressure ($t = 14.7$) (d) Pressure (t =

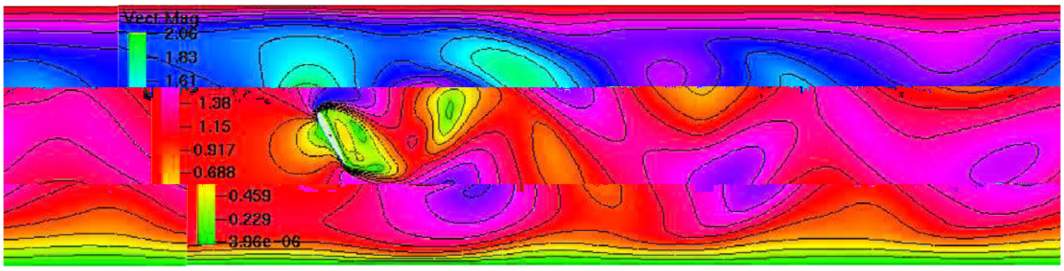
Fig. 12. Local value distributions for the rotating NACA0012 airfoil in a channel.

nodes and 12,288 elements, as well as LEVEL = 7 with 49,665 nodes and 49,152 elements which provide sufficiently accurate results.

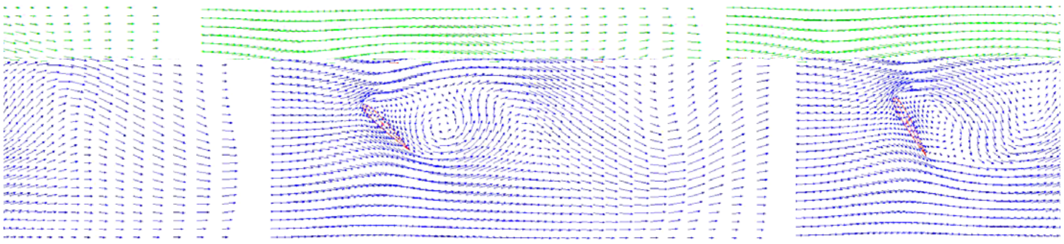
Fig. 7 gives two snapshots at $t = 0.30$ and $t = 0.48$ of the deformed meshes and the corresponding vector fields, respectively. Fig. 8 presents the comparison of the time history of the y-coordinate and v-velocity component of the center of the ball by using equidistant meshes and deformed meshes, each of them are calculated by the mesh levels LEVEL = 6 and LEVEL = 7, respectively, both leading to convergent results with refining the mesh. If we compare these results with those obtained by Glowinski [8], we can see that the results of the deformed meshes are much closer to his results than ones of the equidistant meshes, which proves again that the accuracy is improved when the moving deformed meshes are employed. Fig. 9 shows the ratio (%) of the effective area of the falling ball covered by the underlying fixed equidistant mesh and the adaptively deformed mesh compared with the real area of the disk, respectively. Again we can see that the deformed meshes can much better capture the falling ball than the equidistant meshes, which also explains why the deformed mesh can improve the accuracy of the simulation. Table 3 shows the typical CPU time for one time step and memory (in MByte) needed (DELL Precision Workstation 670, 2.66 GHz) by using equidistant and deformed meshes, respectively. We can also see the linear relation between CPU and storage cost w.r.t. the mesh size due to the optimized multigrid components. The CPU time of the deformed meshes cases is increased by ≈10–20% compared to the equidistant mesh cases, since the deformed mesh generation, mesh velocities



(a) Norm of velocity ($t = 14.7$)



(b) Norm of velocity ($t = 16$)



field ($t = 16$) (c) Local vector field ($t = 14.7$) (d) Local vector

Fig. 13. Induced rotation of a NACA0012 airfoil in a channel.

and system matrix update are needed to implement in every time step. However, the computer memory storage required for both cases is not significantly changed.

5.4. Induced rotation of an airfoil in a channel

The rigid bodies considered so far have been of circular form. The following simulation will show that the presented method can deal with rigid body of more complicated shape very well, and provide better results compared to that without using the moving grid deformation techniques. We consider a NACA0012 airfoil that has a fixed center of mass and is induced to rotate freely around its center of mass due to hydrodynamical forces under the action of an incompressible viscous flow in a channel. The channel has width 20 and height 4.

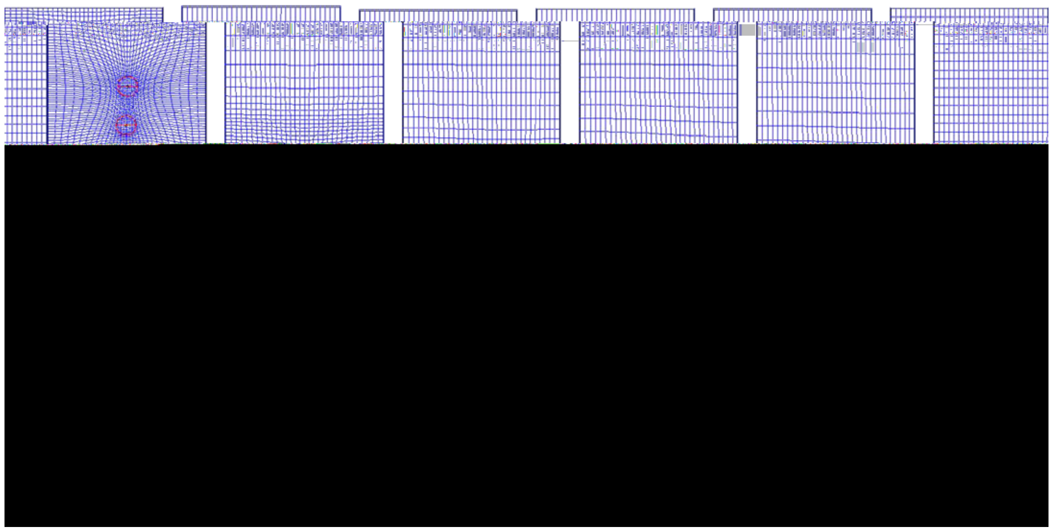


Fig. 14. Deformed meshes for two circular disks falling down in a channel.

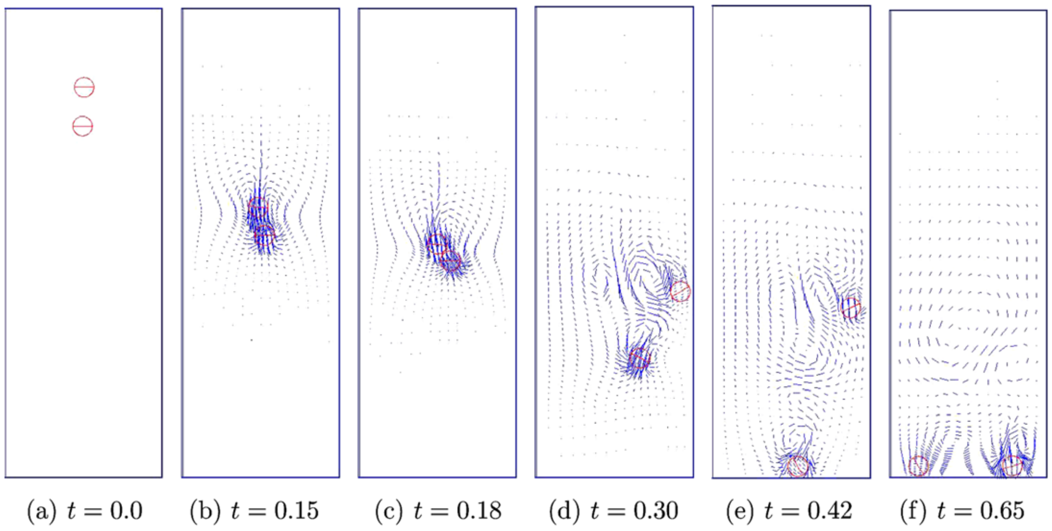
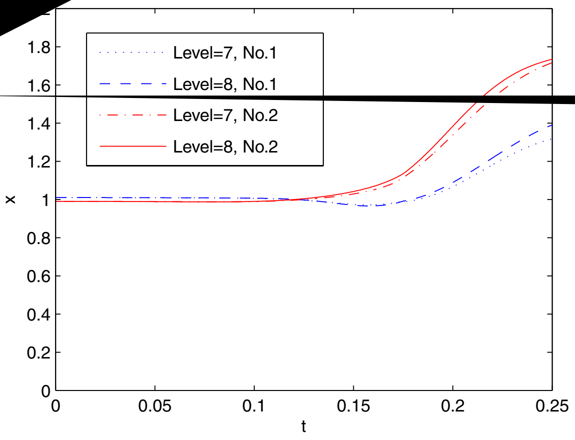


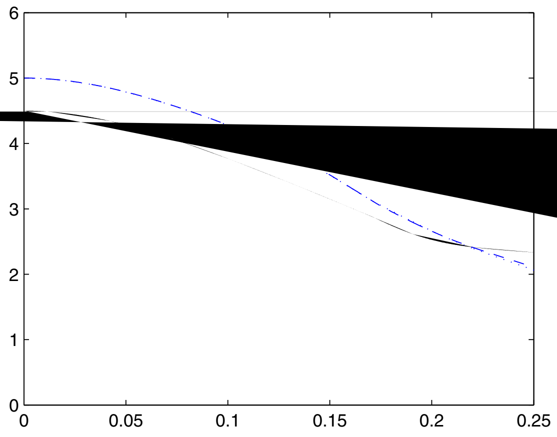
Fig. 15. Vector fields for two circular disks falling down in a channel.

The density of the fluid is $\rho_f = 1$ and the dens[redacted]$= 10^{-2}$. The initial condition for the fluid velocity i[redacted]0 when $y = 0$ or 4 and $\mathbf{u} = 1$ when $x = 0$ or 20 for $t \geqslant$[redacted]ce of the airfoil are zero. The airfoil length is 1.0089304[redacted](0.420516, 2). Hence the Reynolds number is about 101 wit[redacted]low speed. The detailed shape of the NACA0012 is described[redacted]

$$Y = 0.6 \cdot \{0.2969 \cdot \sqrt{X}$$ [redacted] $$.2843 - 0.1015 \cdot X)]]\}.$$

The simulation is re[redacted]eshes and moving deformed meshes, each of them having two different [redacted]1,409 nodes and 40,960 elements, as well as LEVEL = 8 with 164,737 node[redacted]deformed mesh is generated in each time step in order to always keep the grid al[redacted]ce of the induced rotating airfoil. Fig. 10(a) shows the starting equidistant mesh[redacted]ned meshes. In Fig. 10(b) and (c), two deformed meshes at $t = 14.7$ and $t = 16.0$ a[redacted]ocal zooms are illustrated in Fig. 10(d) and (e). Fig. 11 plots the time history





of the angle of incidence θ and the angular velocity ω of the airfoil calculated by using the equidistant and the deformed meshes, each of them is performed on the two levels LEVEL = 7 and LEVEL = 8, respectively. The local vorticity distribution, local pressure contour, norm of velocity and local vector field for $t = 14.7$ and $t = 16.0$ are given in Figs. 12 and 13. From these figures and pictures, we can see that the airfoil quickly reaches a periodic motion and intends to keep its broadside perpendicular to the in-flow direction which is a stable position for a non-circular rigid body settling in a channel at moderate Reynolds numbers. We observe that the results of the deformed meshes converge better to a mesh independent solution than those on the equidistant meshes, and are in excellent agreement with those obtained by Glowinski, Joseph and coauthors [7,8]. The results of the equidistant meshes exhibit much more numerical oscillations and loose stability

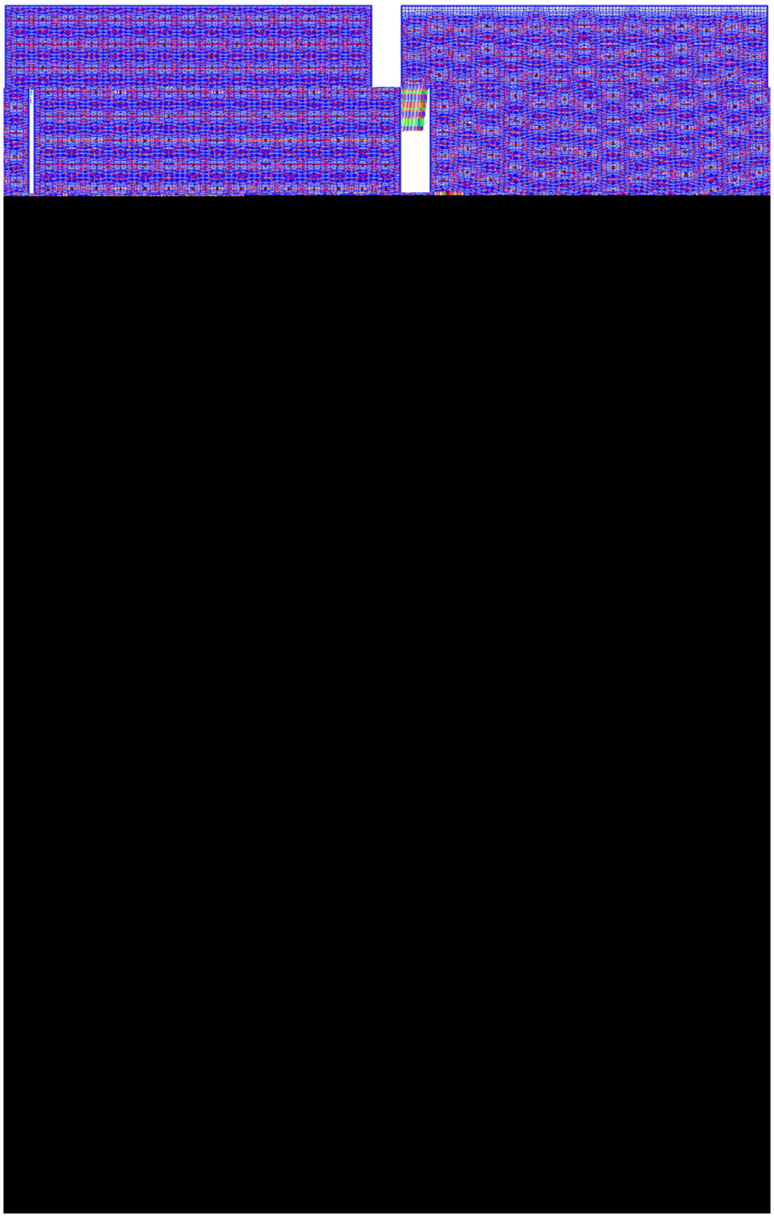


Fig. 17. Deformed meshes for 120 particles falling down in a cavity.

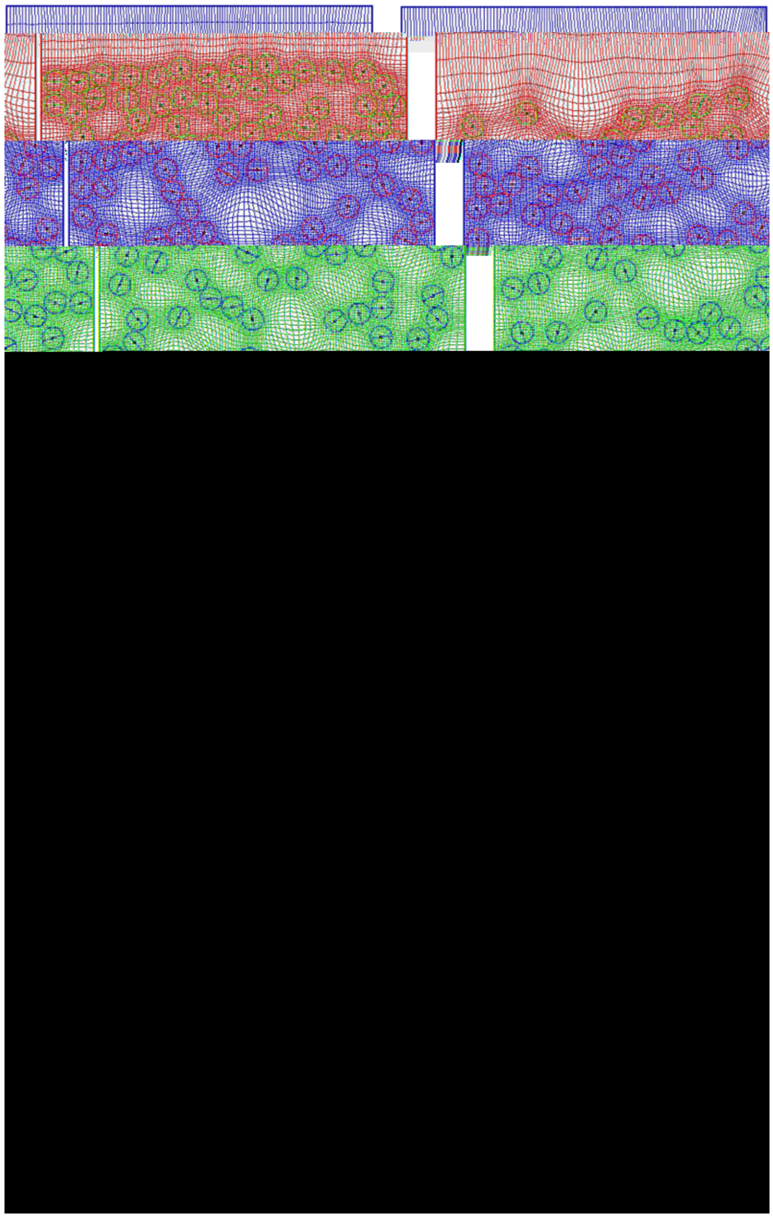


Fig. 17 (*continued*).

since they cannot catch very well the velocity field close to the leading edge of the airfoil, which causes the numerical solution blew up near the leading edge of the airfoil. Obviously, good results and significant accuracy improvements are achieved by using the moving grid deformation techniques. It illustrates that the presented method can easily handle more complex shapes of rigid bodies and can obtain more accurate results than those without employing the moving grid deformation techniques.

5.5. *Drafting, kissing and tumbling of two disks in a channel*

We will carry on the cases of multiple particles in a fluid to show that the presented method can be easily applied to more realistic particulate flows with large number of moving particles.

When two particles are dropped close to each other, they interact by undergoing "drafting, kissing and tumbling" [27], which is often chosen to examine the complete computational model of particulate flows, including the prevention of collisions. Therefore, we also study the sedimentation of two circular particles in a two-dimensional channel, comparing the results w.r.t. two different levels of mesh refinement and regarding the results in [8]. The computational domain is a channel of width 2 and height 6. Two rigid circular disks with diameter $d = 0.25$ and density $\rho_p = 1.5$ are located at (1, 5) (No. 1 disk) and (1, 4.5) (No. 2 disk) at time $t = 0$, and they are falling down under gravity in an incompressible fluid with density $\rho_f = 1$ and viscosity $\nu = 0.01$, with gravitational acceleration constant $g = -980$. We suppose that the disks and the fluid are initially at rest. The simulation is carried out on moving deformed meshes, having two different levels, i.e., LEVEL = 7 with 49,665 nodes and 49,152 elements, as well as LEVEL = 8 with 19,7633 nodes and 196,608 elements which gives sufficiently accurate solutions.

Fig. 14 shows the moving deformed meshes employed in simulation of the two falling disks. The grid lines are always concentrated around the surfaces of the two disks and in the region of the gap between the two disks, and move with the two disks during the computations. Fig. 15 presents the corresponding computed vector fields. From these figures, we can see that the disk in the wake (No. 1 particle) falls more rapidly than the disk No. 2 in front since the fluid forces acting on it are smaller. The gap between them decreases, and they almost touch ("kiss") each other at time $t = 0.15$. After touching, the two disks fall together until they tumble

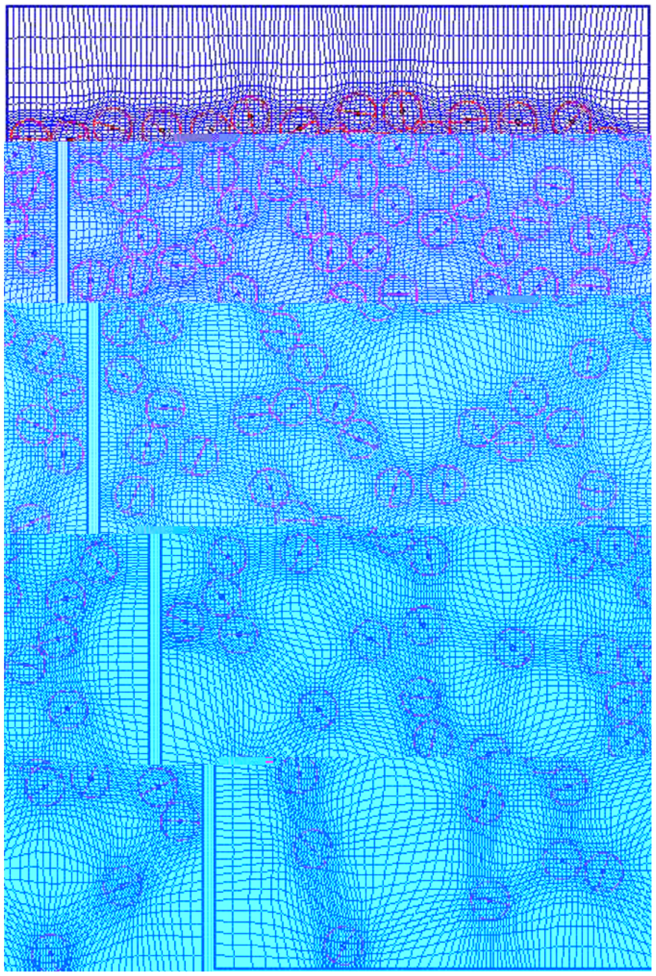


Fig. 18. Zoomed mesh at $t = 1.53$ for 120 particles falling down in a cavity.

($t = 0.18$) and subsequently they separate from each other ($t = 0.30$). The tumbling of the disks takes place because the configuration, when both are parallel to the flow direction, is unstable. The No. 1 disk is touching first the bottom wall at $t = 0.42$, while the No. 2 disk reaches the bottom wall at $t = 0.65$. In Fig. 16, several quantities are plotted. These are the time histories of the x- and y-coordinate of the two disk centers, u- and v-component of the disk translational velocities, obtained for the two deformed meshes, LEVEL = 7 and LEVEL = 8, respectively. We can see that the results computed on the two different deformed meshes are essentially indistinguishable. All results compare qualitatively well with those presented in [6,8,28,29].

5.6. Sedimentation of 120 circular particles

Finally, we consider the sedimentation of 120 circular particles with identical size falling down in a closed rectangular cavity. The width and height of the cavity are 4 and 6. The 120 particles are placed at the top of the cavity with eight rows, while in each row the number of particles is 15. The diameter of the particles is 0.24. The range of the repulsive force is chosen as $\rho = 0.0167$. The position of the particles at time $t = 0$ is shown in Fig. 20(a). The particles and the fluid are at rest at $t = 0$. The density of the fluid is $\rho_f = 1$ and the density of the particles is $\rho_i = 1.1$ ($i = 1, \ldots, 120$). The viscosity of the fluid is $\nu = 10^{-2}$ (all quantities in non-dimensional form). The parameter ϵ_P in the collision model has been taken as 10^{-6}, and $\epsilon_W = \epsilon_P/2$, $\epsilon'_P = \epsilon_P$, $\epsilon'_W = \epsilon_W$.

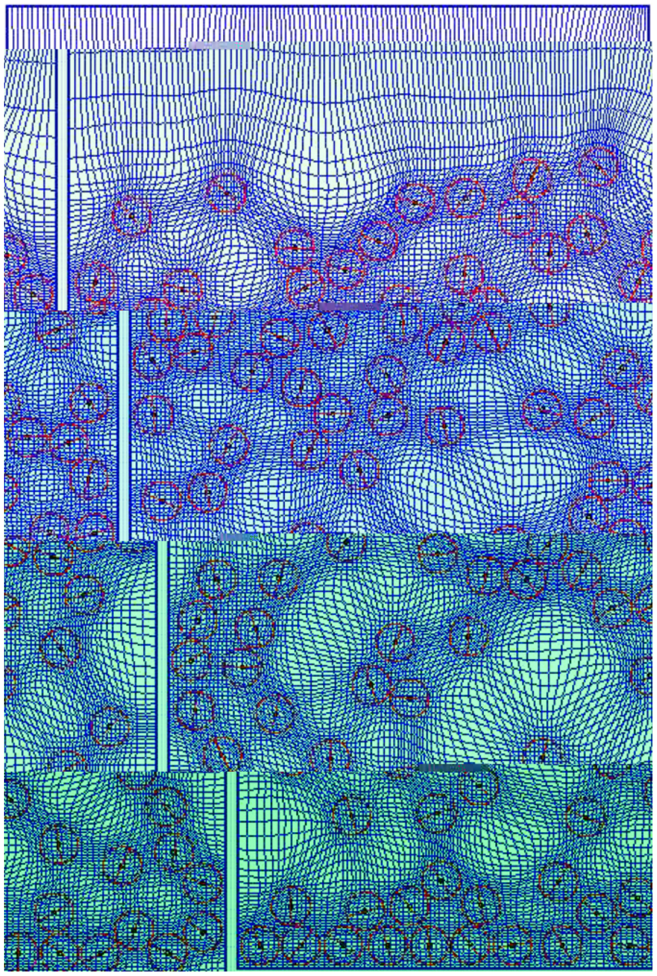
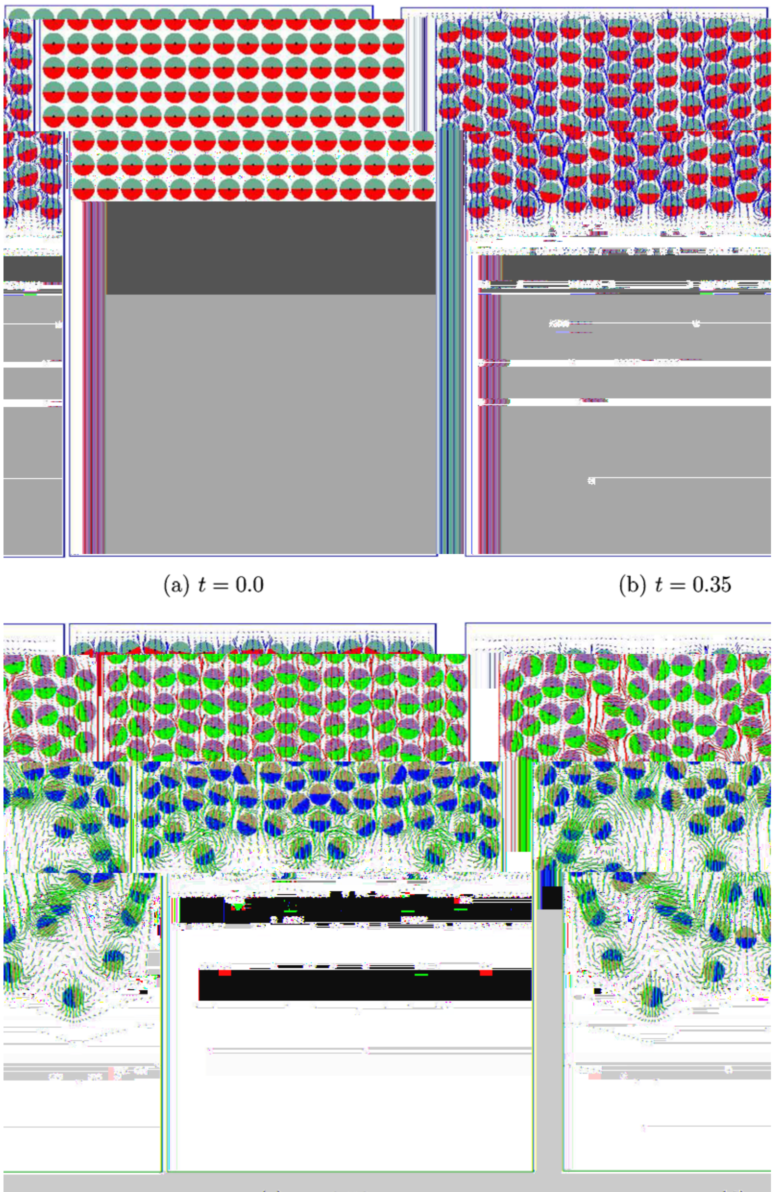


Fig. 19. Zoomed mesh at $t = 2.04$ for 120 particles falling down in a cavity.

The moving meshes are shown in Fig. 17, and two enlarged meshes at $t = 1.53$ and $t = 2.04$ are presented in Figs. 18 and 19, respectively. We can see that grid lines are moved and concentrated around the surfaces of each particle and in the regions of the gap between particles, and can always keep alignment with the surfaces of each particle even when the particle move. The corresponding snapshots for the evolution of the vector fields of the 120 sedimenting circular particles are illustrated in Fig. 20. These figures clearly show the development of the Rayleigh–Taylor instability. This instability develops into a fingering and text-book phenomenon, and many symmetry breaking and other bifurcation phenomena including drafting, kissing and tumbling take place at various scales in space and time. Many complex vortices have been formed which pull the particles downward and mix each other. Finally, the particles settle at the bottom of the cavity and the fluid returns to rest.



(a) $t = 0.0$ (b) $t = 0.35$

$= 1.02$ (c) $t = 0.56$ (d) t

Fig. 20. Velocity fields for 120 particles falling down in a cavity.

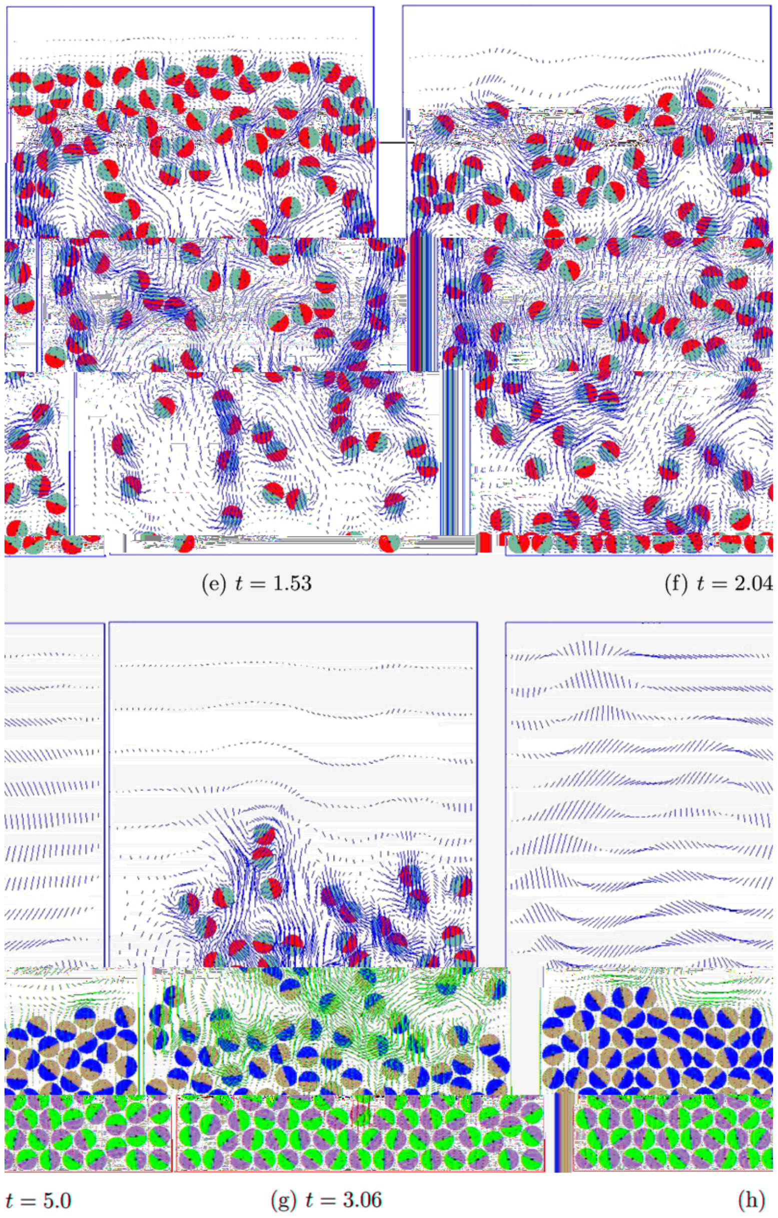


Fig. 20 (*continued*).

6. Conclusions

We have presented the combination of the multigrid fictitious boundary method and a new moving mesh method for the simulation of particulate flows with moving rigid particles of different shape and size. The new approach directly improves the accuracy upon the previous pure multigrid FBM based on static non-aligned meshes. Moreover, it is computationally cheap and simple to implement. Since the size of the problem and the data structure of the moving deformed meshes are fixed, this enables the proposed method to be incorporated into most CFD codes without the need for changing the matrix structures and for adding special interpolation procedures. Numerical tests demonstrate that it is suitable to accurately and efficiently perform the direct numerical simulation of particulate flows with large number of moving particles. One classical CFD benchmark experiment, three numerical examples of single moving particles in a fluid as well as the drafting, kissing and tumbling of two disks in a channel and the sedimentation of 120 particles in a cavity have been presented to show that the new method can significantly improve the accuracy for dealing with the interaction between the fluid and the particles, and can be easily applied to more realistic flow configurations with many moving particles.

In the future, we plan to combine these mesh deformation techniques in the context of the FEM–ALE fictitious boundary approach with special hardware-oriented implementation features for generalized tensor-product meshes: as long as the (local) data structure of such highly structured meshes with different local spacing is preserved, special cache and pipelining-oriented techniques can be developed which can significantly exploit the available supercomputing power of modern processors and which allow special hardware components like GPU-computing. Preliminary results are already available and will be presented in a forthcoming paper.

Acknowledgements

This paper is supported by the Program of Shanghai Pujiang Talent Plans (No. 06PJ14057), Program for New Century Excellent Talents in University (NCET), Chinese National Natural Science Foundation (No. 10672101), and Dortmund University.

References

[1] H.H. Hu, D.D. Joseph, M.J. Crochet, Direct simulation of fluid particle motions, Theor. Comput. Fluid Dyn. 3 (1992) 285–306.

[2] H.H. Hu, N.A. Patankar, M.Y. Zhu, Direct numerical simulations of fluid-solid systems using the arbitrary Lagrangian–Eulerian techniques, J. Comput. Phys. 169 (2001) 427–462.

[3] G.P. Galdi, V. Heuveline, Lift and sedimentation of particles on the flow of a viscoelastic liquid in a channel. Available from: <http://www.iwr.uni-heidelberg.de/organization/sfb359/PP/Preprint2004-34.pdf>.

[4] B. Maury, Direct simulations of 2D fluid-particle flows in biperiodic domains, J. Comput. Phys. 156 (1999) 325–351.

[5] R. Glowinski, T.W. Pan, T.I. Hesla, D.D. Joseph, A distributed Lagrange multiplier/fictitious domain method for particulate flows, Int. J. Multiphase Flow 25 (1999) 755–794.

[6] 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 flows, Int. J. Multiphase Flow 26 (2000) 1509–1524.

[7] 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, J. Comput. Phys. 169 (2001) 363–426.

[8] R. Glowinski, Finite element methods for incompressible viscous flow, in: P.G. Ciarlet, J.L. Lions (Eds.), Handbook of Numerical Analysis, vol. IX, North-Holland, Amsterdam, 2003, pp. 701–769.

[9] S. Turek, D.C. Wan, L.S. Rivkind, The fictitious boundary method for the implicit treatment of Dirichlet boundary conditions with applications to incompressible flow simulations, Challenges in Scientific ComputingLecture Notes in Computational Science and Engineering, vol. 35, Springer, Berlin, 2003, pp. 37–68.

[10] D.C. Wan, S. Turek, L.S. Rivkind, An efficient multigrid FEM solution technique for incompressible flow with moving rigid bodies, Numerical Mathematics and Advanced ApplicationsENUMATH 2003, Springer, Berlin, 2004, pp. 844–853.

[11] D.C. Wan, S. Turek, Direct numerical simulation of particulate flow via multigrid FEM techniques and the fictitious boundary method, Int. J. Numer. Method Fluids 51 (2006) 531–566.

[12] D.C. Wan, S. Turek, An efficient multigrid-FEM method for the simulation of liquid–solid two phase flows, J. Comput. Appl. Math. (in press).

[13] M.J. Berger, P. Collela, Local adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys. 82 (1989) 64–84.

[14] W. Huang, Practical aspects of formulation and solution of moving mesh partial differential equations, J. Comput. Phys. 171 (2001) 753–775.

[15] H.D. Ceniceros, T.Y. Hou, An efficient dynamically adaptive mesh for potentially singular solutions, J. Comput. Phys. 172 (2001) 609–639.

[16] P.B. Bochev, G. Liao, G.C. de la Pena, Analysis and computation of adaptive moving grids by deformation, Numer. Methods Partial Differ. Equat. 12 (1996) 489.

[17] G. Liao, B. Semper, A moving grid finite-element method using grid deformation, Numer. Methods Partial Differ. Equat. 11 (1995) 603–615.

[18] F. Liu, S. Ji, G. Liao, An adaptive grid method and its application to steady Euler flow calculations, SIAM J. Sci. Comput. 20 (3) (1998) 811–825.

[19] X.X. Cai, D. Fleitas, B. Jiang, G. Liao, Adaptive grid generation based on least-squares finite-element method, Comput. Math. Appl. 48 (7–8) (2004) 1077–1086.

[20] M. Grajewski, M. Köster, S. Kilian, S. Turek, Numerical analysis and practical aspects of a robust and efficient grid deformation method in the finite element context. Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 294, FB Mathematik, Universität Dortmund, 2005. Available from: <http://www.mathematik.uni-dortmund.de/lsiii/static/showpdffile_GrajewskiKoester-KilianTurek2005.pdf>.
[21] B. Dacorogna, J. Moser, On a partial differential equation involving the Jacobian determinant, Annales de le Institut Henri Poincare 7 (1990) 1–26.
[22] S. Turek, Efficient Solvers for Incompressible Flow Problems, Springer, Berlin/Heidelberg/New York, 1999.
[23] S. Turek, FEATFLOW-finite element software for the incompressible Navier–Stokes equations: User Manual, Release 1.2, University of Dortmund, 1999.
[24] S. Turek, A comparative study of time stepping techniques for the incompressible Navier–Stokes equations: from fully implicit nonlinear schemes to semi-implicit projection methods, Int. J. Numer. Meth. Fluids 22 (1996) 987–1011.
[25] S. Turek, On discrete projection methods for the incompressible Navier–Stokes equations: an algorithmical approach, Comput. Methods Appl. Mech. Eng. 143 (1997) 271–288.
[26] M. Schäfer, S. Turek, Benchmark computations of laminar flow around cylinder, in: E.H. Hirschel (Ed.), Flow Simulation with High-Performance Computers II. Notes on Numerical Fluid Mechanics, vol. 52, Vieweg, 1996, pp. 547–566.
[27] A. Fortes, D.D. Joseph, T. Lundgren, Nonlinear mechanics of fluidization of beds of spherical particles, J. Fluid Mech. 177 (1987) 483–497.
[28] P. Singh, T.I. Hesla, D.D. Joseph, Distributed Lagrange multiplier method for particulate flows with collisions, Int. J. Multiphase Flow 29 (2003) 495–509.
[29] C. Diaz-Goano, P. Minev, K. Nandakumar, A Lagrange multiplier/fictitious domain approach to particulate flows, in: S. Margenov, P. Yalamov (Eds.), Lecture Notes in Computer Science, vol. 2179, Springer, Berlin, 2001, pp. 409–422.