ELSEVIER

Comput. Methods Appl. Mech. Engrg. 134 (1996) 351–373

Computer methods in applied mechanics and engineering

Simulation of multiple spheres falling in a liquid-filled tube

A.A. Johnson, T.E. Tezduyar*

Aerospace Engineering and Mechanics, Army HPC Research Center, University of Minnesota, 1100 Washington Avenue South, Minneapolis, MN 55415, USA

Received 20 October 1995

Abstract

A new 3D finite element flow simulation capability for fluid–particle interactions is presented and applied to study time-dependent behavior of multiple spheres falling in a liquid-filled tube. This capability is based on the flow simulation strategies such as stabilized space–time formulation for moving boundaries and interfaces, automatic mesh generation with structured layers of elements around the spheres, automatic mesh moving with remesh only as needed, and the implementation of these strategies on massively parallel computing platforms.

Several cases of multiple spheres falling in a liquid-filled tube are studied, with the number of spheres ranging from two to five. In all cases, depending on the number of spheres and their initial arrangement, a stable state is eventually reached with all spheres arranged in a pattern corresponding to that stable state, and with all of them falling with the same terminal velocity.

1. Introduction

The stabilized space–time finite element formulation was shown earlier [1–3] to provide a powerful, general-purpose capability in handling problems involving moving boundaries and interfaces. The finite element function space is extended over both the spatial and temporal coordinates, and thus, any movement of a boundary or interface is accommodated by a deformation of the spatial mesh with respect to the temporal coordinate. This function space in continuous in space but discontinuous in time. Because of this discontinuity, calculations are carried out one space–time slab at a time which makes these computations feasible. This formulation can be used in a wide class of problems that involve moving boundaries and interfaces, such as fluid–structure interactions, fluid–particle interactions, free-surfaces and multiple phases.

Along with the space–time formulation for solving problems with moving boundaries and interfaces comes the additional emphasis on mesh moving algorithms. Methods are needed to move/deform/flow the mesh to accommodate the movements of the mesh boundaries. We have developed several techniques to accomplish this. These are based on using explicitly defined rules to move the mesh [4], moving the mesh automatically by solving an equation system with boundary conditions coming from the movement of the boundary [5], remeshing to accommodate the change in the shape of the domain [3, 5], and using combinations of all of these methods [5]. Also, these methods must take into account that our implementation is a parallel one, and each new mesh involves, in addition to the cost of automatic mesh generation in 3D, the cost of mapping the finite element mesh on to the parallel

* Corresponding author.

SSDI 0045-7825(95)00988-4

processors in the most efficient way. In 3D problems, these costs are significant so it is imperative to minimize remeshing as much as possible.

In this paper, we apply these solution strategies, implemented on a massively parallel computing platform (Thinking Machines CM-5), to 3D simulation of multiple spheres falling in a liquid-filled tube. These 3D simulations involve between two to five spheres. As the spheres fall, they interact with the fluid and each other as they react to the fluid-dynamical forces acting on them. The simulations continue until terminal velocities and stable configurations of the spheres are reached.

The problem of multiple spheres sedimenting in a liquid-filled tube has been previously studied. Jayaweera and Mason [6] performed experiments on clusters of spheres falling in a viscous fluid at fairly low Reynolds numbers ($10^{-4} \leq \mathrm{Re} \leq 10$). A more recent study by Fortes et al. [7] included experiments of two spheres sedimenting in fluidized beds. Recent numerical studies by Hu et al. [8] involved 2D numerical simulation of multiple cylinders falling in a liquid-filled channel. Here, we present full 3D, time-dependent simulations with Reynolds number of the order 100.

The problem of multiple spheres falling in a tube imposes restrictions on the type of mesh generator and mesh update strategy that can be used. We wish to have meshes with an arbitrary number of spheres at arbitrary location within the tube. With these requirements, the only option we have in discretizing the domain is by using an automatic mesh generator. The development of an algebraic mesh generator that satisfies these conditions would be extremely difficult if not impossible.

Automatic mesh generators in 3D are quite costly because of the time it takes to generate a new mesh. Because of this, to handle the motion of the spheres, it would be quite inefficient to generate a new mesh at each time step in the simulation. Also, since our implementation is a parallel one, we have certain set-up costs associated with each new mesh. The main set-up cost is the partitioning of each mesh into sub-domains which are sent to individual processors, and this cost is significant.

Because we are using unstructured meshes generated with an automatic mesh generator, we use the automatic mesh moving scheme introduced earlier in [5] to handle the motion of the mesh in response to the motion of the spheres. In the automatic mesh moving scheme, we solve the modified equations of linear elasticity to determine the internal nodal displacements based on the given boundary conditions (the motion of the spheres). In our implementation, we actually translate the entire mesh vertically (in the direction of gravity which is aligned with the tube axis) with the center of mass of all the spheres involved. This global motion does not deform the mesh in any way, and the automatic mesh moving scheme is used to handle the movement of the spheres in relation to this global displacement. When the movement of the mesh creates too much distortion, we create a new mesh in the current sphere configuration, and project the solution from the old mesh on to the new one. By using the automatic mesh moving scheme, the frequency of remeshing is minimized.

We carried out four cases of spheres falling in a liquid-filled tube. For all cases, all the parameters were the same, except for the number of spheres, their initial configuration, and the duration of the simulation. We performed one simulation with two spheres, one with three spheres, and two simulations with five spheres.

In Section 2 we present the stabilized, space–time finite element formulation of the fluid dynamics part of the problem. In Section 3 we present the formulation for the dynamics of the spheres. In Section 4 we present details of the mesh generation and update strategies, and in Section 5 we present four numerical simulations of spheres falling in a liquid-filled tube.

2. Fluid dynamics formulation

Consider a viscous, incompressible fluid occupying, at time $t \in (0, T)$, a bounded region $\Omega_t \subset \mathbb{R}^{n_{sd}}$ with boundary Γ_t, where n_{sd} is the number of space dimensions. The primary degrees of freedom are velocity and pressure, denoted by $\boldsymbol{u}(\boldsymbol{x}, t)$ and $p(\boldsymbol{x}, t)$. The governing equations represent the momentum balance and the incompressibility constraint:

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

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

where ρ is the density of the fluid, $\boldsymbol{\sigma}$ is the stress tensor, and $\boldsymbol{f}(\boldsymbol{x}, t)$ is body force (such as gravity) per unit mass. The stress tensor $\boldsymbol{\sigma}$ is defined as

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

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

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

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

where $\boldsymbol{e}_i$ is the Cartesian unit vector corresponding to axis i, $\boldsymbol{n}$ is the unit normal vector for the boundary, and $(\Gamma_i)_g$ and $(\Gamma_i)_h$ are complementary subsets of the boundary Γ_i are related to the Dirichlet- and Neumann-type boundary conditions, respectively. The initial condition is a divergence-free velocity field specified over the entire domain:

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

where $\boldsymbol{u}_0$ satisfies Eq. (2).

To write the space–time formulation of (1)–(5), we partition the time interval $(0, T)$ into subintervals $I_n = (t_n, t_{n+1})$, where t_n and t_{n+1} belong to an ordered series of time steps $0 = t_0 < t_1 < \ldots < t_N = T$. Also, let $\Omega_n = \Omega_{t_n}$ and $\Gamma_n = \Gamma_{t_n}$. We define the space–time slab Q_n as the domain enclosed by the surfaces Ω_n, Ω_{n+1} and P_n, where P_n is the surface described by the boundary Γ_t as t traverses I_n. As in the case of Γ_t, 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:

$$(S_{\boldsymbol{u}}^h)_n = \{\boldsymbol{u}^h = [u_i^h]_{i=1}^{n_{\mathrm{sd}}} \mid u_i^h \in H^{1h}(Q_n),\ u_i^h = g_i^h \text{ on } (P_n)_{g_i},\ \forall i = 1 \ldots n_{\mathrm{sd}}\}\,, \tag{7}$$

$$(V_{\boldsymbol{u}}^h)_n = \{\boldsymbol{w}^h = [w_i^h]_{i=1}^{n_{\mathrm{sd}}} \mid w_i^h \in H^{1h}(Q_n),\ w_i^h = 0 \text{ on } (P_n)_{g_i},\ \forall i = 1 \ldots n_{\mathrm{sd}}\}\,, \tag{8}$$

$$(V_p^h)_n = (S_p^h)_n = \{p^h \mid p^h \in H^{1h}(Q_n)\}\,. \tag{9}$$

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

The stabilized space–time formulation can be written as follows: given $(\boldsymbol{u})_n^-$, find $\boldsymbol{u}^h \in (S_{\boldsymbol{u}}^h)_n$ and $p^h \in (S_p^h)_n$ such that $\forall\, \boldsymbol{w}^h \in (V_{\boldsymbol{u}}^h)_n$ and $\forall\, q^h \in (V_p^h)_n$

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

where n_{el} is the number of elements in the mesh. This process is applied sequentially to all space–time slabs $Q_1, Q_2, \ldots, Q_{N-1}$. In Eq. (10), the following notation is being used:

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

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

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

The computations start with

$$(\boldsymbol{u}^h)_0^- = \boldsymbol{u}_0\,. \tag{14}$$

REMARKS

(1) The first three terms in Eq. (10) constitute the Galerkin form of the momentum balance equation, while the fourth term is the Galerkin form of the incompressibility constraint. The fifth term weakly enforces the continuity of the velocity field across space–time slabs.

(2) The sixth term in Eq. (10) is the least-squares form of the momentum balance equation. This term provides the necessary stability for advection-dominated flows. This term also provides stability when equal-order interpolation function spaces are used for velocity and pressure. The definition of the stability parameter τ can be found in [1, 4].

(3) The seventh term in Eq. (10) is the least-squares form of the incompressibility constraint. This term enhances the stability of the formulation at high Reynolds number flows. The definition of the stability parameter δ can be found in [1, 4].

(4) The addition of the stabilizing terms does not compromise the consistency of the formulation since these terms are weighted with the residual of the momentum and mass balance equations which vanish for exact solutions.

3. Sphere dynamics formulation

In the liquid-filled tube, the spheres are allowed to translate and rotate with the gravity and fluid forces acting on them. The equation that governs this motion for each sphere is Newton's laws of motion:

$$\boldsymbol{F} = \boldsymbol{M}\dot{\boldsymbol{V}}\,, \tag{15}$$

where $\boldsymbol{F}$ is the generalized force vector, $\boldsymbol{M}$ is the generalized mass matrix, and $\dot{\boldsymbol{V}}$ is the time derivative of the generalized velocity vector. These are defined as

$$\boldsymbol{F} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ M_1 \\ M_2 \\ M_3 \end{Bmatrix}, \quad \boldsymbol{M} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{11} & I_{12} & I_{13} \\ 0 & 0 & 0 & I_{21} & I_{22} & I_{23} \\ 0 & 0 & 0 & I_{31} & I_{32} & I_{33} \end{bmatrix}, \quad \boldsymbol{V} = \begin{Bmatrix} V_1 \\ V_2 \\ V_3 \\ \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{Bmatrix}. \tag{16}$$

Here, F_i and M_i are the components of the force and moment acting on a sphere, V_i and Ω_i are the components of the linear and angular velocities, m is the mass of the sphere, and I_{ij} are the components of the moments of inertia. Specifically, for a sphere $I_{ij} = I\delta_{ij}$, and with this, all six equations in (15) are decoupled from each other.

For further discussion, let us consider just one of the decoupled equations. The first equation in (15) is

$$F_1 = m\dot{V}_1\,. \tag{17}$$

Since we solve for a whole space–time slab at a time, we take the value of the force acting on the

sphere (which includes the fluid dynamical force) at the middle of the slab in the temporal coordinate. Let F_1^n and F_1^{n+1} denote the forces at the lower and upper temporal levels of the slab. Then,

$$F_1 = \frac{F_1^{n+1} + F_1^n}{2}. \tag{18}$$

We can also discretize the time derivative of velocity by equation

$$\dot{V}_1 = \frac{V_1^{n+1} - V_1^n}{\Delta t}, \tag{19}$$

where Δt is the time step. Eq. (17) then becomes

$$\frac{F_1^{n+1} + F_1^n}{2} = m \frac{V_1^{n+1} - V_1^n}{\Delta t}. \tag{20}$$

In Eq. (20), the forces are known (computed from the flow field at each level n and $n+1$), and V_1^n is known (the velocity of the sphere is taken to be continuous across space–time slabs). By rearranging, Eq. (20) then becomes

$$V_1^{n+1} = V_1^n + \frac{\Delta t}{2m}(F_1^{n+1} + F_1^n), \tag{21}$$

where V_1^{n+1} is our unknown at each non-linear iteration, and this governs the update of the sphere position and henceforth, the mesh coordinates.

The velocity of the sphere is related to its position by

$$\dot{X}_1 = V_1. \tag{22}$$

By expanding $\dot{X}_1$ as we have expanded $\dot{V}_1$ in (19), expanding V_1 as we have expanded F_1 in (18), and rearranging for our unknown, we obtain

$$X_1^{n+1} = X_1^n + \frac{\Delta t}{2}(V_1^{n+1} + V_1^n). \tag{23}$$

As was the case with velocity, the sphere's coordinate is taken to be continuous across space–time slabs, and X_1^n is known. Eq. (21), along with (23), will give us the required movement of the sphere due to the forces acting on it. The angular positions, Θ_i, can be calculated in a similar way by using the moments M_i and angular velocities Ω_i.

3.1. Collisions between spheres

In our implementation, we also allow collisions to take place between spheres if any two spheres get too close to each other. The notation that will be used in formulating a collision can be seen in Fig. 1. In Fig. 1, $\boldsymbol{n}$ is a unit normal vector pointing from the center of Sphere A to the center of Sphere B, and V_A and V_B are the velocities of Spheres A and B before the collision. The velocities of the spheres after

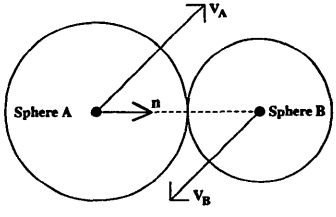

Fig. 1. Notation for formulating a collision.

the collision are unknowns and will be denoted by V_A' and V_B'. We will also denote $V_{nA} = V_A \cdot n$ and $V_{nB} = V_B \cdot n$.

The linear momentum of the system after the collision will be equal to the linear momentum of the system before the collision. The equation imposing this in the normal direction is

$$m_A V'_{nA} + m_B V'_{nB} = m_A V_{nA} + m_B V_{nB} , \tag{24}$$

where m_A and m_B are the masses of Sphere A and Sphere B. An additional equation is needed to determine the unknown velocities, and this equation relates the relative, normal components of velocity before and after the collision. This equation is

$$V'_{nB} - V'_{nA} = e(V_{nA} - V_{nB}) \quad 0 \leq e \leq 1 , \tag{25}$$

where e is the coefficient of restitution. Combining Eqs. (24) and (25), we obtain the expressions for the normal velocities after the collision:

$$V'_{nA} = \frac{m_A - em_B}{m_A + m_B} V_{nA} + \frac{(1+e)m_B}{m_A + m_B} V_{nB} , \tag{26}$$

$$V'_{nB} = \frac{(1+e)m_A}{m_A + m_B} V_{nA} + \frac{m_B - em_A}{m_A + m_B} V_{nB} . \tag{27}$$

In our formulation we need to determine the update to the velocity vector for Spheres A and B (V_A and V_B). Let us consider vector V_A. The update to this vector only occurs in the normal direction (the spheres are assumed to be smooth with no tangential forces acting on them), so the velocity vector after the collision can be written as

$$V_A' = V_A - V_{nA} n + V'_{nA} n , \tag{28}$$

or simply

$$V_A' = V_A + (V'_{nA} - V_{nA}) n . \tag{29}$$

By replacing V'_{nA} with Eq. (26), we obtain

$$V_A' = V_A + \left[\frac{m_A - em_B}{m_A + m_B} V_{nA} + \frac{(1+e)m_B}{m_A + m_B} V_{nB} - V_{nA} \right] n , \tag{30}$$

and rearranging

$$V_A' = V_A + \frac{(1+e)m_B}{m_A + m_B} (V_{nB} - V_{nA}) n . \tag{31}$$

Similarly, we obtain the equation for the velocity vector after the collision for Sphere B

$$V_B' = V_B + \frac{(1+e)m_A}{m_A + m_B} (V_{nA} - V_{nB}) n . \tag{32}$$

In our simulations, we take the value of e equal to 1.0 (perfectly elastic).

In our implementation, we first solve for the position of the spheres at the $n+1$ level in space–time slab by Eq. (23). We then determine if a collision will have taken place between any two spheres in the domain. If so, we solve for a new velocity of the spheres in question at the $n+1$ level by Eqs. (31) and (32). We then update the position of the spheres at the $n+1$ level by Eq. (23).

To predict a collision, we augment the radius of each sphere by a small amount and use this augmented radius (r_c) to determine if a collision is expected to take place. The reason for doing this is because we always keep around each sphere a zone with structured layers of elements. This zone, kind of glued to the sphere, moves with the sphere. Although, in general, it would be quite straightforward to allow this zone to deform as the spheres get too close, in the simulations reported in this paper we chose to keep these zones undeformed.

4. Mesh generation/Update strategy

4.1. Automatic mesh generation

The automatic mesh generator we use is based on Voronoï/Delaunay methods. These methods seem to be the most general and create high quality meshes in 3D. Our implementation also has the capability to create around each sphere zones with structured layers of elements. We create these structured layers of elements to keep full control of the mesh refinement around the spheres and consequently better model the boundary layer features of the flow.

Our 3D automatic mesh generation package is described in [9, 10]. We have developed a special version of this mesh generator to create the meshes used in these simulations. This special version can create a mesh of any number of spheres at any location within a tube of circular crossection, and has a very simple input file format. The creation of this special mesh generator was desirable because of the special geometries involved, and because of the fact that we use this mesh generator in a production environment where we need repetitive mesh generation.

4.2. Automatic mesh moving scheme

In the automatic mesh moving scheme, the mesh 'flow' is governed by the modified equations of linear elasticity. When mesh movement takes place, these equations are solved to determine the internal nodal displacements of the mesh based on the given boundary displacement. By solving this system of equations to determine the displacements, the method works for any mesh type and for any type of movement. The added generality comes at the cost of solving this additional equation system each time the mesh is deformed. The formulation, which was described in [5], will be briefly reviewed here for completeness.

Consider an elastic body occupying a region $\Omega \subset \mathbb{R}^{n_{sd}}$ with boundary Γ. The displacement of the mesh is given by $\boldsymbol{v}(\boldsymbol{x})$ and is governed by the equilibrium equations of elasticity:

$$\nabla \cdot \boldsymbol{\sigma}^* + \boldsymbol{f}^* = \boldsymbol{0}\,, \tag{33}$$

where $\boldsymbol{f}^*$ is a prescribed body force. The stress tensor $\boldsymbol{\sigma}^*$ is related to the strain tensor $\boldsymbol{\epsilon}^*$ by

$$\boldsymbol{\sigma}^* = \lambda(\operatorname{tr}\boldsymbol{\epsilon}^*)\boldsymbol{I} + 2\mu\boldsymbol{\epsilon}^*\,, \tag{34}$$

where λ and μ are the Lamé elastic constants. The strain tensor $\boldsymbol{\epsilon}^*$ is related to the displacement gradients by equation

$$\boldsymbol{\epsilon}^* = \frac{1}{2}(\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^{\mathrm{T}})\,. \tag{35}$$

The Dirichlet- and Neumann-type boundary conditions which enforce either a given displacement or a given normal stress component for the mesh motion, are represented as

$$\boldsymbol{v} \cdot \boldsymbol{e}_i = g_i^* \qquad \text{on } (\Gamma)_{g_i};\quad i = 1 \ldots n_{sd}\,, \tag{36}$$

$$\boldsymbol{n} \cdot \boldsymbol{\sigma}^* \cdot \boldsymbol{e}_i = h_i^* \quad \text{on } (\Gamma)_{h_i};\quad i = 1 \ldots n_{sd}\,, \tag{37}$$

where $(\Gamma)_{g_i}$ and $(\Gamma)_{h_i}$ are complementary subsets of the boundary Γ.

The finite element function spaces are given by

$$(S_v^h) = \{\boldsymbol{v}^h = [v_i^h]_{i=1}^{n_{sd}} \,|\, v_i^h \in [H^{1h}(\Omega)]^{n_{sd}},\ v_i^h \doteq (g_i^*)^h \text{ on } \Gamma_{g_i},\ \forall\, i = 1 \ldots n_{sd}\}\,, \tag{38}$$

$$(V_v^h) = \{\boldsymbol{w}^h = [w_i^h]_{i=1}^{n_{sd}} \,|\, w_i^h \in [H^{1h}(\Omega)]^{n_{sd}},\ w_i^h \doteq 0 \text{ on } \Gamma_{g_i},\ \forall\, i = 1 \ldots n_{sd}\}\,, \tag{39}$$

The automatic mesh moving formulation is then given as: find $\boldsymbol{v}^h \in S_v^h$ such that $\forall\, \boldsymbol{w}^h \in V_v^h$,

$$\int_\Omega \boldsymbol{\epsilon}(\boldsymbol{w}^h) : \boldsymbol{\sigma}^*(\boldsymbol{v}^h)\, \mathrm{d}\Omega - \int_\Omega \boldsymbol{w}^h \cdot \boldsymbol{f}^*\, \mathrm{d}\Omega = \int_{\Gamma_{h_i}} \boldsymbol{w}^h \cdot \boldsymbol{h}^*\, \mathrm{d}\Gamma\,. \tag{40}$$

It is desirable to retain the structure of the mesh in the more refined areas, and have most of the deformation weighted towards the larger element regions of the mesh. To accomplish this, a variable stiffness coefficient is desirable where the small elements are more rigid. We implement this by dropping from Eq. (40) the Jacobian of the transformation from the element domain to the physical one. By doing this, smaller elements which are more susceptible to distortion and located in areas where refinement is important retain their shape better.

We move the structured layers of elements that are created around each sphere together with the sphere. Therefore, these zones of structured layers of elements do not enter into the domain of the mesh deformation, and the outer boundaries of the layers are where boundary conditions for the deforming mesh are specified. If we were to deform these zones of structured layers of elements according to some explicitly defined rules, the situation would still have been the same, because these zones still would not have been part of the domain where the mesh motion is unknown and is governed by the equations of elasticity.

4.3. Remesh procedure

As the spheres move around, in general it will not be possible to continue updating the mesh just by moving the nodes with the automatic mesh moving scheme. The movement of the mesh will eventually result in unacceptable levels of mesh distortion. When this happens, we remesh, which means that we generate a new mesh with the automatic mesh generator, then project the solution from the old mesh on to the new mesh. Once the solution is projected to the new mesh, computations can proceed.

To decide when we remesh, we track measures of mesh distortion throughout the computation. In this simulation, we track changes in element aspect ratio and element volume. The elemental measure of volume and aspect ratio change are given by equations

$$f_V^e = |\log(V^e/V_0^e)|\,, \qquad f_{\mathcal{AR}}^e = |\log(\mathcal{AR}^e/\mathcal{AR}_0^e)|\,, \tag{41}$$

where V^e is the element volume at a given time, V_0^e is the element volume in the initial (underformed) mesh, and $\mathcal{AR}^e$ is the aspect ratio of an element, defined as

$$\mathcal{AR}^e = \frac{(\text{Maximum Edge Length})^3}{V^e}\,. \tag{42}$$

We then define a global factor measuring mesh deformation by

$$(f_V)_{\max} = \max_{1\le e\le n_{el}} (f_V^e)\,, \qquad (f_{\mathcal{AR}})_{\max} = \max_{1\le e\le n_{el}} (f_{\mathcal{AR}}^e)\,. \tag{43}$$

We consider remeshing when either of these parameters $(f_V)_{\max}$ or $(f_{\mathcal{AR}})_{\max}$ becomes greater than $\log(4)$.

In our remesh procedure we use two different computing platforms. The solution is computed on the Thinking Machines CM-5, and the automatic mesh generation is carried out on the Cray C90. In the simulations, at around every 50 time steps we check the levels of mesh distortion. If the mesh is distorted too much, we send the required data to the C90, generate a new mesh for the current computational domain, project the solution on to the new mesh, then send the required data back to the CM-5. The new mesh will be used for the next run, and the projected solution is used as the restart data. This whole procedure is managed within a shell script to automate the process.

4.4. Projection of the solution

There are several methods available to us to project the solution from the old mesh on to the new one. The method we use which seems to be the most natural is by using the 'jump' term within the space–time formulation (i.e. the fifth term in Eq. (10)):

$$\int_{\Omega_n} (w^h)_n^+ \cdot \rho((u^h)_n^+ - (u^h)_n^-)\,\mathrm{d}\Omega\,. \tag{44}$$

The velocity field $(u^h)_n^+$ is defined over the (possible new) mesh corresponding to the current space–time slab, and is one of the unknowns. The velocity field $(u^h)_n^-$, on the other hand, is defined over the (possible old) mesh corresponding to the earlier space–time slab, and is known. The interpolation from the old mesh to the new one is built-in to the integration of this term over the two different meshes. The integration can be carried out over the two different meshes since we are using numerical integration, and the coordinates of the integration points can be found within both meshes.

In the actual implementation, after generating a new mesh, we find the values of $(u^h)_n^-$ at the numerical integration points of the new mesh by a simple linear interpolation over the old mesh. We also compute a velocity field within the new mesh by solving the least-squares problem which is very similar to the jump term in the formulation. This interpolated velocity field is used as an initial guess for $(u^h)_n^+$ and $(u^h)_{n+1}^-$ in the iterative solution. We save the values of the velocity field at the integration points in a file which is sent to the CM-5. For the first time step after remeshing, we read in the values at the integration points and use them to perform the integration within the jump term of the formulation.

This method of interpolation seems to work quite well, but we add one other modification to this method. When a projected solution is used in the formulation, the incompressibility criteria may be slightly violated. This violation causes a spike to appear in the forces as the pressure reacts to this violation of incompressibility. Since we determine that some slight violation of incompressibility will be introduced when remeshing, and the pressure variable will react to this, we do not use this pressure when determining the forces acting on the spheres for this first time step after remeshing. We use the interpolated pressure coming from the least-squares interpolation to determine these forces.

5. Numerical simulations

We carried out four different cases of multiple spheres falling in a liquid-filled tube. The cases differ by the number of spheres, the initial configuration of the spheres, and the duration of the simulation. All other parameters are identical in all cases. The size of the tube and the spheres (R and r denote the radius of the tube and the sphere, respectively), the viscosity and density of the fluid (denoted by μ and ρ), the magnitude of the gravity (denoted by g), and the mass and moment of inertia of the spheres (denoted by m and I) remain the same in all cases. These parameters, in their non-dimensional form, are given in Table 1. The time step Δt and the collision radius r_c are also given in Table 1. The mass is set so that one sphere alone in the tube will fall at a Reynolds number of roughly 100. In all simulations, the tube axis is aligned with the gravity vector which is in the $-X_2$ direction in the Cartesian frame. In each simulation, the spheres and the fluid are at rest in the initial configuration. Finally, the nodal refinement around each sphere, and on the tube, are exactly the same for each mesh and each simulation. We create three structured layers of thin elements around each sphere, and each layer has a thickness of 0.013.

All simulations were performed on the 512 processing node CM-5 at the Army HPC Research Center. Both the flow solver and the automatic mesh moving scheme are implemented in parallel

Table 1
Parameters for all simulations

Sphere:	$m = 6.15$	$I = 2.46$	$r = 1.0$
Fluid:	$\rho = 1.0$	$\mu = 0.02$	$g = 1.0$
Other:	$R = 5.0$	$\Delta t = 0.1$	$r_c = 1.05$

Table 2
Case 1: initial configuration of the spheres

	X_1	X_2	X_3
Sphere 1:	−1.0	−2.0	0.0
Sphere 2:	1.0	2.0	0.0

[11, 12], an an iterative solution strategy with diagonal preconditioner and GMRES update technique is used to solve the large linear equation system encountered at each non-linear iteration. All mesh generation is performed on the Cray C90 at the Minnesota Supercomputer Center.

Case 1: Two spheres

In this case, we have two spheres in an initially staggered configuration. The initial positions of the spheres are given in Table 2. This simulation is performed to capture the phenomena of spheres drafting, kissing and tumbling described in [7]. It is expected that the trailing sphere (Sphere 2 in this

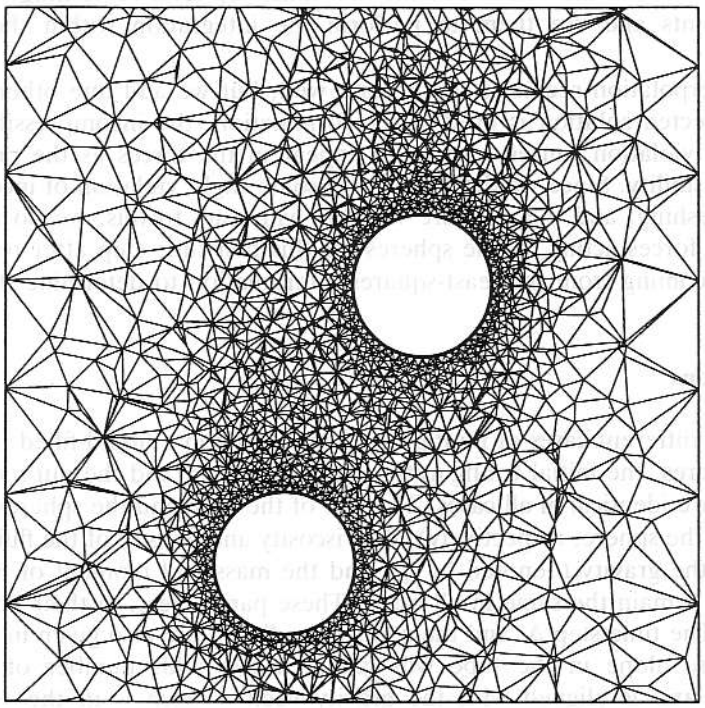

Fig. 2. Case 1: initial mesh (23 961 nodes and 140 541 elements).

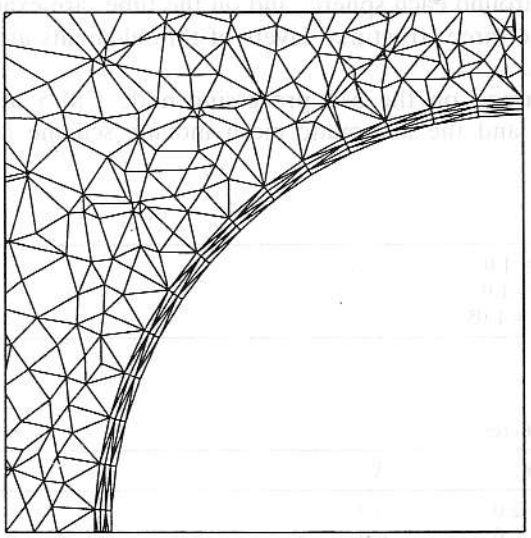

Fig. 3. Case 1: close up view of the initial mesh.

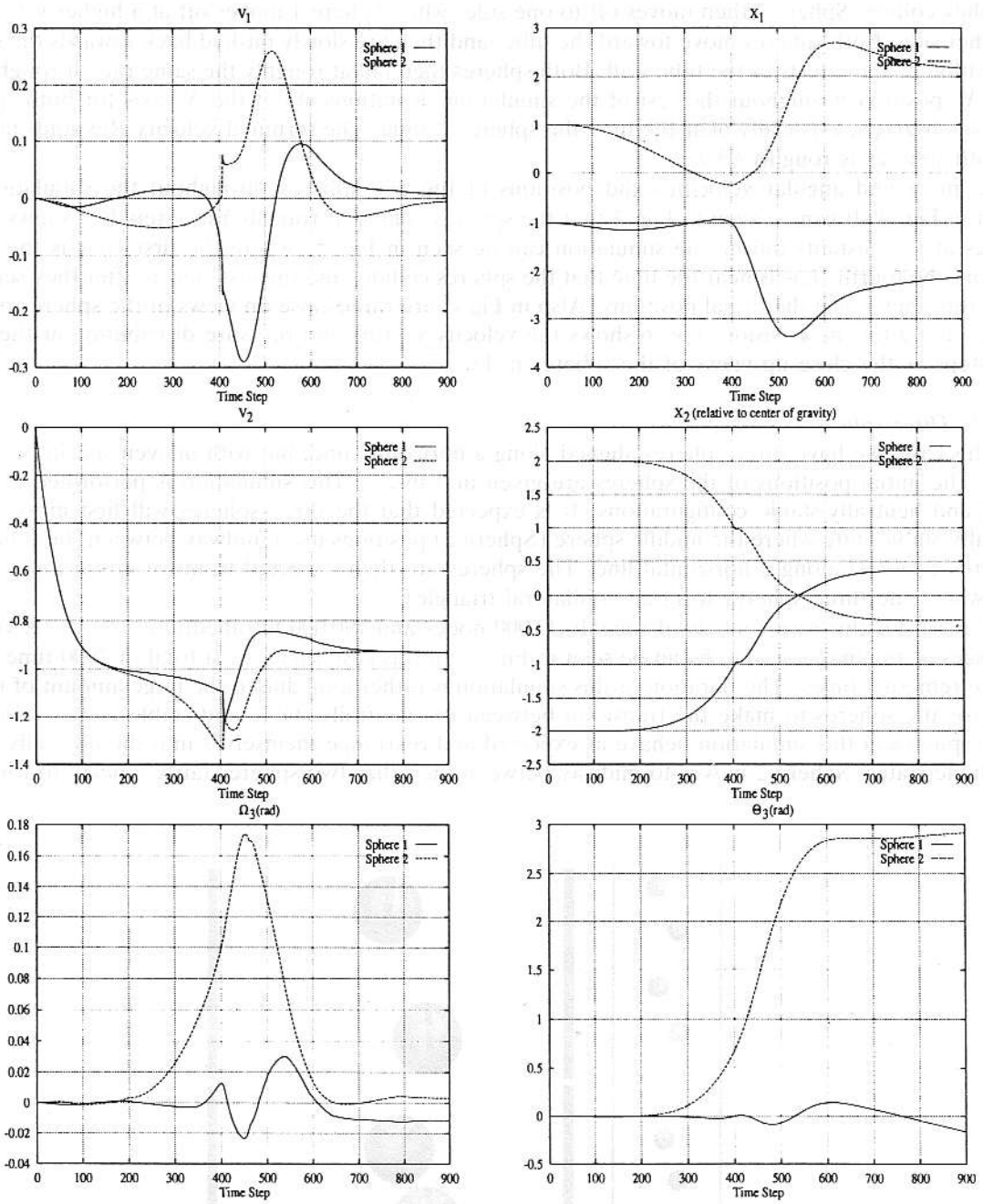

Fig. 4. Case 1: velocity and orientation histories of the spheres.

case) will be attracted to the low pressure region in the wake of the leading sphere (Sphere 1). The spheres will eventually collide and then separate. The two spheres will then continue to fall together side by side.

The mesh for this problem consist of roughly 24 000 nodes and 140 000 tetrahedral elements. A view of a slice of the initial 3D mesh can be seen in Fig. 2, and a close up view of the area around one of the spheres can be seen in Fig. 3. In this case there are a total of 900 time steps, and we remesh 12 times.

The spheres in this simulation do exhibit the behavior of drafting, kissing and tumbling. Sphere 2 is slowly pulled horizontally (in the X_1 direction) towards Sphere 1. As Sphere 2 gets closer to the horizontal position of Sphere 1, it accelerates downward (the negative X_2 direction) toward Sphere 1

until they collide. Sphere 2 then moves off to one side, while Sphere 1 moves off at a higher velocity to the other side. Both spheres move toward the tube, and then are slowly pushed back towards the center of the tube by the effects of the tube wall. Both spheres then fall at roughly the same rate at roughly the same X_2 position throughout the rest of the simulation. Rotations about the X_3 axis for both spheres are also observed, especially near the time the spheres collide. The terminal velocity Reynolds number for both spheres is roughly 93.1.

The linear and angular velocities and positions of the two spheres throughout the simulation are shown in Fig. 4. It can be seen in Fig. 4 that the spheres collide at roughly time step 400. Views of the spheres at five instants during the simulation can be seen in Fig. 5, where the first view is the initial position, the fourth view is near the time that the spheres collide, and the last view is after they separate but before they reach their final positions. Also in Fig. 5 are three close up views of the spheres before, during, and after the collision. Fig. 6 shows the velocity vectors and pressure distribution at the same time steps as the close up views of the spheres in Fig. 5.

Case 2: Three spheres

In this case, we have three spheres aligned along a horizontal line, but with uneven spacing between them. The initial positions of the spheres are given in Table 3. This simulation is performed to study stable and neutrally-stable configurations. It is expected that the three spheres will first move into a neutrally-stable state where the middle sphere (Sphere 2) positions itself midway between the other two (Spheres 1 and 3) along a horizontal line. The spheres are then expected to move into a more stable state where the three spheres form an equilateral triangle.

The mesh for this case consists of roughly 32 000 nodes and 190 000 tetrahedral elements. A view of two slices of the initial 3D mesh can be seen in Fig. 7. In this case there are a total of 2900 time steps, and we remesh 7 times. The duration of this simulation is rather long due to the large amount of time it takes for the spheres to make the transition between the neutrally-stable and stable configurations.

The spheres in this simulation behave as expected and rearrange themselves into the neutrally-stable and stable states. Sphere 2 moves to midway between the other two spheres fairly quickly to form the

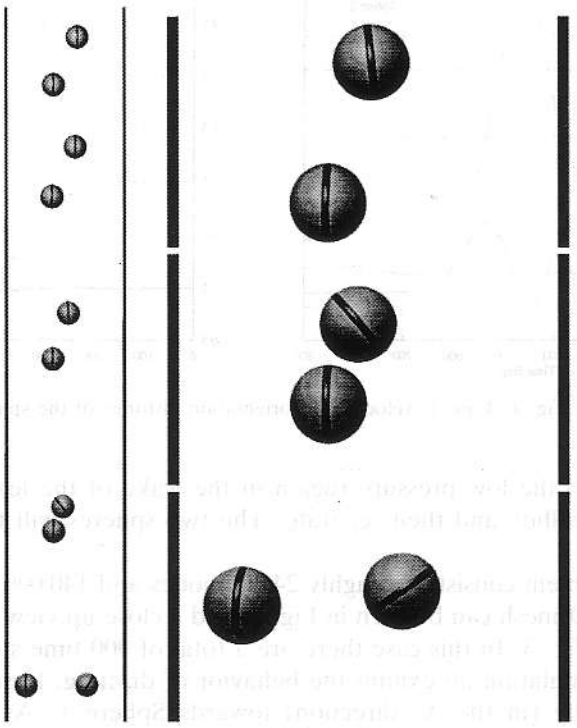

Fig. 5. Case 1: spheres at five instants during the simulation, along with close up views at three instants.

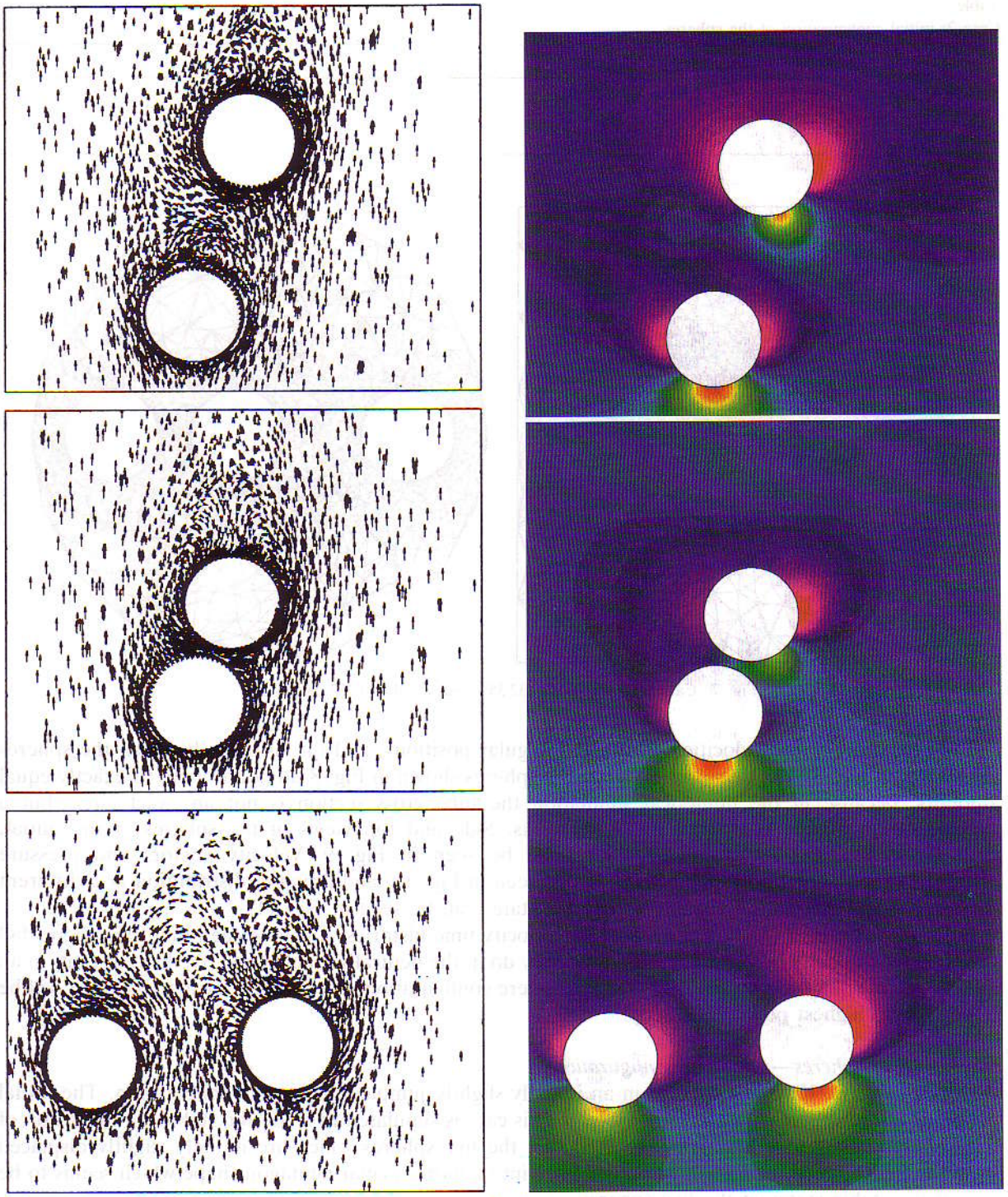

Fig. 6. Case 1: velocity vectors and pressure distribution before, during and after the collision.

neutrally-stable state. The Reynolds number during this time period is 83. After this time, Sphere 2 slowly moves in the X_3 direction (out of line) over a long period of time and eventually the three spheres form an equilateral triangle (this configuration is also observed in [6]). The Reynolds number during the stable configuration is 86.1. Rotations of the spheres are also noted during both configurations.

Table 3
Case 2: initial configuration of the spheres

	X_1	X_2	X_3
Sphere 1:	−3.25	0.0	0.0
Sphere 2:	0.5	0.0	0.0
Sphere 3:	3.25	0.0	0.0

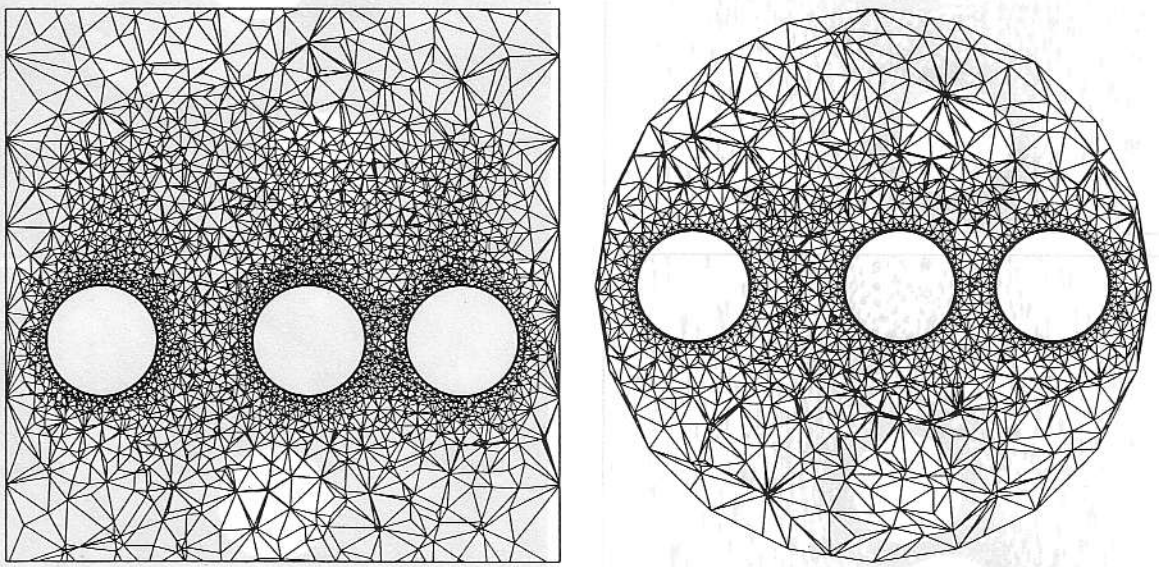

Fig. 7. Case 2: initial mesh (32 357 nodes and 189 253 elements).

Time histories of the velocities, linear and angular positions, and the distance between the spheres are shown in Fig. 8. The distances between the spheres shown in Fig. 8 do not all tend to exactly equal amounts, because in the numerical simulation the tube cross section is not an exact circle but a representation of one composed of 20 segments. Side and top views of the spheres at the initial, neutrally-stable, and stable configurations can be seen in Fig. 9. Velocity vectors and pressure distribution at the neutrally-stable state can be seen in Fig. 10, and pressure distribution at a different cross section at the neutrally-stable and stable states can be seen in Fig. 11.

From an analysis of the flow field data and velocity time histories, it can be seen that the spheres fall with a higher velocity in the stable state than they do in the neutrally-stable state. We believe that, in all likelihood, it will always be the case that any sphere configuration that is in the most stable state will be falling at the highest possible velocity.

Case 3: Five spheres—pentagon configuration

In this case, we have five spheres in an initially slightly jumbled pentagon configuration. The initial positions of the spheres are given in Table 4. This case is simulated to study the stable configuration of five spheres falling together. It is expected that the five spheres which are initially slightly displaced from the pentagon configuration will move back into a more regular pentagon shape which seems to be one of the stable states of the five spheres.

The mesh for this case consists of roughly 47 000 nodes and 276 000 tetrahedral elements. A view of a slice of the initial 3D mesh can be seen in Fig. 12. In this case there are a total of 500 time steps, and we remesh 4 times.

The spheres in this simulation did eventually form an exact pentagon configuration with all of the spheres aligned vertically with the same X_2 coordinate. Again, this configuration is observed in [6]. Rotations of the spheres are also observed. The Reynolds number at the final configuration is 72.1.

Time histories of velocities, positions, and the distance between the spheres are shown in Fig. 13.

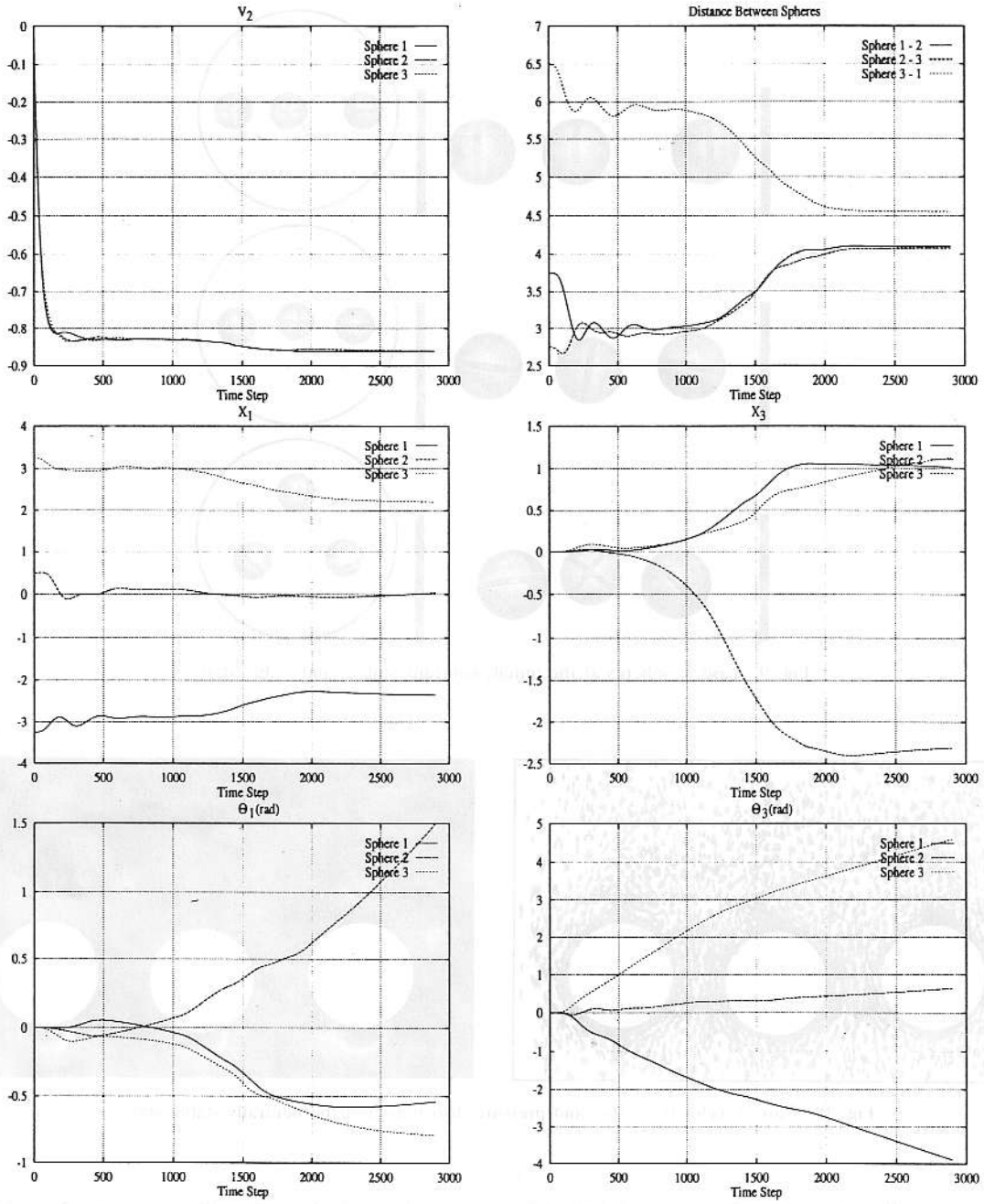

Fig. 8. Case 2: velocity and orientation histories of the spheres.

Views of the spheres at five instants during the simulation can be seen in Fig. 14, where the first one is the initial configuration and the last one is the final configuration. Also in Fig. 14 are top views of the spheres at the initial and final states. Pressure distribution at the final state can be seen in Fig. 15.

Case 4: Five spheres—pyramid configuration

In this case, we have five spheres, four of them aligned in an exact square configuration, with the fifth one in the center and above the other spheres (at the apex of the pyramid). The initial positions of the spheres are given in Table 5. This simulation is performed to study alternate stable configurations of

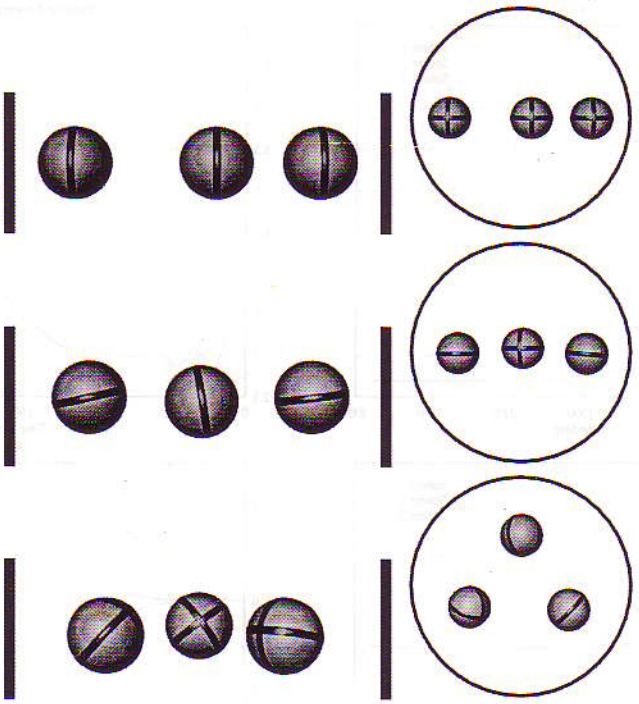

Fig. 9. Case 2: spheres at the initial, neutrally-stable, and stable states.

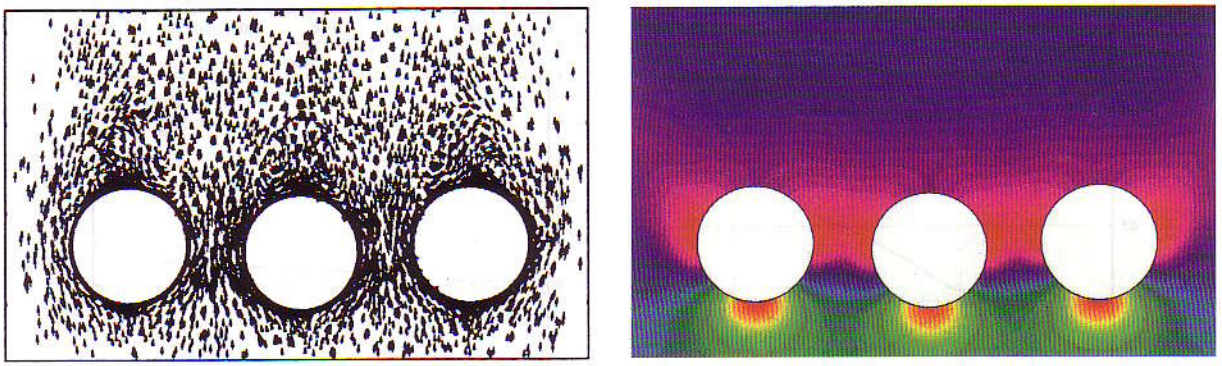

Fig. 10. Case 2: velocity vectors and pressure distribution at the neutrally-stable state.

five spheres falling together. It is expected that the rogue sphere (Sphere 5) will move into the center of the other four to form a ⊗ configuration.

The mesh for this case consists of roughly 55 000 modes and 321 000 tetrahedral elements. A view of two slices of the initial 3D mesh can be seen in Fig. 16. In this case there are a total of 1000 time steps, and we remesh 2 times.

The spheres in this simulation do eventually form the ⊗ shape with all spheres aligned vertically with the same X_2 coordinate. During the simulation, Sphere 5 drops through the center of the other four spheres to a slightly lower level, but then eventually settles back to the same level as the others. The Reynolds number at the final sphere configuration is 71.6.

Time histories of the velocities, linear and angular positions, and the distance between the spheres are shown in Fig. 17. Views of the spheres at five instants during the simulation can be seen in Fig. 18, where the first one is the initial configuration and the last one is the final configuration. Also in Fig. 18

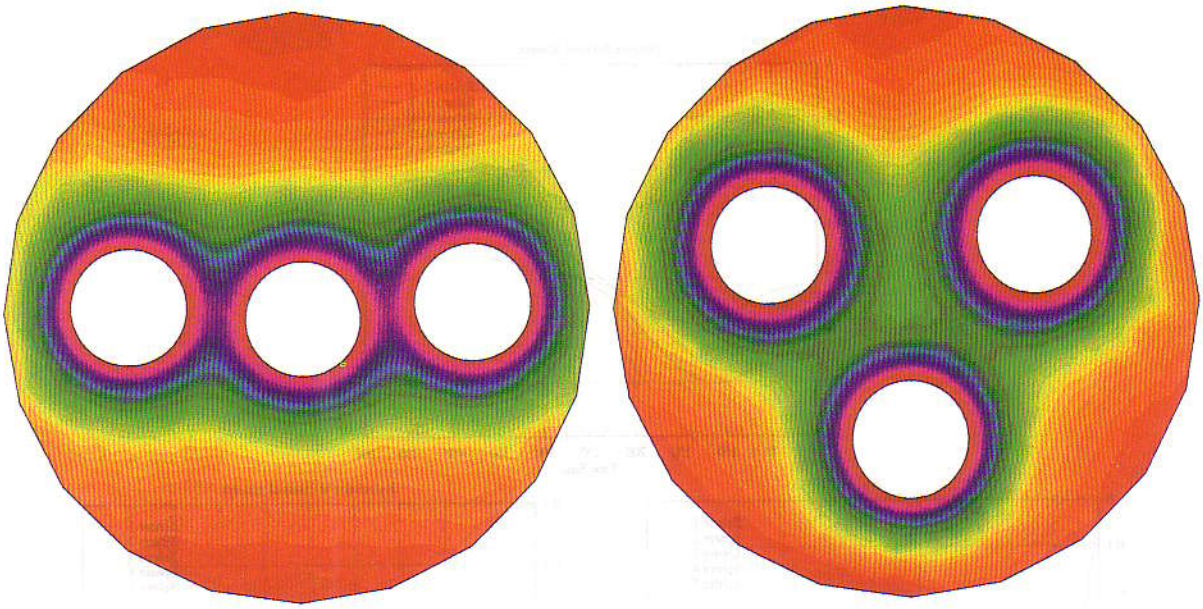

Fig. 11. Case 2: pressure distribution at the neutrally-stable and stable states.

Table 4
Case 3: initial configuration of the spheres

	X_1	X_2	X_3
Sphere 1:	0.316	−0.096	2.525
Sphere 2:	2.545	−0.264	0.649
Sphere 3:	1.653	0.296	−2.341
Sphere 4:	−1.957	0.027	−2.557
Sphere 5:	−2.276	0.038	1.257

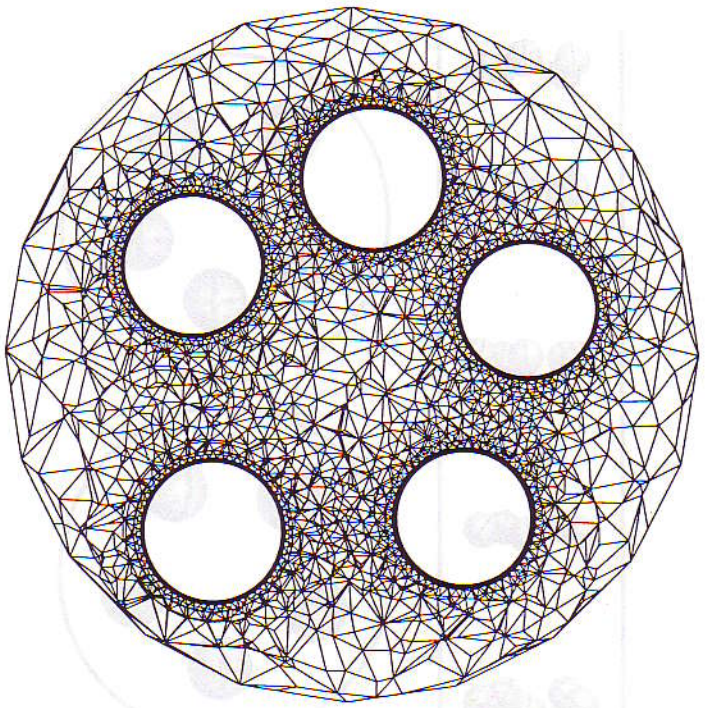

Fig. 12. Case 3: initial mesh (46 649 nodes and 270 718 elements).

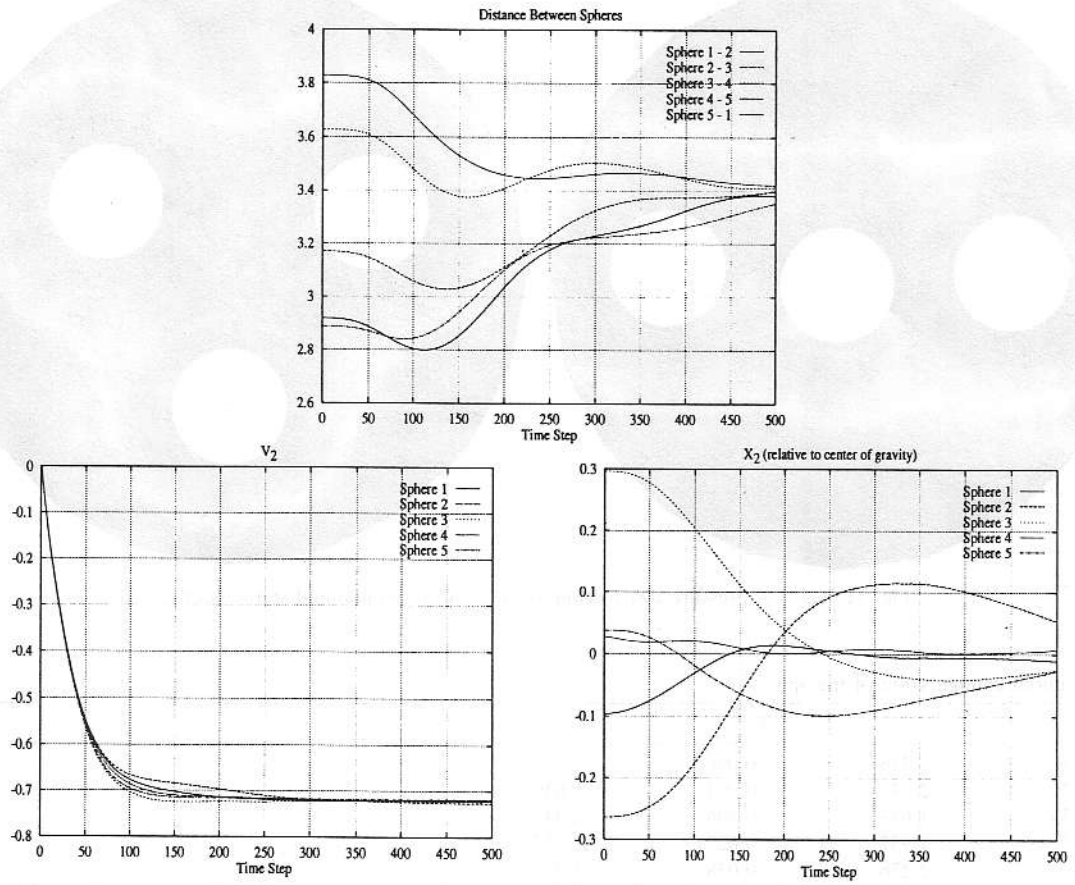

Fig. 13. Case 3: velocity and orientation histories of the spheres.

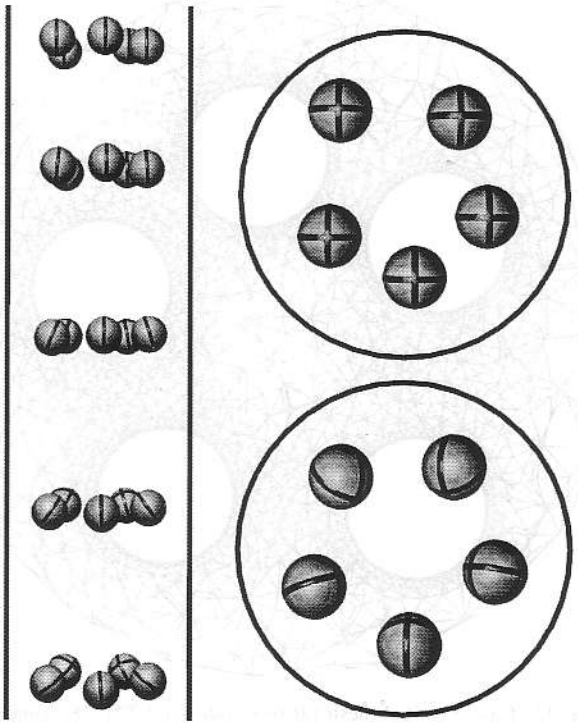

Fig. 14. Case 3: spheres at five instants during the simulation, along with top views of the initial and final configurations.

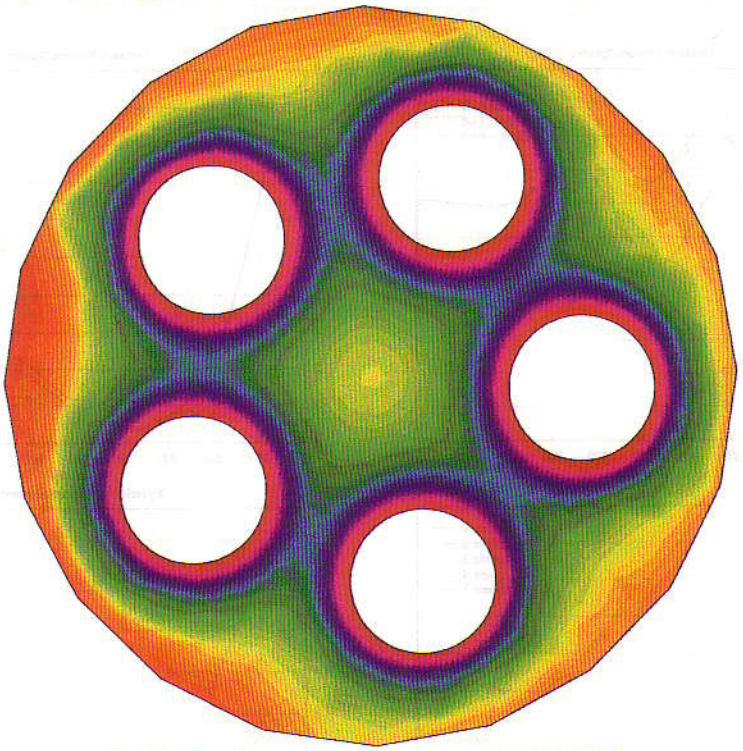

Fig. 15. Case 3: pressure distribution at the final state.

Table 5
Case 4: initial configuration of the spheres

	X_1	X_2	X_3
Sphere 1:	1.768	−0.8	1.768
Sphere 2:	1.768	−0.8	−1.768
Sphere 3:	−1.768	−0.8	−1.768
Sphere 4:	−1.768	−0.8	1.768
Sphere 5:	0.0	3.2	0.0

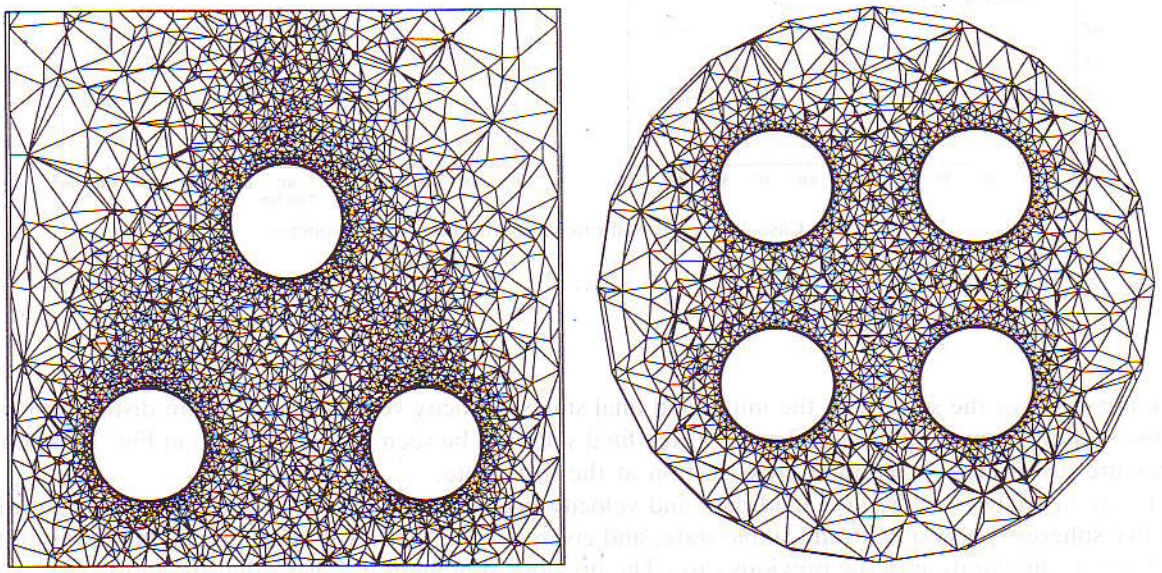

Fig. 16. Case 4: initial mesh (54 574 nodes and 321 236 elements).

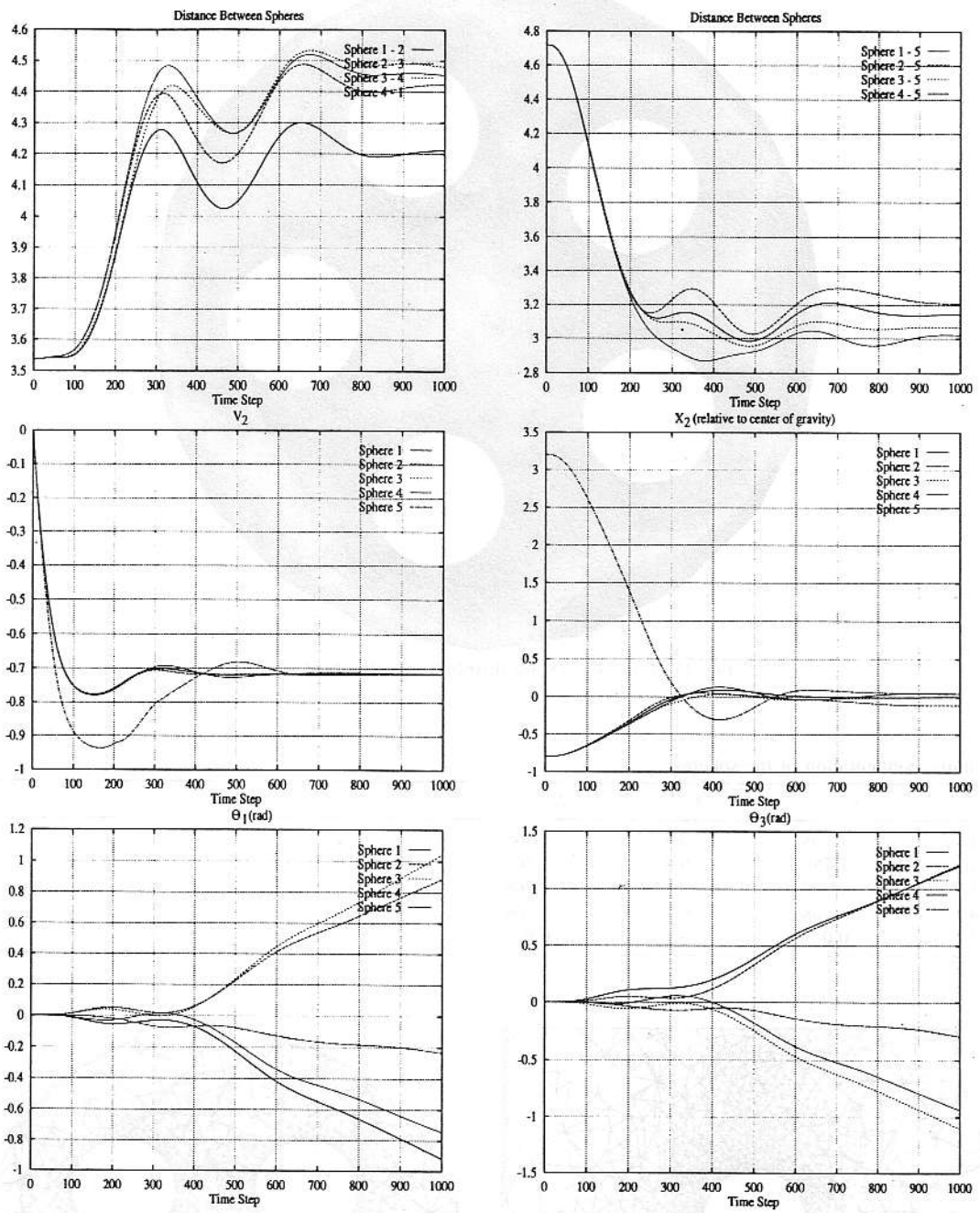

Fig. 17. Case 4: velocity and orientation histories of the spheres.

are top views of the spheres at the initial and final states. Velocity vectors and pressure distribution at a cross section through Spheres 1, 3 and 5 at the final state can be seen in Fig. 19. Also in Fig. 19 is shown pressure distribution at another cross section at the final state.

From an analysis of the flow field data and velocity time histories, we believe that this configuration of five spheres is only a neutrally-stable state, and given enough time, will rearrange themselves into the pentagon configuration of the previous case. The previous statement in Case 3 that the most stable state

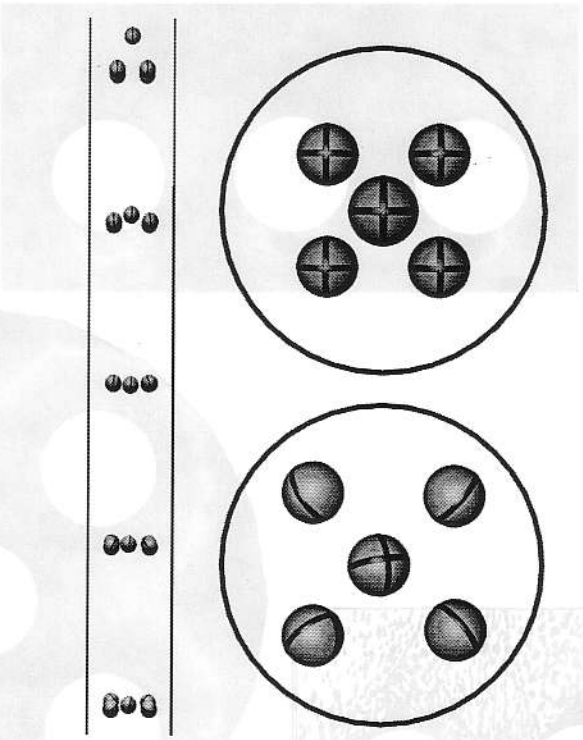

Fig. 18. Case 4: spheres at five instants during the simulation, along with top views of the initial and final configurations.

is the one with the highest terminal velocity may also point to the conclusion that this is only a neutrally-stable state. The Reynolds number at terminal velocity for this case is 71.6, while for the pentagon shape, the Reynolds number was 72.1.

6. Conclusions

In this paper we have applied our 3D parallel finite element simulation strategies for fluid–particle interaction to study multiple spheres falling in a liquid-filled tube. We used the stabilized space–time finite element formulation due to its capability in handling problems involving moving boundaries and interfaces. To handle the motion of the mesh as the spheres move, we used a combination of several methods. We combined a global translation of the mesh along with the automatic mesh moving scheme as the main mechanism of mesh motion. When the distortion of the mesh became too high, we employed remeshing techniques to continue the simulation with a new, undeformed mesh. These simulations were carried on a massively parallel computing platform which allows the completion of these large 3D computations in a reasonable amount of time.

These simulations of multiple spheres falling in a liquid-filled tube have yielded several interesting results. We have been that the multiple spheres tend to form stable geometric arrangements, and the velocity that the spheres falls is influenced by the number of spheres and their arrangement. We have also reproduced the phenomenon of two spheres drafting/kissing/tumbling in a 3D numerical simulation.

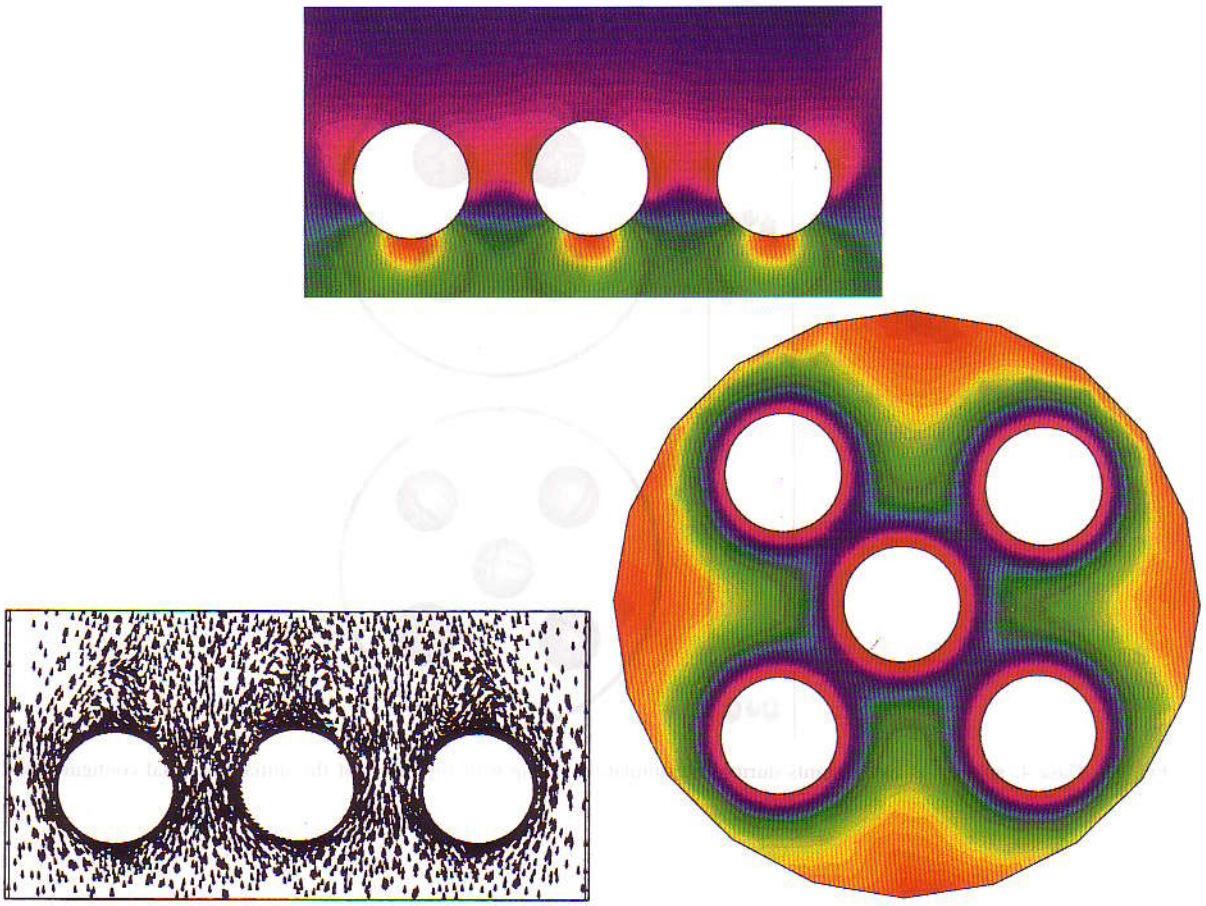

Fig. 19. Case 4: velocity vectors and pressure distribution at the final state.

Acknowledgments

This research was sponsored by ARPA under NIST contract 60NANB2D1272, 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/contract number DAAH04-94-C-0008, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. Cray C90 time was provided in part by the University of Minnesota Supercomputer Institute.

References

[1] T. Tezduyar, M. Behr and 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, Comput. Methods Appl. Mech. Engrg. 94 (1992) 339–351.

[2] T. Tezduyar, M. Behr, S. Mittal and 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, Comput. Methods Appl. Mech. Engrg. 94 (1992) 353–371.

[3] T. Tezduyar, M. Behr, S. Mittal and A. Johnson, Computation of unsteady incompressible flows with the finite element method—space–time formulations, iterative strategies and massively parallel implementations, in: P. Smolinski, W.K. Liu, G. Hulbert and K. Tamma, eds., New Methods in Transient Analysis, AMD-Vol. 143 (ASME, New York, 1992) 7–24.

[4] S. Mittal and T. Tezduyar, Massively parallel finite element computation of incompressible flows involving fluid-body interactions, Comput. Methods Appl. Mech. Engrg., 112 (1994) 253–282.

[5] A. Johnson and T. Tezduyar, Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces, Comput. Methods Appl. Mech. Engrg. 119 (1994) 73–94.
[6] K.O.L.F. Jayaweera and B.J. Mason, The behavior of clusters of spheres falling in a viscous fluid, J. Fluid Mech. 20 (1964) 121–128.
[7] A. Fortes, D. Joseph and T. Lundgren, Nonlinear mechanics of fluidization of beds of spherical particles, J. Fluid Mech. 117 (1987) 467–483.
[8] H.H. Hu, D.D. Joseph and M.J. Crochet, Direct simulation of fluid particle motions, Theoret. Comput. Fluid Mech. 3 (1992) 285–306.
[9] A. Johnson, Mesh generation and update strategies for parallel computation of flow problems with moving boundaries and interfaces, Ph.D. Thesis, University of Minnesota, 1995.
[10] A. Johnson and T. Tezduyar, Parallel computation of incompressible flows with complex geometries, Int. J. Numer. Methods Fluids, in press.
[11] M. Behr, A. Johnson, J. Kennedy, S. Mittal and T. Tezduyar, Computation of incompressible flows with implicit finite element implementations on the Connection Machine, Computer Meth. Appl. Mech. Engrg. 108 (1993) 99–118.
[12] J. Kennedy, M. Behr, V. Kalro and T. Tezduyar, Implementation of implicit finite element methods for incompressible flows on the CM-5, Comput. Methods Appl. Mech. Engrg. 119 (1994) 95–111.