Journal of Computational Physics **155,** 348–379 (1999)
Article ID jcph.1999.6340, available online at http://www.idealibrary.com on IDEAL®

A Computational Model for Suspended Large Rigid Bodies in 3D Unsteady Viscous Flows

Feng Xiao

Computational Science Laboratory, Institute of Physical and Chemical Research (RIKEN), Hirosawa 2-1, Wako, Saitama 351-0198, Japan
E-mail: xiao@atlas.riken.go.jp

Received September 8, 1998; revised June 15, 1999

A 3D numerical model for computing large rigid objects suspended in fluid flow has been developed. Rather than calculating the surface pressure upon the solid body, we evaluate the net force and torque based on a volume force formulation. The total effective force is obtained by summing up the forces at the Eulerian grids occupied by the rigid body. The effects of the moving bodies are coupled to the fluid flow by imposing the velocity field of the bodies to the fluid. A Poisson equation is used to compute the pressure over the whole domain. The objects are identified by color functions and calculated by the PPM scheme and a tangent function transformation which scales the transition region of the computed interface to a compact thickness. The model is then implemented on a parallel computer of distributed memory and validated with Stokes and low Reynolds number flows. © 1999 Academic Press

Key Words: computational algorithm; fluid dynamics; large suspended objects; multiphase flows; low Reynolds number; parallel computing.

1. INTRODUCTION

Solid objects contained in fluid interact nonlinearly with the surrounding flow and create complex flow patterns. By solid (or rigid) body (or object), we mean a mass of materials which does not experience any distortion in shape, and by "large body (or object)" we mean an object having a volume that covers at least several computational grids. In only very few cases can analytical results be obtained from simplified models. Most practical problems appear to be difficult to tackle without the help of modern computing facilities or sophisticated computational techniques. Numerical studies for solid/fluid flow have so far been conducted by various numerical approaches. The continuum approach, which has long been used to simulate the dynamics of a uniform fluid, has recently been extended to solid/fluid flows. The computational models for the solid/fluid flows can be divided into two sorts: the averaged particulate flow model based on mixture theory [9, 29] and the direct numerical simulation model [15, 8, 22]. In the averaged particulate flow model, conservation laws similar to those conventional to one-fluid dynamics are used along with some new terms to reflect the interactions between fluid and solid. An extra parameterization is needed to close the system. The averaged models usually appear easy to compute, but do not give the details of the interaction between the fluid and the solid. The direct numerical simulation approach, on the other hand, takes the physics between the flow and the solid into account by explicitly computing both the force and the torque acting on the solid and the solid driving force exerted on the suspending flow.

In [15, 8], an arbitrary Lagrangian–Eulerian (ALE) technique was incorporated with a finite element formulation to simulate moving solid bodies in suspension flow. These methods are able to deal with the translational and rotational motion of solids and appear convenient for finding the hydrodynamic force upon the body surface. Hu [8] derived a general Galerkin finite element formulation by combining the fluid and particle equations into a single variational equation, which cancels the force and the moment terms from fluid and solid. This makes computation more efficient. However, ALE and finite element-based methods need remeshing at every step to fit the moving boundaries. This may become quite computationally intensive in large 3D calculations.

Finite difference methods on a fixed mesh provide no computational cost for grid generation. This advantage appears more significant for time-dependent problems where the finite element methods or body-fitted coordinate methods on an ALE base need to remesh the computational grids to conform to the moving interface at each time step. However, a separate problem needs to be solved for those finite difference methods: how does one compute moving or free interfaces on a fixed computational mesh? To solve this kind of problem, numerical methods for tracking or capturing a moving interface or a free boundary on a fixed mesh have been developed. Among the most widely used are the works of Unverdi and Tryggvason [23], Youngs [28], and Osher and Sethian [16].

In this study, moving interfaces are computed on a fixed, finite difference mesh. Color functions, which are valued 1 or 0 depending on whether a grid falls into the region of a solid body or not, are used to identify the moving objects. A scheme, namely PPM-TFT (PPM with tangent function transformation), which is based on the PPM method [2] and a tangent transformation, is used to advance the color function. The tangent function can scale the values falling in the transition layer and then modify the slope of the discontinuity, which allows efficient control over the thickness of the transition region and avoidance of the numerical diffusion.

As a previous work on the computation of fluid-suspended large rigid objects, we developed a numerical model for computing rigid objects suspended in a gravitationally stratified flow in 2D [25]. The underlying idea is to calculate the net force and torque exerted on the object by a volume force formulation. The force is first computed at all the computational grids and then the integrated force and torque are evaluated through a summation over the space occupied by the rigid body. There is no need to calculate the information about the surface of the object, which also is generally not a trivial computation on a fixed mesh.

In this paper, we constructed a 3D computing model for unsteady flows containing rigid moving bodies, based on the fundamental consideration of [25]. The hydrodynamic equations are first computed over all of the grids. The net force driving the solid object is then calculated by summing over all grids within the solid body. The driving effect from the moving body on the surrounding fluid is taken into account by replacing the velocity field of the fluid with that of the solid body in the region occupied by the body. As opposed to an iterative procedure [17, 18], this treatment of the solid/fluid coupling does not require an extra computational step.

This paper is arranged as follows. The treatment of a sharp interface on a fixed 3D Cartesian mesh is presented in Section 2. The physical and computational aspects of the solid/fluid dynamics are described in Section 3. The solution procedure is outlined in Section 4. Section 5 briefly discusses the parallel implementation of the code. Numerical validations and testing computations are given in Section 6, and Section 7, a short conclusion, ends the paper.

2. MOVING INTERFACE TREATMENT

There are some existing methods for numerically tracking the sharp interface of a multiphase flow based on Eulerian representation. The VOF method [7, 28, 11, 12] and the level set method [21], for example, are among those widely used. For the current problem, however, we can make use of a more efficient numerical treatment to compute the sharp surface of the rigid body. The numerical technique involved is just a high-resolution scheme for a hyperbolic equation (in the present work, the PPM method is used) and a tangent transformation.

Consider L kinds of materials occupying closed areas that are embedded in the computational domain $\{\Omega_l(t) \in \mathbf{R}^3, l = 1, 2, \ldots, L\}$. We identify all the objects with density functions or color functions $\{\phi_l(x, y, z, t), l = 1, 2, \ldots, L\}$ by the definition

$$\phi_l(x, y, z, t) = \begin{cases} 1, & (x, y, z) \in \Omega_l(t), \\ 0, & \text{otherwise.} \end{cases}$$

The color functions are then predicted according to the advection equation

$$\frac{\partial \phi_l}{\partial t} + \mathbf{u} \cdot \nabla \phi_l = 0, \qquad l = 1, 2, \ldots, L, \tag{1}$$

where $\mathbf{u}$ is the local velocity. It is commonly known that almost all the finite difference schemes based on Eulerian representation tend to produce numerical diffusion. We have observed from our previous experience that the PPM method [2] has the ability to eliminate spurious oscillation and preserve the geometry of the object when used as an advection solver. However, like any other finite difference method, the PPM produces numerical diffusion that smears the initial sharpness of the discontinuity and then cannot maintain the compactness of the interface. In an earlier work [27] we used a tangent function to scale the dependent variable before solving it with the advection scheme. We found that such a transformation also works well with the PPM scheme. The resulting computational procedure is quite simple and gives a transition layer of compact thickness for the color function.

We solve the advection equation for the tangent transformation of the color function rather than the color function itself and then go back to the original field variable by an inverse transformation. This procedure, which we call the PPM-TFT method, can be simply described as the follows:

- transform ϕ_l into $F(\phi_l)$ for all l by

$$F(\phi_l) = \tan[(1 - \epsilon)\pi(\phi_l - 0.5)]$$

- solve $F(\phi_l)$ for all l by the PPM method

- invert $F(\phi_l)$ back to ϕ_l for all l by

$$\phi_l = \tan^{-1} F(\phi_l)/[(1-\epsilon)\pi_l] + 0.5,$$

where ϵ is a small positive.

We found that the tangent function transformation locally improves the spatial resolution near the large gradients, and then the sharp discontinuity can be described quite easily.

The free parameter ϵ needs to be chosen artificially before calculation. According to the feather of the tangent function, a smaller ϵ results in a numerically sharper slope across the transition layer. However, as ϵ goes to zero the tangent function approaches infinity around $\phi_l = 0$ or 1; one has to make a choice among finite positives to avoid the arithmetic instruction failure in computation. From numerical experiments, we found that a value smaller than 0.05 produced a sharp transition layer which usually covers no more than two computational grid points. In the current study, $\epsilon = 0.01$ was used for all the calculations.

Concerning the computational efficiency of this method, a comparison is given in [25]. We observed that the PPM-TFT method takes only 62.8% of the time consumed by the VOF(PLIC) method [11] and only 72.2% of the time of the level set method with reinitialization [21] in the 2D case.

As a 3D testing problem of the PPM-TFT scheme, we used the example of a rotating notched brick, which was also used in [19, 12] to evaluate the performance of the SLIC method and the PLIC method. A brick of initial shape as shown in Fig. 1 is partitioned into $24 \times 20 \times 15$ volume cells and calculated on a 40^3 mesh. Figure 2 is the result after one circle of rotation with only the PPM. We observe that the object has been smeared significantly. Figure 3 shows the result computed by the PPM-TFT method ($\epsilon = 0.05$).

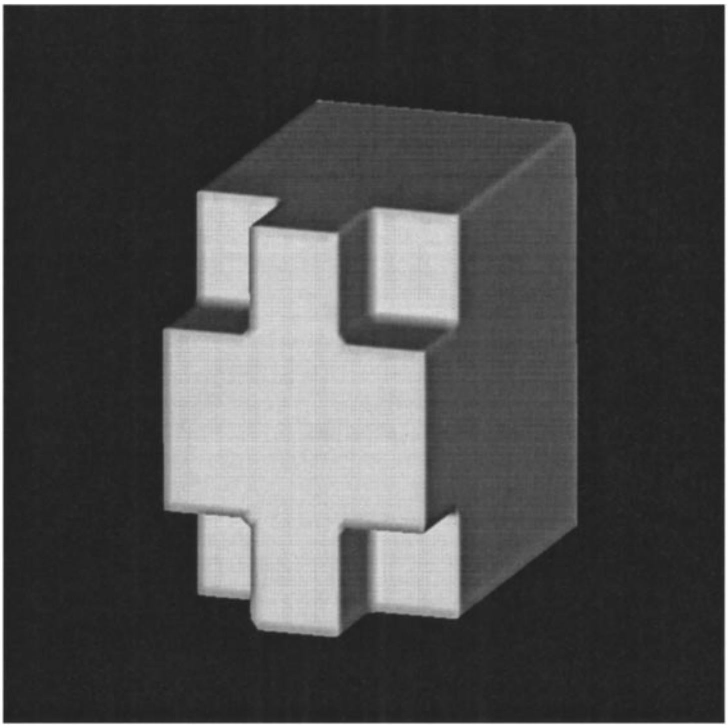

FIG. 1. Initial surface of a notched brick. The color function is valued 1 inside the brick and 0 otherwise.

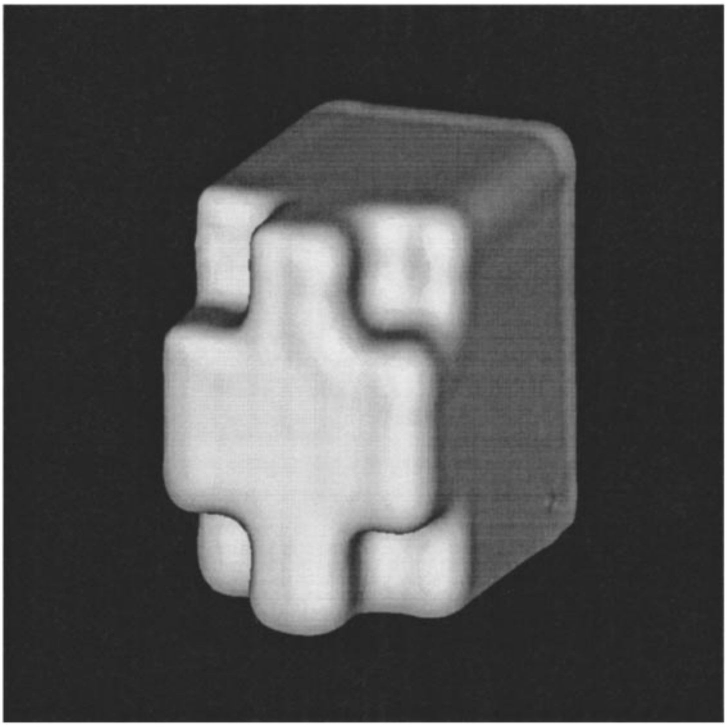

FIG. 2. The 0.5 isosurface of the notched brick after one revolution computed by the PPM method.

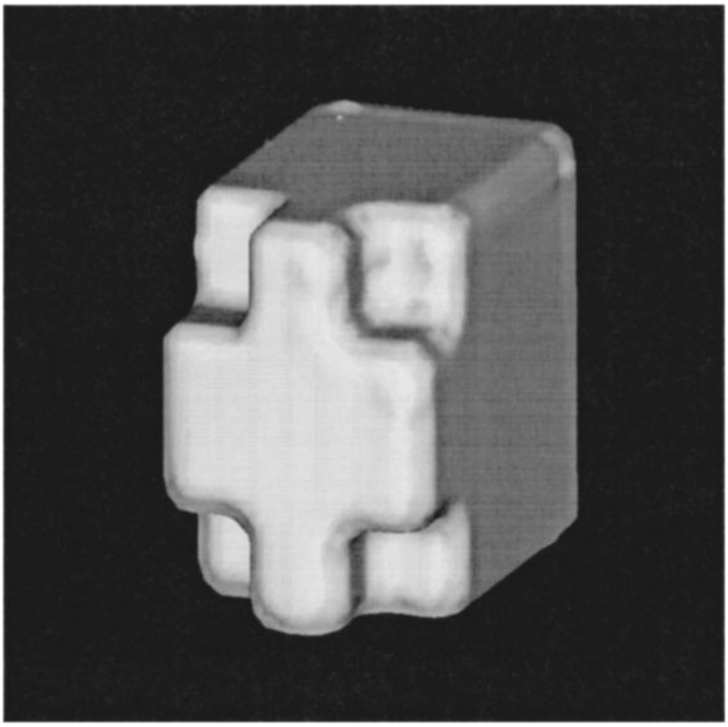

FIG. 3. Same as Fig. 2 but by PPM-TFT with $\epsilon = 0.05$.

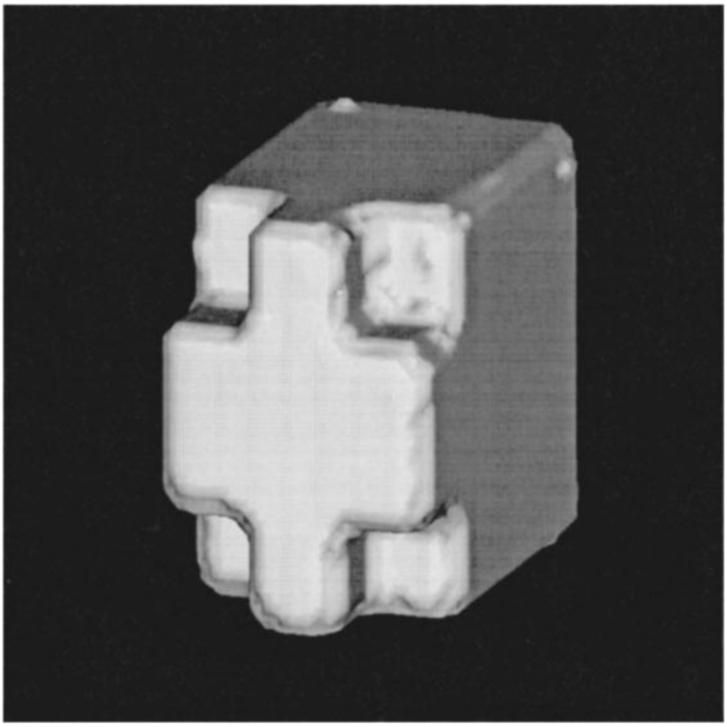

FIG. 4. Same as Fig. 2 but by PPM-TFT with $\epsilon = 0.01$.

The PPM-TFT scheme produced a diffusionless solution, and furthermore the solution is more topologically preserved when compared with those from the VOF(PLIC) method. A result for $\epsilon = 0.01$ is shown in Fig. 4. With ϵ decreasing, the transition layer becomes even narrower and the solution looks even "tougher." Figure 5 displays the contours on the cross section cutting through the notched head. As mentioned above, the PPM scheme produced a diffused profile, while the PPM-TFT method resolves the transition layer within a width of just one or two mesh units.

3. THE NUMERICAL MODEL

3.1. *The Governing Equations*

A set of hydrodynamical equations is used to model the suspending flow. We start from a set of equations for a general Newtonian viscous fluid

$$\frac{\partial \rho}{\partial t} + (\mathbf{u} \cdot \nabla)\rho = -\rho \nabla \cdot \mathbf{u}, \tag{2}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = \frac{1}{\rho} \nabla \cdot (-p\mathbf{I} + \tau) + \mathbf{F}, \tag{3}$$

$$\frac{\partial e}{\partial t} + (\mathbf{u} \cdot \nabla)e = -\frac{1}{\rho} p \nabla \cdot \mathbf{u} + \mathbf{\Phi}, \tag{4}$$

where the inner energy e is defined as $e = c_v T$. $\tau = \lambda(\nabla \cdot \mathbf{u})\mathbf{I} + 2\mu\mathbf{s}$ is the viscous stress tensor. $\mathbf{I}$ is a unit tensor. λ and μ are the two coefficients of viscosity. $\mathbf{s}$ is the tensor for rates

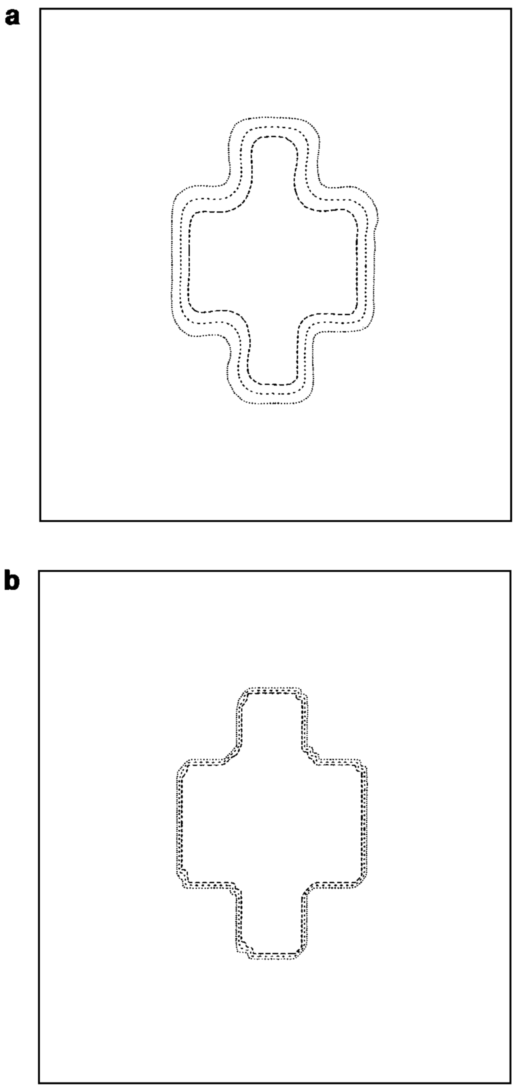

FIG. 5. (a) Contours on the cross section of the notched head calculated by the PPM scheme. The lines indicate the values of 0.1, 0.5, and 0.9, respectively. (b) The same as (a) but by the PPM-TFT method ($\epsilon = 0.01$).

of deformation. **F** is the body force, and $\mathbf{\Phi} = \lambda(\nabla \cdot \mathbf{u})^2 + 2\mu\mathbf{s} \cdot \mathbf{s}$ is the dissipation function. The thermal conduct is neglected.

Equations (2)–(4) are for compressible flow. However, as we will see later it is possible to treat incompressible flow, computationally, as a limit case of compressible flow. In order to derive a more general computational formulation that covers both compressible and incompressible cases, a fractional step procedure is used to solve the governing equation.

Applying time splitting reduces the equations to the following three parts:

1. Advection phase:

$$\frac{\partial \rho}{\partial t} + (\mathbf{u} \cdot \nabla)\rho = 0, \tag{5}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = 0, \tag{6}$$

$$\frac{\partial e}{\partial t} + (\mathbf{u} \cdot \nabla)e = 0. \tag{7}$$

2. Nonadvection phase (i):

$$p = p(e, \rho), \tag{8}$$

$$\frac{\partial \rho}{\partial t} = -\rho \nabla \cdot \mathbf{u}, \tag{9}$$

$$\frac{\partial \mathbf{u}}{\partial t} = -\frac{1}{\rho}\nabla p, \tag{10}$$

$$\frac{\partial e}{\partial t} = -\frac{p}{\rho}\nabla \cdot \mathbf{u}. \tag{11}$$

3. Nonadvection phase (ii):

$$\frac{\partial \mathbf{u}}{\partial t} = \frac{1}{\rho}\nabla \cdot \tau + \mathbf{F}, \tag{12}$$

$$\frac{\partial e}{\partial t} = \mathbf{\Phi}. \tag{13}$$

The physical variables are updated at each fractional step by using the provisional results from the previous step. Supposing the new values of dependent variables (ρ^{n+1}, p^{n+1}, e^{n+1}, $\mathbf{u}^{n+1}$) are computed from the known values at the previous step (ρ^n, p^n, e^n, $\mathbf{u}^n$), we make use of the following fractional steps on time interval $[t_n, t_{n+1}]$

$$\frac{\partial \chi^*}{\partial t} = \mathcal{L}_\chi^{AD}(t, \mathbf{x}, \chi^n), \tag{14}$$

$$\frac{\partial \chi^{**}}{\partial t} = \mathcal{L}_\chi^{N-AD1}(t, \mathbf{x}, \chi^*) \tag{15}$$

and

$$\frac{\partial \chi^{n+1}}{\partial t} = \mathcal{L}_\chi^{N-AD2}(t, \mathbf{x}, \chi^{**}) \tag{16}$$

where χ represents each of ρ, p, e, u, v, and w; $\mathcal{L}_\chi^{AD}$, $\mathcal{L}_\chi^{N-AD1}$, and $\mathcal{L}_\chi^{N-AD2}$ represent the spatial differencing for the *advection phase, nonadvection phase (i),* and *nonadvection phase (ii),* respectively. We will find, as follows, that this fractional step treatment will introduce a global error on the order of $O(\Delta t)$.

Suppose that we use one-step methods for all the fractional parts, and assume an exact starting data χ^n; then any nonfractional step method on a uniform mesh spacing can be written as

$$\begin{aligned}\chi^{n+1} - \chi^n = {}& \Delta t \mathcal{L}_\chi^{AD}(t, \mathbf{x}, \chi^n) + \Delta t \mathcal{L}_\chi^{N-AD1}(t, \mathbf{x}, \chi^n) + \Delta t \mathcal{L}_\chi^{N-AD2}(t, \mathbf{x}, \chi^n) \\ & + O\left(\Delta t^{(p_{AD}+1)} + \Delta t h^{(q_{AD}+1)}\right) + O\left(\Delta t^{(p_{N-AD1}+1)} + \Delta t h^{(q_{N-AD1}+1)}\right) \\ & + O\left(\Delta t^{(p_{N-AD2}+1)} + \Delta t h^{(q_{N-AD2}+1)}\right),\end{aligned} \tag{17}$$

where p_{AD}, q_{AD}, p_{N-AD1}, q_{N-AD1}, and p_{N-AD2}, q_{N-AD2} are the leading truncation orders of the time and space discretizations for each fractional step. Δt is the time increment and h the mesh width.

By assuming a smoothness of $\mathcal{L}_\chi^{AD}$, $\mathcal{L}_\chi^{N-AD1}$, and $\mathcal{L}_\chi^{N-AD2}$ with respect to χ, we observe that the fractional step treatments (14)–(16) have truncated errors as

$$\chi^* - \chi^n = \Delta t \mathcal{L}_\chi^{AD}(t, \mathbf{x}, \chi^n) + O\left(\Delta t^{(p_{AD}+1)} + \Delta t h^{(q_{AD}+1)}\right), \tag{18}$$

$$\begin{aligned} \chi^{**} - \chi^* = {} & \Delta t \mathcal{L}_\chi^{N-AD1}(t, \mathbf{x}, \chi^n) + O(\Delta t^2) \mathcal{L}_\chi^{AD} \frac{\partial \mathcal{L}_\chi^{N-AD1}}{\partial \chi}\bigg|^n \\ & + O\left(\Delta t^{(p_{N-AD1}+1)} + \Delta t h^{(q_{N-AD1}+1)}\right), \end{aligned} \tag{19}$$

$$\begin{aligned} \chi^{n+1} - \chi^{**} = {} & \Delta t \mathcal{L}_\chi^{N-AD2}(t, \mathbf{x}, \chi^*) + O(\Delta t^2) \mathcal{L}_\chi^{N-AD1} \frac{\partial \mathcal{L}_\chi^{N-AD2}}{\partial \chi}\bigg|^* \\ & + O\left(\Delta t^{(p_{N-AD2}+1)} + \Delta t h^{(q_{N-AD2}+1)}\right) \\ = {} & \Delta t \mathcal{L}_\chi^{N-AD2}(t, \mathbf{x}, \chi^n) + O(\Delta t^2) \mathcal{L}_\chi^{AD} \frac{\partial \mathcal{L}_\chi^{N-AD2}}{\partial \chi}\bigg|^n \\ & + O(\Delta t^2) \mathcal{L}_\chi^{N-AD1} \frac{\partial \mathcal{L}_\chi^{N-AD2}}{\partial \chi}\bigg|^* + O\left(\Delta t^{(p_{N-AD2}+1)} + \Delta t h^{(q_{N-AD2}+1)}\right). \end{aligned} \tag{20}$$

Combining Eqs. (18)–(20) and comparing the resulting expression with (17), one finds that each fractional step introduces an $O(\Delta t^2)$ local error to the solution. Even if it is possible to obtain a higher order splitting by some extra manipulations, for example, the symmetrical Strang splitting [20], the fractional steps in the present model are arranged in an order of *nonadvection phase (i)*, *nonadvection phase (ii)*, and *advection phase* for simplicity. The Euler integration method (both explicit and implicit) are used for each substep; thus then we end up with a solution with $O(\Delta t)$ global error.

This splitting technique permits a large variety of solution methods for the equations in the different phases. One can generally divide the nonadvection phase into the equation of state-related part, the constitution equation-related part, and the source-related part. Some implicit numerical formulations can be employed for stabilizing the computation in case stiffness appears . By the scheme developed in [26], the equations of the *advection phase* can be solved on a compact stencil. As the equation of state-related part, *nonadvection phase (i)* reflects the aspect of fluid compressibility. By treating this part separately, we can obtain a formulation to deal with flows of quite different compressibilities, from compressible fluid to nearly incompressible fluid. We will next derive a formulation for computing pressure from the equations of the *nonadvection phase (i)*.

From Eq. (10), we get a calculation formula as

$$\nabla \cdot \left(\frac{\nabla p}{\rho} \right) = -\frac{\partial}{\partial t}(\nabla \cdot \mathbf{u}) = \frac{1}{\Delta t}[-(\nabla \cdot \mathbf{u})^{**} + (\nabla \cdot \mathbf{u})^*]. \tag{21}$$

The superscripts * and ** indicate the provisional values before and after the calculation of the *nonadvection phase (i)*. By operating on (8) with $\partial/\partial t$ and considering the continuity relation Eq. (9), we have

$$\frac{\partial p}{\partial t} = \frac{\partial p}{\partial e}\frac{\partial e}{\partial t} + \frac{\partial p}{\partial \rho}\frac{\partial \rho}{\partial t} = \frac{\partial p}{\partial e}\frac{\partial e}{\partial t} - \rho \frac{\partial p}{\partial \rho} \nabla \cdot \mathbf{u}. \tag{22}$$

From (22) and (11), there is

$$\begin{aligned}\frac{\partial p}{\partial t} &= \frac{\partial p}{\partial e}\left(-\frac{p}{\rho}\nabla\cdot\mathbf{u}\right) - \rho\frac{\partial p}{\partial \rho}\nabla\cdot\mathbf{u}\\ &= -\left(\frac{p}{\rho}\frac{\partial p}{\partial e} + \rho\frac{\partial p}{\partial \rho}\right)\nabla\cdot\mathbf{u}. \end{aligned}\tag{23}$$

Making use of a temporally implicit velocity in above expression, we have

$$(\nabla\cdot\mathbf{u})^{**} = -\frac{\partial p}{\partial t}\Big/\left(\frac{p}{\rho}\frac{\partial p}{\partial e} + \rho\frac{\partial p}{\partial \rho}\right). \tag{24}$$

Combining Eq. (24) with Eq. (21), one gets

$$\nabla\cdot\left(\frac{\nabla p}{\rho}\right) = \frac{1}{\Delta t}\left(\frac{\partial p}{\partial t}\Big/\left(\frac{p}{\rho}\frac{\partial p}{\partial e} + \rho\frac{\partial p}{\partial \rho}\right) + \frac{1}{\Delta t}(\nabla\cdot\mathbf{u})^{*}. \tag{25}$$

Equation (25) is an evolution equation of pressure p. One candidate of the time integration formula is

$$\nabla\cdot\left(\frac{\nabla p}{\rho}\right)^{**} = (p^{**} - p^{*})\Big/\left[\Delta t^2\left(\frac{p}{\rho}\frac{\partial p}{\partial e} + \rho\frac{\partial p}{\partial \rho}\right)\right]^{*} + \frac{1}{\Delta t}(\nabla\cdot\mathbf{u})^{*}. \tag{26}$$

We then obtain a Poisson type equation for calculating pressure. The first term on the RHS of Eq. (26) reflects the effects of the assumed local thermodynamic equilibrium, by which the contributions from compressibility are included. In fact, $[(\frac{p}{\rho}\frac{\partial p}{\partial e} + \rho\frac{\partial p}{\partial \rho})/\rho]^{1/2} = c$ represents the speed of sound for any given equation of state (EOS). As a limit case, one can choose an equation of state so that c goes to infinity. The first term on the RHS of (26) then vanishes, and Eq. (26) approaches

$$\nabla\cdot\left(\frac{\nabla p}{\rho}\right)^{**} = \frac{1}{\Delta t}(\nabla\cdot\mathbf{u})^{*}. \tag{27}$$

By projecting the velocity with the pressure field computed from (27), one gets a divergence-free velocity field. This is the case of incompressible flow.

In general, EOS data can be input to the simulations using either tabular data libraries or prescribed functions. When large gradients of discontinuities appear, smoothed EOS data are needed to take numerical derivatives. The quotidian equation of state discussed in [14] is an example of the equations of state for practical simulations. In the present study, the materials involved are rigid bodies, liquid, and gas. The liquid and the gas are both assumed to have an equation of state in the form of $p = \gamma\rho e$, but with quite a different speed of sound. $\gamma = 0.4$ was used for the gas, and $\gamma = 2.0 \times 10^4$ for the liquid.

Hence the equations for *nonadvection phase (i)* can be rewritten as

$$\nabla\cdot\left(\frac{\nabla p}{\rho}\right) = \frac{1}{\Delta t}\left(\frac{\partial p}{\partial t}\right)\Big/\left(\frac{p}{\rho}\frac{\partial p}{\partial e} + \rho\frac{\partial p}{\partial \rho}\right) + \frac{1}{\Delta t}(\nabla\cdot\mathbf{u}), \tag{28}$$

$$\frac{\partial \mathbf{u}}{\partial t} = -\frac{1}{\rho}\nabla p, \tag{29}$$

$$\frac{\partial \rho}{\partial t} = -\rho \nabla \cdot \mathbf{u}, \tag{30}$$

$$e = e(p, \rho). \tag{31}$$

By (31) we mean that the internal energy should be evaluated at this stage from the resulting ρ and p through the equation of state.

The Poisson equation of p is solved over all the computational domain, so the pressure gradient force in the form of volume force is available at each grid within the solid body for calculating the total force exerted on that body.

Because of the existence of different material components, jumps in dependent variables are expected to appear across material surface. In practice, it is important to make the jumps be bounded with finite values. This was done in our model by averaging both the EOS and the velocity field with the color function. Integrating (26) over a volume containing an object surface with a thickness which becomes infinitesimally small, we know that if one assumes a bounded jump for the terms on the RHS of (26) across the material surface, the resulting pressure gives a continuous distribution of $\frac{1}{\rho}\frac{\partial p}{\partial n}$ (n indicates a path normal to the object surface) across the interface and even ρ has a large jump across the boundary. Thus, one can use Eq. (26) to evaluate the pressure over the entire computational domain and get a reasonable pressure distribution across the material boundary. In the current study, we did not specify an EOS separately for any rigid object but used the same EOS as the surrounding fluid, and the velocity distribution was averaged over the interface, based on the color function.

We have noticed that a more sophisticated treatment for such a problem can be derived from known jump conditions across the boundary [13]. Using the given jump conditions, one could attempt to get an overall second-order accuracy. Imposing the known jump conditions on the algebraic equations from the discretization of (26), however, makes the solution procedure more complicated. Another boundary treatment for Cartesian grid methods was also reported in [4], where a boundary is treated as a symmetry line and a inviscid flow field can be completely defined at the ghost cells via a reflecting boundary condition. Then, the boundary cells can be treated like regular cells. This method, however, needs to compute the orientation of the object surface and does not give a flow field over all the object regions.

The averaging we use in the current model is relatively simple, but, as we will see in Section 6, gives adequate results for a wide spectrum of problems of medium or low Reynolds number flows.

The advection equation for color function ϕ_l

$$\frac{\partial \phi_l}{\partial t} + \mathbf{u}_{b(l)} \cdot \nabla \phi_l = 0, \qquad l = 1, 2, \ldots, L \tag{32}$$

is added to the advection phase and solved by the above PPM-TFT method to recognize all the solid bodies, where $\mathbf{u}_{b(l)}$ is the velocity field for object l. The motion of the object l is determined by

$$\mathbf{u}_{b(l)} = \bar{\mathbf{u}}_l + \mathbf{r} \times \bar{\boldsymbol{\Omega}}_l, \tag{33}$$

where $\bar{\mathbf{u}}_l$ is the translational speed of the mass center of object l, $\bar{\boldsymbol{\Omega}}_l$ the angular speed, and $\mathbf{r}$ the distance to the mass center. These quantities are predicted by the following Newton's law of motion,

$$\frac{d\bar{\mathbf{u}}_l}{dt} = \bar{\mathbf{f}}_l \tag{34}$$

and

$$\frac{d}{dt}\left(\bar{\mathbf{\Pi}}_{(l)}\bar{\mathbf{\Omega}}_l\right) = \bar{\mathbf{\Gamma}}_l, \tag{35}$$

with $\bar{\mathbf{f}}_l$ being the net force and $\bar{\mathbf{\Gamma}}_l$ the torque for object l. $\bar{\mathbf{\Pi}}_{(l)}$ is the tensor of inertia moment and

$$\bar{\mathbf{\Pi}}_{(l)} = \begin{pmatrix} \Pi_{(l)xx} & \Pi_{(l)xy} & \Pi_{(l)xz} \\ \Pi_{(l)yx} & \Pi_{(l)yy} & \Pi_{(l)yz} \\ \Pi_{(l)zx} & \Pi_{(l)zy} & \Pi_{(l)zz} \end{pmatrix}$$

is a symmetric second-order tensor.

3.2. *The Coupling between Solid and Fluid*

The impact forces from an accelerated solid body on the fluid are taken into account by imposing a change of motion on the boundaries through a no-slip boundary condition. This means that the fluid on the rigid body surface should move with the body. We modify the velocity distribution at every time step as

$$\tilde{\mathbf{u}}_{i,j,k} = \left(1 - \phi_{(l)i,j,k}\right)\mathbf{u}_{i,j,k} + \phi_{(l)i,j,k}\mathbf{u}_{b(l)i,j,k}, \tag{36}$$

where $\phi_{(l)i,j,k}$ is the color function and $\mathbf{u}_{b(l)i,j,k}$ the velocity of the solid body l at grid point (i, j, k).

Imposing a change in velocity will obviously cause a response in the pressure field. In order to clarify how the pressure responds to the velocity change, we can examine the disturbed pressure field $\tilde{p}$ caused by the imposed velocity $\tilde{\mathbf{u}}$ as follows.

Replacing $\mathbf{u}$ with $\tilde{\mathbf{u}}$ in the equations of the *nonadvection phase (i)*, we get a modified Poisson equation for pressure as

$$\nabla \cdot \left(\frac{\nabla \tilde{p}}{\rho}\right)^{**} = (\tilde{p}^{**} - p^{*}) \Big/ \left[\Delta t^2 \left(\frac{p}{\rho}\frac{\partial p}{\partial e} + \rho \frac{\partial p}{\partial \rho}\right)\right]^{*} + \frac{1}{\Delta t}(\nabla \cdot \tilde{\mathbf{u}})^{*}. \tag{37}$$

Letting $\acute{p} = \tilde{p}^{**} - p^{**}$ and recalling Eq. (36), we can recast Eq. (37) into

$$\begin{aligned} &\nabla \cdot \left(\frac{\nabla p}{\rho}\right)^{**} + \nabla \cdot \left(\frac{\nabla \acute{p}}{\rho}\right) \\ &\quad = (p^{**} - p^{*}) \Big/ \left[\Delta t^2 \left(\frac{p}{\rho}\frac{\partial p}{\partial e} + \rho \frac{\partial p}{\partial \rho}\right)\right]^{*} + \frac{1}{\Delta t}(\nabla \cdot \mathbf{u})^{*} \\ &\qquad + \acute{p} \Big/ \left[\Delta t^2 \left(\frac{p}{\rho}\frac{\partial p}{\partial e} + \rho \frac{\partial p}{\partial \rho}\right)\right]^{*} + \frac{1}{\Delta t}\nabla \phi_{(l)} \cdot \left(\mathbf{u}_{b(l)} - \mathbf{u}^{*}\right). \end{aligned} \tag{38}$$

Comparing with Eq. (26), we have

$$\nabla \cdot \left(\frac{\nabla \acute{p}}{\rho}\right) = \acute{p} \Big/ \left[\Delta t^2 \left(\frac{p}{\rho}\frac{\partial p}{\partial e} + \rho \frac{\partial p}{\partial \rho}\right)\right]^* + \frac{1}{\Delta t} \nabla \phi_{(l)} \cdot \left(\mathbf{u}_{b(l)} - \mathbf{u}^*\right). \qquad (39)$$

Since the interface remains sharp, the function $\phi(\mathbf{x})$ is a Heaviside type function and has a generalized derivative $\phi'(\mathbf{x})$ defined by

$$\langle \phi'(\mathbf{x}), \psi(\mathbf{x})\rangle = -\langle \phi(\mathbf{x}), \psi'(\mathbf{x})\rangle, \qquad \forall \psi \in C_0^\infty$$

with

$$\langle a(\mathbf{x}), b(\mathbf{x})\rangle = \int_{-\infty}^{\infty} a(\mathbf{x}) b(\mathbf{x})\, d\mathbf{x}.$$

Thus, we have $\phi'(\mathbf{x}) = \delta(\mathbf{x})$ and consequently $\nabla\phi_{(l)} = -\mathbf{n}(\mathbf{x})\delta[\mathbf{n}(\mathbf{x}_{b(l)}) \cdot (\mathbf{x} - \mathbf{x}_{b(l)})]$, $\mathbf{x}_{b(l)}$ represents any point on the surface of object l, and $\mathbf{n}(\mathbf{x})$ is the outgoing normal vector. Therefore

$$\nabla \cdot \left(\frac{\nabla \acute{p}}{\rho}\right) = \acute{p} \Big/ \left[\Delta t^2 \left(\frac{p}{\rho}\frac{\partial p}{\partial e} + \rho \frac{\partial p}{\partial \rho}\right)\right]^* \\ - \frac{1}{\Delta t}\delta\left[\mathbf{n}\left(\mathbf{x}_{b(l)}\right) \cdot \left(\mathbf{x} - \mathbf{x}_{b(l)}\right)\right]\mathbf{n}(\mathbf{x}) \cdot \left(\mathbf{u}_{b(l)} - \mathbf{u}^*\right). \qquad (40)$$

Neumann type boundary conditions were used for the pressure field, and the rigid bodies were positioned far enough from the domain boundaries in our simulations to reduce the direct effect of the boundary conditions on the rigid bodies. We examined $\acute{p}$ on the domain boundaries for some cases, and observed that $\acute{p}$ was negligibly small on the domain boundaries during one time step. Assuming that $\acute{p}$ vanishes on the boundaries of the computational domain, we find that the velocity discontinuity in the direction normal to the body surface, i.e., the second term on the RHS of Eq. (40), is the only contribution to the pressure change. Therefore if the imposed velocity produces a velocity discontinuity normal to the body surface, an impulsive change in pressure will be introduced. For an accelerated body, a positive pressure disturbance will be created in the fluid ahead of it and a negative one will appear on the lee side, and vice versa for a decelerated body. This disturbed pressure will then build up a pressure gradient field that produces a fluid force back to the solid body. This induced pressure field meanwhile produces a change in fluid velocity and makes the motion of fluid consistent to the moving body.

We use an explicit formulation to compute the viscosity in the *nonadvection phase (ii)*. Similarly, a no-slip condition on the rigid objects is imposed by Eq. (36) at every step. A viscous fluid field will then develop while the motion of the fluid elements on the body surface coincides with the motion of the moving body.

3.3. *Formulation of a Rigid Object*

By using color functions we can easily treat solid bodies or objects in any complex shape or having heterogeneous density distributions. Supposing that solid objects are separable into different subregions where density remains constant, we treat object l as a combination of the joint parts represented by color functions $\phi_{(lq)}$ $(q = 1, 2, \ldots, Q_{(l)})$. $Q_{(l)}$ denotes the number of components that make up object l. $\phi_{(lq)}$ is valued initially 1 for each subregion and 0 otherwise and each $\phi_{(lq)}$ indicates a region where the density $\rho_{(lq)}$ is uniform. According to the definitions of $\phi_{(l)}$ and $\phi_{(lq)}$, it is obvious that

$$\phi_{(l)i,j,k} = \sum_{q=1}^{Q_{(l)}} \phi_{(lq)i,j,k}.$$

The motion of object l can be decomposed into a translation $\bar{\mathbf{u}}_l = (\bar{u}_l, \bar{v}_l, \bar{w}_l)$ of the mass center and a mean rotation $\bar{\boldsymbol{\Omega}}_l = (\bar{\omega}_{lx}, \bar{\omega}_{ly}, \bar{\omega}_{lz})$. Next, we derive the computational formula for these quantities based on the color functions.

The mass center of the lth object can be computed directly from a volume integration formulation as

$$\bar{x}_l = \frac{1}{M_{(l)}} \sum_{i,j,k} \sum_{q=1}^{Q_{(l)}} x_i \phi_{(lq)i,j,k} \rho_{(lq)} \Delta x_i \Delta y_j \Delta z_k, \tag{41}$$

$$\bar{y}_l = \frac{1}{M_{(l)}} \sum_{i,j,k} \sum_{q=1}^{Q_{(l)}} y_j \phi_{(lq)i,j,k} \rho_{(lq)} \Delta x_i \Delta y_j \Delta z_k \tag{42}$$

and

$$\bar{z}_l = \frac{1}{M_{(l)}} \sum_{i,j,k} \sum_{q=1}^{Q_{(l)}} z_k \phi_{(lq)i,j,k} \rho_{(lq)} \Delta x_i \Delta y_j \Delta z_k, \tag{43}$$

where (i, j, k) indicates the computational grid, and (x_i, y_j, z_k) represents the coordinate of point (i, j, k) in 3D space. $M_{(l)}$ denotes the total mass of the lth object and is calculated by

$$M_{(l)} = \sum_{i,j,k} \sum_{q=1}^{Q_{(l)}} \phi_{(lq)i,j,k} \rho_{(lq)} \Delta x_i \Delta y_j \Delta z_k. \tag{44}$$

Since all the forces (including both the body force and the fluid stress) are calculated at all grids in a volume force form as

$$\mathbf{f}_{i,j,k} = -\nabla p_{i,j,k} + (\nabla \cdot \tau)_{i,j,k} + (\rho \mathbf{F})_{i,j,k}, \tag{45}$$

it is convenient to compute the net force upon the mass center of object l by summing up all the forces over the object volume. That is,

$$\bar{\mathbf{f}}_l = \frac{1}{M_{(l)}} \sum_{i,j,k} \sum_{q=1}^{Q_{(l)}} [-\nabla p_{i,j,k} + (\nabla \cdot \tau)_{i,j,k} + \rho_{lq} \mathbf{F}_{i,j,k}] \phi_{(lq)i,j,k} \Delta x_i \Delta y_j \Delta z_k. \tag{46}$$

After the net force $\bar{\mathbf{f}}_l$ is calculated from (46), the translation motion of the mass center is evaluated by (34).

By Eqs. (34) and (46), we evaluate the total force acting on the solid body in terms of "volume force." We calculate the net force by summing up all the forces at the computation grids over the volume of the solid body. Different from the so-called “surface force” formulation, the volume force-based schemes, as we used here, do not need the information of the body surface, such as the orientations and the surface areas, which appear in other difficult problems in computation.

Supposing that there is no support or fixed point to a solid body, we can include the rotation of a rigid body by considering an angular motion to its mass center. With the mass center of a solid body known, the elements of the tensor of the moment of inertia are calculated as

$$\Pi_{(l)xx} = \sum_{i,j,k} \sum_{q=1}^{Q_{(l)}} \left[(y_j - \bar{y}_l)^2 + (z_k - \bar{z}_l)^2 \right] \Delta m_{i,j,k}, \tag{47}$$

$$\Pi_{(l)yy} = \sum_{i,j,k} \sum_{q=1}^{Q_{(l)}} \left[(z_k - \bar{z}_l)^2 + (x_i - \bar{x}_l)^2 \right] \Delta m_{i,j,k}, \tag{48}$$

$$\Pi_{(l)zz} = \sum_{i,j,k} \sum_{q=1}^{Q_{(l)}} \left[(x_i - \bar{x}_l)^2 + (y_j - \bar{y}_l)^2 \right] \Delta m_{i,j,k}, \tag{49}$$

$$\Pi_{(l)xy} = \Pi_{(l)yx} = -\sum_{i,j,k} \sum_{q=1}^{Q_{(l)}} [(x_i - \bar{x}_l)(y_j - \bar{y}_l)] \Delta m_{i,j,k}, \tag{50}$$

$$\Pi_{(l)xz} = \Pi_{(l)zx} = -\sum_{i,j,k} \sum_{q=1}^{Q_{(l)}} [(x_i - \bar{x}_l)(z_k - \bar{z}_l)] \Delta m_{i,j,k}, \tag{51}$$

$$\Pi_{(l)yz} = \Pi_{(l)zy} = -\sum_{i,j,k} \sum_{q=1}^{Q_{(l)}} [(y_j - \bar{y}_l)(z_k - \bar{z}_l)] \Delta m_{i,j,k}, \tag{52}$$

where $\Delta m_{i,j,k} = \phi_{(lq)i,j,k} \rho_{(lq)} \Delta x_i \Delta y_j \Delta z_k$ is the mass of one volume unit of the rigid object.

The components of moment of force (torque) in x, y, and z are computed as

$$\bar{\Gamma}_{lx} = \sum_{i,j,k} \sum_{q=1}^{Q_{(l)}} [(y_j - \bar{y}_l) f_{zi,j,k} - (z_k - \bar{z}_l) f_{yi,j,k}] \phi_{(lq)i,j,k} \Delta x_i \Delta y_j \Delta z_k, \tag{53}$$

$$\bar{\Gamma}_{ly} = \sum_{i,j,k} \sum_{q=1}^{Q_{(l)}} [(z_k - \bar{z}_l) f_{xi,j,k} - (x_i - \bar{x}_l) f_{zi,j,k}] \phi_{(lq)i,j,k} \Delta x_i \Delta y_j \Delta z_k \tag{54}$$

and

$$\bar{\Gamma}_{lz} = \sum_{i,j,k} \sum_{q=1}^{Q_{(l)}} [(x_i - \bar{x}_l) f_{yi,j,k} - (y_j - \bar{y}_l) f_{xi,j,k}] \phi_{(lq)i,j,k} \Delta x_i \Delta y_j \Delta z_k. \tag{55}$$

For convenience of computation, the equation of rotational motion of rigid body l (35) is written in the form

$$\sum_{\beta} \left(\Pi_{(l)\alpha\beta} \frac{d\bar{\omega}_{l\beta}}{dt} \right) = \bar{\Gamma}_{l\alpha} - \sum_{\beta} \left(\bar{\omega}_{l\beta} \frac{d\Pi_{(l)\alpha\beta}}{dt} \right) \qquad (\alpha, \beta = x, y, z), \tag{56}$$

where $d\Pi_{(l)\alpha\beta}/dt$ is approximated by differentiating the moment of inertia with respect to

t. The resulting relations read

$$\frac{d\Pi_{(l)xx}}{dt} = \sum_{i,j,k}\sum_{q=1}^{Q_{(l)}}[2(y_j - \bar{y}_l)(v_{i,j,k} - \bar{v}_l) + 2(z_k - \bar{z}_l)(w_{i,j,k} - \bar{w}_l)]\Delta m_{i,j,k}, \tag{57}$$

$$\frac{d\Pi_{(l)yy}}{dt} = \sum_{i,j,k}\sum_{q=1}^{Q_{(l)}}[2(z_k - \bar{z}_l)(w_{i,j,k} - \bar{w}_l) + 2(x_i - \bar{x}_l)(u_{i,j,k} - \bar{u}_l)]\Delta m_{i,j,k}, \tag{58}$$

$$\frac{d\Pi_{(l)zz}}{dt} = \sum_{i,j,k}\sum_{q=1}^{Q_{(l)}}[2(x_i - \bar{x}_l)(u_{i,j,k} - \bar{u}_l) + 2(y_j - \bar{y}_l)(v_{i,j,k} - \bar{v}_l)]\Delta m_{i,j,k}, \tag{59}$$

$$\frac{d\Pi_{(l)xy}}{dt} = \frac{d\Pi_{(l)yx}}{dt} = -\sum_{i,j,k}\sum_{q=1}^{Q_{(l)}}[(x_i - \bar{x}_l)(v_{i,j,k} - \bar{v}_l) + (y_j - \bar{y}_l)(u_{i,j,k} - \bar{u}_l)]\Delta m_{i,j,k}, \tag{60}$$

$$\frac{d\Pi_{(l)xz}}{dt} = \frac{d\Pi_{(l)zx}}{dt} = -\sum_{i,j,k}\sum_{q=1}^{Q_{(l)}}[(x_i - \bar{x}_l)(w_{i,j,k} - \bar{w}_l) + (z_k - \bar{z}_l)(u_{i,j,k} - \bar{u}_l)]\Delta m_{i,j,k}, \tag{61}$$

$$\frac{d\Pi_{(l)yz}}{dt} = \frac{d\Pi_{(l)zy}}{dt} = -\sum_{i,j,k}\sum_{q=1}^{Q_{(l)}}[(y_j - \bar{y}_l)(w_{i,j,k} - \bar{w}_l) + (z_k - \bar{z}_l)(v_{i,j,k} - \bar{v}_l)]\Delta m_{i,j,k}. \tag{62}$$

From (56), we have the following evolution equations for $\bar{\omega}_{lx}$, $\bar{\omega}_{ly}$, and $\bar{\omega}_{lz}$,

$$\frac{d}{dt}(\bar{\omega}_{lx}) = \frac{\det \bar{\mathbf{\Lambda}}_{lx}}{\det \bar{\mathbf{\Pi}}_{(l)}}, \tag{63}$$

$$\frac{d}{dt}(\bar{\omega}_{ly}) = \frac{\det \bar{\mathbf{\Lambda}}_{ly}}{\det \mathbf{\Pi}_{(l)}} \tag{64}$$

and

$$\frac{d}{dt}(\bar{\omega}_{lz}) = \frac{\det \bar{\mathbf{\Lambda}}_{lz}}{\det \mathbf{\Pi}_{(l)}}, \tag{65}$$

where

$$\bar{\mathbf{\Lambda}}_{lx} = \begin{pmatrix} \eta_{lx} & \Pi_{(l)xy} & \Pi_{(l)xz} \\ \eta_{ly} & \Pi_{(l)yy} & \Pi_{(l)yz} \\ \eta_{lz} & \Pi_{(l)zy} & \Pi_{(l)zz} \end{pmatrix},$$

$$\bar{\mathbf{\Lambda}}_{ly} = \begin{pmatrix} \Pi_{(l)xx} & \eta_{lx} & \Pi_{(l)xz} \\ \Pi_{(l)yx} & \eta_{ly} & \Pi_{(l)yz} \\ \Pi_{(l)zx} & \eta_{lz} & \Pi_{(l)zz} \end{pmatrix},$$

$$\bar{\mathbf{\Lambda}}_{lz} = \begin{pmatrix} \Pi_{(l)xx} & \Pi_{(l)xy} & \eta_{lx} \\ \Pi_{(l)yx} & \Pi_{(l)yy} & \eta_{ly} \\ \Pi_{(l)zx} & \Pi_{(l)zy} & \eta_{lz} \end{pmatrix}$$

with

$$\eta_{lx} = \overline{\Gamma}_{lx} - \left(\frac{d\Pi_{(l)xx}}{dt}\overline{\omega}_{lx} + \frac{d\Pi_{(l)xy}}{dt}\overline{\omega}_{ly} + \frac{d\Pi_{(l)xz}}{dt}\overline{\omega}_{lz} \right),$$
$$\eta_{ly} = \overline{\Gamma}_{ly} - \left(\frac{d\Pi_{(l)xy}}{dt}\overline{\omega}_{lx} + \frac{d\Pi_{(l)yy}}{dt}\overline{\omega}_{ly} + \frac{d\Pi_{(l)yz}}{dt}\overline{\omega}_{lz} \right)$$

and

$$\eta_{lz} = \overline{\Gamma}_{lz} - \left(\frac{d\Pi_{(l)zx}}{dt}\overline{\omega}_{lx} + \frac{d\Pi_{(l)zy}}{dt}\overline{\omega}_{ly} + \frac{d\Pi_{(l)zz}}{dt}\overline{\omega}_{lz} \right).$$

Equations (34) and (63)–(65) are then integrated by the forward Euler method. Once the translational and rotational speeds have been determined, we end up with the velocity of the *l*th moving rigid object as

$$u_{b(l)i,j,k} = \overline{u}_l + \overline{\omega}_{ly}(z_k - \overline{z}_l) - \overline{\omega}_{lz}(y_j - \overline{y}_l), \tag{66}$$
$$v_{b(l)i,j,k} = \overline{v}_l + \overline{\omega}_{lz}(x_i - \overline{x}_l) - \overline{\omega}_{lx}(z_k - \overline{z}_l) \tag{67}$$

and

$$w_{b(l)i,j,k} = \overline{w}_l + \overline{\omega}_{lx}(y_j - \overline{y}_l) - \overline{\omega}_{ly}(x_i - \overline{x}_l). \tag{68}$$

It is obvious that the resulting velocity field for advancing the rigid body will never cause any distortion on the body. According to the velocity field of the moving body (66)–(68), we used the sharp interface tracking scheme discussed in Section 2 to update the profile of the rigid body. In spite of a few minor and local deformations on the body surface due to numerical errors, a overall geometrically faithful solution to the moving object was achieved.

4. BRIEF SUMMARY OF THE CALCULATION PROCEDURES

For simulating an unsteady flow suspending rigid moving objects, we need to calculate alternately the net forces and torques acting on the objects, and the effects from the moving objects on the fluid. The solid bodies and the fluid should be coupled during the time integration. The evolutionary solutions to the problems presented in this paper are computed via the following fractional steps:

- Find the forces and the moment of forces exerted on the rigid objects and determine their velocities by the numerical processes discussed in Section 3.3.
- Advance the solid bodies, according to the velocities obtained from the preceding step, to their updated positions by solving Eq. (32) with the interface tracking technique discussed in Section 2.
- Modify, according to the updated color functions, the velocity $\mathbf{u}_{i,j}$ as

```
$do\ 1, l$
$\mathbf{u}_{i,j,k} = (1 - \phi_{(l)i,j,k})\mathbf{u}_{i,j,k} + \phi_{(l)i,j,k}\mathbf{u}_{b(l)i,j,k}$
$enddo.$
```

- Solve the Poisson equation (28). The matrix resulting from a central difference discretization is computed by the Bi-CGSTAB method [24] with a tridiagonal approximation factorization preconditionner [3].
- Calculate the velocity $\mathbf{u}$ and the density ρ by the equations in the *nonadvection phase (i)* (29) and (30).
- Calculate the inner energy e by equation of state (31).
- Compute the *nonadvection phase (ii)* for $\mathbf{u}$ and e.
- Advance the physical variables for the *advection phase* by the scheme presented in [26].
- Repeat the procedure.

5. PARALLELIZATION ON A DISTRIBUTED MEMORY MACHINE

Directly and realistically simulating multimaterial dynamics of a large scale in 3D requires high-speed hardware with a large memory space. It is now widely recognized that only parallel processing offers the potential of solving such a problem. Therefore, an important requirement is that a practical numerical code should be parallelable on various distributed environments with moderate effort. We found that there is no substantial difficulty in porting the present model to a parallel architecture. Some aspects of its parallel implementation on the Fujitsu VPP/500 system at the Institute of Physical and Chemical Research (RIKEN) will be discussed in this section.

Being a DM-MIMD (distributed memory and multiple instruction stream multiple data) machine, the RIKEN VPP500 system has 28 processor elements (PEs). Each PE has 256 MB of memory and is equipped with both scalar and vector processing units, which gives a peak speed of 1.6 GFLOPS. Interprocessor communication is realized through a cross bar network.

The computational domain was divided into several subdomains by a grid-partitioning algorithm. In the present computation, 1D partitioning was used to decompose the computational grids into nonoverlapping blocks, and each block is assigned to one processor. Each processor stores its own data while communicating with the other processors during the numerical solution procedure. To distribute the grid data to each processor and exchange computed values among the processors both a local/local and a local/global communication are used. The local/global communication is done by mapping the global variable values onto the corresponding local ones. Except for the Poisson pressure equation, all the rest of the numerical model is written in an explicit sense. Therefore, we need only exchange the data at the boundary regions between neighboring subdomains. Due to the compactness of the advection solver in the present code, only one stencil of values next to the subdomain boundaries need to be communicated for the explicit parts. The elliptic Poisson equation for pressure, which may contain discontinuous coefficients or singular sources, is currently solved by the Bi-CGSTAB method [24] with a tridiagonal approximation factorization preconditionner [3]. This method guarantees, in most cases, the convergence of iteration, but suffers from some communication overhead. When calculating the net force and torque exerted on a solid body, operations of summation need to be conducted over the entire computational domain. This inner product-like procedure also causes communication across the different PEs.

In our computation, a 1D partition as shown in Fig. 6 was used. Along the z direction, the computational domain is subdivided equally into subdomains and each subdomain is

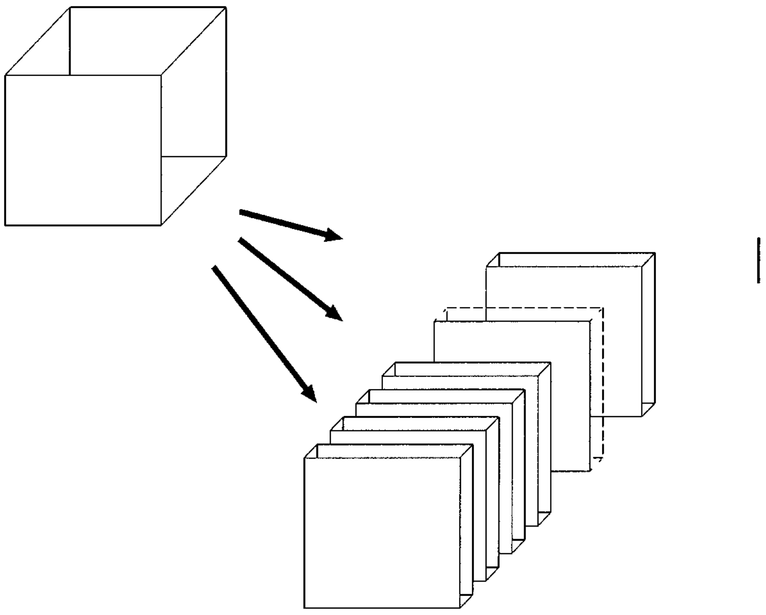

FIG. 6. A 1D partitioning of computational domain. Each block is assigned to one processor element.

assigned to one processor. Data reside separately in different processor elements and the sequence of operations is identical across the processors.

As mentioned before, most of the calculations are conducted by explicit algorithms. For these explicit parts, one needs only transfer the data to the halo cells from neighboring processors, and do the operations separately within each PE. In the present model only one halo cell is required at each interface.

The parallel computing procedure of the Bi-CGSTAB method with a preconditioner of tridiagonal approximation, used in the present model, is briefly discussed below.

For linear system $A\mathbf{x} = \mathbf{b}$, the Bi-CGSTAB algorithm with preconditioning matrix K reads as follows:

```
Set:
    $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, $\mathbf{x}_0$ is the initial guess;
    $\bar{\mathbf{r}}_0 = \mathbf{r}_0$;
    $\rho_0 = \alpha_0 = \omega_0 = 1$;
    $\mathbf{v}_0 = \mathbf{p}_0 = 0$;
for $i = 1, 2, 3, \ldots$
    $\rho_i = (\mathbf{r}_i, \bar{\mathbf{r}}_i)$; $\beta_{i-1} = (\rho_i/\rho_{i-1})/(\alpha_{i-1}/\omega_{i-1})$;
    $\mathbf{p}_i = \mathbf{r}_i + \beta_{i-1}(\mathbf{p}_{i-1} - \omega_{i-1}\mathbf{v}_{i-1})$;
    **SOLVE** $\hat{\mathbf{p}}$ from $K\hat{\mathbf{p}} = \mathbf{p}_i$;
    $\mathbf{v}_i = A\hat{\mathbf{p}}$;
    $\alpha_i = \rho_i/(\mathbf{v}_i, \bar{\mathbf{r}}_0)$;
    $\mathbf{s} = \mathbf{r}_i - \alpha_i\mathbf{v}_i$;
    **SOLVE** $\mathbf{q}$ from $K\mathbf{q} = \mathbf{s}$;
    $\mathbf{u} = A\mathbf{q}$;
    $\omega_i = (\mathbf{u}, \mathbf{s})/(\mathbf{u}, \mathbf{u})$;
    $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \hat{\mathbf{p}} + \omega_i \mathbf{q}$;
    $\mathbf{r}_{i+1} = \mathbf{s} - \omega_i \mathbf{u}$;
    if $\|\mathbf{r}_{i+1}\|/\|\mathbf{b}\| \leq \epsilon_T$ then quit.
end.
```

The convergence tolerance $\epsilon_T = 10^{-10}$ was used in all the calculations. This method shows good convergence behavior, even where the resulting Poisson equation has a large jump in coefficients or a singular source. Even though the number of iterations needed for convergence varies with simulation problems and stages of physical processes, less than 40 iterations in general could reach the tolerance in our calculations.

As four inner product operations are involved for one iteration, global communication is relatively significant. This is the common feature of all the iterative methods based on Krylov subspace. For implementation on a massively parallel environment, it might be necessary to make use of other types of solvers such as the multigrid method.

The preconditioning matrix K is formulated as a tridiagonal approximation in this study. For a linear system arising from a standard finite difference discretization of the Poisson pressure equation, matrix A can be written as

$$A = D + A_x + A_y + A_z,$$

where A_x, A_y, and A_z represent the elements in the x, y, and z directions, respectively.

The corresponding preconditioning matrix K is written as

$$K = \left(D + A_x^L + A_x^U\right) D^{-1} \left(D + A_y^L + A_y^U\right) D^{-1} \left(D + A_z^L + A_z^U\right).$$

Similar to an ADI type algorithm, each component in the x, y, or z direction forms a tridiagonal system. So a complete LU decomposition can easily be cast as

$$K = \left(D_x^{-1} + A_x^L\right)\left(I + D_x A_x^U\right) D^{-1} \left(D_y^{-1} + A_y^L\right)\left(I + D_y A_y^U\right) D^{-1} \left(D_z^{-1} + A_z^L\right)\left(I + D_z A_z^U\right).$$

Thus, the computation of $K\mathbf{x} = \mathbf{y}$ can be carried out as

$$\left(D_x^{-1} + A_x^L\right)\left(I + D_x A_x^U\right)\mathbf{x}_1 = \mathbf{y},$$

(parallelizable in the y and z directions);

$$D^{-1}\left(D_y^{-1} + A_y^L\right)\left(I + D_y A_y^U\right)\mathbf{x}_2 = \mathbf{x}_1,$$

(parallelizable in the x and z directions);

$$D^{-1}\left(D_z^{-1} + A_z^L\right)\left(I + D_z A_z^U\right)\mathbf{x} = \mathbf{x}_2,$$

(parallelizable in the x and y directions).

For 1D partitioning in the z direction, four sweeps of global access for $\mathbf{x}$ are required to change the partition direction per Bi-CG iteration.

6. PRELIMINARY NUMERICAL TESTS

We first computed a solid object undergoing steady translation at a low speed $\mathbf{v}_\mathrm{s}$ in Stokes flows. Some similar samples can also be found in the works of Pan and Banerjee [17, 18]. As discussed in [10], some analytical solutions are available for Stokes flows. Consider a sphere translating with a steady velocity $\mathbf{v}_\mathrm{s}$ in a Stokes flow. The velocity field induced by this moving particle can be expressed by

$$\mathbf{v}(\mathbf{x}) = 3a\mathbf{v}_\mathrm{s} \cdot \left(1 + \frac{a^2}{6}\nabla^2\right)\frac{\boldsymbol{\Psi}(\mathbf{x})}{4}, \tag{69}$$

where a is the radius of the sphere. $\boldsymbol{\Psi}(\mathbf{x})$ is the Oseen tensor, given by

$$\boldsymbol{\Psi}(\mathbf{x})_{ij} = \frac{1}{r}\delta_{ij} + \frac{3}{r^3}x_i x_j. \tag{70}$$

We calculated this problem by setting a particle of radius of only two grid spacings moving downward at a speed $\mathbf{v}_\mathrm{s} = -w_s\mathbf{k}$ in a quiescent flow (as shown in Fig. 7). The Reynolds number $R = 2aw_s/\nu$ is 0.006. This value makes the flow close to a Stokes flow. The calculation was continued until the flow became steady with the frame moving at the speed of the particle. The velocities in the direction parallel to the motion of the particle are plotted in Fig. 8 against the analytical velocities. We observed that the present model produces results very close to the analytical solution. In Fig. 9 we compared the velocity component perpendicular to the velocity of the particle to the analytical velocities at different

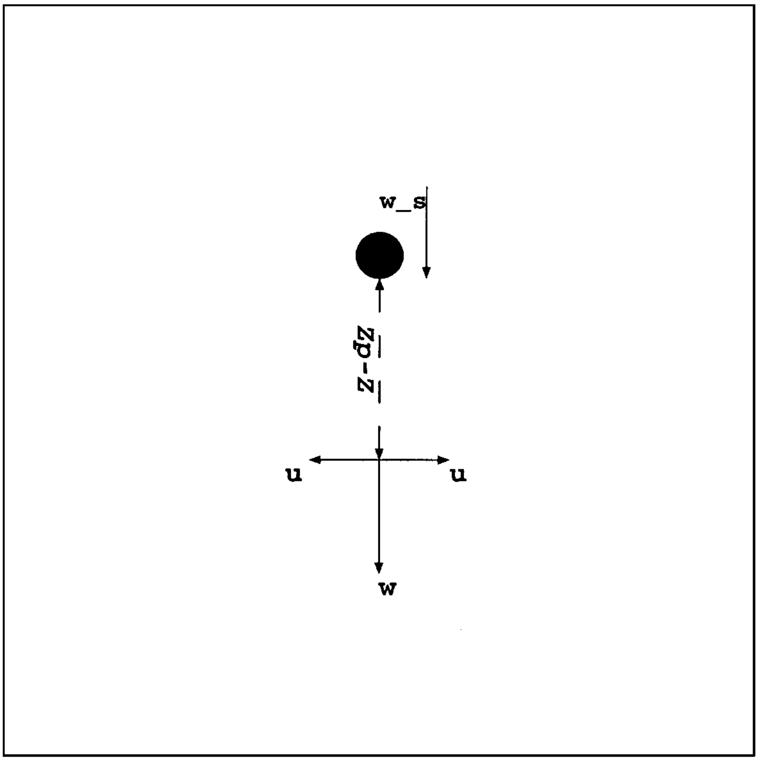

FIG. 7. A steadily moving sphere in Stokes flow.

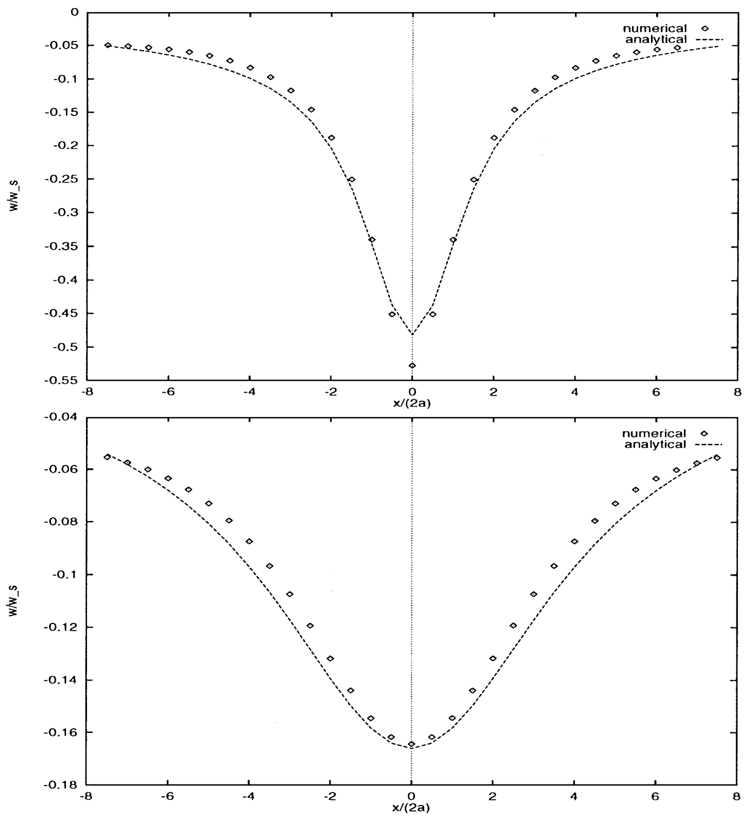

FIG. 8. The velocity component parallel to the moving speed of the particle w_s at distances of $(z_p - z)/(2a) = 1.5$ (upper) and $(z_p - z)/(2a) = 4.5$ (lower) below the particle. z_p is the particle location and a the radius of the particle.

levels apart from the particle, and here as well we found an agreement between the numerical results and the analytical solutions.

The second example we used to validate this model is the drag coefficient of a sphere in a viscous flow. As given in [18], an empirical formula for the drag coefficient of a sphere $C_{\rm d}$ can be written as

$$C_{\rm d} = \frac{24}{R_b} + \frac{6}{1 + \sqrt{R_b}} + 0.4, \qquad 0 \le R_b \le 2 \times 10^5, \tag{71}$$

where $R_b = U d_b / \nu$ is the Reynolds number, U the velocity of the main stream, d_b the diameter of the sphere, and ν the kinematic viscosity.

A sphere with a diameter $d_b = 10 \times$ gridspacing is centered in a $70 \times 70 \times 104$ 3D computational mesh. The numerical outputs of the drag coefficient for different Reynolds numbers are plotted against those predicted with (71) in Fig. 10. We observed that the present model gives reasonable results for a Reynolds number less than 10^3. In the present calculations, the solid body is represented by the color function defined on a rectangular mesh, and no manipulation is used for reconstructing the surface of the solid body. For a viscous flow of low or moderate Reynolds number, the overall fluid flow seems insensitive to the solid surface.

As another test computation, the interactions between a log and the suspending fluid were simulated. Nearly incompressible liquid was put hydrostatically in a tank with gravity pointing downward. A layer of gas laid above the liquid was also computed using numerical

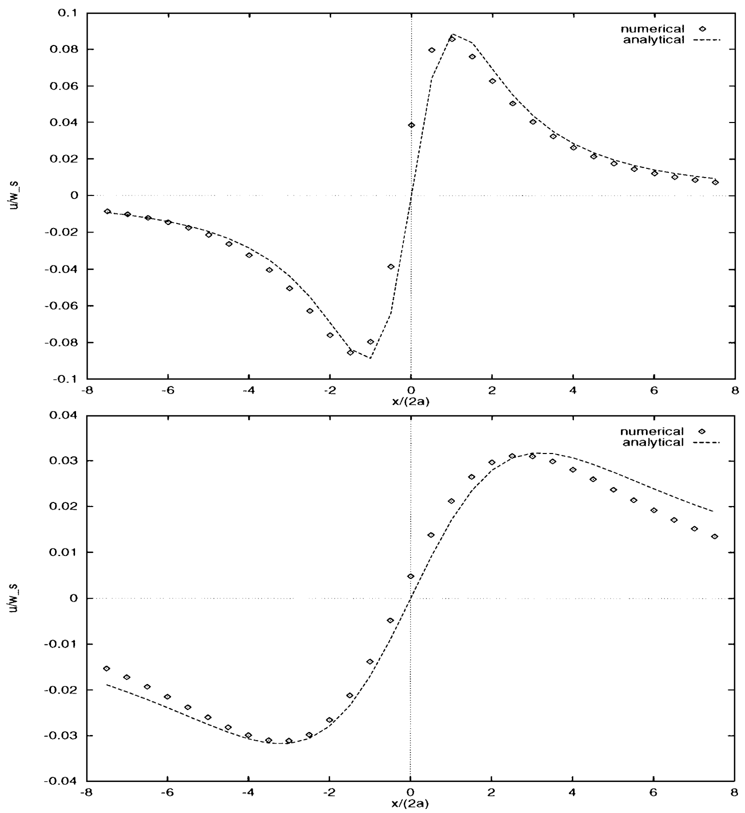

FIG. 9. The velocity component normal to the moving speed of the particle w_s at distances of $(z_p - z)/(2a) = 1.5$ (upper) and $(z_p - z)/(2a) = 4.5$ (lower) below the particle. z_p is the particle location and a the radius of the particle.

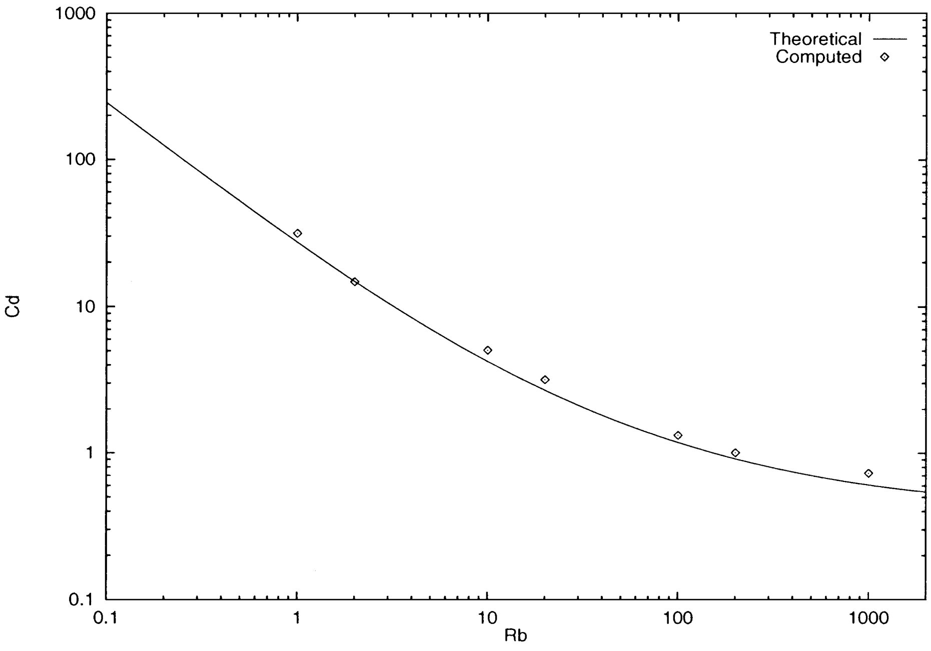

FIG. 10. The drag coefficient of a sphere.

algorithms identical to those for the liquid in a unified procedure, and no extra treatment was needed across the gas/liquid interface (this applies also to the following sample simulation). A log was initially released from above the fluid surface, and fell down through the fluid under the effect of gravity. A 3D mesh of 150^3 was used. Figure 11 shows the time evolution

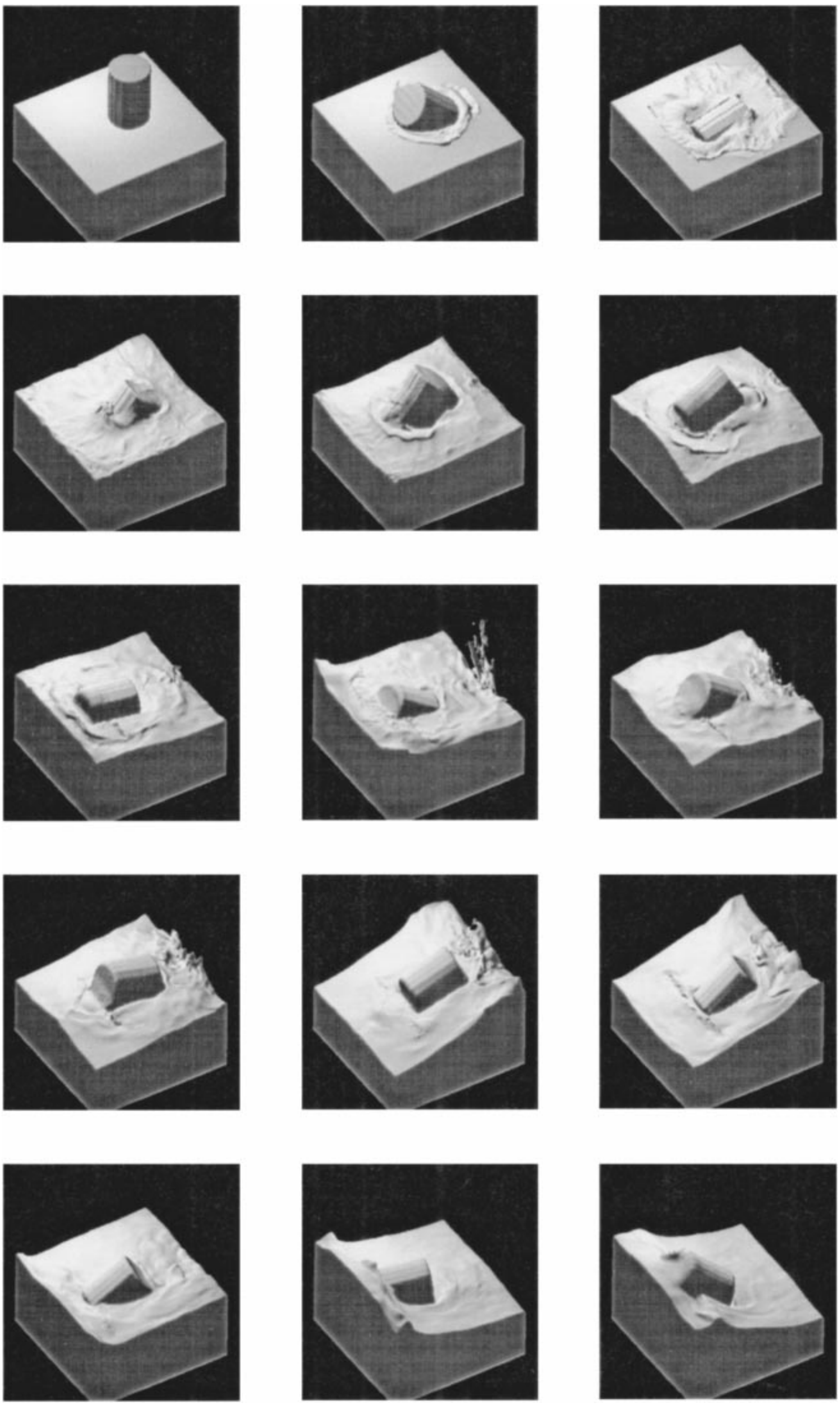

FIG. 11. Time evolves from left to right and from top to bottom at an increment of 0.1. The density ratio of the fluid and the log is $\rho_L : \rho_S = 1.0 : 5.0 \times 1.0^{-1}$.

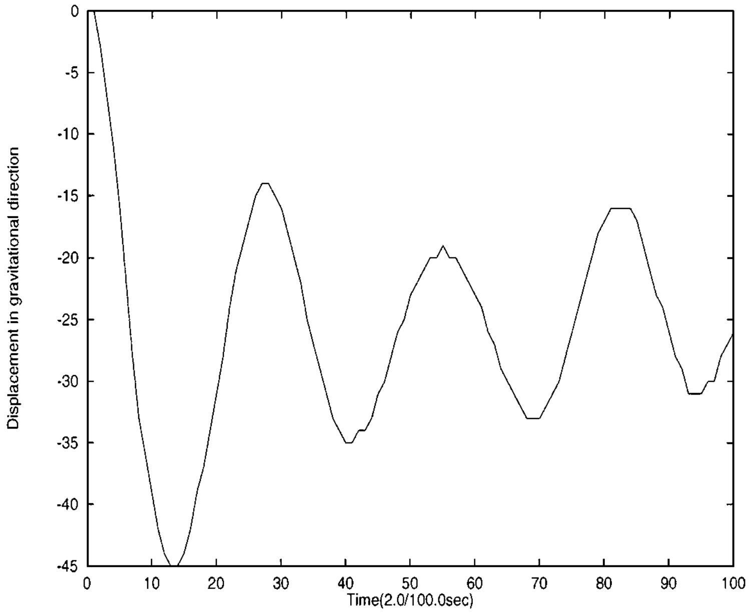

FIG. 12. The displacement in the gravitational direction of the mass center of the log.

of the log and the suspending flow.[1] Fluid motion was caused by the falling log, and then interacted with the log as well as the tank wall. The log changed its motion according to the net force and torque it received from the suspending fluid. The log finally lay in an orientation along the fluid surface. Since the log had a density less than that of the liquid, it experienced an oscillation in the gravity direction under the force of floating. Figure 12 shows the displacement in the gravitational direction of the mass center of the log. Even if the fluid flow appears quite complex, the oscillation period of the mass center experiences no significant change.

As an example of treating complex geometry, we simulated a rotating spherical cage rising from liquid under the force of floating. The cage is a hollow sphere with six holes on the surface as shown in Fig. 13a. It gives a too complex geometry for many numerical methods to deal with. From Fig. 13b, we observe a faithfully presented geometry of the cage. The cage has a density 10% that of the liquid, and rotated initially along the gravity direction. Figure 14 shows the time development of the process.[2] The cage rose from under water and drove out the surrounding fluid. Part of the fluid was carried up by the cage and then leaked out from the holes. The cage then approached its equilibrium state and stayed on the fluid's surface. The simulation result appears reasonable.

As an application to the water entry problem, the present numerical model was used to simulate the impact of a circular disk entering the free surface of fluid in a low-speed regime. The reasons why some animals of moderate size can support their body weight on the water surface were discussed by Glasheen and McMahon [5]. The forces the animals obtain to support themselves are considered to be produced by slapping and stroking the water with their feet. To investigate the hydrodynamic forces of low-speed water entry, Glasheen and McMahon [6] systematically conducted experiments to measure directly the impact and drag

[1] An "mpeg" movie can be found on http://atlas.riken.go.jp/~xiao/3D-SOLIDinLIQUID.

[2] An "mpeg" movie can be found on http://atlas.riken.go.jp/~xiao/3D-SOLIDinLIQUID.

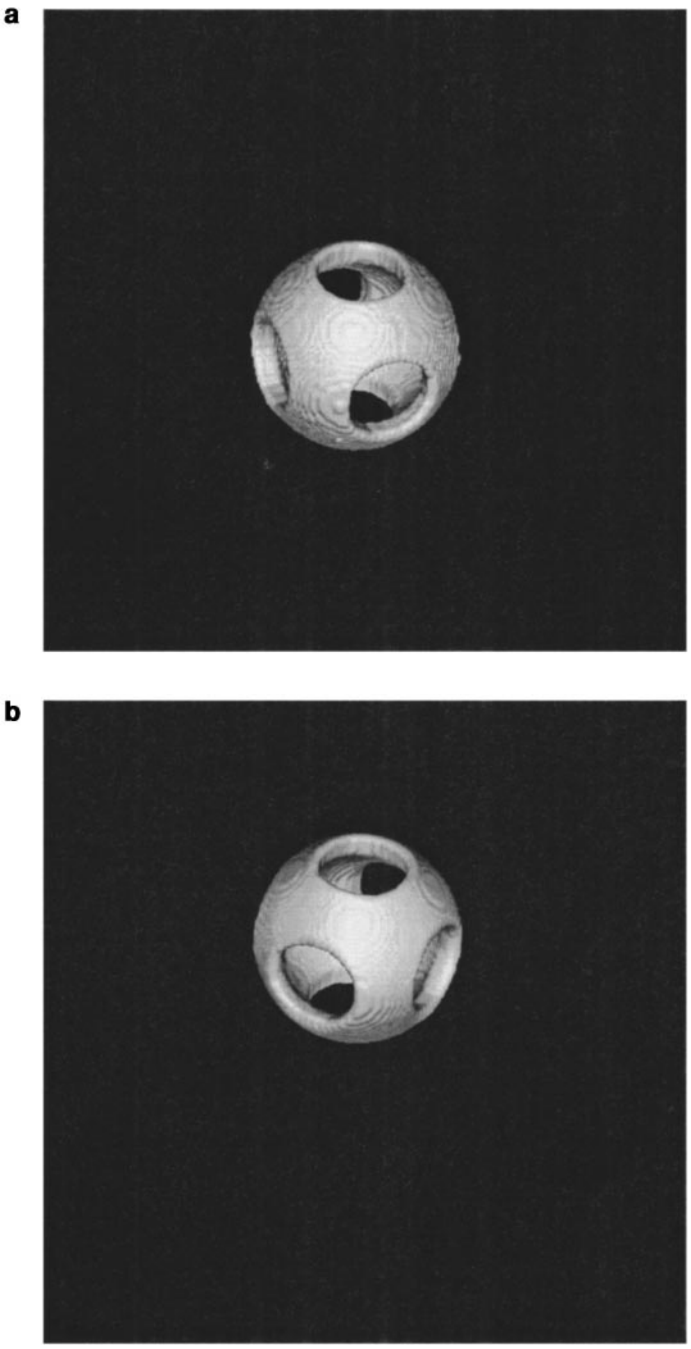

FIG. 13. (a) The initial cage. (b) The cage at the end of the calculation.

forces for circular disks dropped in water at low Froude numbers ($u[\text{disk}]^2/gr = 1 - 80$, where r is the radius of a disk, g the gravitational acceleration, and u[disk] the velocity of the disk).

We simulated the water entry of a circular disk at various low Froude numbers. Similar to the experimental conditions described in [6], two layer fluids with the density of air and water were initially in a hydrostatic balance. A circular disk with a downward impact

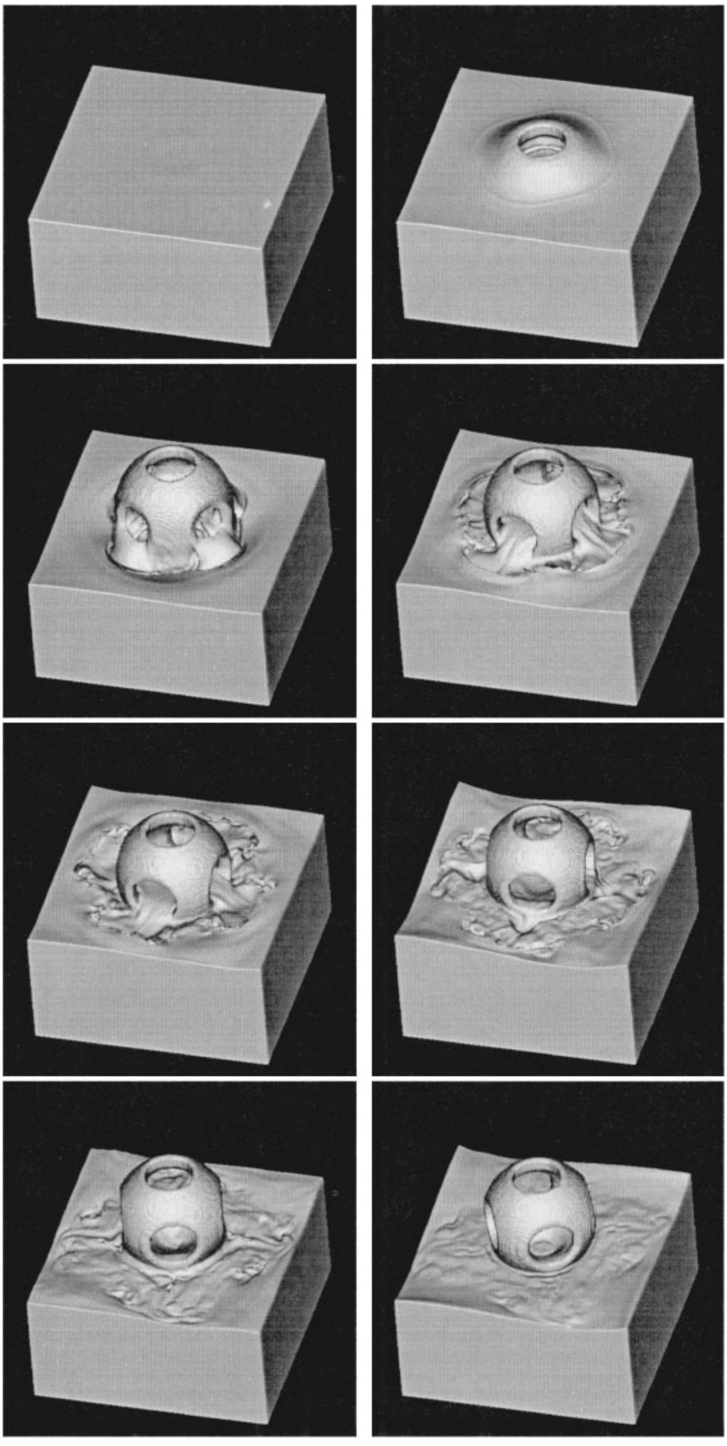

FIG. 14. A spherical cage floating up from under water with an initial rotation along the gravitational direction. Time increases from left to right and from top to bottom.

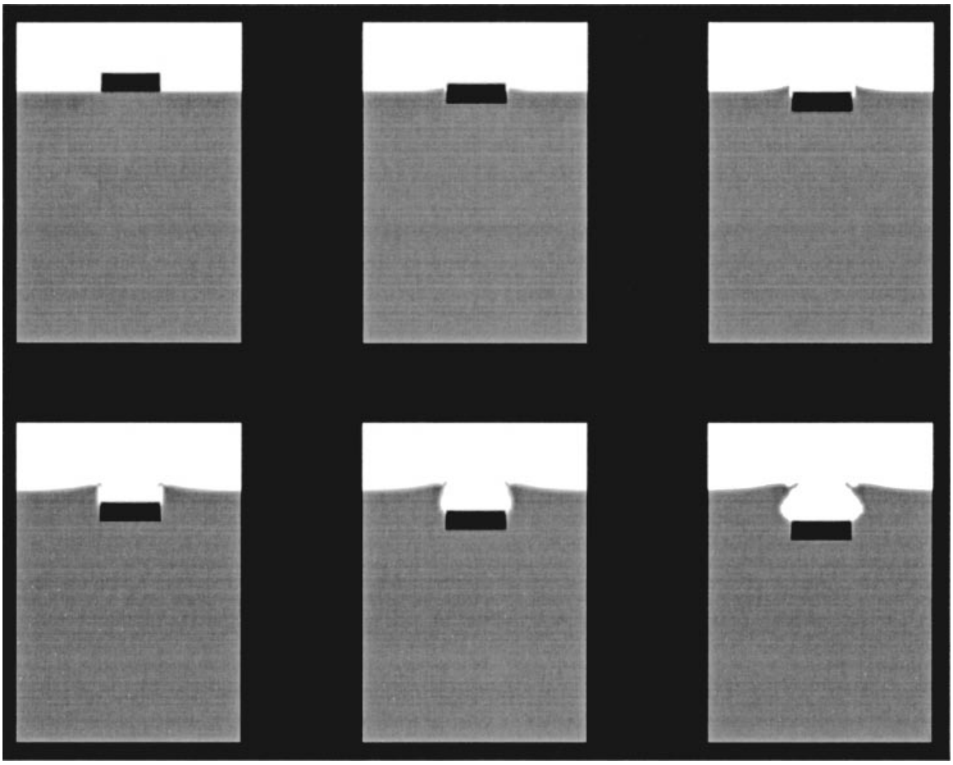

FIG. 15. Water entry of a circular disk at a low Froude number ($u[\text{disk}]^2/gr = 25.51$). Time increases from left to right and from top to bottom at an increment of 3.6×10^{-4} s.

speed ($u[\text{disk}]_{\text{impact}}$) was initially put on the surface of the heavier fluid. Figure 15 is the cross-section views of the disk and the surrounding fluids at a Froude number of 25.51. The disk entering the water drives away the fluid and creates a cavity behind it. When the inertial effects induced by the disk impact are balanced by the pressure in the surrounding water, the fluid will be driven back toward the open cavity and an isolated air bubble can be produced behind the disk.

A sudden change in the vertical velocity of a disk entering the liquid surface, $\Delta u[\text{disk}]$, was observed during the impact. In [6], $\Delta u[\text{disk}]$ was measured for the cases in which the relative change in disk momentum ($\Delta u[\text{disk}]/u[\text{disk}]_{\text{impact}}$) was less than 7% to make the data comparable to the theoretical estimates of Birkhoff and Zarantonello [1].

In order to make a quantitative comparison with the experimental results, calculations were carried out with different Froude numbers by changing disk radius. A density 10 times that of the liquid was used for the circular disk to ensure that the changes in disk velocity during the period of impact were less than 8% in the numerical results. Simulations were conducted with different radii for the circular disks. Figure 16 shows the vertical speed profiles of the disks with an entering speed of 1 m/s. In all the cases, a sudden deceleration in disk velocity occurs during the impact which is dominant in the early stage of the entry. The velocity slows down more significantly as the radius increases.

According to [6], the virtual mass (mass of fluid that accelerates during impact, m_{virtual}) can be calculated as

$$m_{\text{virtual}} = m_{\text{disk}} \Delta u[\text{disk}]/u[\text{disk}]_{\text{final}}, \tag{72}$$

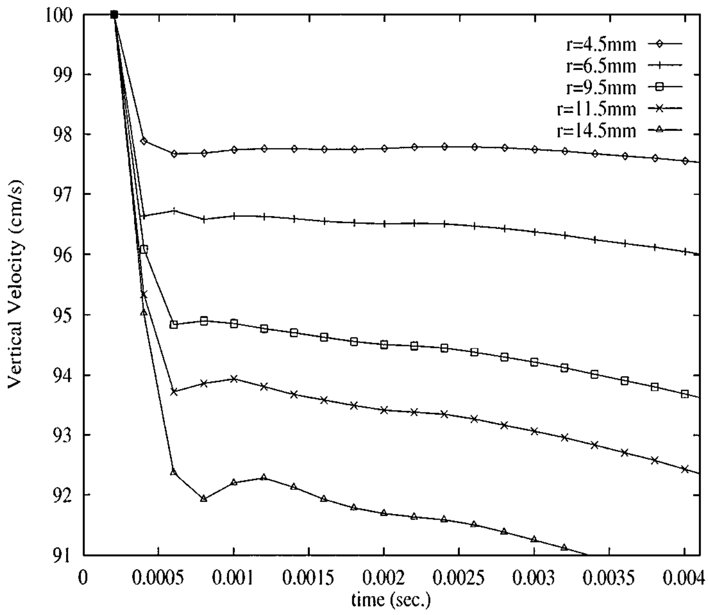

FIG. 16. Downward velocity profiles for different disk radii. All the disks have the same thickness (7 mm) and the same density ($10 \times \rho_{\text{liquid}}$).

where m_{disk} is the disk's mass and $u[\text{disk}]_{\text{final}}$ is computed by

$$u[\text{disk}]_{\text{final}} = u[\text{disk}]_{\text{impact}} - \Delta u[\text{disk}].$$

Then, a dimensionless virtual mass M can be defined as

$$M = \frac{m_{\text{virtual}}}{(4/3)\pi\rho_{\text{liquid}} r^3}. \tag{73}$$

From the experiments, Glasheen and McMahon concluded that the virtual mass of the fluid accelerated during impact increased linearly with $\rho_{\text{liquid}} r^3$ along a slope of 1.42 and the impact impulse rose linearly with $u[\text{disk}]_{\text{final}}$. The dimensionless virtual mass M has a value of 0.34.

As plotted in Fig. 17, our calculated results agree well with Glasheen and McMahon's observation.

We also investigated numerically the disks with lighter mass which are expected to experience larger changes in the falling velocity during impact. A density which is the same as that of the liquid was used for the disk. The impact speed was 2 m/s. The velocity profiles of the disks are plotted in Fig. 18. Similarly, sudden changes in disk velocities are observed. With smaller inertia mass, a lighter disk is decelerated to a larger extent. For the cases in which the change in speed is larger than 20%, we plotted again the virtual mass m_{virtual} in Fig. 19 against $\rho_{\text{liquid}} r^3$. The virtual mass increases along a slope which appears to be less linear and slightly steeper than the slope of 1.42. This preliminary result suggests that a modification to the linear scaling relation between the virtual mass and the product of the mass density of liquid and the cube of the disk radius should be taken in account for an entry in which the falling velocity of a disk is largely changed during impact.

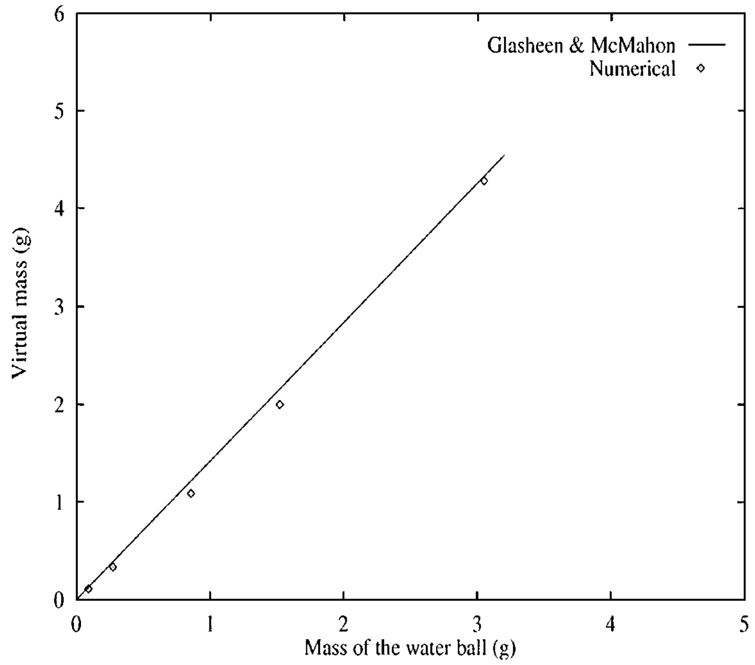

FIG. 17. The virtual mass of a disk, with a radius r, rises as the mass of the water ball ($\rho_{\text{liquid}}r^3$) increases. The symbols are the simulated results. The solid line indicates the 1.42 slope of Glasheen and McMahon.

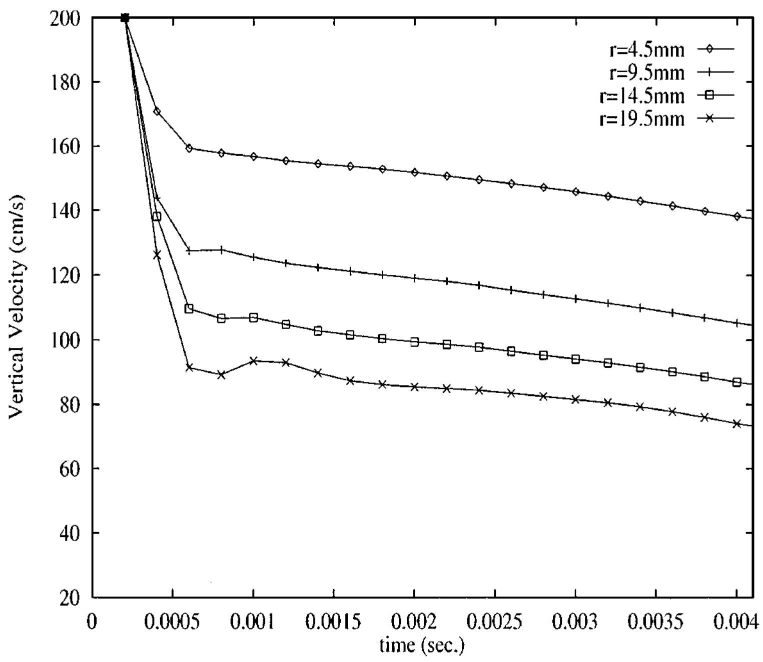

FIG. 18. Downward velocity profiles for different disk radii. All the disks have the same thickness (7 mm) and the same density (ρ_{liquid}).

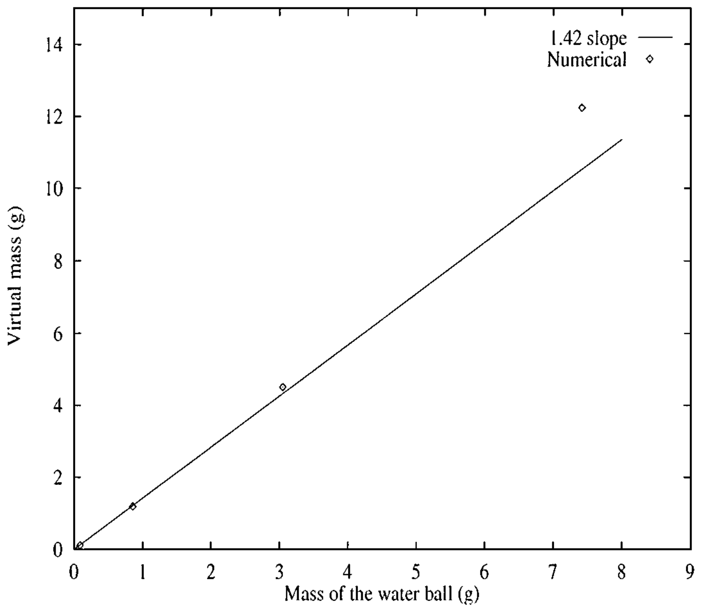

FIG. 19. Same as Fig. 17 but for the disks having a density of ρ_{liquid}.

7. CONCLUSIONS

We have presented a computational model for flows suspending large rigid objects. The moving bodies are represented by color functions and solved by an advection solver that gives geometrically faithful results with transition regions of compact thickness. The pressure Poisson equation is computed over the entire computational domain. Volume forces are then calculated at every grid point, and the net force and torque are obtained by volume integration for all the rigid objects.

The code has been implemented on a parallel environment of distributed memory. It was tested with some solid/fluid suspension problems, and reasonable solutions were thereby obtained.

This model should be applicable to many practical applications, such as floating structure simulation, water entry dynamics, and direct simulation of particulate flow.

ACKNOWLEDGMENTS

The author thanks Professor T. Yabe and Professor T. Ebisuzaki for helpful discussions. The author thanks C. W. Stern for making English corrections to the manuscript. All the computations were carried out at the computer center of the Institute of Physical and Chemical Research (RIKEN), Japan.

REFERENCES

1. G. Birkhoff and E. H. Zarantonello, *Jets, Wakes and Cavities* (Academic Press, New York, 1957).
2. P. Colella and P. R. Woodward, The piecewise parabolic method (PPM) for gas-dynamical simulations, *J. Comput. Phys.* **54**, 174 (1984).
3. S. Doi and N. Harada, Tridiagonal factorization algorithm: A preconditioner for nonsymmetric system solving on vector computers, *J. Inform. Proc.* **11**, 38 (1987).

4. H. Forrer and R. Jeltsch, A higher-order boundary treatment for Cartesian-grid methods, *J. Comput. Phys.* **140**, 259 (1998).
5. J. W. Glasheen and T. A. McMahon, A hydrodynamic model of locomotion in the Basilisk lizard, *Nature* **380**, 340 (1996).
6. J. W. Glasheen and T. A. McMahon, Vertical water entry of disks at low Froude numbers, *Phys. Fluids* **9**, 2078 (1996).
7. C. W. Hirt and B. D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, *J. Comput. Phys.* **39**, 201 (1981).
8. H. H. Hu, Direct simulation of flows of solid–liquid mixtures, *Int. J. Multiphase Flow* **22**, 335 (1996).
9. D. D. Joseph and T. S. Lundgren, Ensemble averaged and mixture theory equations for incompressible fluid particle suspension, *Int. J. Multiphase Flow* **16**, 35 (1990).
10. S. Kim and S. J. Karrila, *Microhydrodynamics: Principles and Selected Applications* (Butterworth–Heinmann, Boston, 1991).
11. D. B. Kothe and W. J. Rider, *Comments on Modeling Interfacial Flows with Volume of Fluid Methods*, Technical Report LA-UR-94-3384 (Los Alamos National Laboratory, 1994).
12. D. B. Kothe, W. J. Rider, S. J. Mosso, and J. S. Brock, Volume tracking of interface having surface tension in two and three dimension, AIAA paper 96-0859 (1996).
13. R. J. LeVeque and Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, *SIAM J. Numer. Anal.* **31**, 1019 (1994).
14. R. M. More, K. H. Warren, D. A. Young, and G. B. Zimmerman, A new quotidian equation of state (QEOS) for hot dense matter, *Phys. Fluids* **31**, 3059 (1988).
15. T. Nomura and T. J. R. Hughes, An arbitrary Lagrangian–Eulerian finite element method for interaction of fluid and a rigid body, *Comput. Methods Appl. Mech. Eng.* **95**, 115 (1992).
16. S. Osher and J. A. Sethian, Front propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations, *J. Comput. Phys.* **79**, 12 (1988).
17. Y. Pan and S. Banerjee, Numerical simulation of particle interactions with wall turbulence, *Phys. Fluids* **8**, 2733 (1996).
18. Y. Pan and S. Banerjee,Numerical investigation of the effects of large particles on wall-turbulence, *Phys. Fluids* **9**, 3786 (1997).
19. E. G. Puckett and J. S. Saltzman, A 3D adaptive mesh refinement algorithm for multimaterial gas dynamics, *Physica D* **60** 84 (1992).
20. G. Strang, On the construction and comparison of difference schemes, *SIAM J. Numer. Anal.* **5**, 506 (1968).
21. M. Sussman, P. Smereka, and S. Osher, A level set approach for computing solutions to incompressible two-phase flow, *J. Comput. Phys.* **114**, 146 (1994).
22. T. Tezduyar, A. Aliabadi, and M. Behr, Enhanced-discretization interface-capturing technique, in *Proceedings, ISAC'97: High Performance Computing on Multiphase Flows, Tokyo, Japan, July 1997*, edited by Matsumoto and Prosperetti, p. 1.
23. S. O. Unverdi and G. Tryggvason, A front-tracking method for viscous, incompressible, multi-fluid flows, *J. Comput. Phys.* **100**, 25 (1991).
24. H. A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* **13**, 631 (1992).
25. F. Xiao, T. Yabe, T. Ito, and M. Tajima, An algorithm for simulating solid objects suspended in stratified flow, *Comput. Phys. Commun.* **102**, 147 (1997).
26. F. Xiao, T. Yabe, G. Nizam, and T. Ito, Constructing a multi-dimensional oscillation preventing scheme for the advection equation by a rational function, *Comput. Phys. Commun.* **94**, 103 (1997).
27. T. Yabe and F. Xiao, Description of complex and sharp interface with fixed grids in incompressible and compressible fluid, *Comput. Math. Appl.* **29**, 15 (1995).
28. D. L. Youngs, Time-dependent multi-material flow with large fluid distortion, in *Numerical Methods for Fluids Dynamics*, edited by K. W. Morton and M. J. Baines (1982), p. 273.
29. D. Z. Zhang and A. Prosperetti, Averaged equations for inviscid disperse two-phase flow, *J. Fluid Mech.* **267**, 185 (1994).