ELSEVIER

Comput. Methods Appl. Mech. Engrg. 190 (2001) 3201–3221

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Methods for 3D computation of fluid–object interactions in spatially periodic flows

Andrew Johnson [a], Tayfun Tezduyar [b,*]

[a] *Network Computing Services, 1200 Washington Avenue South, Minneapolis, MN 55415, USA*

[b] *Mechanical Engineering and Materials Science, Rice University [1] – MS 321, 6100 Main Street, Houston, TX 77005-1892, USA*

Received 12 February 1999

Abstract

We present computational methods for 3D simulation of fluid–object interactions in spatially periodic flows. These methods include a stabilized space-time finite element formulation for incompressible flows with spatial periodicity, automatic mesh generation and update techniques for fluid–object mixtures with spatial periodicity, and parallel implementations. The methods can be applied to uni-periodic (i.e., periodic in one direction), bi-periodic, or tri-periodic flows. The methods are described here in the context of tri-periodic flows with fluid–object interactions, and are applied to the simulation of sedimentation of particles in a fluid. We present several case studies where the results obtained provide notable insight into the behavior of fluid–particle mixtures during sedimentation.

1. Introduction

The deformable-spatial-domain/stabilized space-time (DSD/SST) formulation [1,2] developed earlier gave us a powerful computational capability for simulation of a wide class of complex flow problems with moving boundaries and interfaces, including fluid–object interactions. Although we had some method development and simulation effort in the past for spatially periodic flows [3], these studies were limited to 2D problems, and were based on the vorticity-stream function equations of incompressible flows. Furthermore, in these studies the objects (cylinders) were fixed.

In our earlier fluid–object interaction studies with the DSD/SST formulation, we did not address spatially periodic flows. In all those studies, the flow problem involved a number of spheres falling in a liquid-filled tube. In these class of problems, in September 1994 we completed the simulations for several cases of 2–5 spheres at Reynolds number 100 (see [4]). We then proceeded to improve continuously our numerical simulation techniques and implementation strategies, and completed in June 1996 the simulation of 101 spheres, again at Reynolds number 100 (see [5]). Finally, in December 1997 we completed the simulation of 1000 spheres at Reynolds number 10 (see [6]). All these computations were performed on Thinking Machines CM-5. These simulations, while yielding many interesting flow features, were mainly used to test the methods and tools we have been developing for large-scale computation of fluid–object interactions as a more general class of problems. Although fluid–particle interactions were the applications we were focusing on while we were developing these numerical methods, this class of methods can also be and have been applied to other applications involving moving objects (see [7,8]).

* Corresponding author. Tel.: +1-713-348-6051; fax: +1-713-348-5423.

E-mail address: tezduyar@rice.edu (T. Tezduyar).

[1] http://www.mems.rice.edu/TAFSM/

0045-7825/01/$ - see front matter
PII: S0045-7825(00)00389-3

In this paper, we describe the computational methods we developed for the purpose of simulating 3D spatially periodic flows, specifically targeting fluid–particle mixtures. We believe that this is the next, natural step for us in terms of developing simulation and modeling methods and tools for fluid–particle mixtures. We expect that with this new capability we will be able to consider models which are more relevant to real-world applications. We will be able to handle models that are uni-periodic (i.e., periodic only in 1D), bi-periodic, or tri-periodic, where the flow rate is specified in the direction of periodicity. In our past simulations of spherical particles in a liquid-filled tube, the particles existed only within a finite length of the tube and the tube did not have particles everywhere else. If we performed these simulations with a uni-periodic model with the periodicity along the tube axis, we would be modeling the same problem but with particles everywhere within the tube. This would be the case in a number of applications involving the transport of a fluid–particle mixture. If we perform simulations with bi-periodic models, we would be simulating fluid–particle mixtures constrained by two parallel plates.

The case studies we report in this paper are all tri-periodic flows, where a continuous fluid–particle mixture that extends to infinity in all three directions is being modeled. In most fluid–particle mixture applications, particles are very small and millions of them are involved. If we zoom into a small portion of a large-scale fluid–particle mixture, simulation of the behavior of a small "representative sample" of the mixture might be sufficient to obtain significant information about the behavior of the entire mixture. In such cases, the size of this representative sample and the number of particles being modeled become important factors, and these are some of the issues we briefly explore in this paper.

The fluid–particle mixture problems of the type considered here are very challenging to simulate due to the complex geometry involved and the unrestricted (and also unknown) motion of the objects. We developed a new version of the DSD/SST formulation which is applicable to spatially periodic flows. Because this is a space-time formulation, where finite element functions are also used to discretize the time domain, the formulation automatically accounts for the changes in the shape of the spatial domain occupied by the fluid. To handle the complex geometries involved, we employ automatic mesh generation techniques (see [6,9]) based on the Delaunay method, and create 3D meshes with tetrahedral elements. In some cases, we create thin layers of elements around the spheres in order to model better the boundary layer features of the flow. Adding the capability for spatial periodicity makes the mesh generation process even more complicated. Furthermore, to complete a simulation in a reasonable amount of time, distributed-memory parallel computing techniques are employed [9–11]. To update the mesh as the particles move around, we employ an automatic mesh moving scheme with remeshing as needed [4,12]. Along with remeshing comes techniques to accurately and efficiently project the computed solution between two meshes. These methods, together with other computational strategies such as multi-platform computing, yield a powerful capability to simulate one of the most challenging classes of flow problems.

In Section 2, we present our new version of the DSD/SST formulation for spatially periodic flows. In Section 3, we provide an overview of the techniques we developed for automatically generating meshes for periodic fluid domains, and in Section 4 we briefly outline the computation strategy and features of our new fluid–particle interaction solver. Section 5 contains details of our parallel implementation of the formulation for spatially periodic flows. Several numerical examples of fluid–particle flows in tri-periodic domains are reported in Section 6, and concluding remarks are presented in Section 7.

2. Governing equations and formulation

We consider a viscous, incompressible fluid occupying at time $t \in (0, T)$ a bounded region $\Omega_t \subset \mathbb{R}^{n_{\mathrm{sd}}}$, where n_{sd} is the number of spatial dimensions. The boundary of Ω_t is Γ_t. The primary degrees of freedom are velocity and pressure, denoted by $\mathbf{u}(\mathbf{x}, t)$ and $p(\mathbf{x}, t)$. The governing equations are the momentum and mass balance equations

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} - \mathbf{f}\right) - \nabla\cdot\boldsymbol{\sigma} = \mathbf{0} \text{ on } \Omega_t, \quad \forall t \in (0, T), \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0 \text{ on } \Omega_t, \quad \forall t \in (0, T), \tag{2}$$

where ρ is the fluid density, $\boldsymbol{\sigma}$ the stress tensor, and $\mathbf{f}(\mathbf{x},t)$ is the body force such as gravity. The stress and strain-rate tensors are defined as

$$\boldsymbol{\sigma}(p,\mathbf{u}) = -p\mathbf{I} + 2\mu\boldsymbol{\varepsilon}(\mathbf{u}), \qquad \boldsymbol{\varepsilon}(\mathbf{u}) = \frac{1}{2}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^{\mathrm{T}}\right), \tag{3}$$

where μ is the viscosity and $\mathbf{I}$ is the identity tensor. The Dirichlet- and Neumann-type boundary conditions are written as

$$\mathbf{u}\cdot\mathbf{e}_i = g_i \text{ on } (\Gamma_t)_{g_i}, \quad i = 1,\ldots,n_{\mathrm{sd}}, \tag{4}$$

$$\mathbf{n}\cdot\boldsymbol{\sigma}\cdot\mathbf{e}_i = h_i \text{ on } (\Gamma_t)_{h_i}, \quad i = 1,\ldots,n_{\mathrm{sd}}, \tag{5}$$

where $\mathbf{e}_i$ is the cartesian unit vector corresponding to axis i, and $\mathbf{n}$ is the unit normal vector for the boundary. Boundaries $(\Gamma_t)_{g_i}$ and $(\Gamma_t)_{h_i}$ are complementary subsets of the boundary Γ_t. The initial condition is a divergence-free velocity field specified over the entire domain

$$\mathbf{u}(\mathbf{x},t) = \mathbf{u}_0 \text{ on } \Omega_0. \tag{6}$$

Before writing the space-time formulation of Eqs. (1)–(5), we partition the time interval $(0,T)$ into sub-intervals denoted by $I_n = (t_n, t_{n+1})$, where t_n and t_{n+1} belong to an ordered series of time levels $0 = t_0 < t_1 < \cdots < t_N = T$. Also, we define $\Omega_n = \Omega_{t_n}$ and $\Gamma_n = \Gamma_{t_n}$. We define the space-time slab (Q_n) as the domain enclosed by surfaces Ω_n, Ω_{n+1} and P_n, where P_n is the surface described by boundary Γ_t as t traverses I_n. Surface P_n is decomposed into $(P_n)_{g_i}$ and $(P_n)_{h_i}$, $i = 1,\ldots,n_{\mathrm{sd}}$. For each space-time slab, we define the following finite element interpolation function spaces:

$$(S^h_{\mathbf{u}})_n = \left\{\mathbf{u}^h = \left[u^h_i\right]_{i=1}^{n_{\mathrm{sd}}} \middle| u^h_i \in H^{1h}(Q_n), u^h_i \doteq g^h_i \text{ on } (P_n)_{g_i}, \quad i = 1,\ldots,n_{\mathrm{sd}}\right\}, \tag{7}$$

$$(V^h_{\mathbf{u}})_n = \left\{\mathbf{w}^h = \left[w^h_i\right]_{i=1}^{n_{\mathrm{sd}}} \middle| w^h_i \in H^{1h}(Q_n), w^h_i \doteq 0 \text{ on } (P_n)_{g_i}, \quad i = 1,\ldots,n_{\mathrm{sd}}\right\}, \tag{8}$$

$$(V^h_p)_n = (S^h_p)_n = \left\{p^h \mid p^h \in H^{1h}(Q_n)\right\}. \tag{9}$$

In these definitions, $H^{1h}(Q_n)$ represents the finite-dimensional function space over the space-time slab. Within the element domain, this space is formed using first-order polynomials in both space and time. Globally, these interpolation functions are continuous in space, but are discontinuous in time.

The DSD/SST formulation can be written as follows: given $(\mathbf{u}^h)^-_n$, find $\mathbf{u}^h \in (S^h_{\mathbf{u}})_n$ and $p^h \in (S^h_p)_n$ such that $\forall \mathbf{w}^h \in (V^h_{\mathbf{u}})_n$ and $\forall q^h \in (V^h_p)_n$

$$\begin{aligned}
&\int_{Q_n} \mathbf{w}^h\cdot\rho\left(\frac{\partial\mathbf{u}^h}{\partial t} + \mathbf{u}^h\cdot\nabla\mathbf{u}^h - \mathbf{f}^h\right)\mathrm{d}Q + \int_{Q_n}\boldsymbol{\varepsilon}(\mathbf{w}^h):\boldsymbol{\sigma}(p^h,\mathbf{u}^h)\,\mathrm{d}Q - \int_{(P_n)_{\mathbf{h}}}\mathbf{w}^h\cdot\mathbf{h}^h\,\mathrm{d}P + \int_{Q_n} q^h\nabla\cdot\mathbf{u}^h\,\mathrm{d}Q \\
&\quad + \int_{\Omega_n}(\mathbf{w}^h)^+_n\cdot\rho\left((\mathbf{u}^h)^+_n - (\mathbf{u}^h)^-_n\right)\mathrm{d}\Omega + \sum_{e=1}^{(n_{\mathrm{el}})_n}\int_{Q^e_n}\frac{\tau}{\rho}\left[\rho\left(\frac{\partial\mathbf{w}^h}{\partial t} + \mathbf{u}^h\cdot\nabla\mathbf{w}^h\right) - \nabla\cdot\boldsymbol{\sigma}(q^h,\mathbf{w}^h)\right] \\
&\quad \cdot\left[\rho\left(\frac{\partial\mathbf{u}^h}{\partial t} + \mathbf{u}^h\cdot\nabla\mathbf{u}^h - \mathbf{f}^h\right) - \nabla\cdot\boldsymbol{\sigma}(p^h,\mathbf{u}^h)\right]\mathrm{d}Q = 0,
\end{aligned} \tag{10}$$

where n_{el} is the number of elements. This process is applied sequentially to all space-time slabs $Q_0, Q_1, \ldots, Q_N$. In Eq. (10), the following notation is being used:

$$(\mathbf{u}^h)^{\pm}_n = \lim_{\epsilon\to 0}\mathbf{u}(t_n \pm \epsilon), \tag{11}$$

$$\int_{Q_n}(\cdots)\,\mathrm{d}Q = \int_{I_n}\int_{\Omega_t}(\cdots)\,\mathrm{d}\Omega\,\mathrm{d}t, \tag{12}$$

$$\int_{P_n}(\cdots)\,\mathrm{d}P = \int_{I_n}\int_{\Gamma_t}(\cdots)\,\mathrm{d}\Gamma\,\mathrm{d}t. \tag{13}$$

Remarks.

1. *The first three terms of Eq.* (10) *are the Galerkin form of the momentum balance equation, and the fourth term constitutes the Galerkin form of the mass balance equation. We use the fifth term to weakly enforce the continuity of the velocity field across space-time slabs.*
2. *The sixth term in Eq.* (10) *is a least-squares form of the momentum balance equation. This term provides the necessary stability for advection-dominated flows, as well as provide stability when equal-order interpolation function spaces are used for both velocity and pressure. The definition of* τ *can be found in* [1,13].

By extending this formulation to spatially periodic flows, we first define our notation using a sample 2D computational domain (see Fig. 1). This figure represents a rectangular domain (width $\times$ height $= L \times H$) with N circular objects, denoted by Γ_κ where $\kappa = 1, 2, \ldots, N$. The four outer boundaries are denoted by Γ_{I}, Γ_{II}, Γ_{III}, and Γ_{IV}. For simplicity in the following discussion, we will drop the superscript h and the subscript n.

We want to derive a formulation applicable to uni-periodic (i.e., periodic in one direction), bi-periodic and tri-periodic flows, where the total volumetric flow rate in each periodic direction is prescribed. We will first derive the formulation for uni-periodic flows, and then extend it to bi-periodic and tri-periodic cases.

First we look in Eq. (10) the terms relating to the stress $\boldsymbol{\sigma}$

$$\int_Q \boldsymbol{\varepsilon}(\mathbf{w}) : \boldsymbol{\sigma}(p, \mathbf{u})\, \mathrm{d}Q = \int_{P_{\mathrm{I}}} \mathbf{w} \cdot \mathbf{h}_{\mathrm{I}}\, \mathrm{d}P + \int_{P_{\mathrm{III}}} \mathbf{w} \cdot \mathbf{h}_{\mathrm{III}}\, \mathrm{d}P. \tag{14}$$

We assume periodicity only in the x_1 direction, and the total volumetric flow rate we wish to prescribe in this direction is denoted by $\dot{V}$. We define a continuous, periodic velocity field $\mathbf{u}^*$ and an associated weighting function $\mathbf{w}^*$ such that

$$\mathbf{u}^*|_{P_{\mathrm{I}}} = \mathbf{u}^*|_{P_{\mathrm{III}}}, \tag{15}$$

$$\mathbf{w}^*|_{P_{\mathrm{I}}} = \mathbf{w}^*|_{P_{\mathrm{III}}}. \tag{16}$$

With this definition, Eq. (14) reduces to

$$\int_Q \boldsymbol{\varepsilon}(\mathbf{w}^*) : \boldsymbol{\sigma}(p, \mathbf{u}^*)\, \mathrm{d}Q = \int_{P_{\mathrm{III}}} (\mathbf{w}^* \cdot \mathbf{e}_1)(p_{\mathrm{I}} - p_{\mathrm{III}})\, \mathrm{d}P, \tag{17}$$

where $\mathbf{e}_1$ is the cartesian unit vector corresponding to the x_1 axis. We define the term J as $J(t) = p_{\mathrm{III}} - p_{\mathrm{I}}$ which is the pressure jump across the domain in the x_1 direction. This pressure jump term, which is now an additional unknown, drives the flow in the direction of periodicity, and needs to be solved for. Eq. (17) can then be rewritten as

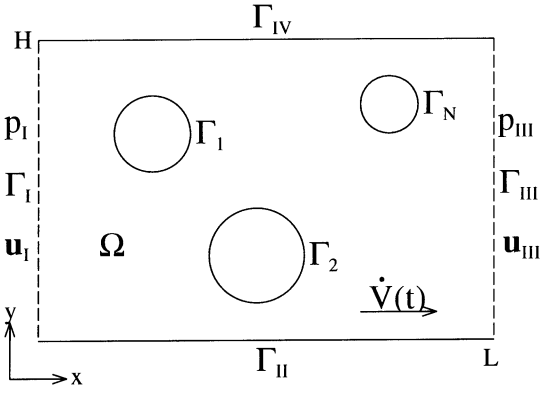

Fig. 1. Computational domain.

$$\int_Q \boldsymbol{\varepsilon}(\mathbf{w}^*) : \boldsymbol{\sigma}(p, \mathbf{u}^*)\, \mathrm{d}Q = -\int_{P_{\mathrm{III}}} (\mathbf{w}^* \cdot \mathbf{e}_1) J\, \mathrm{d}P. \tag{18}$$

Now we look at the mass conservation terms. Prescribing the total volumetric flow rate in the periodic direction introduces the constraint

$$\dot{V} - \int_{\Gamma_{\mathrm{III}}} (\mathbf{u}^* \cdot \mathbf{e}_1)\, \mathrm{d}\Gamma = 0. \tag{19}$$

We include this term along with the mass conservation term in Eq. (10) as follows:

$$\int_Q q(\nabla \cdot \mathbf{u}^*)\, \mathrm{d}Q + \int_I K \left[\dot{V} - \int_{\Gamma_{\mathrm{III}}} (\mathbf{u}^* \cdot \mathbf{e}_1)\, \mathrm{d}\Gamma \right] \mathrm{d}I = 0, \tag{20}$$

where K is the variation of the pressure jump J, and I represents the time domain.

In order to simplify the formulation and the subsequent implementation, we introduce a space-reduction procedure. We decompose pressure p and its associated weighting function q into two parts

$$p = p^* + \frac{J}{L} x_1, \tag{21}$$

$$q = q^* + \frac{K}{L} x_1. \tag{22}$$

As is the case with $\mathbf{u}^*$ and $\mathbf{w}^*$, p^* and q^* represent continuous forms across the periodic boundaries, while the discontinuity is now associated with the J and K terms.

Now we introduce this new form of p into Eq. (18)

$$\int_Q \boldsymbol{\varepsilon}(\mathbf{w}^*) : \boldsymbol{\sigma}(p^*, \mathbf{u}^*)\, \mathrm{d}Q - \int_Q (\nabla \cdot \mathbf{w}^*) \frac{J}{L} x_1\, \mathrm{d}Q = -\int_{P_{\mathrm{III}}} (\mathbf{w}^* \cdot \mathbf{e}_1) J\, \mathrm{d}P. \tag{23}$$

After integrating the second term by parts, this equation becomes

$$\int_Q \boldsymbol{\varepsilon}(\mathbf{w}^*) : \boldsymbol{\sigma}(p^*, \mathbf{u}^*)\, \mathrm{d}Q - \int_P (\mathbf{n} \cdot \mathbf{w}^*) \frac{J}{L} x_1\, \mathrm{d}P + \int_Q \frac{J}{L} (\mathbf{w}^* \cdot \nabla x_1)\, \mathrm{d}Q = -\int_{P_{\mathrm{III}}} (\mathbf{w}^* \cdot \mathbf{e}_1) J\, \mathrm{d}P. \tag{24}$$

After reducing the second term further, it cancels out with the right-hand side, and the periodic stress terms of Eq. (10) becomes

$$\int_Q \boldsymbol{\varepsilon}(\mathbf{w}^*) : \boldsymbol{\sigma}(p^*, \mathbf{u}^*)\, \mathrm{d}Q + \int_Q \frac{J}{L} (\mathbf{w}^* \cdot \mathbf{e}_1)\, \mathrm{d}Q = 0. \tag{25}$$

Now we introduce the new form of q into the mass conservation terms we derived in Eq. (20). We get

$$\int_Q q^* (\nabla \cdot \mathbf{u}^*)\, \mathrm{d}Q + \int_Q \frac{K}{L} x_1 (\nabla \cdot \mathbf{u}^*)\, \mathrm{d}Q + \int_I K \left[\dot{V} - \int_{\Gamma_{\mathrm{III}}} (\mathbf{u}^* \cdot \mathbf{e}_1)\, \mathrm{d}\Gamma \right] \mathrm{d}I = 0. \tag{26}$$

By integrating by parts the second term, we obtain

$$\int_Q q^* (\nabla \cdot \mathbf{u}^*)\, \mathrm{d}Q + \int_P \frac{K}{L} x_1 (\mathbf{n} \cdot \mathbf{u}^*)\, \mathrm{d}P - \int_Q \frac{K}{L} (\mathbf{e}_1 \cdot \mathbf{u}^*)\, \mathrm{d}Q + \int_I K \left[\dot{V} - \int_{\Gamma_{\mathrm{III}}} (\mathbf{u}^* \cdot \mathbf{e}_1)\, \mathrm{d}\Gamma \right] \mathrm{d}I = 0. \tag{27}$$

After splitting up the second term with respect to individual boundaries, the term related to integration over Γ_{III} cancels out with the same term embedded in the last term of Eq. (27). We now have

$$\int_Q q^*(\nabla\cdot\mathbf{u}^*)\,\mathrm{d}Q+\sum_{\kappa=1}^{N}\int_{P_\kappa}\frac{K}{L}x_1(\mathbf{n}\cdot\mathbf{u}^*)\,\mathrm{d}P+\int_I K\left[\dot{V}-\frac{1}{L}\int_\Omega(\mathbf{u}^*\cdot\mathbf{e}_1)\,\mathrm{d}\Omega\right]\mathrm{d}I=0. \tag{28}$$

By evaluating the second term of Eq. (28), for a circular or spherical particle, this equation reduces to our new form for the conservation of mass for a periodic domain

$$\int_Q q^*(\nabla\cdot\mathbf{u}^*)\,\mathrm{d}Q+\frac{1}{L}\int_I K\left[L\dot{V}+\sum_{\kappa=1}^{N}V_\kappa(U_1)_\kappa-\int_\Omega(\mathbf{u}^*\cdot\mathbf{e}_1)\,\mathrm{d}\Omega\right]\mathrm{d}I=0, \tag{29}$$

where V_κ and $(U_1)_\kappa$ are the volume and velocity of sphere κ, respectively. This new third term means that we should include the motion of the particles as part of the total volumetric flow rate that we wish to impose.

By introducing this space-reduction procedure, we have removed all surface integrations across the periodic boundary and all integrations are now volume integrations over the entire domain. This will help in implementing this formulation. Also, our functions are now continuous across the periodic boundary and do not involve a jump in value across this surface. Actually, the exact location of the periodic boundary is now irrelevant. How we use this to our advantage in the type of periodic mesh that we use will be discussed in Section 3.

With these changes to the formulation, we can now restate the entire DSD/SST finite element formulation for spatially periodic flows, where the formulation is applicable to uni-, bi- or tri-periodic flows. In this general formulation, for the flow rate and pressure jump vectors $\dot{\mathbf{V}}$ and $\mathbf{J}$, as well as for the vector $\mathbf{K}$, we set to zero the components not corresponding to the directions of periodicity. This general formulation is stated as follows: given $(\mathbf{u}^*)_n^-$, find $\mathbf{u}^*\in(S_{\mathbf{u}^*})_n$ and $p^*\in(S_{p^*})_n$ such that $\forall\mathbf{w}^*\in(V_{\mathbf{u}^*})_n$ and $\forall q^*\in(V_{p^*})_n$

$$\begin{aligned}
&\int_Q\mathbf{w}^*\cdot\rho\left(\frac{\partial\mathbf{u}^*}{\partial t}+\mathbf{u}^*\cdot\nabla\mathbf{u}^*-\mathbf{f}\right)\mathrm{d}Q+\int_Q\boldsymbol{\varepsilon}(\mathbf{w}^*):\boldsymbol{\sigma}(p^*,\mathbf{u}^*)\,\mathrm{d}Q+\frac{1}{L}\int_Q(\mathbf{w}^*\cdot\mathbf{J})\,\mathrm{d}Q-\int_{(P_n)_{\mathbf{h}^*}}\mathbf{w}^*\cdot\mathbf{h}^*\,\mathrm{d}P\\
&+\int_{Q_n}q^*\nabla\cdot\mathbf{u}^*\,\mathrm{d}Q+\frac{1}{L}\int_{\mathrm{I}}\mathbf{K}\cdot\left[L\dot{\mathbf{V}}+\sum_{\kappa=1}^{N}V_\kappa\mathbf{U}_\kappa-\int_\Omega\mathbf{u}^*\,\mathrm{d}\Omega\right]\mathrm{d}I+\int_{\Omega_n}(\mathbf{w}^*)_n^+\cdot\rho\big((\mathbf{u}^*)_n^+-(\mathbf{u}^*)_n^-\big)\,\mathrm{d}\Omega\\
&+\sum_{e=1}^{(n_{\mathrm{el}})_n}\int_{Q_n^e}\frac{\tau}{\rho}\left[\rho\left(\frac{\partial\mathbf{w}^*}{\partial t}+\mathbf{u}^*\cdot\nabla\mathbf{w}^*\right)-\nabla\cdot\boldsymbol{\sigma}(q^*,\mathbf{w}^*)+\frac{\mathbf{K}}{L}\right]\cdot\left[\rho\left(\frac{\partial\mathbf{u}^*}{\partial t}+\mathbf{u}^*\cdot\nabla\mathbf{u}^*-\mathbf{f}\right)-\nabla\cdot\boldsymbol{\sigma}(p^*,\mathbf{u}^*)+\frac{\mathbf{J}}{L}\right]\mathrm{d}Q=0.
\end{aligned} \tag{30}$$

3. Automatic mesh generation for spatially periodic domains

For flow problems with spatially periodic domains and complex geometry (as is the case with the examples presented in this paper), we use a special automatic mesh generation procedure to construct the tri-periodic unstructured meshes. We presented this automatic mesh generation procedure first in [6], and here we briefly review it for completeness. This procedure is presented here for tri-periodic domains, but can easily be applied to cases with periodicity in only one or two directions.

Generation of a periodic unstructured mesh is mainly based on the definition of the periodic mesh and the way it is represented. We only allow nodes to exist within the periodic domain (usually a $L\times L\times L$ box), but an element can reference the "images" of these nodes in adjacent periodic domains. A proper data structure is used to define each element with both global node number references and information on which periodic domain this global node is referenced in. For example, if a node of an element is referenced in the adjacent periodic domain in the positive x_1 direction, as far as that element is concerned, the x_1 coordinate of that node is $x_1\leftarrow x_1+L$. In this way, elements can straddle the periodic boundary of the domain. An example of 2D periodic mesh is shown in Fig. 2. In this figure, the solid round nodes are the real nodes, and all others are just images of these nodes in other periodic domains. The shaded elements are the real elements, and all others are just copies and are only shown here to help visualize the mesh.

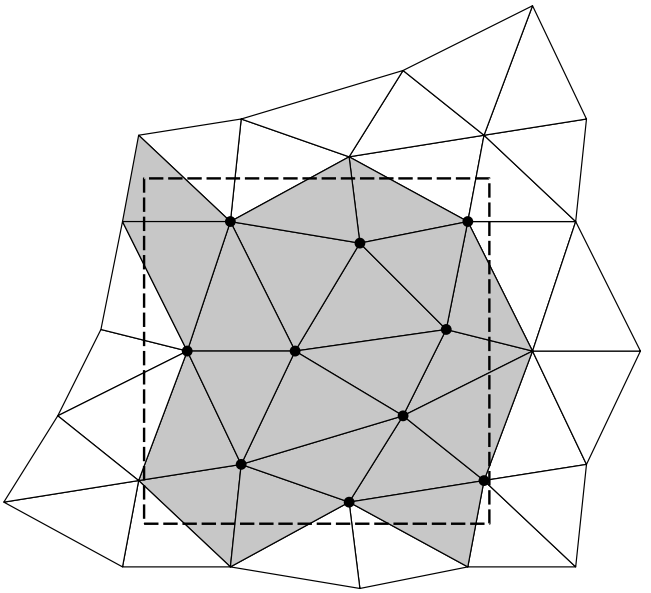

Fig. 2. An example of unstructured periodic mesh (10 nodes and 20 elements).

In formulating the finite element matrices, the exact location of the element does not matter. Only the shape and size of the element, as well as the relation of that element to the others, is all that is used in forming these matrices. In Section 2, we removed from the finite element formulation all surface integrations over the periodic boundary, and we have only the volume integrals. Therefore, it can be seen that the actual location of the periodic boundaries is irrelevant as long as the mesh is represented properly by our data structure.

With this definition of an unstructured periodic mesh and the use of proper data structures to store this information, a regular automatic mesh generation procedure can be used to actually construct the mesh. The one we use is based on the Delaunay method and incorporate face swapping techniques [9]. Our implementation is very fast; we can construct a tri-periodic mesh with 1.27 million tetrahedral elements in approximately 7 min on a modern PC (400 MHz-PentiumII). While this speed (0.18 million elements/min) is still rather high, it is lower than what we get in generating a nonperiodic mesh, which is approximately 0.67 million elements/min. This is due to the extra data structures and more complex search algorithms needed in generating a periodic mesh. Views of an example of tri-periodic mesh with eight particles is shown in Fig. 3. The resolution of the mesh in this figure is lower than those used for the numerical simulations of Section 6. See [6] for a more comprehensive description of this automatic mesh generation procedure.

4. Simulation technique

For the simulations presented here, we employ the solution technique we developed earlier (see [4,5]) for fluid–particle interaction problems. This solution technique includes an automatic mesh moving scheme to move the mesh as the particles move around, and remeshing as needed. Even though the simulation technique used here is essentially the same as the one we used in our earlier simulations, the software components used in each stage of the simulation are very different. This overhaul of the simulation software was motivated by the need to make the simulations more efficient and by the migration of our flow solver from the data-parallel programming environment of the Thinking Machines CM-5 to the message-passing programming environment of the CRAY T3E-1200.

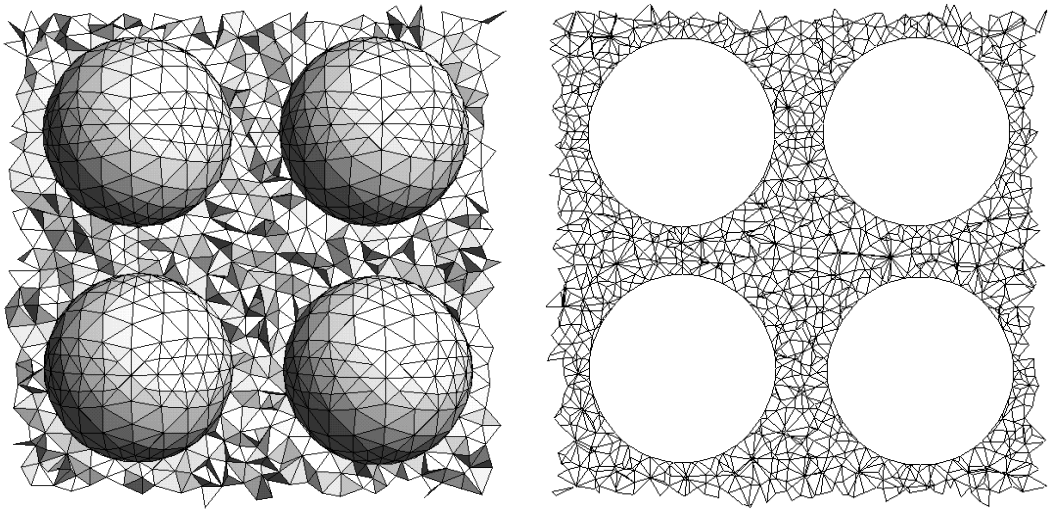

Fig. 3. Example of tri-periodic mesh (11,730 nodes and 69,219 elements). Left: particles with an artificial interior surface (surface of the mesh if it is split down the middle) and right: a 2D cross-section.

We now provide a brief overview of the components of the solution technique.

(1) *Flow solver*. Full discretization of the DSD/SST finite element formulation described in Section 2 leads to a coupled, nonlinear equation system that needs to be solved at every time step of the simulation. This nonlinear system is solved iteratively with the Newton–Raphson method. The coupled, linear equation system that needs to be solved at every step of the Newton–Raphson sequence is also solved iteratively. In parallel implementation, the METIS mesh partitioning package [14] is employed to distribute the elements to different processors. More information on the parallel implementation can be found in Section 5. In iterative solution of the linear equation system, we use a diagonal preconditioner and the GMRES update technique [15].

(2) *Particle motion*. At each nonlinear iteration of the flow solver, the forces and moments acting on the particles are computed, and, based on these fluid forces and gravity, the motion of the particles are calculated. These calculations include a collision model to account for the collision between the particles. The equations governing these calculations were presented in [5]. The discretization techniques used here are based on those presented in [5], with some improvements to increase accuracy.

(3) *Mesh movement*. As the particles move around, at each nonlinear iteration the mesh needs to be updated to accommodate the motion of the particles. We do this by solving a set of modified equations of linear elasticity that governs the motion of the nodes. This equation system is solved in parallel by using the same implementation techniques as those used for solving the flow equations. While adding somewhat to the run-time of the computations, the automatic mesh moving scheme allows us to handle the complex particle motions. More details on this technique can be found in [4,12].

(4) *Remeshing*. The mesh moving process needs to be interrupted when the elements become tangled or too distorted for the computations to proceed. When this happens, the flow solver automatically orders remeshing, where a new mesh is generated with our automatic mesh generator and the solution is projected from the old mesh to the new one. After that, the flow computations resume. Remeshing involves two steps.

(a) *Automatic mesh generation*. The Delaunay algorithm used in our automatic mesh generator (see Section 3) is an inherently serial algorithm and cannot be performed in parallel with a desirable parallel efficiency. While all other tasks of the flow simulation (including flow computations and mesh movements) are performed on the parallel computer CRAY T3E, the mesh generation takes place on a PentiumII-based PC running a Unix operating system. The data is transferred back and forth between the PC and the CRAY T3E using a standard Internet protocol (RCP). The two computers are in separate buildings, but are connected with a sufficiently high-speed network, so the data transfer times are not very large.

(b) *Projection of the solution*. In our earlier simulations, after each remesh, to project the solution from the old mesh to the new one we used a shared-memory parallel computing method, executed on a computer different than the main parallel computer (see [5]). This method was based on a 1D walking algorithm, and was taking a rather significant amount of time for some of our past simulations (see [6]). We now use a new algorithm and implementation to project the solution. This new approach is based on the message-passing programming model (MPI) and is executed on the CRAY T3E, just like all other tasks of the simulation except for mesh generation. This new parallel projection technique is based on recursive center bisection partitioning [16], and an Octree data structure is used for the actual point search. With this technique, the projection for a mesh with 1.75 million elements can be performed in 1.83 min on eight processors of the T3E.

4.1. Other features of the simulation technique

The fluid–particle interaction technique has been rewritten to support the message-passing programming model on the CRAY T3E. This programming model has been implemented using the MPI library, and because MPI runs on almost all architectures, the implementation is very portable.

The residual computations can be performed with either an element-vector-based (matrix-free) method or a sparse matrix computation technique [11].

The mesh partitioning and redistribution routines built-in to the flow solver are very fast and usually take only a few seconds out of the total simulation time.

Our new fluid–particle interaction solver is also very flexible and can be used for other classes of fluid–object interaction problems.

5. Parallel implementation of the spatially periodic formulation

Parallel implementation of our finite element flow solvers using unstructured meshes has been in place for several years (see [9–11]). It involves mesh partitioning to distribute the elements amongst the parallel processors and gather/scatter functions to exchange information between the element level and node level data sets. The scatter operation can be thought of as a standard finite element assembly operation, while the gather operation is simply a transfer of the global node data to the processor level. It is at this gather/scatter operation where inter-processor communication takes place, and we currently use the message passing interface (MPI) to facilitate these data transfers. The solution of the linear equation systems is based on diagonal preconditioning with a GMRES update technique. For residual computations during the solution of the linear equation systems, we can use two options. In the element-vector-based (matrix-free) mode, the left-hand side matrices are never formed or stored (not even at the element level), but their influence on the equation system is recomputed as needed. In the sparse-matrix storage mode, the left-hand side matrices are formed and stored in a sparse storage form and used in the matrix–vector multiplications (see [11] for more information on our iterative solvers). Of course, the matrix-free mode uses much less memory, but may involve more overall computations.

In these simulations, we use a CRAY T3E-1200 as our computing platform. It has 256 processors, but we typically use between 8 and 32 processors at a time. Each processor has 512 MBytes of dedicated memory. Our parallel implementation is very efficient and the total cost involved in inter-processor communication ranges typically between 2% and 6% of the total run time.

For periodic flows, depending on the number of directions of periodicity, we have 1–3 additional equations that are coupled to the other equations that need to be solved. These additional equations, which are related to flow rate constraints (i.e., the pressure jump equations), are global to the entire domain and are not associated with any particular node. Our parallel implementation is based on the partitioning of the elements and nodes and the associated equations that go along with each node. To incorporate these additional equations within the parallel implementation of the overall solver, we had to introduce some minor modifications to our algorithm. We add an extra fictitious node to the last processor that we are using on the parallel computer. We "assign" the pressure jump equations to this new node. This fictitious node is not connected to any element and does not participate in the normal gather/scatter operations that

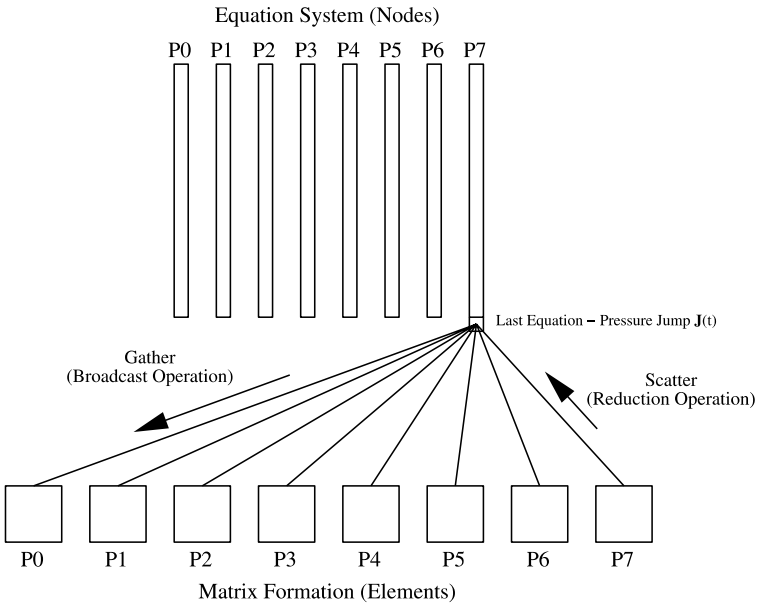

Fig. 4. Modification of the parallel implementation to account for flow rate constraints (i.e., pressure jump equations).

take place within the equation solver. We have to create additional gather/scatter operations which only operate on these few extra equations. The extra gather operation that must take place involves a global broadcast of the pressure jump equation data from the last processor to all of the other processors. This is depicted in Fig. 4. Once each processor has a copy of this pressure jump data, it can be used in forming the finite element equation system. The extra scatter step involves a global assembly operation where each processor adds its contribution to the global value being stored on the last processor. This is also depicted in Fig. 4. Once the values for these extra equations are assembled, the solution procedure can proceed with these extra equations included.

We should also note here that the elimination of the surface integrals across a periodic boundary simplifies the parallel implementation of the formulation significantly.

6. Numerical examples

To demonstrate this new solution technique for spatially periodic flows with fluid–particle interactions, as numerical examples we present several cases of sedimentation in continuous (i.e., infinite) fluid–particle mixtures. In each case, the computational domain consists of a $L \times L \times L$ tri-periodic box with N spheres. This is depicted in Fig. 5 where $N = 4$. As was explained in Section 3, the computational domain used does not strictly conform to the perceived periodic planar boundaries that are depicted in Fig. 5. Instead an unstructured mesh is used where elements do cross these planar boundaries. However, every point located within the $L \times L \times L$ periodic box (outside the particles) can be mapped to a single element within the computational domain. In the figures presented later in this section, it can be seen that the shape of the domain does not strictly conform to a box.

We consider two different sequences of simulations. In Sequence A, the only parameter we vary is the number of spheres within the tri-periodic cell. In Sequence B, we keep the number of particles constant and vary the particle density. We define the particle density as the percentage of the total volume that is occupied by particles. Aside from the number of particles and particle density, all other parameters for each simulation are held constant. These base simulation parameters are set such that a single particle at the particle density of Sequence A (26.8%) falls at a terminal Reynolds number of approximately 10. We define the terminal Reynolds number as

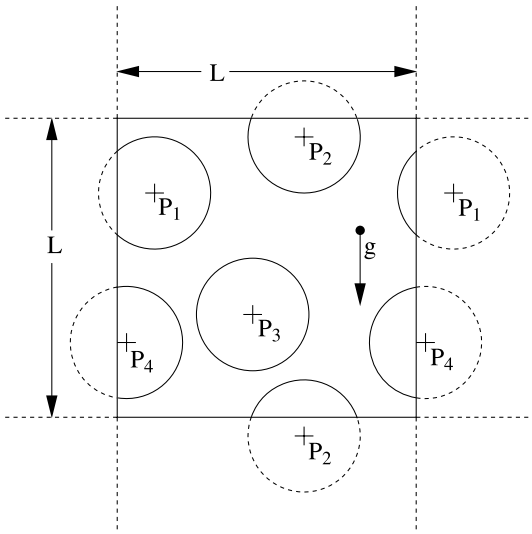

Fig. 5. Problem setup for sedimentation in a continuous fluid–particle mixture.

$$Re_{\mathrm{T}} = \frac{\rho \bar{U} \bar{D}}{\mu}, \tag{31}$$

where $\bar{U}$ is the average speed of the particles and $\bar{D}$ is the average diameter. The simulation parameters are as follows. radius of each sphere = 0.4, fluid density = 1.0, fluid viscosity = 0.08, particle mass density = 16.2, gravitational force = 1.25, and time step size = 0.04. In all simulations, the spheres are initially at rest in a regular 3D grid arrangement.

We set all three components of the volumetric flow rate ($\dot{\mathbf{V}}$) to 0.0, and this is appropriate for modeling sedimentation. For sedimentation problems, we may assume that at some far distance, there is a surface where the particles are accumulating as they fall, such as the bottom of a closed container which holds the fluid–particle mixture (see Fig. 6). As the particles fall, fluid must rise elsewhere in order to maintain global conservation of volume. To apply this global condition locally within the periodic domain, we set the total volumetric flow rate to zero, which means that as particles fall due to gravity in the tri-periodic domain, fluid will be rising elsewhere. By applying this zero total volumetric flow rate also in the two directions perpendicular to gravity, we are simply constraining the flow such that we do not get any sort of net motion or drift in these two directions.

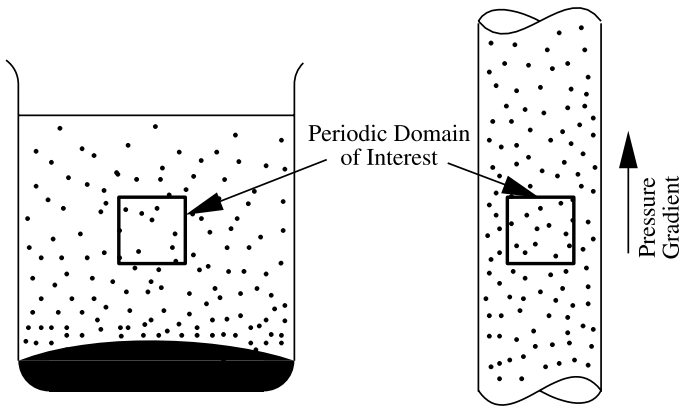

Fig. 6. Typical fluid–particle sedimentation problems.

6.1. Sequence A

In this sequence of simulations, we hold the particle density constant at 26.8% and vary the number of particles being simulated in each case. As more particles are added to the representative periodic domain, the simulations become more accurate at representing the behavior of a fluid–particle mixture. We had five simulations, with 1, 8 (initially arranged in a regular grid of $2 \times 2 \times 2$), 27 ($3 \times 3 \times 3$), 64 ($4 \times 4 \times 4$), and 125 particles, respectively. We compare these five simulations to observe changes in flow behavior due to the increase in simulation "resolution" and size. The periodic box size (L) for these simulations is 1.0, 2.0, 3.0, 4.0, and 5.0, respectively. The mesh sizes for these five simulations are approximately 0.012, 0.120, 0.400, 0.930, and 1.8 million tetrahedral elements, respectively. These are only approximate numbers since due to remeshing (which takes place on average every 6–8 time steps) mesh sizes change during the simulation.

Both qualitatively and quantitatively, each of the five simulations yield very similar results. The quantitative results (average settling speed, volumetric fluid flow rate, particle drag, and pressure jump values) are strikingly similar considering the differences in simulation resolution and size. This is most likely due to the fact that at such low Reynolds numbers and high particle density, these gross flow measurements are mainly dependent upon the particle density alone.

In Figs. 7–9, we show at a vertical cross-section, the velocity vectors, the vertica l component of the velocity, and pressure (p^*). In each of these three figures, an image from each of the five simulations is shown. To make it easier to compare the images with each other, we include periodic copies (i.e., show more than a single periodic domain) for the simulations with lower number of particles such that each image shows a domain at least as large as the one in Simulation 5 (125 particles). For example, for Simulation 2 (8 particles), the domain size L is only 2, so we actually show nine computational domains patched together in a 3×3 grid so that we show a "total" domain size of 6, which is larger than what we have in Simulation 5. In Fig. 10, we show some additional qualitative results only for Simulation 5.

In Fig. 11, we plot for each simulation the time history of the average settling speed (i.e., vertical component of the velocity). It can be seen in this figure that the settling speed for each simulation is similar. These curves are not very smooth due to the numerous collisions that take place and in some cases due to the solution projection errors that occur when remeshing takes place. It can also be seen in these curves that the initial grid arrangement of the particles seems to break up at around time step 200 for all simulation.

We also calculate the time-average of several basic flow parameters to represent the gross behavior of this fluid–particle mixture. The initial acceleration of the particles from their resting state is not included in these time-averages. We present the following parameters in Table 1: $\bar{U}_2$ is the average of the particle settling speeds (gravity is aligned with the x_2 axis); S.D. the standard deviation of U_2; $(\dot{V}_2)_{\mathrm{F}}$ the fluid volumetric flow rate in the vertical direction; $(\dot{V}_2)_{\mathrm{F}}/V_{\mathrm{F}}$ is in effect the average vertical speed of the fluid; $\bar{F}_2$ the average vertical force on the particles; J_2 the pressure jump; and J_2/L is a measure of the pressure gradient across the domain. It can be seen in this table that the five simulations are compared very well. Of course, due to the flow rate constraint, the normalized fluid volumetric flow rate $(\dot{V}_2)_{\mathrm{F}}/V_{\mathrm{F}}$ is proportional to the particle velocity, and the force on the particles should be (and is) equal to the particle weight. The standard deviation gives some idea about how much mixing takes place as the particle falls.

Table 1
Basic flow parameters for Sequence A. All values are the result of time-averaging. Values pertaining to particle quantities are also averaged over the particles

Simulation	$\bar{U}_2$	S.D. U_2	$(\dot{V}_2)_{\mathrm{F}}$	$(\dot{V}_2)_{\mathrm{F}}/V_{\mathrm{F}}$	$\bar{F}_2$	J_2	J_2/L
1	−0.996	0.000	0.267	0.365	5.430	−6.004	−6.004
2	−0.915	0.168	1.957	0.334	5.299	−11.486	−5.743
3	−0.912	0.216	6.576	0.333	5.280	−17.103	−5.701
4	−0.990	0.243	16.911	0.361	5.284	−22.806	−5.702
5	−1.039	0.292	34.594	0.378	5.322	−28.672	−5.734

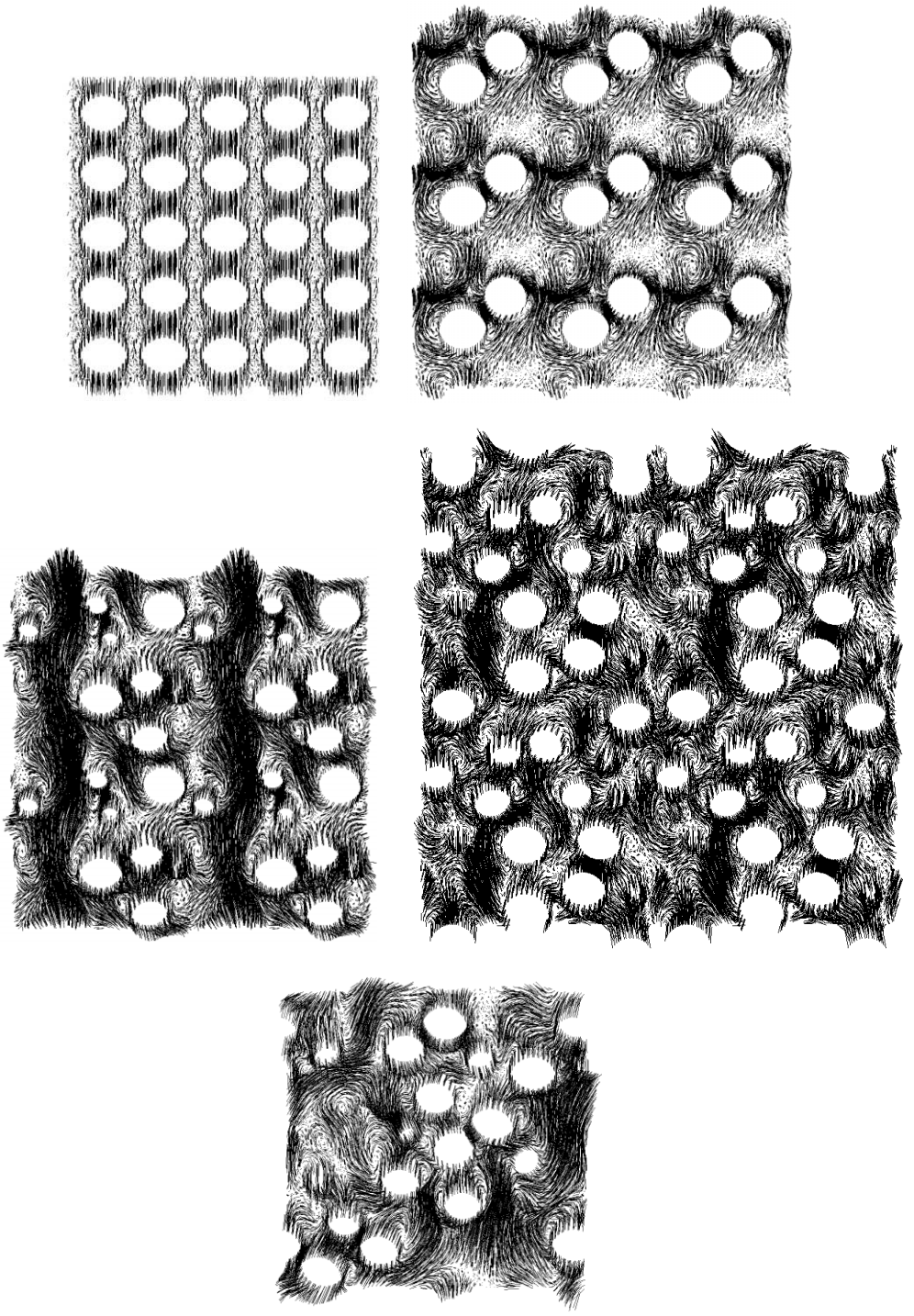

Fig. 7. Velocity vectors at a cross-section for the five simulations of Sequence A. Top left: Simulation 1 with 25 domains (5×5) shown, top right: Simulation 2 with nine domains (3×3), middle left: Simulation 3 with four domains (2×2), middle right: Simulation 4 with four domains (2×2), and bottom: Simulation 5 with only a single domain.

6.2. Sequence B

In this sequence of four simulations, we hold the number of particles in a periodic domain constant and vary the particle density. The number of particles in each case is 27, initially arranged in a $3 \times 3 \times 3$ grid. In

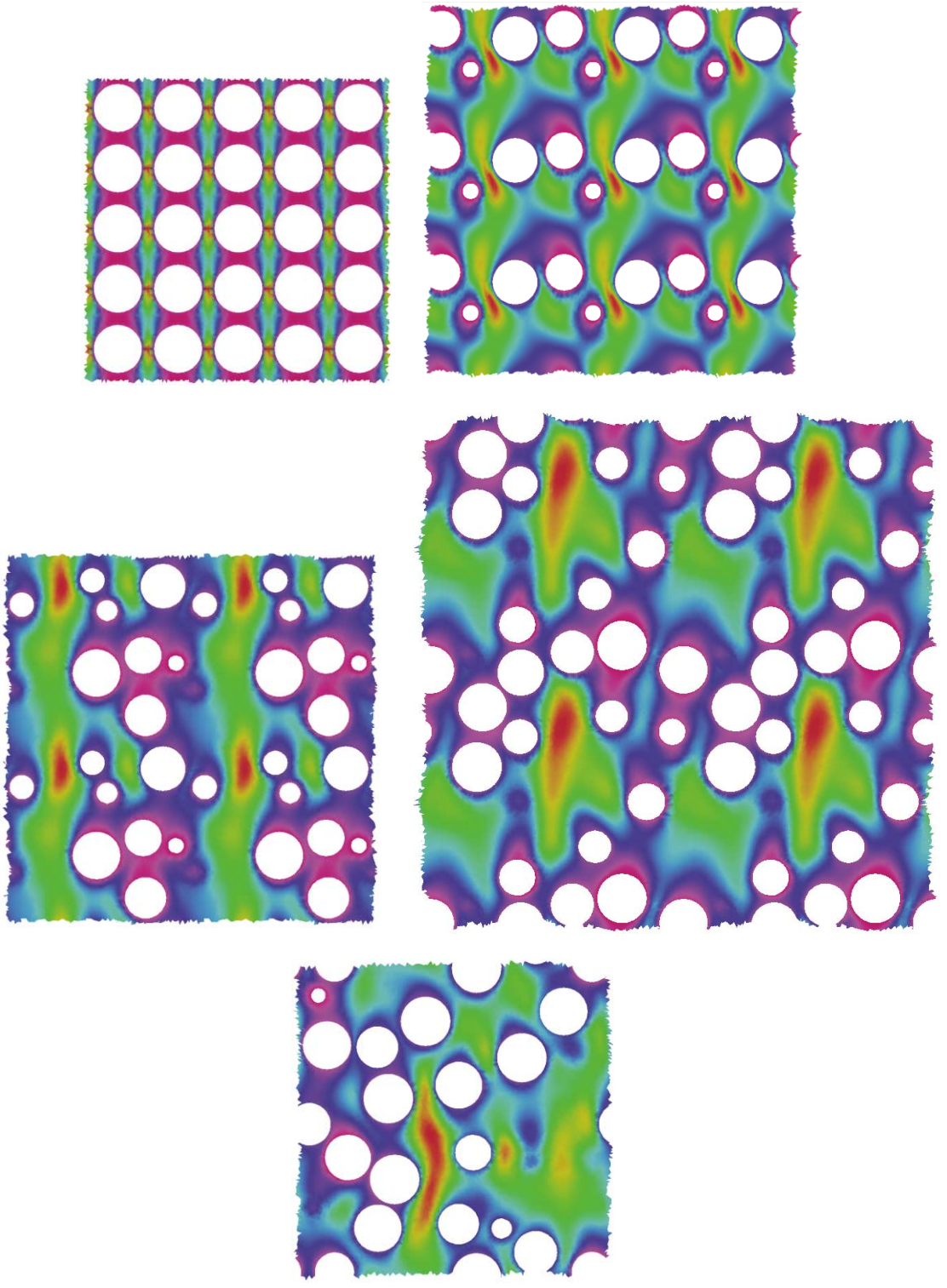

Fig. 8. Vertical component of the velocity at a cross-section for the five simulations of Sequence A. Red color represent zones of rising fluid, and the blue color represent zones of descending flow. The arrangement of the images is the same as the one used in Fig. 7.

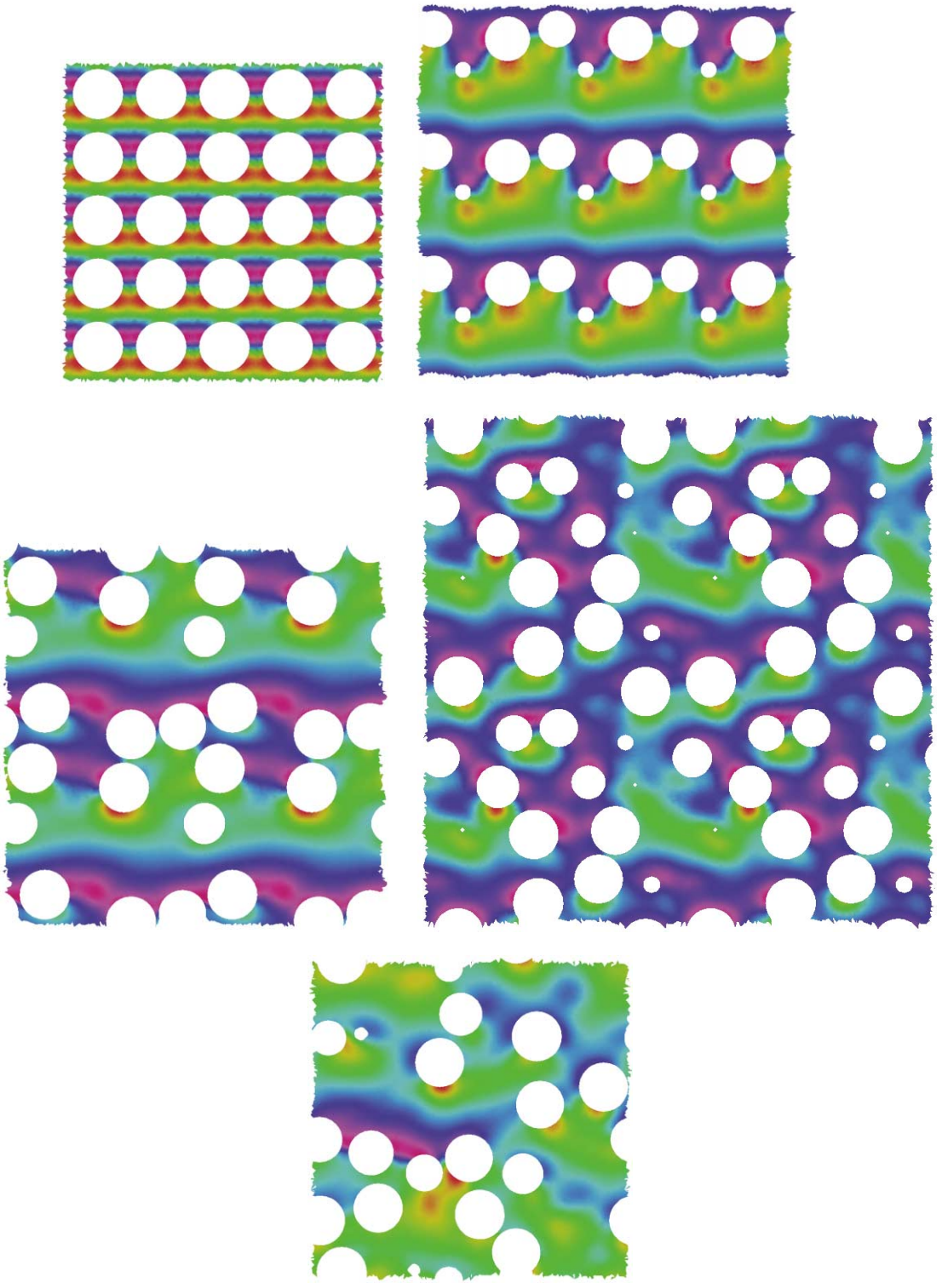

Fig. 9. Pressure at a cross-section for the five simulations of Sequence A. The arrangement of the images is the same as the one used in Fig. 7.

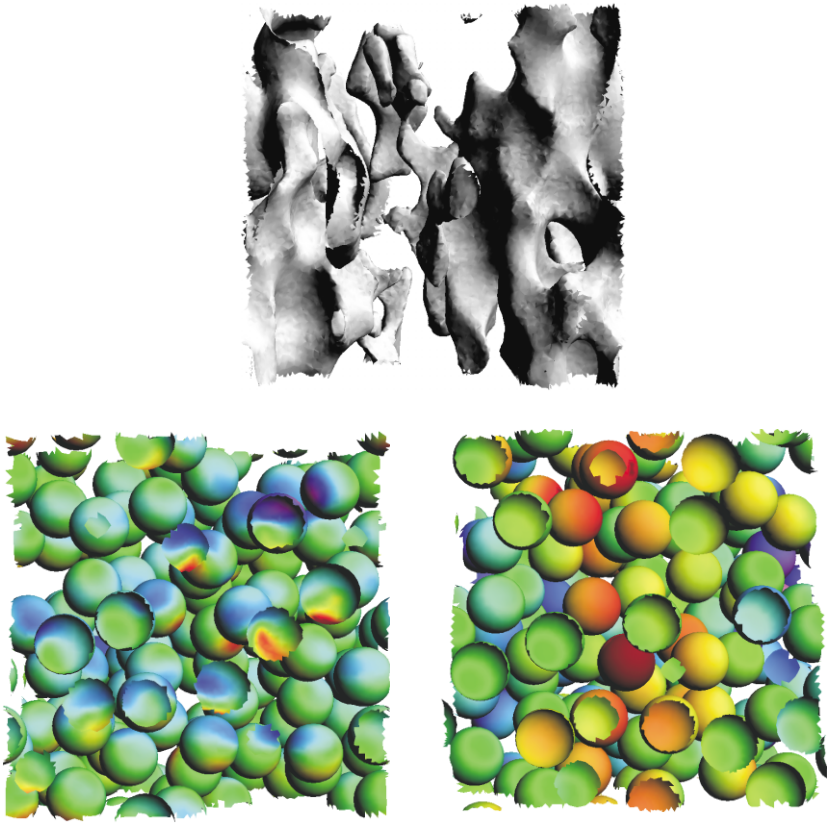

Fig. 10. Results from Simulation 5 (125 particles) of Sequence A. Top: iso-surface of the vertical component of the velocity corresponding to the value of 1.0 (this image depicts the zones of rising fluid), bottom left: pressure on the surface of the particles, and bottom right: particle speeds as indicated by their colors.

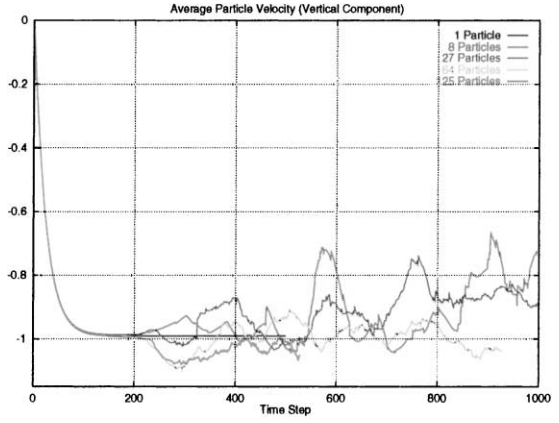

Fig. 11. Time-history of the average settling speed for all the five simulations of Sequence A.

Simulation 1, the particle density is 26.8% (this was the value used in Sequence A). In Simulation 2, 3 and 4, the particle density is 20.1% (3/4 of Simulation 1), 13.4% (1/2 of Simulation 1), and 6.7% (1/4 of Simulation 1). In these simulations, L is 3.00, 3.30, 3.78 and 4.76, respectively. The mesh sizes, respectively, are 0.4, 0.5, 0.7 and 1.0 million tetrahedral elements. We perform these simulations to observe changes in the flow patterns and particle settling speeds due to the differences in particle densities.

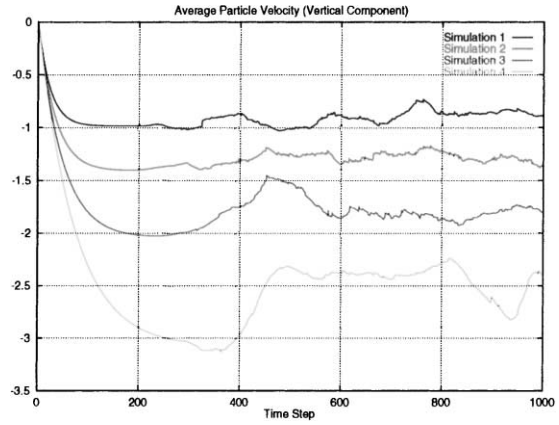

Fig. 12. Time-history of the average settling speed for all the four simulations of Sequence B.

Table 2
Basic flow parameters for Sequence B. All values are the result of time-averaging. Values pertaining to particle quantities are also averaged over the particles

Simulation	$\bar{U}_2$	S.D. U_2	$(\dot{V}_2)_F$	$(\dot{V}_2)_F/V_F$	$\bar{F}_2$	J_2	J_2/L
1	−0.912	0.216	6.576	0.333	5.280	−17.103	−5.701
2	−1.289	0.346	9.301	0.323	5.241	−14.119	−4.276
3	−1.833	0.438	13.233	0.283	5.273	−10.902	−2.884
4	−2.468	0.531	17.822	0.177	5.327	−7.016	−1.473

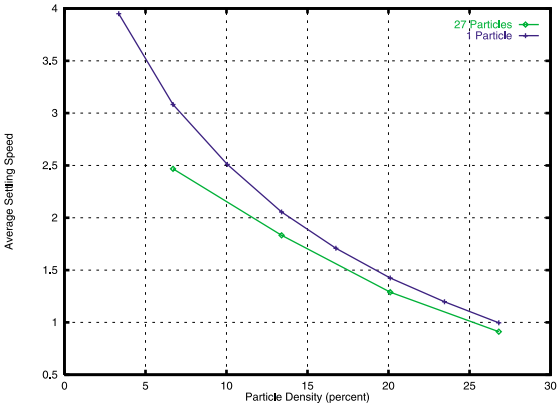

Fig. 13. Average settling speed as a function of particle density, for a single particle and for 27 particles.

It is expected that as the particle density decreases, the flow will be less constrained and the particles will fall at a faster rate. This is exactly what happens in this case. In Fig. 12, we see the time-history of the average settling speed for all four simulations of Sequence B. Time-averages of the basic flow parameters representing the gross behavior of the fluid–particle mixture are shown in Table 2. It can be seen in this table that the flow trends are as expected, whereas the particle volume fraction decreases, the settling speed increases and pressure gradient decreases. Also, as the volume fraction decreases and the speeds increase, the amount of particle mixing increases as indicated by the standard deviation of the particle velocities.

In Fig. 13, we plot the particle settling speed as a function of particle density. In this figure, we also plot the settling speed for a single particle at different particle densities. We observe that 27 particles fall at a

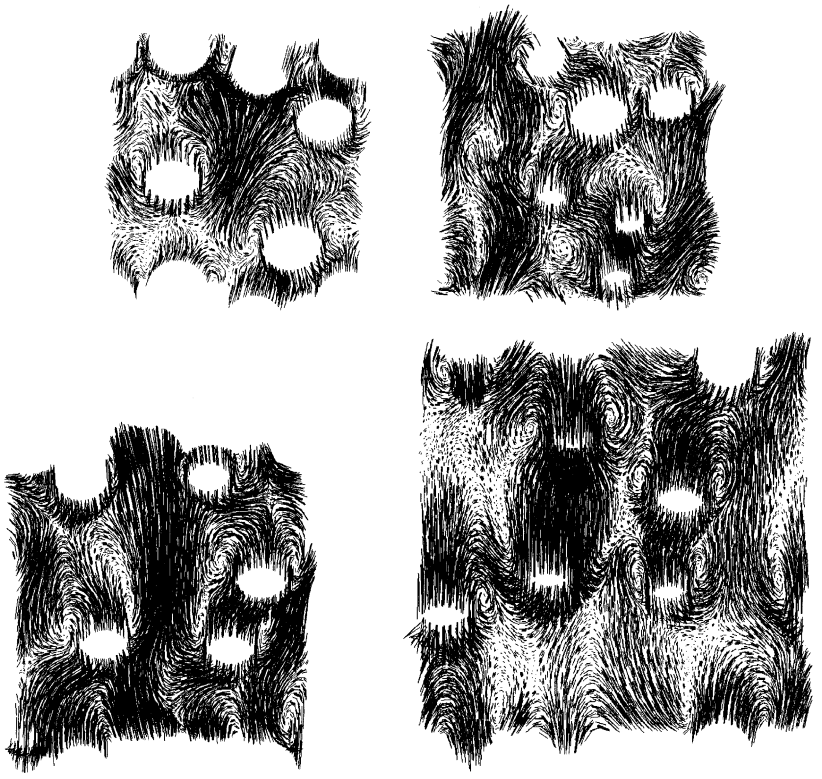

Fig. 14. Velocity vectors at a vertical cross-section for the four simulations of Sequence B. Top left: Simulation 1 (26.8% particle density), top right: Simulation 2 (20.1%), bottom left: Simulation 3 (13.4%), and bottom right: Simulation 4 (6.7%). Only one domain is shown in each image.

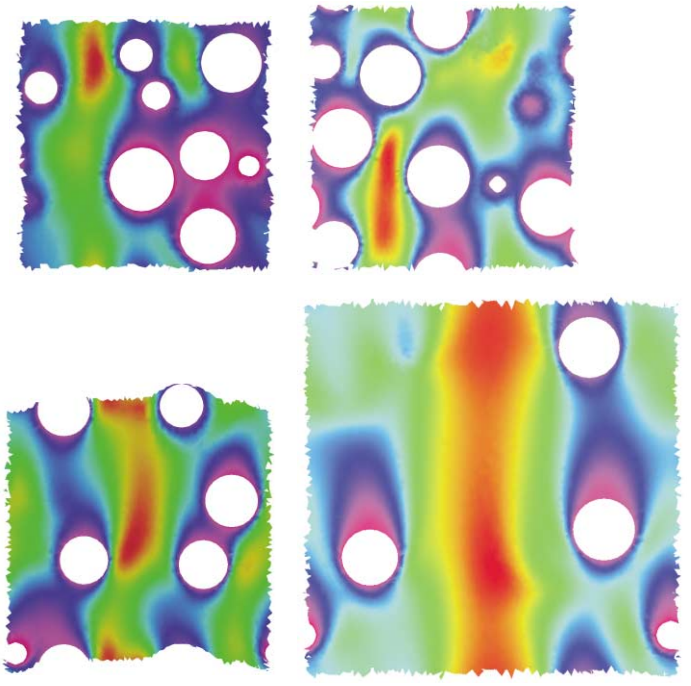

Fig. 15. Vertical component of the velocity at a vertical cross-section for the four simulations of Sequence B. Red color represent rising fluid zones and blue represents the descending flow. The arrangement of the images is the same as the one used in Fig. 14.

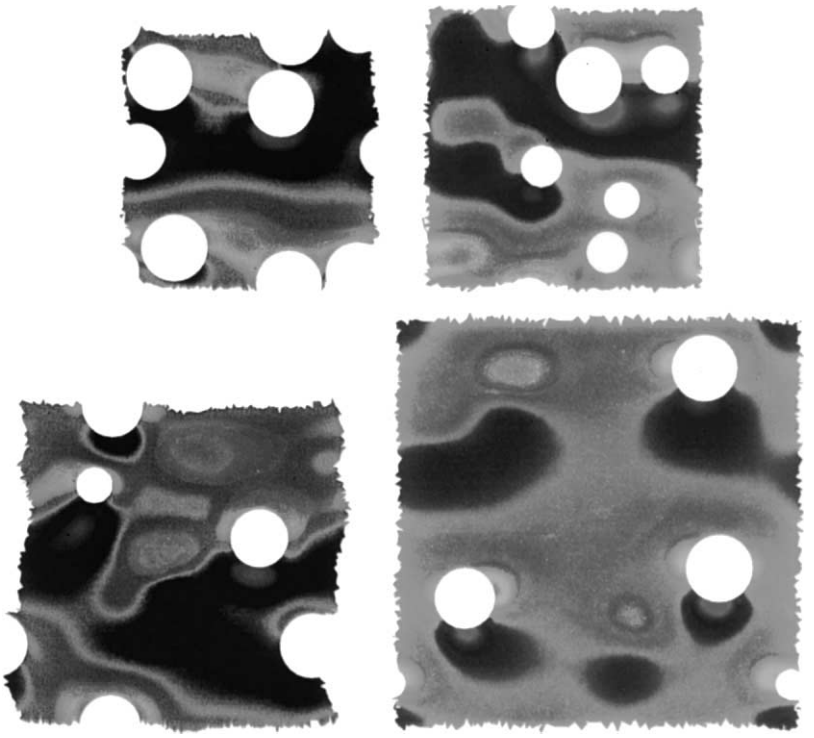

Fig. 16. Pressure at a vertical cross-section for the four simulations of Sequence B. The arrangement of the images is the same as the one used in Fig. 14.

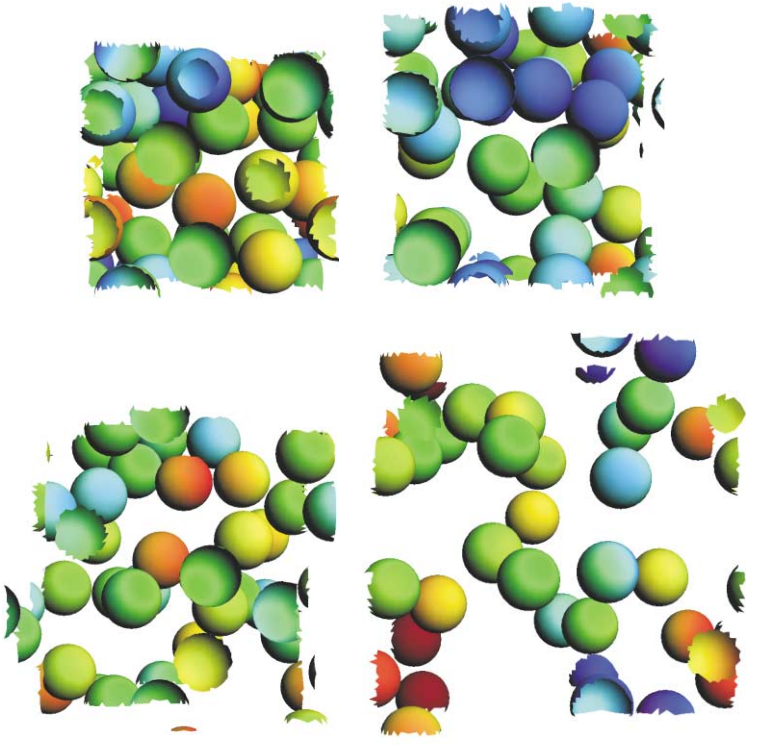

Fig. 17. Particle arrangement at a later time step for the four simulations of Sequence B. The colors represent the particle speed.

slower rate. This is because a single particle simulation is always locked into a regular grid arrangement with its periodic copies. We notice in Fig. 12 that the 27 particles also fall faster initially, until their initial grid arrangement breaks up, and after that they fall at a slower rate. If you take a measurement of the very initial settling speed with which these particles fall before symmetry is broken, the speeds match almost exactly with the results of the single particle simulations.

In Fig. 14, we show at a cross-section the velocity vectors, and in Figs. 15 and 16 we show, at vertical cross-sections, the vertical component of the velocity and the pressure. In Fig. 17, we show the particles at a later time step with the colors representing their speed. In all these figures, only one periodic domain is shown.

7. Concluding remarks

We have presented a new DSD/SST finite element formulation for spatially periodic flows and have demonstrated it through several examples of fluid–particle sedimentation in tri-periodic domains. We also discussed some of the implementation issues such as automatic mesh generation strategies and parallel programming techniques. All flow simulations were carried out on a CRAY T3E-1200 using an entirely new flow solver software, and the details on these new capabilities are also briefly covered.

The simulations yielded many interesting flow patterns and confirmed expected flow behavior. These simulations show that a reasonable number of particles must be simulated within the periodic domain to obtain reasonable results, but some of the gross flow features are somewhat insensitive to this number. We also presented some observations on how the flow behavior is affected by the varying particle density.

Acknowledgements

This research was sponsored by NSF (grant CTS-9896278) and by the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAH04-95-2-0003 and contract number DAAH04-95-C-0008. The content does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

[1] T.E. Tezduyar, M. Behr, J. Liou, A new strategy for finite element computations involving moving boundaries and interfaces – the deforming-spatial-domain/space-time procedure: I. The concept and the preliminary tests, Computer Methods in Applied Mechanics and Engineering 94 (1992) 339–351.

[2] T.E. Tezduyar, M. Behr, S. Mittal, J. Liou, A new strategy for finite element computations involving moving boundaries and interfaces – the deforming-spatial-domain/space-time procedure: II. Computation of free-surface flows two-liquid flows and flows with drifting cylinders, Computer Methods in Applied Mechanics and Engineering 94 (1992) 353–371.

[3] A.A. Johnson, T.E. Tezduyar, J. Liou, Numerical simulation of flows past periodic arrays of cylinders, Computational Mechanics 11 (1993) 371–383.

[4] 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.

[5] A.A. Johnson, T.E. Tezduyar, 3D simulation of fluid–particle interactions with the number of particles reaching 100, Computer Methods in Applied Mechanics and Engineering 145 (1997) 301–321.

[6] A.A. Johnson, T.E. Tezduyar, Advanced mesh generation and update methods for 3D flow simulations, Computational Mechanics 23 (1999) 130–143.

[7] K. Stein, R. Benney, V. Kalro, T. Tezduyar, J. Leonard, M. Accorsi, Parachute fluid–structure interactions: 3-D computation, Computer Methods in Applied Mechanics and Engineering 190 (2000) 373–386.

[8] S.E. Ray, T. Tezduyar, Fluid–object interactions in interior ballistics, Computer Methods in Applied Mechanics and Engineering 190 (2000) 363–372.

[9] A.A. Johnson, T.E. Tezduyar, Parallel computation of incompressible flows with complex geometries, International Journal for Numerical Methods in Fluids 24 (1997) 1321–1340.

[10] J.G. Kennedy, M. Behr, V. Kalro, T.E. Tezduyar, Implementation of implicit finite element methods for incompressible flows on the CM-5, Computer Methods in Applied Mechanics and Engineering 119 (1994) 95–111.

[11] V. Kalro, T. Tezduyar, Parallel iterative computational methods for 3D finite element flow simulations, Computer Assisted Mechanics and Engineering Sciences 5 (2) (1998) 173–183.
[12] A.A. Johnson, T.E. Tezduyar, Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces, Computer Methods in Applied Mechanics and Engineering 119 (1994) 73–94.
[13] S. Mittal, T.E. Tezduyar, Massively parallel finite element computation of incompressible flows involving fluid–body interactions, Computer Methods in Applied Mechanics and Engineering 112 (1994) 253–282.
[14] G. Karypis, V. Kumar, Metis 4.0: Unstructured graph partitioning and sparse matrix ordering system, Technical report, Department of Computer Science, University of Minnesota, 1998, available on the WWW at URL http://www.cs.umn.edu/~metis.
[15] Y. Saad, M. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal of Scientific and Statistical Computing 7 (1986) 856–869.
[16] G. Karypis, April 1998, personal communications.