Journal of Computational Physics **174,** 345–380 (2001)
doi:10.1006/jcph.2001.6916, available online at http://www.idealibrary.com on IDEAL®

A Sharp Interface Cartesian Grid Method for Simulating Flows with Complex Moving Boundaries

H. S. Udaykumar,[*] R. Mittal,[†] P. Rampunggoon,[‡] and A. Khanna[*]

[*]*Department of Mechanical Engineering, University of Iowa, Iowa City, Iowa 52242;* [†]*Department of Mechanical and Aerospace Engineering, The George Washington University, Washington, DC 20052; and* [‡]*Department of Mechanical Engineering, University of Florida, Gainesville, Florida 32611*
E-mail: mittal@seas.gwu.edu

Received March 4, 2001; revised July 19, 2001

A Cartesian grid method for computing flows with complex immersed, moving boundaries is presented. The flow is computed on a fixed Cartesian mesh and the solid boundaries are allowed to move freely through the mesh. A mixed Eulerian–Lagrangian framework is employed, which allows us to treat the immersed moving boundary as a sharp interface. The incompressible Navier–Stokes equations are discretized using a second-order-accurate finite-volume technique, and a second-order-accurate fractional-step scheme is employed for time advancement. The fractional-step method and associated boundary conditions are formulated in a manner that properly accounts for the boundary motion. A unique problem with sharp interface methods is the temporal discretization of what are termed “freshly cleared” cells, i.e., cells that are inside the solid at one time step and emerge into the fluid at the next time step. A simple and consistent remedy for this problem is also presented. The solution of the pressure Poisson equation is usually the most time-consuming step in a fractional step scheme and this is even more so for moving boundary problems where the flow domain changes constantly. A multigrid method is presented and is shown to accelerate the convergence significantly even in the presence of complex immersed boundaries. The methodology is validated by comparing it with experimental data on two cases: (1) the flow in a channel with a moving indentation on one wall and (2) vortex shedding from a cylinder oscillating in a uniform free-stream. Finally, the application of the current method to a more complicated moving boundary situation is also demonstrated by computing the flow inside a diaphragm-driven micropump with moving valves. © 2001 Elsevier Science

1. INTRODUCTION

In recent years there has been a renewal of interest in numerical methods that compute flowfields with complex stationary and/or moving immersed boundaries on fixed Cartesian grids. The obvious advantage of these methods over the conventional body-conformal approach is that irrespective of the geometric complexity of the immersed boundaries, the computational mesh remains unchanged. Cartesian grid methods free the underlying structured computational mesh from the task of adapting to the moving boundary, thus allowing large changes in the geometry due to boundary evolution.

Methods that simulate flows with complex immersed boundaries on Cartesian grids can be broadly classified into two categories:

1. In representing the effect of the immersed boundary on the surrounding fluid phase, the immersed boundary is represented as a "diffuse" interface of finite thickness. The thickness of the boundary is usually on the order of the local grid spacing. This category of methods includes most methods that employ body and/or surface forces and/or mass sources in order to represent the effect of the immersed boundary [3, 14, 15, 36, 49, 50] as well as methods which use the volume-of-fluid [6] and phase-field [9, 51, 55] approaches.
2. The boundary is tracked as a sharp interface, either explicitly as curves or as level sets on fixed meshes. The communication between the moving boundary and the flow solver is usually accomplished directly by modifying the computational stencil near the immersed boundary [2, 4, 11, 18, 27, 35, 37, 47, 58].

The methods in the second category share an important property with conventional body-conformal methods in that the boundary is represented as a sharp interface irrespective of the grid resolution. In these sharp interface methods, the interface clearly demarcates two regions of the computational domain and retains the jumps in material and flow quantities as sharp discontinuities. On the other hand, in diffuse interface methods, the boundary is eventually treated as a special region in a single fluid, which occupies the entire computational domain, and the discontinuities across the interface are smoothed. The influence of the boundary in these methods is transmitted to the fluid through source terms in the transport equations. Typically, this boundary effect is distributed over a few mesh cells [3], surface forces are converted to volume forces [6], and jump discontinuities are enforced only in an integral sense. An additional issue present in diffuse interface methods is the presence of parasitic flows, which can be problematic when the source term representing the interfacial effects (such as capillary forces) becomes stiff [56]. In sharp interface methods on the other hand, the effect of the boundary is accounted for through direct application of the appropriate boundary condition(s) on the immersed boundary and parasitic flows are not created [27].

It is important to make a distinction between the methodology used to track the boundary motion and that used to incorporate the influence of the boundary on the fluid phase. There are diffuse interface methods that retain the diffuse nature of the interface both in tracking the boundary as well as in solving the flowfield, such as the volume-of-fluid [6] and phase-field methods [55]. These can be considered as purely Eulerian methods. The level-set method [33] in which the interface is tracked by advecting a distance function is one Eulerian tracking method in which a sharp interface treatment can be devised for boundary representation in solving the flow equations, as in Hou *et al.* [17]. Thus, such level-set–based methods can be considered to be "mixed"; i.e., they possess the characteristics of Eulerian (for tracking) as well as sharp interface (for boundary interaction with the flowfield) approaches. There are other mixed methods in which the boundary is tracked as a sharp, Lagrangian entity, while it is treated as diffuse in accounting for the effect of the immersed boundary on the fluid phase. An example in this category is the immersed boundary method [36, 49]. Thus, boundary tracking and boundary representation must be considered distinct issues in classifying methods for solving moving boundary problems. However, in order to qualify as a true sharp interface method, a method has to track the interface as a sharp entity and also treat it as such when discretizing the flow equations in the presence of the moving immersed boundary.

Methods in all the aforementioned categories have been used successfully for simulating a variety of thermal transport and fluid flow problems including solidification [22, 24, 44, 48], bubble dynamics [42, 47, 49], cell mechanics [12, 23], fluid–structure interaction [41], and complex turbulent and transitional flows [15, 50]. Diffuse interface methods have been the primary choice for solving flow problems with evolving fluid–fluid boundaries. The exception is the immersed interface method applied by Leveque and Li [27] for fluid–fluid boundaries evolving in creeping flows. This method tracks the immersed boundaries as sharp interfaces and accounts explicitly for jump discontinuities at the immersed boundaries in the discretization of the elliptic equations. In the presence of immersed solid boundaries, where boundary layers form on the immersed boundaries, and for flows that are driven primarily by the boundary motion, it is especially important to minimize the discretization error near the boundaries. For such flows, sharp interface methods have an advantage since one source of error, namely that incurred in boundary representation, is eliminated. Additionally, the boundary conditions are applied exactly as in body-fitted grid formulations and thus there is no spreading of the interface effects.

The advantage of representing the immersed boundary as a sharp interface in solidification problems has been clearly demonstrated in Udaykumar *et al.* [47] where a finite-difference, sharp interface, Cartesian grid method was developed for simulating the evolution of solid–fluid phase boundaries driven by diffusion of heat. Complex, dendritic crystal structures were computed and a careful analysis of the errors accruing during the calculations was performed. It was demonstrated there that the field equations were computed to second-order accuracy while the interface evolution was captured with first-order accuracy. Despite the use of a finite-difference formulation, which does not explicitly conserve fluxes, it was found that the interface dynamics was simulated in an accurate manner and this was attributed to the dominance of diffusion in the process and to the sharp representation of the interface. Following this, in Ye *et al.* [58], a finite-volume-based, sharp interface Cartesian grid method which was designed to simulate convection-dominated flows with complex, stationary, immersed boundaries was presented. The switch to a finite-volume technique was prompted by the need to compute accurately the transport of mass and momentum in thin boundary layers that formed on the immersed boundaries in these flows. It was demonstrated that the flow was computed to second-order-accuracy in space and the solution procedure was validated by simulating a number of different flows and comparing with available experimental and computational results.

In the method presented in this paper, the Cartesian grid method of Ye *et al.* [58] is extended in order to allow for the motion of the immersed boundaries. Thus, the spatial and temporal discretization scheme in the current method is for the most part, identical to that presented in Ye *et al.* [58]. The moving boundary is represented as a sharp interface using an Eulerian–Lagrangian approach and the interface tracking procedure is adopted from Udaykumar *et al.* [48]. However, representation of a moving immersed boundary as a sharp interface in these flow simulations leads to some unique problems in terms of accuracy, complexity, and conditioning of discrete operators, and these have to be addressed in order to develop a method which is robust and accurate. Discussion of these issues and validation of the numerical results against experiments forms the subject of this paper.

The first issue that emerges is the proper implementation of the fractional-step method and associated boundary conditions in the context of the current sharp interface method. It is found that the correct splitting of the Navier–Stokes equations for the current approach is similar to that which would be employed in a purely Lagrangian method. The pressure boundary condition, which is required for the solution of the pressure Poisson equation, is also reformulated in a manner consistent with the Lagrangian nature of the interface. One problem that is unique to all sharp interface methods is the appearance of what can be termed as "freshly cleared" cells. These are cells that were in the solid at one time step but emerge into the fluid at the next time step as a result of the boundary motion. These cells do not have any history in the fluid and time derivatives for these cells cannot be constructed in a straightforward manner [5, 28]. This situation is not encountered in Lagrangian methods since the grid is confined to the fluid region and moved as the boundary moves, so that no computational points lie inside the solid at any time. However, in Lagrangian methods, upon grid adaptation, computational points do change position, and the field is then projected from the old grid locations to the new. The information at these newly introduced points is drawn from old computational points lying only in the fluid. The freshly cleared cell situation is also not encountered in diffuse interface methods since there is no clear distinction between the cells on either side of the immersed boundary and all cells have a well-defined, continuous, albeit smoothened, time history. Thus, for sharp interface methods, a systematic and consistent discretization procedure needs to be developed for such cells which change from solid to fluid; this will be discussed in the current paper.

This paper also addresses the issue of speedup of the pressure Poisson equation (PPE) solver in the presence of arbitrary moving boundaries on the fixed mesh. In Ye *et al.* [58] a Line-SOR preconditioned BiCGSTAB (biconjugate gradient stabilized) algorithm was used and was found to be adequate for complex stationary boundary problems. However, for moving boundary calculations, the convergence acceleration of this method was found to be inadequate. The multigrid method, which is most effective for speedup of the elliptic PPE, would seem a logical choice for the current structured grid solver. However, straightforward implementation of the multigrid method requires the reconstruction of the immersed boundary at every coarse multigrid level, which can significantly increase the complexity of the multigrid scheme. We overcome this problem by using a volume–fraction approach to discretize the Poisson equation at the coarse level, while retaining a sharp interface at the finest level.

The applications targeted with this method are wide-ranging and include fluid–structure interaction, multiphase flows, solidification dynamics, and cell mechanics. However, in the current paper, the solver is validated by simulating two flows, both of which involve moving solid boundaries. The computed results are then compared with available experimental and numerical data. In addition, a case with multiple moving solid boundaries is simulated in order to demonstrate the capabilities of the current numerical method.

2. THE NUMERICAL METHOD

The key aspects of the algorithm include:

1. A fractional-step scheme [10, 13], which results in a fast solution of unsteady flows.
2. Adoption of a compact interpolation scheme [58] near the moving immersed boundaries, which allows us to retain second-order-accuracy and conservation property of the solver.
3. A full approximation storage multigrid technique [7, 52] with line-SOR smoothing, which substantially accelerates the convergence of the pressure Poisson equation (PPE) with/without immersed boundaries in the domain.

These aspects are described in detail in the following sections.

2.1. Governing Equations and Flow Configuration

The schematic in Fig. 1 shows a solid with a curved boundary moving through a fluid, which illustrates the typical flow problem of interest here. The equations solved are the incompressible Navier–Stokes equations. The nondimensionalized, integral form of these equations is given by

$$\int_S \vec{u} \cdot \hat{n}\, dS = 0 \tag{1}$$

$$\mathrm{St}\frac{\partial}{\partial t}\int_V \vec{u}\, dV + \int_S \vec{u}\,(\vec{u}\cdot\hat{n})\, dS = -\int_S p\hat{n}\, dS + \frac{1}{\mathrm{Re}}\int_S \nabla\vec{u}\cdot\hat{n}\, dS, \tag{2}$$

where $\vec{u}$ and p are the nondimensional velocity and pressure, respectively; St and Re are the Strouhal number and Reynolds number, respectively, which are defined as $\mathrm{St} = \omega L \big/ U_o$; and $\mathrm{Re} = U_o L/\nu$, where ω is an imposed frequency, L the length scale, U_o the velocity scale, and ν the kinematic viscosity. In the above equations, subscript V and S denote the volume

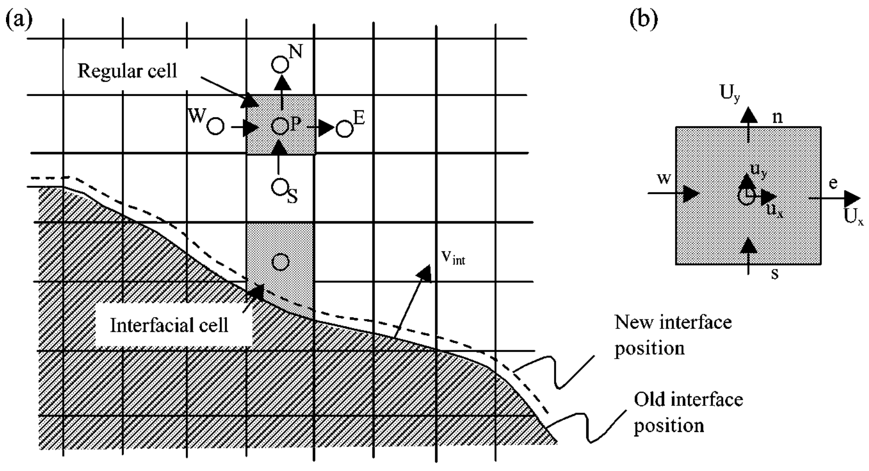

FIG. 1. (a) Illustration of a moving boundary cutting through a fixed mesh. Cells traversed by the interface are called interfacial cells and are trapezoidal in shape. Cells away from the interface are regular cells. (b) A regular cell showing the cell-face nomenclature and cell-center and cell-face velocities.

and surface of the control volume and $\hat{n}$ is a unit vector normal to the surface of the control volume. The above equations are to be solved with $\vec{u}(\vec{x}, t) = \vec{u}_\partial(\vec{x}, t)$ on the boundary of the flow domain where $\vec{u}_\partial(\vec{x}, t)$ is the prescribed boundary velocity, including that at the moving immersed boundary. The above equations with the moving immersed boundary are to be discretized and solved on a Cartesian mesh shown in Fig. 1. The discretization of the above equations in the context of a stationary immersed boundary is described first. With this as the basis, the discretization scheme in the presence of moving boundaries is described subsequently.

2.2. Flow Solver with Stationary Immersed Boundaries

As the first step, the curved immersed boundary is represented using marker particles which are connected by piecewise quadratic curves parameterized with respect to the arclength s. Details regarding interface representation, evaluation of derivatives along the interface to obtain normals, curvatures, and so forth, have been presented in previous papers [48, 58] and are not repeated here. Also described in earlier papers are details regarding the projection of the immersed boundary onto the underlying fixed Cartesian mesh. This includes determining the intersection of the boundary with the mesh; identifying the phase (solid or fluid) of each cell; and determining procedures for obtaining a mosaic of control volumes, which are clearly demarcated by the immersed boundary. This results in the formation of control volumes adjacent to the immersed boundary that are trapezoidal in shape, as shown in Fig. 1. Depending on the location and local orientation of the immersed boundary, trapezoidal cells of varying aspect ratio are formed. It should be pointed out that due to the cell-merging operation [58], the nominal aspect ratio of the trapezoidal cells is limited to a range between 0.5 and 1.5, which is advantageous from the point of view of conditioning of the discrete operators. With the boundary-adjacent grid cells reconstructed in this manner, we now turn to the discretization of the governing Eqs. (1) and (2) on this grid.

A two-step, mixed explicit–implicit fractional step scheme [31] is used for advancing the solution of the above equations in time. The Navier–Stokes equations are discretized using a cell-centered, colocated (nonstaggered) arrangement [39, 59] of the primitive variables $(\vec{u}, p)$. In addition to the cell-center velocities, which are denoted by $\vec{u}$, face-center velocities $\vec{U}$are also computed. In a manner similar to a fully staggered arrangement, only the component normal to the cell-face is computed and stored (see Fig. 1b). The face-center velocity is used for computing the volume flux from each cell in the current finite-volume discretization scheme. The advantage of separately computing the face-center velocities has been discussed in the context of the current method in [58]. The solution is advanced from time level t to $t + \Delta t$ through an intermediate advection–diffusion step where the momentum equations without the pressure gradient terms are first advanced in time. A second-order Adams–Bashforth scheme is employed for the convective terms, and the diffusion terms are discretized using an implicit Crank–Nicolson scheme. This eliminates the viscous stability constraint, which can be quite severe in simulation of viscous flows. The discretized form of the advection–diffusion equation for each cell shown in Fig. 1 can therefore be written as

$$\mathrm{St}\,\frac{\vec{u}^* - \vec{u}^t}{\Delta t}\Delta V = -\frac{1}{2}\sum_f [3\,\vec{u}^t(\vec{U}^t\cdot\hat{n}_f) - \vec{u}^{t-\Delta t}(\vec{U}^{t-\Delta t}\cdot\hat{n}_f)]\Delta S_f + \frac{1}{2\,\mathrm{Re}}\sum_f [\nabla u^* + \nabla u^t]\cdot\hat{n}_f\Delta S_f, \tag{3}$$

where $\vec{u}^*$ is the intermediate cell-center velocity and subscript f denotes one face of the control volume. This equation is solved with the final velocity imposed as the boundary condition; i.e., $\vec{u}^*_\partial(\vec{x}) = \vec{u}_\partial(\vec{x}, t + \Delta t)$. The intermediate face-center velocities $\vec{U}^*$ are obtained at this point by interpolating the intermediate cell-center velocities $\vec{u}^*$.

The advection–diffusion step is followed by the pressure–correction step in which the following integral equation is discretized:

$$St \int_V \frac{\vec{u}^{t+\Delta t} - \vec{u}^*}{\Delta t} dV = -\int_V \nabla p^{t+\Delta t} dV. \tag{4}$$

By requiring a divergence-free velocity field at the end of the time step the following elliptic equation for pressure is obtained:

$$\sum_f \nabla p^{t+\Delta t} \cdot \hat{n}_f \Delta S_f = \frac{\mathrm{St}}{\Delta t} \sum_f \vec{U}^* \cdot \hat{n}_f \Delta S_f. \tag{5}$$

With stationary, nonporous boundaries, a homogeneous Neumann boundary condition for pressure results in a consistent approximation of the Navier–Stokes equations [43]. Once the pressure is obtained by solving Eq. (5), both the cell-center and face-center velocities, $\vec{u}$ and $\vec{U}$, are updated separately as

$$\vec{u}^{t+\Delta t} = \vec{u}^t - \Delta t (\nabla p^{t+\Delta t})_{cc} \tag{6}$$

$$\vec{U}^{t+\Delta t} = \vec{U}^t - \Delta t (\nabla p^{t+\Delta t})_{fc}, \tag{7}$$

where subscripts *cc* and *fc* indicate evaluation at the cell-center and face-center locations, respectively. Further discussion regarding the adoption of cell-center and face-center velocities can be found in Zang *et al.* [59] and in the context of the present method in Ye *et al.* [58].

The key element in the finite-volume discretization of Eqs. (3)–(5) in the context of the current method is the evaluation of fluxes and derivatives at the faces of each control volume. These include momentum, mass, and diffusive fluxes and gradients of pressure. A detailed discussion of this aspect, including validation of the accuracy of the solution procedure, has been presented in Ye *et al.* [58]. For the regular Cartesian cells away from the immersed boundary, the fluxes and pressure gradients on the face centers can be computed to second-order accuracy by assuming a linear variation between adjoining cell centers. This is not the case for a trapezoidal boundary cell since the center of some of the faces of such a cell may not lie halfway between neighboring cell centers. This is seen from Figs. 2b and 2c, where the locations where fluxes are evaluated are indicated by the filled arrows. A linear approximation would not provide a second-order-accurate estimate of the gradients. Furthermore, some of the neighboring cell centers do not even lie on the same side of the immersed boundary and therefore cannot be used in the differencing procedure. Thus, a different approach is needed in order to discretize the equations in these cells.

To maintain second-order-accurate discretization in the boundary cells [58], we employ a compact two-dimensional polynomial interpolating function which allows us to obtain the fluxes and gradients on the cell faces of the trapezoidal to second-order-accuracy. For

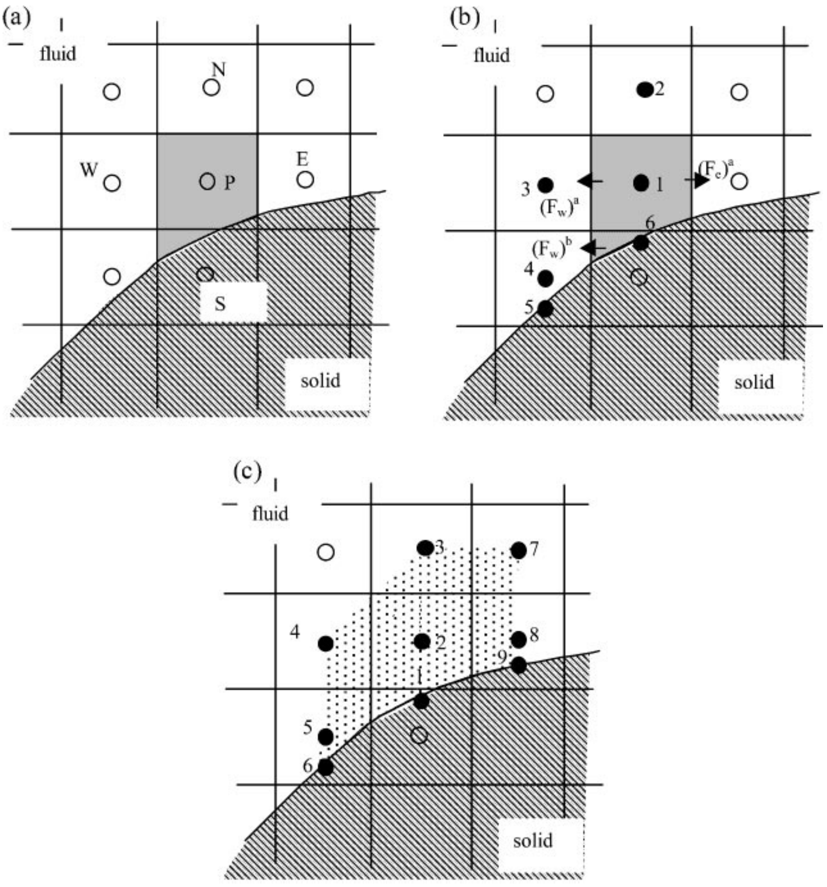

FIG. 2. Illustration of stencils for evaluation of cell face fluxes. (a) Interfacial cell nomenclature showing flux components required in the discrete form of the conservation laws. F represents the flux (convective/diffusive) at each cell face. (b) Stencil points for linear–quadratic interpolation to obtain the flux F_w and F_e. (c) All the stencil points used to calculate fluxes for the control volume P.

instance, for a typical trapezoidal cell shown in Fig. 2a the interpolating function for a generic variable ϕ has the form

$$\phi = c_1 x y^2 + c_2 y^2 + c_3 x y + c_4 y + c_5 x + c_6. \tag{8}$$

The unknown coefficients in this interpolant can be expressed in terms of surrounding nodal and boundary values. Using this, the fluxes and gradients on the cell faces can also be expressed in terms of the neighboring nodal and boundary values. For instance, for the lower portion of the west face of the trapezoidal boundary cell shown in Fig. 2b, the value and gradient of ϕ at the face center can be expressed as

$$\phi_w^b = \sum_{j=1}^{6} \alpha_j \phi_j \quad \text{and} \quad \left(\frac{\partial \phi}{\partial x}\right)_w^b = \sum_{j=1}^{6} \beta_j \phi_j, \tag{9}$$

where the coefficients α and β depend on the geometry of the cell and couple the value at the face center to four nodal and two boundary values. Similar expressions can be constructed for the fluxes on the other faces of the trapezoidal boundary cells. These expressions are incorporated into the discrete representation of Eqs. (3)–(5) and the final discrete equation for any given cell P is of the form as

$$\sum_M A_M^P \, \phi_M = R^P, \tag{10}$$

where ϕ is the variable under consideration (velocity or pressure), R is the source term for the corresponding equation, M is the size of the stencil, and the As are the known coefficients that depend on the geometry of the cell and other flow parameters. For a regular Cartesian cell $M = 5$, whereas for a trapezoidal boundary cell $M = 9$. A typical 9-point stencil for a boundary cell is shown in Fig. 2c. Furthermore, as the stencil in Fig. 2c indicates, the boundary conditions are directly incorporated into the flux calculation procedure. Cells that lie inside the solid are treated within the framework of Eq. (10) simply by putting $R^P \equiv 0$ and zeroing out all the coefficients on the left-hand side of Eq. (10) that couple the value of cell P with the neighboring values.

This interpolation scheme coupled with the finite-volume formulation guarantees that the accuracy and conservation property of the underlying algorithm are retained even in the presence of arbitrary-shaped immersed boundaries [58]. As pointed out earlier, in convection-dominated flows relatively thin boundary layers are expected to be generated in the vicinity of the immersed boundary. These boundary layers not only are regions of high gradients but often are the most important features of the flow field. Therefore, accurate representation of the conservation laws is especially important in this region. The combination of a finite-volume approach and a locally second-order discretization that is employed here is therefore well suited to such flows. This method has now been extended to include moving boundaries, and the modification and additions in the algorithm required to accomplish this are described in the following sections.

2.3. Flow Solver with Moving Immersed Boundaries

The objective in the following sections is to describe the Cartesian grid methodology in the presence of moving solid boundaries. The first element in such cases is the determination of the boundary motion and the procedure for coupling the boundary motion with the fluid flow. As mentioned before, the immersed boundary is defined by “marker particles” distributed on the boundary surface with a spacing which is of the same order of magnitude as the grid spacing. Translating each marker particle with a prescribed velocity produces boundary motion. Subsequently, at any time instant, for a given location of these marker particles, a smooth representation of the entire boundary can be constructed by fitting piecewise quadratic polynomials through these particles.

As in the stationary immersed boundary case, a mixed-explicit scheme is used for time advancement of the governing equations where the convection terms are treated explicitly and the viscous terms implicitly. In cases where there is a two-way interaction between the flow and the moving boundary, a choice also needs to be made regarding the treatment of this coupling. One choice is explicit treatment where the boundary motion and the time advancement of the flow equations are carried out in a sequential manner. The alternative is implicit treatment where the boundary and flow are advanced in time simultaneously in a fully coupled manner. The primary advantage of the implicit approach is that it removes any stability constraints associated with the boundary motion [18, 46]. This can be crucial in problems where the boundary motion is highly sensitive and closely coupled to the flowfield such as in curvature-driven solidification and capillarity-driven flows. In fact, in the previous work of Udaykumar *et al.* [48], which focused on using a Cartesian grid method for solving diffusion controlled dendritic growth, implicit coupling was employed and this resulted in a robust solution technique. Implicit coupling, however, provides no significant advantage when the boundary motion is prescribed independent of the flowfield since in this case, the boundary motion is not subject to errors that can grow in time. In fluid–structure interaction problems, where the boundary motion depends on the flow (for instance, in flow-induced vibrations), explicit coupling results in a convective-type stability constraint on the boundary motion, which can be relieved by employing an implicit treatment of the boundary motion. However, since an explicit scheme is being used here for the convective terms in the transport equations, there is no significant advantage to using implicit coupling between the boundary motion and the fluid flow. In the current methodology an explicit boundary update is therefore employed. However, as shown in Udaykumar *et al.* [48], if needed, a predictor–corrector approach can be employed to implement an implicit coupling in a straightforward manner.

The first step in the time-advancement procedure is to update the location of the boundary. At any given time t, the immersed boundary is denoted by $\vec{x} = \vec{\psi}(s, t)$, where s is the arc length along the boundary measured from a reference point. This is accomplished by advecting each marker particle with the prescribed velocity as

$$\vec{X}_i(t + \Delta t) = \vec{X}_i(t) + \Delta t \vec{u}_i(t + \Delta t), \tag{11}$$

where i corresponds to the index of the marker particle and $\vec{u}_i$ is the velocity of this marker particle. The updated position of the marker particles is then used to reconstruct the boundary at time $(t + \Delta t)$. Note that in advancing the boundary, the boundary velocity at $(t + \Delta t)$ is used. As will be pointed out later, this allows consistency between the boundary velocity and the velocity boundary condition for the fluid.

With the boundary location at $(t + \Delta t)$ now known, the discretized advection–diffusion equation, Eq. (3), can be rewritten as

$$\begin{aligned} \mathrm{St}\left(\frac{\vec{u}^* - \vec{u}^{t+\Delta t}}{\Delta t}\right)\Delta V^{t+\Delta t} &= -\sum_f \frac{1}{2}[3\vec{u}^t(\vec{U}^t \cdot \hat{n}^{t+\Delta t}) - \vec{u}^{t-\Delta t}(\vec{U}^{t-\Delta t} \cdot \hat{n}^{t+\Delta t})]_f \Delta S_f^{t+\Delta t} \\ &\quad + \frac{1}{2\,\mathrm{Re}} \sum_f [(\nabla \vec{u}^t + \nabla \vec{u}^{t+\Delta t}) \cdot \hat{n}^{t+\Delta t}]_f \, dS_f^{t+\Delta t}, \end{aligned} \tag{12}$$

where the superscript $t + \Delta t$ on the cell volume (ΔV), surface areas (ΔS), and normals $(\hat{n})$ indicates that the values corresponding to the boundary location at time level $t + \Delta t$, i.e., $\vec{x} = \vec{\psi}(s, t + \Delta t)$, are used in advancing the advection–diffusion equation from t to $t + \Delta t$. Equation (5) for the pressure is reformulated as

$$\sum_f [\nabla p^{t+\Delta t} \cdot \vec{n}^{t+\Delta t}]_f \Delta S_f^{t+\Delta t} = \frac{\mathrm{St}}{\Delta t} \sum_f [\vec{U}^* \cdot \hat{n}^{t+\Delta t}]_f \Delta S_f^{t+\Delta t}. \tag{13}$$

As before, the summations in the above equations run over the sides of the given computational cell. Subsequently, the pressure correction is added to the intermediate velocity as shown in Eqs. (6) and (7).

In keeping with the stationary boundary algorithm, the advection–diffusion equation (12) is solved with a boundary condition corresponding to the final velocity, i.e., $\vec{u}_\partial(\psi(s, t + \Delta t),\ t + \Delta t)$. This boundary condition is consistent with the boundary motion since the boundary is also moving at this same velocity as shown in Eq. (11). Therefore, the no-slip, no-penetration condition is properly imposed at every time step even in the moving boundary case.

The pressure boundary condition also needs to be reformulated for the moving boundary case. For stationary immersed boundaries, $\partial p/\partial n = 0$ is applied on the immersed boundary. This boundary condition is consistent with the inviscid nature of the pressure correction step and is found to work adequately in most cases. In the moving boundary case, the corresponding pressure boundary condition can be obtained from projecting the inviscid momentum equation in a direction normal to the boundary. This gives

$$\frac{\partial p}{\partial n} = -\left(\mathrm{St}\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u}\right) \cdot \hat{n}$$

as the boundary condition for pressure. A convenient means of implementing this boundary condition in the current solver comes from recognizing that the boundary condition can be recast as

$$\frac{\partial p}{\partial n} = -\mathrm{St}\frac{D\vec{u}}{Dt} \cdot \hat{n}.$$

The material derivative of the velocity (denoted byD/Dt) can then be approximated directly from the known boundary velocities and this obviates the approximation of the convective term. For small boundary acceleration, which corresponds to $\mathrm{St} \ll 1$, this term causes little deviation from the homogeneous Neumann condition for pressure.

In the present framework, when stationary solid boundaries are embedded in the domain, as with any pressure correction methodology, explicit mass conservation is enforced over the domain boundaries. When deformable solid boundaries are present inside the computational domain, the mass conservation enforced at the domain boundaries should take account of the net mass flux at the moving boundaries caused by the boundary deformation. The net mass flux arising at the moving interfaces is given by

$$\dot{m}_{\mathrm{int}} = \sum_{j=1}^{nb} \int_{S_j} \rho \vec{u}_\partial \cdot \hat{n}\, dS, \tag{14}$$

where *nb* is the number of immersed boundaries and the integral is performed over the surface of the immersed boundary. At each step, the mass deficit over the domain boundaries is evaluated as follows:

$$\dot{m}_{deficit} = \sum_{j=1}^{ninlets} \dot{m}_j - \sum_{j=1}^{noutlets} \dot{m}_j - \dot{m}_{\mathrm{int}}. \tag{15}$$

This mass deficit is distributed evenly at the designated outflow boundary through adjustment of the intermediate velocity boundary condition. In the context of the current paper, this deficit correction is applied in the problems involving the moving indentation and the diaphragm-driven pump as presented under Results, Sections 3.2 and 3.4, respectively. In the case of the oscillating cylinder problem in Section 3.3, no net efflux of mass results at the moving boundary and hence the moving boundary does not have any impact on global mass conservation.

It is worth pointing out that the form of Eqs. (12) and (13) is virtually identical to (3) and (5). The primary difference is that for the trapezoidal boundary cells, the cell volume, surface area, and directions of the surface normals are now functions of time. Furthermore, the boundary conditions have to be reformulated in order to account for the time-dependant motion of the immersed boundary. However, unlike Lagrangian methods, time derivatives of the cell volume and surface area do not appear in the equation. In this respect, the simplicity of the Eulerian approach is retained. In the context of the current method, this implies that the discretization of Eqs. (12) and (13) at any given time step is virtually identical to the stationary boundary case. Thus, the spatial discretization methodology described in Section 2.2 can be used even with moving boundaries. The primary difficulty in the moving boundary algorithm comes from the appearance of "freshly cleared" cells (this issue is discussed in the next section).

2.4. "Freshly Cleared" Cells

In sharp interface methods, such as the present one or those presented by Bayyuk *et al.* [4] and Leveque and co-workers [5, 27], the issue of change of material needs to be addressed. This arises when a computational point (as in Fig. 3), which was in the solid at one time step, emerges into the fluid at the next time step. In Leveque and Li [27], the issue of a discontinuity in time at cross-over is dealt with by applying a jump condition in time to the time-derivative term on the LHS of equations such as Eq. (12). For certain problems this temporal jump condition is physically clear. For instance, in solidification problems, the temporal jump condition for the enthalpy in a given control volume is simply the latent heat released within that volume. In the present fluid-structure interaction problem, such a physics-based jump condition is not available for the velocity field. Since the cell *P*

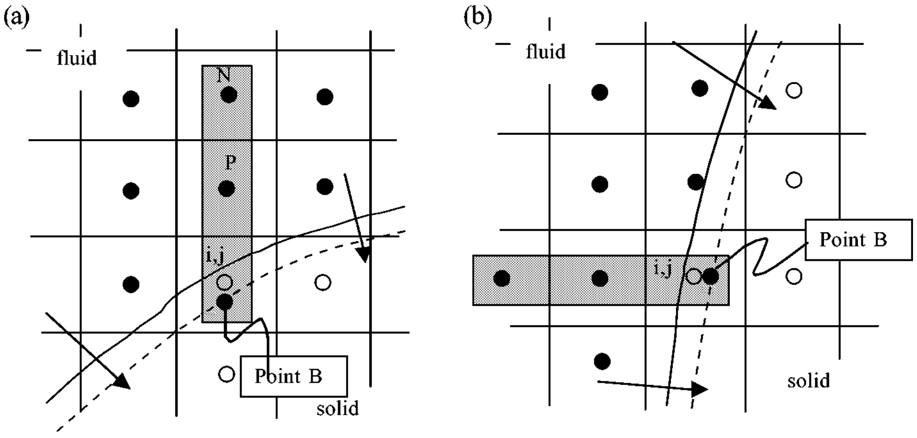

FIG. 3. Change of material as the immersed boundary traverses the mesh. (a) Configuration in which the interface is nearly horizontal and (b) Configuration in which the interface is nearly vertical.

was previously in the solid and it had no history in the fluid phase, there is no physically realizable value of $\vec{u}^t_{i,j}$. Thus, Eq. (14) is not applicable for such cells.

In the current method, an approach similar to that used in the finite-difference method of Udaykumar *et al.* [48] is employed. This consists of temporarily merging the freshly cleared cell with a neighboring cell and is analogous to the approach taken in moving grid formulations when a new cell is inserted following mesh refinement. In the current finite-volume-based methodology, this cell-merging is realized through a simple one-dimensional interpolation operation and this can be explained here for the particular freshly cleared cell shown in Fig. 3a. For this cell, the following interpolant is used,

$$u^*(y) = a_0 + a_1 y + a_2 y^2, \tag{16}$$

where, the coefficients a_0, a_1, and a_2 are coefficients that depend on the value of $\vec{u}^*$ at P, N, and the boundary location B, and the corresponding locations of these points. A similar procedure can be followed for the situation illustrated in Fig. 3b. The above dependence can therefore be recast in the form

$$\sum_M B^P_M u^*_M = 0, \tag{17}$$

and for this cell, Eq. (17) replaces the discretized advection–diffusion equation (12). Note that Eq. (17) is applied only at the instant when there is a crossover. Following that instant, Eq. (12) is again used to time step the $\vec{u}^*$ field. Note also that since pressure depends only on the $\vec{u}^*$ field, Eq. (13) is still valid for these cells as long as $\vec{u}^*$ can be computed through some appropriate means, such as Eq. (17).

2.5. Fast Solution of Discretized Equations

The general form of the discretized equations to be solved at each time step is given in (10). The term on the left-hand side of this equation represents a discretized Helmholtz operator in the case of the advection–diffusion equation and a Laplacian operator in the case of the pressure Poisson equation. The standard alternating-direction line successive–overrelaxation (SOR) proves extremely effective for the solution of the discretized advection–diffusion equation and the residual can be reduced to acceptable levels within a few iterations. However, the discretized pressure Poisson equation (PPE) exhibits significantly slower convergence. In fact, due to its slow convergence, the solution of the discretized pressure equation is usually the most time-consuming part of a fractional-step algorithm. In the presence of immersed boundaries, this behavior of the pressure equation can be further exacerbated since the discrete operator for the trapezoidal cells requires a stencil that has significant dependence on neighbors which are not included in the line-SOR sweeps. For example, in Fig. 2c the coupling of the cell with its northeast and southwest neighbor is not included directly in any of the line-sweeps. Furthermore, discretization in the irregularly shaped boundary cells results in weaker diagonal dominance than in the regular cells and this has a deleterious effect on the convergence of any iterative scheme.

In Ye *et al.* [58], a line-SOR preconditioned BiCGSTAB iterative method was employed and was found to be significantly faster than a simple line-SOR algorithm. Furthermore, for the stationary boundary problems that were simulated there, it was found that this iterative method allowed us to obtain the solution of most problems in a reasonable amount of time. However, BiCGSTAB is inadequate when applied to moving boundary problems. As the boundaries move, the geometry of the domain changes and the elliptic nature of the PPE induces global readjustments of the pressure field. Thus, changes in the pressure field for moving boundary problems can be more significant than for the stationary boundary cases. Thus, the pressure from the previous time step is a much poorer guess for boundary cells in the moving boundary case than it is in the stationary boundary case. Consequently, the starting residual for the PPE in the boundary cells is generally higher in the moving boundary case and considerable effort is therefore required to reduce these to acceptable levels.

One of the most effective methods devised for these types of problems is the multigrid method [52]. The key elements of a multigrid procedure are (i) an appropriate "smoother," (ii) grid coarsening, (iii) restriction, (iv) prolongation, and (v) multigrid schedule. A number of alternatives are available for each of these steps and the reader is referred to Wesseling [52] and Briggs [7] for surveys. The implementation of a standard multigrid algorithm into the current solver would at first glance also seem straightforward given the simple mesh topology. However, the presence of a sharp immersed boundary presents a unique difficulty in this implementation. This issue is explained here with the help of the schematic shown in Fig. 4.

An integral part of the multigrid procedure is grid coarsening. In structured grid methods, the grid is typically coarsened by a factor of 2 or more at each multigrid level. However, representation of geometrical features such as that shown in Figs. 4a and 4b requires appropriate resolution in the underlying Cartesian grid. Our experience with this method indicates that if the radius of curvature of a geometrical feature is r, then acceptable representation of the feature requires a grid spacing, which is smaller than $r/2$. This provides an upper limit on the grid spacing in the vicinity of any geometrical feature, and the grid spacing is always chosen to ensure adequate representation of all geometrical features. In the current Cartesian grid method, there then arises the possibility that a given geometrical feature of the immersed boundary will not be resolved on one or more of the coarse-grid levels. This will happen in the case where the grid spacing of a coarse grid is greater than $r/2$ in the vicinity of a geometrical feature with radius of curvature r. To some extent, a similar problem would in principle also exist for a body-conformal, structured grid. However, one simple remedy there would be to resort to semi-coarsening [52], where the coarsening is not done along the family of grid lines that is aligned with the boundary. This remedy, however, is not available for the current solver since there is no unique family of grid lines that is aligned with the immersed boundary.

One alternative approach would be to develop a multiscale representation of the immersed boundary such that as the grid is coarsened, the geometrical features of the immersed boundary are also appropriately coarsened. However, this approach has a number of undesirable features. First, developing a robust algorithm for multiscale representation of the boundary is a nontrivial proposition because of the variety of situations that would have to be tackled. One simple case where difficulty may arise is when there are multiple distinct features (or bodies) that are relatively close to each other. On a coarse grid, where the grid spacing is larger than the distance between these distinct boundaries, a coarse representation of such boundaries would necessarily lead to merging of these boundaries. Just how such a merging would be implemented into the multigrid algorithm is unclear. Second, even if a multiscale representation of the boundary could be constructed, the prolongation and restriction operations would be extremely complicated for the boundary cells. Furthermore, a wide variety

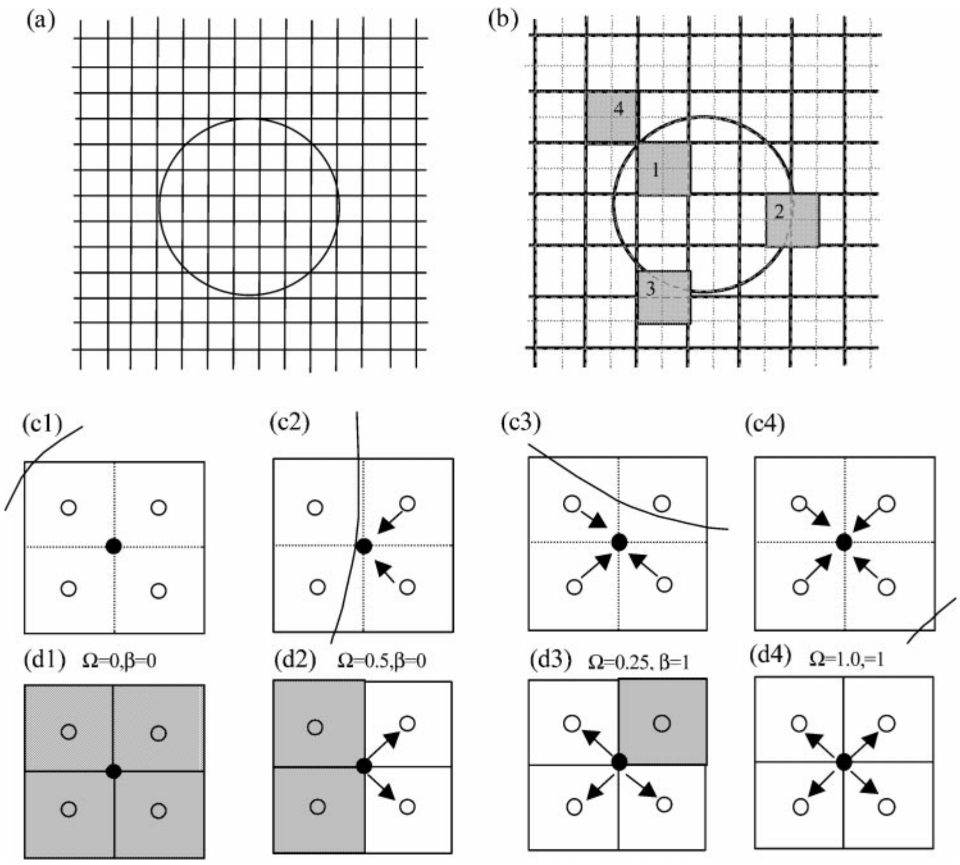

FIG. 4. Illustration of the effect of coarsening of the mesh on the interface representation in the multigrid solution of the Poisson equation. Arrows show information transfer. Filled circle is the coarse mesh point. Open circles are fine mesh points. (a) The fine mesh level with sharp interface embedded. (b) The coarse mesh with embedded boundary. Four sample cells are shown shaded with different behavior of the sharp interface with respect to the coarse mesh. (c1) to (c4) Illustration of the interaction of the sharp interface with the fine mesh for cells marked 1 to 4 in (b). (d1) to (d4) Representation of the interface on the coarse mesh corresponding to the cases (c1) to (c4). Shaded fine cells are taken to be solid cells.

of such cells will be encountered thereby eliminating the possibility of devising systematic restriction and prolongation procedures for such cells.

It should be pointed out that there have been previous efforts at developing multigrid methods in domains with internal structure, such as when the coefficients for the transport are discontinuous [1]. However, the problems tackled in the present paper cannot make use of these methods since, in the present case, the pressure field in not computed within the solid regions. Therefore, the present method is not a one-domain solution of an elliptic equation with discontinuous coefficients, but rather the solution of an elliptic equation in a domain with embedded “holes.” Webster [53, 54] has presented an algebraic multigrid for a Cartesian grid flow solver in which arbitrary internal geometries are introduced by cell blocking. However, in that method, the internal geometries are aligned with the Cartesian mesh and therefore that method is also not appropriate in the current case. Johansen and Colella [21] also have provided a brief account of a multigrid method for solving the Poisson equation in the presence of immersed boundaries. In the following we describe in detail the implementation of the multigrid in the presence of the embedded boundaries as adapted to the finite-volume approach. In the results section, we demonstrate the convergence behavior of the method described below.

The primary complexity in applying a multigrid technique in the current solver is associated with retaining the immersed boundary as a sharp interface at the coarse grid levels. Motivated by this, a multigrid algorithm has been developed wherein the boundary is represented as a sharp interface only at the finest grid level. At the coarser levels, the presence of the boundary is accounted for only in an approximate sense through the volume fraction of the coarse cells and no explicit reconstruction of the immersed boundary is done at these levels. Although conceptually this approach is relatively straightforward, the key is to implement it in a systematic manner so that it is applicable for the wide variety of situations that could be encountered. Furthermore, it is also essential to ensure that this approach does not significantly degrade the convergence properties of the multigrid algorithm. In the following, the implementation of this multigrid technique is described and in a later section, the convergence acceleration of this technique is demonstrated.

In order to simplify the following discussion, a uniform grid is assumed in both directions. However, the actual algorithm has been applied to the general nonuniform case. For the bulk of the flow domain, i.e., away from the immersed boundaries, a standard multigrid with V- or W-cycle is used. Coarsening of the grid is performed as for simple Cartesian meshes without regard to the immersed boundaries, so that the grid spacing at level k is given by $h^k = 2h^{k-1}$. For regular cells, away from the immersed boundary, the multigrid solve proceeds in a standard way that involves smoothing (level k),

$$\nabla^2\phi^k = S^k, \tag{18}$$

residual computation at level k,

$$\nabla^2\phi^k - S^k = \varepsilon^k, \tag{19}$$

and restriction,

$$\bar{\varepsilon}^{k+1} = \sum_M \varepsilon_M^k, \tag{20}$$

where M goes over the four fine mesh cells surrounding the particular coarse mesh cells (Fig. 4c). This is followed by coarse mesh solve,

$$\nabla^2\phi^{k+1} = S^{k+1} \quad \text{where } S^{k+1} = -\bar{\varepsilon}^{k+1}, \tag{21}$$

and finally, prolongation

$$\phi^k = \phi^k + \tilde{\phi}^{k+1}, \tag{22}$$

where

$$\tilde{\phi}^{k+1} = \sum_M \lambda_M\, \phi_M^{k+1}. \tag{23}$$

In this summation, M goes over the surrounding coarse nodes (see Fig. 4d1) and λ denotes weights corresponding to distance-weighted interpolation as shown in Fig. 4d4.

The above procedure applies in the case where there are no immersed boundaries or to cells that are distant from the immersed boundary. Now consider a boundary defining an immersed solid object on the finest level mesh (i.e., one where the solution is sought) and subsequent coarsening of the mesh with the boundary shape retained, as shown in Figs. 4a and 4b. Some situations, which may arise upon coarsening the mesh in the presence of an immersed boundary, are indicated in Figs. 4c1 to 4c4. As shown, the fine cells comprising the coarse cell may lie in solid or fluid phase depending on the manner in which the sharp interface passes through the fine mesh. In defining the Laplacian operator and also in affecting transfer between successive mesh levels, care has to be exercised in cells cut by the immersed boundary. At the finest level, the operators are assembled as detailed in Section 2.3 above. At this level, full information on the sharp immersed boundary is retained as in Figs. 4c1 to 4c4. In the restriction step, residuals from the fine mesh are to be interpolated to the coarse mesh. At the coarser levels, to avoid coarsening of the embedded geometry, the geometry is represented using a volume fraction field. In the following discussion, the implementation procedure is described with primary focus on cells that are close to the immersed boundary.

Any given coarse mesh cell at level k, such as the one shown in Fig. 4b, consists of four finer cells of level $k-1$. Assuming that, based on a set of rules [58], it has already been determined if a given cell at level $k-1$ is in the fluid or in the solid, a procedure needs to be developed in order to make this determination for cells at the current level k. For each cell at every grid level, a boolean variable β, which has a value of one if the cell is a fluid cell, and zero if the cell is a solid cell, is defined. Furthermore, a weighting factor ω is also defined for each contributing fine-mesh cell, which corresponds to the fractional volume contributed by that fine cell to the coarse cell. For a uniform mesh, $\omega = 1/4$ for any fine mesh cell. Given these definitions, the volume fraction of fluid in a coarse cell at level k is estimated as (Figs. 4d1 to 4d4)

$$\Omega^k = \sum_{M=1}^{4} \omega_M^{k-1} \beta_M^{k-1}, \tag{24}$$

where the summation runs over the four fine mesh points surrounding the coarse mesh point. Therefore, for a coarse cell comprising of all four contributing finer level cells in the fluid phase, the volume fraction $\Omega^k = 1.0$; otherwise $0 \le \Omega^k < 1.0$. The Boolean variable β can now be defined at level k based on the volume fraction of that cell as follows:

$$\begin{aligned} \beta_{ij}^k &= 1 \quad \text{if } 0.5 < \Omega_{ij}^k \le 1.0 \\ \beta_{ij}^k &= 0 \quad \text{if } 0.0 < \Omega_{ij}^k \le 0.5. \end{aligned} \tag{25}$$

The above definition holds for all levels $k > 1$. For the finest level ($k = 1$) β and Ω are determined simply based on whether the cell center lies in the fluid or solid region as follows:

$$\begin{aligned} \beta_{ij}^1 &= \Omega_{ij}^1 = 0 \quad \text{if cell center lies in solid} \\ \beta_{ij}^1 &= \Omega_{ij}^1 = 1 \quad \text{if cell center lies in fluid.} \end{aligned} \tag{26}$$

Thus, at the finest level, the exact location of the sharp boundary is employed in constructing the cells, whereas at the coarser levels, the presence of the boundary is incorporated into the discretization in an approximate manner by using the fluid volume fraction information. Equation (25) in effect implies that a coarse cell whose fluid fraction is less than 0.5 is to be considered a solid cell. This threshold volume fraction value of 0.5 used in determining the phase of coarse cells is not chosen arbitrarily. In almost all cases it is found that the normalized volume of a trapezoidal boundary cell (of level $k = 1$) is greater than 0.5. Thus, the construction of the coarse cells near the boundary is consistent with that of the cells at the finest level.

The volume fraction Ω not only allows us to construct the stencils for all mesh levels near the boundary but also facilitates the systematic development of the restriction and prolongation operations at all levels. This is accomplished by defining another variable γ for each cell at every grid level, which derives its value from Ω in the following manner:

$$\begin{aligned}
\gamma_{ij}^{k} &= 1, \quad \text{if } \ \Omega^{k} = 1, \text{ i.e., for a complete fluid cell} \\
\gamma_{ij}^{k} &= 2, \quad \text{if } \ 0.5 < \Omega^{k} < 1.0, \text{ i.e., for a partially fluid cell} \\
\gamma_{ij}^{k} &= 0, \quad \text{if } \ 0 \le \Omega^{k} \le 0.5, \text{ i.e., for a solid cell.}
\end{aligned} \tag{27}$$

The restriction operation can now be accomplished at every grid level by modifying Eq. (20) using the variables β, Ω, and ω as

$$\overline{\varepsilon}^{k} = \frac{1}{\Omega^{k}} \sum_{i,j} \varepsilon_{ij}^{k-1} \beta_{ij}^{k-1} \omega_{ij}^{k-1}. \tag{28}$$

Thus, only the residuals from the finer-level cells that are in the fluid phase are used in the restriction operation. Furthermore, the contribution of each finer cell is weighted by its fractional volume contribution ω.

For the fine cells, the Laplace operator is assembled as explained in Section 2.3, by using the compact linear–quadratic interpolant in the boundary cells. For coarse cells, the operator is modified based on the value of β_{ij}^{k} to accommodate the volume fraction information. The standard five-point central-difference discretization of the Laplacian is used on the coarse level ($k > 1$),

$$\alpha_{i,j}\phi_{i,j}^{k} + \alpha_{i+1,j}\phi_{i+1,j}^{k} + \alpha_{i-1,j}\phi_{i-1,j}^{k} + \alpha_{i,j+1}\phi_{i,j+1}^{k} + \alpha_{i,j-1}\phi_{i,j-1}^{k} = -\overline{\varepsilon}^{k}, \tag{29}$$

where $\overline{\varepsilon}^{k}$ is computed from Eq. (21). The nominal values of the coefficients α in Eq. (29) correspond to those arising from central-difference discretization of the Laplace operator on a regular Cartesian mesh. For cells that do not adjoin the immersed boundary, these remain unchanged. In coarse cells that are partially solid or have neighbors that are partially solid, these have to be modified to account for the presence of the (coarsened) immersed boundary. The volume-fraction information is used in the coarse cell discretization to construct the coefficients α in Eq. (29) in the manner described below. The volume-fraction-based coarse cell representation of the four cells shown in Figs. 4c1 to 4c4 is illustrated in Figs. 4d1 to 4d4. The corresponding values of β_{ij}^{k}, computed from Eq. (25), are also illustrated in the figure. For a coarse cell *ij*,

$$\text{if } \beta_{ij}^{k} = 0, \text{ then } \alpha_{i,j} = 1, \alpha_{i+1,j} = \alpha_{i-1,j} = \alpha_{i,j-1} = \alpha_{i,j+1} = 0 \ \text{ and } \ \overline{\varepsilon}^{k} = 0. \tag{30}$$

Thus, the corrections $\phi^k_{i,j}$ are set to zero in the coarse cells with fluid fractions $\Omega^k_{ij} \leq 0.5$. These cells are treated as solid cells at the coarse grid level and therefore the correction field on the coarse mesh is not computed in such coarse cells. On the coarse mesh, the corrections are computed only in those computational cells whose fluid fraction $\Omega^k_{ij} > 0.5$. Whether or not the corrections are computed in a given cell is determined by the computational flags γ^k_{ij}, set in Eq. (27). Now, some of the coarse fluid cells have neighboring cells in the solid phase, i.e., with $\Omega \leq 0.5$. For such coarse cells, the standard 5-point stencil in Eq. (30) is modified to impose a Neumann boundary condition on the adjoining face. This is achieved through the following condition:

If $\beta^k_{ij} = 1$,

$$\alpha_{l,m} = \alpha_{l,m} \quad \text{if } \gamma^k_{lm} \neq 2 \ \text{ for all } (l,m),\ l \neq i, m \neq j, \tag{31a}$$

$$\alpha_{l,m} = 0, \qquad \text{if } \gamma^{k+1}_{lm} = 2, \ \text{ for all } (l,m), l \neq i, m \neq j. \tag{31b}$$

The first condition implies that if the neighbor to a side of the cell is a fluid cell, as given by the condition in Eq. (25), then the usual central-difference coefficient is retained for that side. If the neighbor is in the solid phase, then the coefficient for the neighbor on that side is set to zero. This applies a Neumann condition for the correction field on that cell face. Note that in the coarse mesh, by this procedure, the volume-fraction representation results in a stair-stepped treatment of the embedded solid boundary. This rough representation of the immersed boundary on the coarser meshes is an implicit coarsening of the embedded geometry accomplished through the volume fraction variable at all mesh levels except at the finest level, where the shape of the interface is accounted for exactly. Once the coefficients of the neighboring cells have been computed, the coefficient of the (i, j) cell is assembled as

$$\alpha_{i,j} = -\sum_{l,m} \alpha_{l,m}, \ \text{ with summation over } l \neq i, m \neq j. \tag{31c}$$

Once the correction field at level k is obtained, the prolongation of these corrections to the next fine level $k-1$ is performed as follows. For a fine cell at level $k-1$ in the fluid phase, i.e., a cell for which $\beta \neq 0$, the prolonged correction $\tilde{\phi}^k_{ij}$ is obtained using distance-weighted interpolation from the surrounding coarse mesh cells as

$$\tilde{\phi}^k_{ij} = \frac{1}{\Delta} \sum_{l,m} \kappa_{lm} \phi^k_{lm}, \tag{32}$$

where the summation goes over all neighboring coarse cells,

$$\kappa_{lm} = 0, \qquad \text{if } \gamma^k_{lm} = 0 \tag{33a}$$

$$\kappa_{lm} = \sqrt{\left(x^k_l - x^{k-1}_i\right)^2 + \left(y^k_m - y^{k-1}_j\right)^2}, \quad \text{if } \gamma^k_{lm} \neq 0 \tag{33b}$$

$$\text{and } \ \Delta = \sum_{l,m} \kappa_{lm}. \tag{33c}$$

If $\beta^{k-1}_{ij} = 0$, no correction is performed since that fine-level cell lies in the solid phase. The procedure described above leads to a multigrid algorithm that successfully avoids coarsening of solid immersed boundaries, while executing information transfer from fine to coarse meshes and vice-versa. The coarse grid calculations account for the boundary only in an approximate manner and the effect of this on the overall convergence behavior is examined in Section 3.1.

2.6. Overall Solution Procedure

Given the velocity and pressure field and interface position at time t, the overall solution procedure to advance the solution to $t + \Delta t$ is as follows:

1. Advance the immersed boundary to its position at $t + \Delta t$ as described in Section 2.3.
2. Determine the intersection of the updated immersed boundary with the Cartesian mesh and using this information, reshape the trapezoidal boundary cells.
3. Update the discrete expressions corresponding to (10) for the boundary cells including freshly cleared cells.
4. Advance the discretized equations in time:

   a. Solve for intermediate velocity field $\vec{u}^*$.
   b. Solve for pressure from Eq. (13) using the multigrid technique.
   c. Correct the velocity field as in Eq. (6)–(7) to obtain velocity field at $t + \Delta t$.

This completes the description of the current simulation methodology. It should be pointed out that although the methodology has been described only in the context of 2-D geometries, the approach developed here can be extended to three dimensions. The key aspects to be addressed in extending this methodology to 3-D are efficient methods for representing curved 3-D interfaces and reconstructing boundary cells [20, 25]. Apart from these aspects, the discretization procedure described here carries over into 3-D in a straightforward way. In the following sections the focus is on examining the performance of the multigrid algorithm, validating the solution methodology by comparing computed results with experiments, and demonstrating the capabilities of the method for simulating flows with complex immersed moving solid boundaries.

3. RESULTS

3.1. Multigrid Performance with Immersed Boundaries

The performance of the multigrid algorithm outlined above is first examined in the presence of immersed boundaries by comparing it to a case which does not include immersed boundaries. This is accomplished by computing flow through a channel with increasing geometric complexity introduced by inserting immersed boundaries into the domain (see Fig. 5). The first case (Fig. 5a) is that of a simple channel flow with no internal immersed boundaries. The complexity of the domain is increased by introducing cylinders into the channel and the second, third, and fourth cases include 1, 5, and 11 cylinders (each of radius 0.05) in the channel, respectively. A staggered-array type configuration is chosen and the geometry of this array is shown in Fig. 5b. Note that in the present Cartesian grid method, the mesh remains unchanged as immersed boundaries are successively embedded in the domain. The inlet flow for all cases corresponds to a Reynolds number of 100 defined on the channel height (H) and inlet velocity. The streamwise length of the channel is equal to $5H$ and in all cases a uniform 400×80 grid is used. The convergence acceleration of the multigrid scheme is compared for these various cases for the first time step, given an

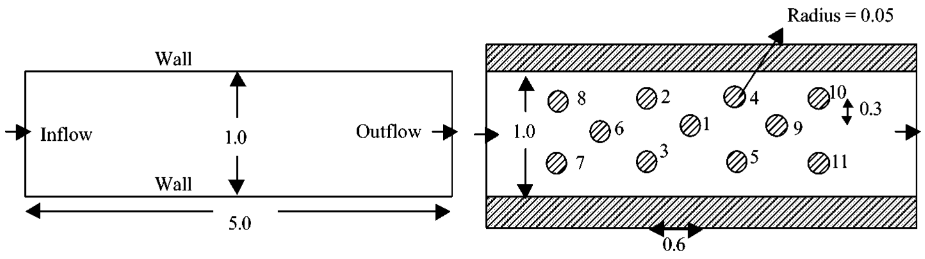

FIG. 5. Illustration of the setup for channel flow computation as a test of the multigrid algorithm. (a) A channel without immersed boundaries. (b) A channel with immersed boundaries. First, the channel walls are immersed in the computational domain. Thereafter, cylindrical obstructions are successively added in the domain.

initial condition of $\vec{u} = (1, 0)$ and $p = 0$. In each case, we report on the behavior of a multigrid run with a V-cycle consisting of one LSOR iteration at the finest level and three iterations at each coarse level. Through numerical experimentation, this schedule was found to yield the optimal convergence behavior for a number of different flow configurations. The convergence history for cases 1 (no cylinders) and 4 (11 cylinders) is shown in Fig. 6 and results for all cases are presented in Tables I and II. Table I contains information about the actual speedup (CPU time) produced by the multigrid technique whereas in Table II the numbers of iterations at the finest level are presented. In both tables, all values are normalized by the corresponding value observed for the convergence without multigrid for the particular case.

Figure 6a shows the convergence history for the first case, where the only immersed boundaries are the channel walls. This is the baseline case, which should exhibit the performance of a standard multigrid algorithm. The pressure Poisson equation has been solved with one-, two-, and three-level multigrid schedules and Fig. 6a clearly shows the convergence acceleration achieved by the multigrid algorithm. The corresponding numbers in Tables I and II show that the CPU time required for two- and three-level multigrid schedules is only 21 and 7.8%, respectively, of the CPU time required by the basic LSOR scheme. In terms of iterations, the two- and three-level multigrid schedules require only 17 and 5.6% of the number of iterations required by the baseline LSOR scheme. The slight mismatch between the iteration reduction and the CPU reduction is due to the CPU time associated with the computation at the coarse grid levels. Figure 6b shows the corresponding convergence behavior for the most complex case, where there are 11 cylinders, and a number of observations can be made regarding this figure. First, the convergence acceleration with increasing grid level for this case is comparable to that

TABLE I
CPU Times for the Multigrid Solver for the Pressure Poission Equation

Levels	Channel (no cylinders)	Channel with 1 cylinder	Channel with 5 cylinders	Channel with 11 cylinders
1	1.000	1.000	1.000	1.000
2	0.210	0.163	0.136	0.158
3	0.078	0.045	0.034	0.035

Note. The CPU times in each case have been normalized by the single-grid LSOR time.

TABLE II
Number of Fine-Level Iterations for the Multigrid Solver for the Pressure Poisson Equation

Levels	Channel (no cylinders)	Channel with 1 cylinder	Channel with 5 cylinders	Channel with 11 cylinders
1	1.000	1.000	1.000	1.000
2	0.170	0.137	0.118	0.153
3	0.056	0.035	0.028	0.030

Note. The iterations in each case have been normalized by the single-grid LSOR iterations.

observed for the simple channel case. This is indeed confirmed in Tables I and II, where it is shown that the CPU times required by the two- and three-level multigrid schedules for this case are about 16 and 3.5%, respectively, of the baseline LSOR scheme. A similar convergence acceleration behavior is also observed for the cases where there are one and five cylinders. This numerical experiment therefore indicates that increasing geometrical complexity and the associated approximate correction procedure, adopted at the coarse levels, do not degrade the convergence behavior of the underlying multigrid method.

It is also observed in Fig. 6b that, overall, the number of iterations required for the case with no cylinders is much lower than that required in the case where there are 11 cylinders. This is because increasing complexity in the domain can be expected to always increase the effort required to converge the pressure Poisson equations since the pressure field has to satisfy the Neumann boundary condition at multiple boundaries in the domain. Note that comparison of convergence rates for a given case with increasing grid level is straightforward. On the other hand, comparison *between* the various cases has to be done with the realization that the distribution of the starting residual, even with the same initial guess, might be quite different. However, the objective of the current set of calculations is

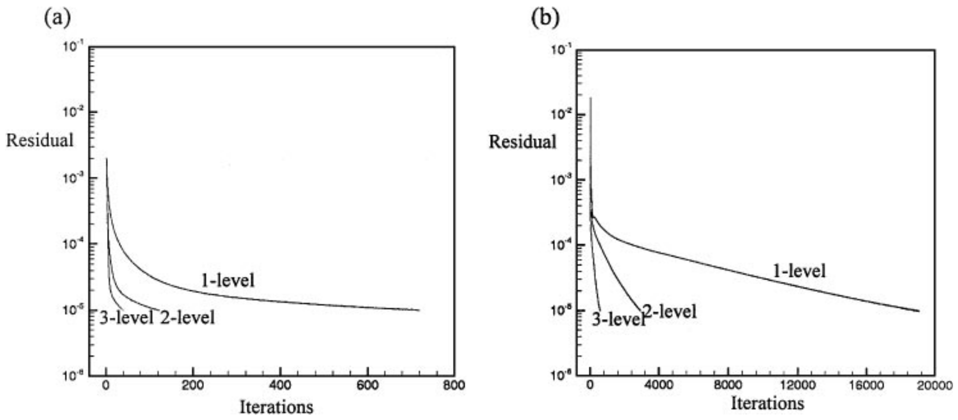

FIG. 6. Covergence path for the PPE using LSOR preconditioned multigrid. The number of levels is indicated. The convergence residual was specified to be $\varepsilon = 10^{-5}$. (a) The case of channel flow with channel walls as immersed boundaries. (b) The case of channel walls as immersed boundaries and with 11 immersed cylindrical obstructions.

to compare the *relative* convergence behavior of the multigrid with increasing complexity, and this comparison can be made clearly despite the difference in the starting residuals.

In addition to the cases presented in this section, several cases with complex domains and moving boundaries, including those presented in the following sections, have been computed using the present multigrid technique. In each case the multigrid has yielded substantial acceleration of the solution to the Poisson equation.

3.2. Convergence with Mesh Refinement for a Moving Boundary Calculation

We now present an analysis of the variation of solution error with grid refinement in order to establish the overall order of accuracy of the numerical method. This study is performed for the case where a solid moving boundary is immersed in a fluid enclosed within a domain with solid no-slip walls. The flow situation can be visualized from Fig. 7a. The cylinder of diameter $D = 1.37$ units was placed initially at the center of the box of dimension 2.7×2.7 units and oscillated horizontally with a nondimensional time period of 1 and an amplitude of $0.25D$. The oscillation was effected by moving the cylinder as a rigid body with velocity given by

$$\nu_x = 0.25\pi \sin(2\pi t); \quad \nu_y = 0.$$

The flow for this moving boundary problem is simulated using the current solver for a Reynolds number (with respect to the cylinder diameter) of 100. The following sequence of grid sizes was employed in performing the error analysis: 20×20, 60×60, 100×100, and 270×270 grid points. In the absence of an exact analytical solution to this flow problem, the results on the 270×270 mesh was taken to be the “exact” solution in order to obtain the error distribution for each of the coarser meshes.

The cylinder motion was impulsively started with the above velocity function. A small time step of $\Delta t = 10^{-4}$ was chosen for all these simulations in order to minimize the effect of temporal errors on the solution. The simulations were carried out for one oscillation period and the velocity components at each grid point were recorded for all the meshes under consideration at the end of this period. For a given $N \times N$ mesh, the nth norm of the error was computed as

$$\varepsilon_n = \left(\frac{1}{N^2} \sum_{j=1}^{N^2} \left| \phi_j^{(N)} - \phi_j^{(270)} \right|^n \right)^{1/n},$$

where $\phi_j^{(N)}$ denotes a generic computed variable (x-velocity component in this case) on the $N \times N$ mesh, and j is the index of the grid cell on this mesh.

The results of the analysis are presented in Fig. 7. Figure 7a shows the flowfield developed due to the moving cylinder in the box, at a time corresponding to 1/10 of the period of oscillation, at which time the cylinder is moving to the right. This flow calculation was performed on the 270×270 mesh. The streamlines of the flow induced by the moving cylinder and the x-velocity contours are shown in Fig. 7a. The presence of significant velocity gradients in the boundary layers adjoining the moving cylinder can be seen in the contour plot. Furthermore, due to the closed domain in which the cylinder moves and to the incompressibility constraint, a recirculating flow is created with streamlines emerging from the leading side of the cylinder and attaching at the trailing side. In Fig. 7b we

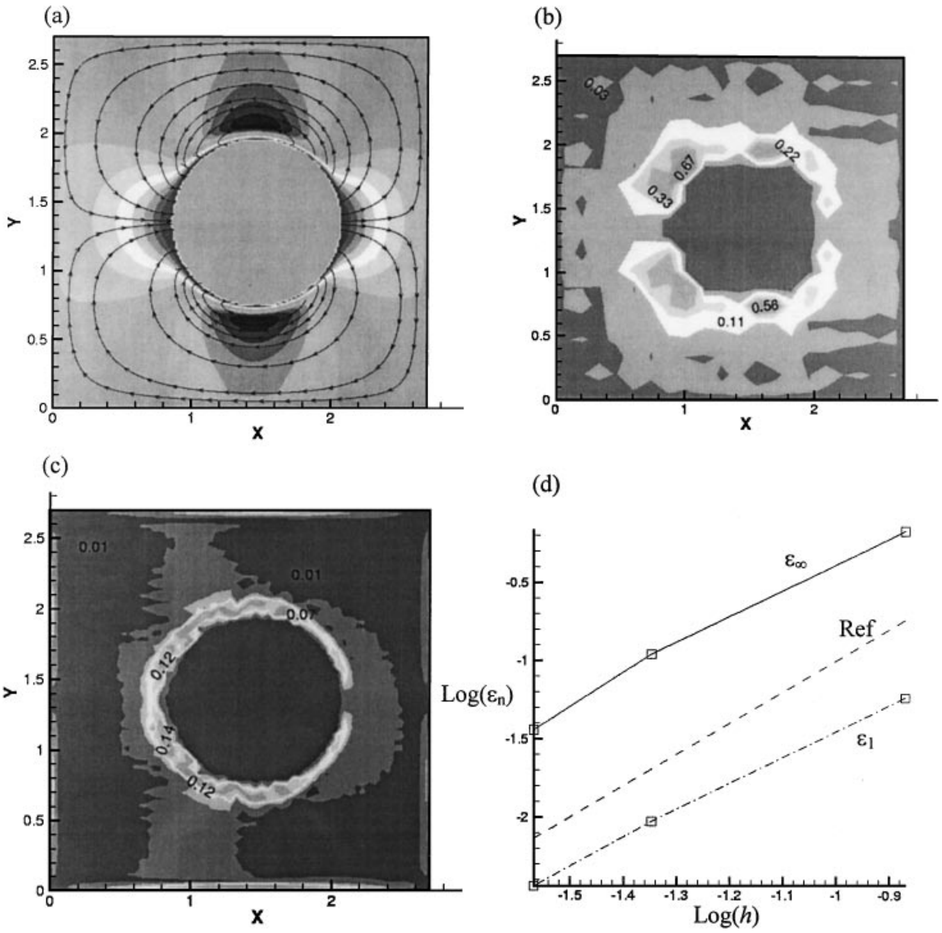

FIG. 7. (a) Computational domain, streamlines, and x-velocity contours for a cylinder oscillating in a box computed on the 270×270 mesh. (b) The error distribution in x-velocity for a 20×20 mesh. (c) The error distribution for the 60×60 mesh. (d) The convergence behavior of the error norms. The L_∞ and L_1 norms are shown along with the reference (dashed) line corresponding to second-order convergence.

show the distribution of the local error $|\phi_j^{(N)} - \phi_j^{(270)}|$ for the coarsest 20×20 mesh. The magnitudes of errors are labeled on a few contours and it can be observed that the errors are clearly concentrated in the region surrounding the cylinder where significant gradients exist. Figure 7c shows the error distribution for the 60×60 mesh. Again, the errors appear to be concentrated in the boundary layer region; however, as expected, the magnitude of the error is lower than that in Fig. 7b. Figure 7d shows the convergence behavior of the error norms for the three meshes (20×20, 60×60, 100×100) with respect to the finest reference mesh solution. Logs of both $\varepsilon_1$and ε_∞ are plotted against $\log(h)$, where h is the grid spacing. Also plotted is a reference line with a slope of 2 corresponding to second-order convergence. As can be noted, the convergence rate of error in the simulations is close to the reference line, indicating nearly second-order-accuracy. It should be pointed out that similar behavior in the error was observed when error was analyzed at intermediate times during the oscillation cycle. It should be noted that exact second-order-accuracy is not expected in this test primarily because the errors are not computed based on an exact

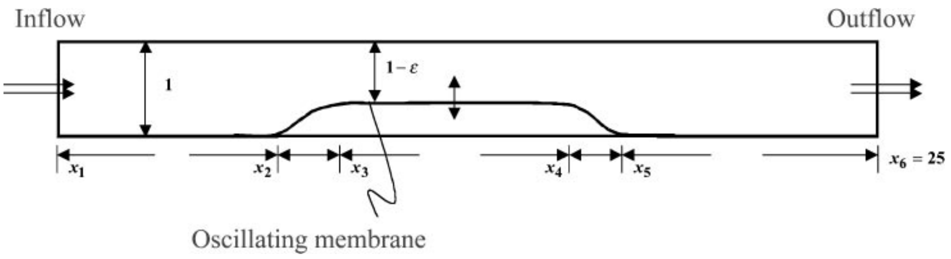

FIG. 8. Configuration used to validate the current method. Flow in a channel with a moving indentation. The inflow condition is a Poiseuslle flow, Re = 507.

solution. In conclusion, even in the case of a moving boundary, the second-order spatial accuracy inherent in the spatial discretization is maintained.

3.3. Flow in a Channel with a Moving Indentation

There are a few problems with moving solid boundaries interacting with flowfields where detailed experimental and numerical results have been documented for use as benchmarks. One such problem is that involving the flow in a channel with a moving indentation performed by Pedley and co-workers [34, 38]. This is a model for the collapse of a blood vessel and has relevance in cardiovascular flows. The arrangement is as shown in Fig. 8. Poisuelle inflow condition and channel dimensions are specified in line with the experiment. The Reynolds number is

$$\mathrm{Re} = \frac{U_o d}{\nu} = 507.$$

($U_0 = \frac{Q_0}{d}$ is the velocity scale, where Q_0 is the volumetric flow rate per unit channel depth and d is the channel height.) The motion of the indentation is imposed in experiments by means of a piston pushing against a rubber diaphragm. The velocity of the interface is given by [38]:

$$v_y(x,t) = g(x)h(t),$$

where

$$h(t) = \frac{1}{2}(1 - \cos(2\pi t)). \tag{34}$$

The spatial variation is given by

$$g(x) \begin{aligned} &= 0 && x_1 < x < x_2 \\ &= \frac{1}{2}\varepsilon(1 + \tan\mathrm{h}\,\beta(x - x_o)) && x_2 < x < x_3 \\ &= \varepsilon && x_3 < x < x_4 \\ &= \frac{1}{2}\varepsilon(1 - \tan\mathrm{h}\,\beta x) && x_4 < x < x_5 \\ &= 0 && x_5 < x < x_6, \end{aligned} \tag{35}$$

where locations x_1 to x_6 are as shown in Fig. 8. In Ralph and Pedley [38], the stream-function vorticity method has been used along with a body-fitted moving mesh to solve the problem. In the present method, primitive variables are used and the mesh remains fixed. We will compare our results for this case with experiments [34] and numerical results [38]. For low Strouhal numbers, the flow behaves in a quasi-steady fashion and the sequence of events in time as the indentation grows, corresponds to those encountered when steady-state solutions are obtained at the different geometries and juxtaposed. The physics of the problem is interesting for high enough Strouhal numbers, when the quasi-steady assumption no longer holds and the full unsteady, viscous dynamics needs to be captured to reproduce the experimental results.

For the Strouhal number chosen here, i.e., for the frequency of oscillation of the indentation, the flow in the channel downstream of the indentation falls in the unsteady regime. The squeezing of the flow in the channel propagates waves downstream of the constriction leading to the formation and advection of vortices on both the top and the bottom surfaces of the channel. The streamlines are shown in Fig. 9 at various instants in one cycle of motion of the indentation, for the case where $\mathrm{St} = 0.037$, $\varepsilon = 0.38$, and $\beta = 4.14$. Our results reproduce almost exactly the sequence of events occurring in the experiments and agree with the numerical results of Ralph and Pedley [38]. In Fig. 9 we show the development of a series of eddies downstream of the moving indentation. These eddies are formed successively in time as the disturbance propagates through the channel. In agreement with the numerical [38] and experimental [34] results, we find that the eddy B at the top wall splits at a nondimensional time between $t = 0.6$ and $t = 0.65$ into three eddies. This behavior of eddy B is shown in detail in Fig. 10. In Fig. 11, we compare the wave propagation characteristics quantitatively with numerical and experimental results of Pedley and coworkers. We show in that figure the location of the crests and troughs associated with each of eddies B, C, and D with time. The slope of this curve gives the phase velocity of the disturbances downstream of the indentation. Note that for eddies B and D, we measure trough position,

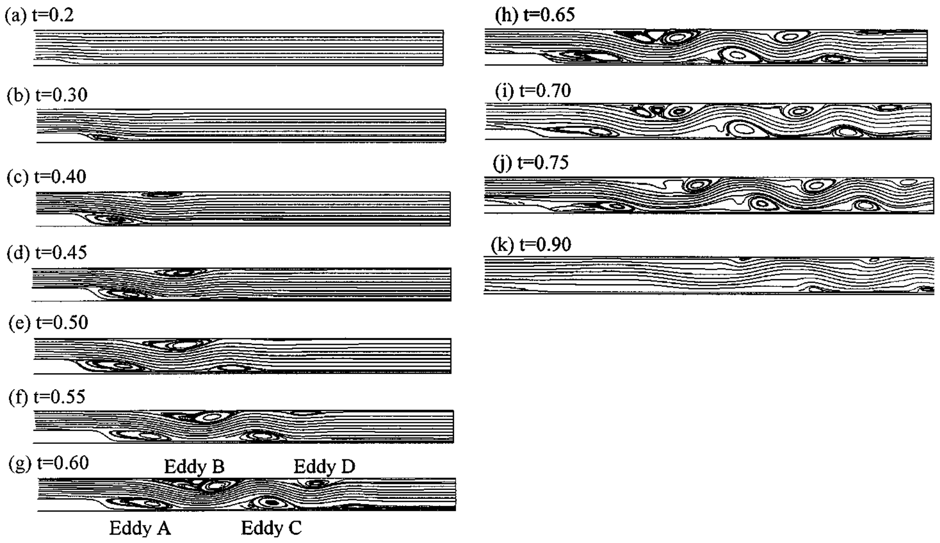

FIG. 9. Wave formation and propagation downstream of a moving indentation computed with the present numerical method. The formation of the various eddies downstream of the indentation is indicated. The nondimensional time instants at which the streamlines are shown are also indicated.

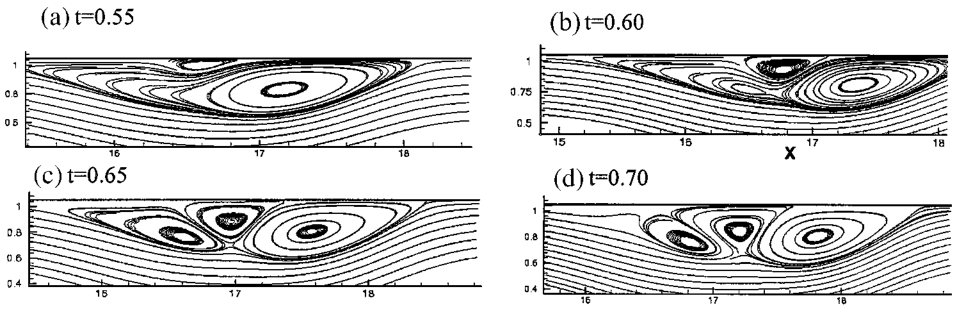

FIG. 10. Splitting of eddy B downstream of the indentation. The eddy splits between nondimensional time $t = 0.6$ and $t = 0.65$, in agreement with the experiments [34] and numerics of Ralph and Pedley [38].

i.e., x-location of the lowest point of the eddy, for C the crest, i.e., x-location of highest point of the eddy. As seen from the figure, the phase speed of the disturbances, as given by the slope of the x-t curves, is captured accurately by the present method. The position and phase velocity of the waves computed by the present method appear to match with experimental results better than the numerical results of Ralph and Pedley [38]. The trifurcation of the curves in Fig. 11 corresponds to the occurrence of multiple crests and troughs due to the splitting of eddies B and C. The numerical results for the time and location of the splitting of the eddies agree with the experimental results. Thus, the numerical method is successful in capturing the full unsteady, viscous effects in the flow caused by the moving indentation.

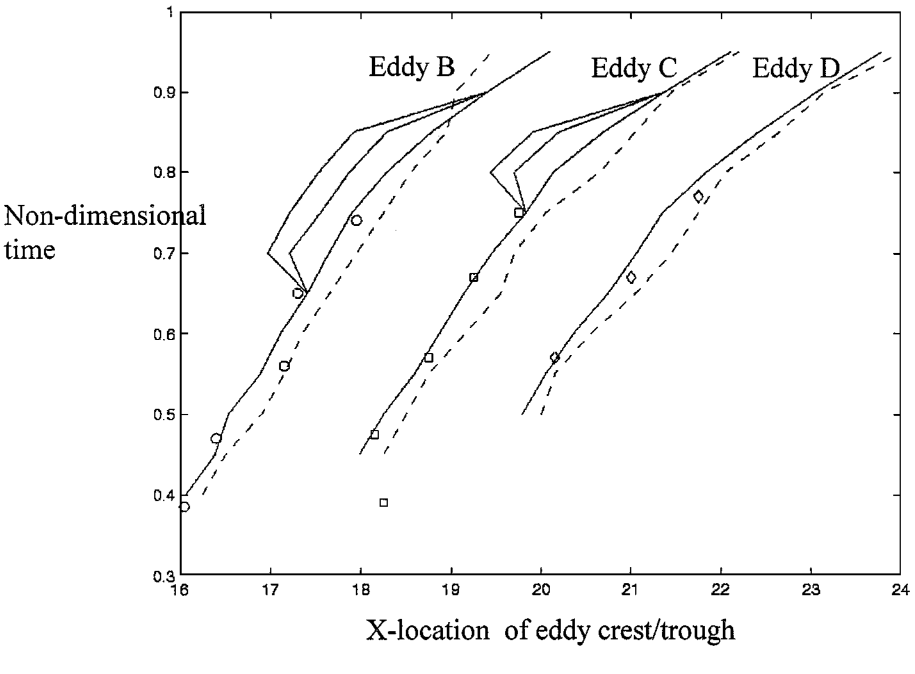

FIG. 11. x–t curves for the eddies labeled B, C, and D in Fig. 9g. Solid lines—present calculation; dashed lines—numerics of Ralph and Pedley [38]; symbols—experiments of Pedley and Stephanoff [34]. The x-locations correspond to the respective crests and troughs of the eddies downstream of the moving indentation. The trifurcation of eddy B is also tracked and the locations of the three eddies formed after splitting are also plotted.

3.4. Vortex Shedding from an Oscillating Cylinder in Uniform Free Stream

The second flow configuration that has been chosen to validate the current computational technique is flow past a transversely oscillating circular cylinder and the associated phenomenon of vortex shedding "lock-on." Vortex shedding "lock-on" is a classical phenomenon that is observed in the wake of bluff bodies and refers to the situation where the frequency of vortex shedding in the wake synchronizes with or locks on to the frequency of an imposed perturbation. The perturbation could be imposed through pulsation of the incoming flow [16] or by free [30] or forced vibration [26] of the bluff body immersed in a steady oncoming flow. In particular, vortex-shedding lock-on past a transversely oscillating cylinder has been studied extensively and is a good benchmark case to validate the current methodology. Here we have simulated flow past a cylinder at Re = 200 undergoing sinusoidal transverse oscillation over a range of amplitudes and frequencies, and a direct comparison of the computed results with the experiments of Koopmann [26] and simulations of Meneghini and Bearman [29] is made.

The computational domain and grid used for the current simulations are shown in Fig. 12. All lengths here have been normalized by the cylinder diameter D. As can be seen in Fig. 12, a relatively large (30×30) computational domain size is used for the current simulation and the mean location of the cylinder center (x_o, y_o) is (10, 15) relative to the left bottom corner of the domain. A uniform free stream velocity U_∞ is prescribed on the inflow (left) and top and bottom boundaries and a convective boundary condition employed at the exit (right) boundary. The cylinder is oscillated sinusoidally such that the location of its center (x_c, y_c) is given by $x_c(t) = x_o;\ y_c(t) = y_o + A\sin(2\pi f_f t)$, where t is the time nondimensionalized by D/U_∞ and A and f_f are the nondimensional amplitude and frequency of the oscillation, respectively. As shown in Fig. 12, a nonuniform mesh is used in the simulation wherein enhanced resolution is provided in the cylinder vicinity and in the wake. In the vertical direction, enhanced resolution is provided up to three diameters on either side of the nominal cylinder location, which is adequate to cover the near wake for all the oscillation amplitudes studied here. The cylinder is immersed and oscillates through the fixed, nonuniform, Cartesian mesh.

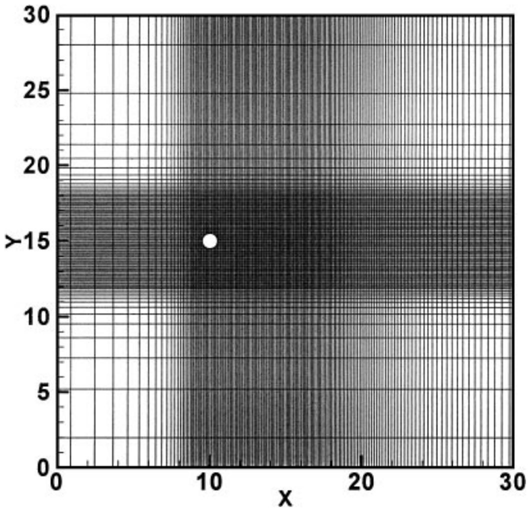

FIG. 12. Nonuniform mesh used in the simulations. Only every other grid line is shown in both directions.

As a first step, flow past a stationary circular cylinder at $\mathrm{Re} = 200$ has been simulated. The flow at this Reynolds number exhibits classical Karman vortex shedding and the current simulation has been continued for about 40 shedding cycles beyond the point where it reaches a stationary state. The vortex-shedding frequency was computed from the variation of the velocity components in the near wake and based on this, a nondimensional vortex shedding frequency of (or Strouhal number) $f_0 = 0.198$ was obtained. This value matches well with the experiments of Williamson [57], who obtained a Strouhal number of 0.197. Furthermore, the average value of the computed drag coefficient was 1.38 and this matches well with the numerically calculated value in Braza *et al.* [8]. The flowfield from this stationary cylinder simulation was used as the initial condition for all oscillating cylinder simulations.

In the current study, two sequences of simulations have been carried out at fixed amplitudes (A) of 0.1 and 0.2 and the frequency has been varied systematically over a range around the natural vortex shedding frequency f_0. Each of these simulations is integrated in time for about 200 nondimensional time units, which is sufficient to reach a stationary state. Subsequently, the equations are integrated further for about $200D/U_\infty$ and the vortex shedding frequency is determined by computing the frequency spectra of the velocity fluctuation in the near wake over this period of time. The solid line in Fig. 13a shows the vortex shedding lock-on region in the space defined by the oscillation frequency (x-axis) and amplitude (y-axis), as determined experimentally by Koopmann [26] for $\mathrm{Re} = 200$. According to this figure, lock-on is observed for amplitudes higher than 0.05 and the frequency range over which lock-on occurs increases with oscillation amplitude.

Figure 13b shows the variation of the vortex shedding frequency in the wake of the cylinder for $A = 0.1$ for a range of cylinder oscillation frequencies. The horizontal line in the plot corresponds to the natural shedding frequency and the dashed inclined line represents the situation of lock-on where the shedding frequency matches the oscillation frequency. Our simulations indicate that vortex shedding lock-on occurs for oscillation frequencies

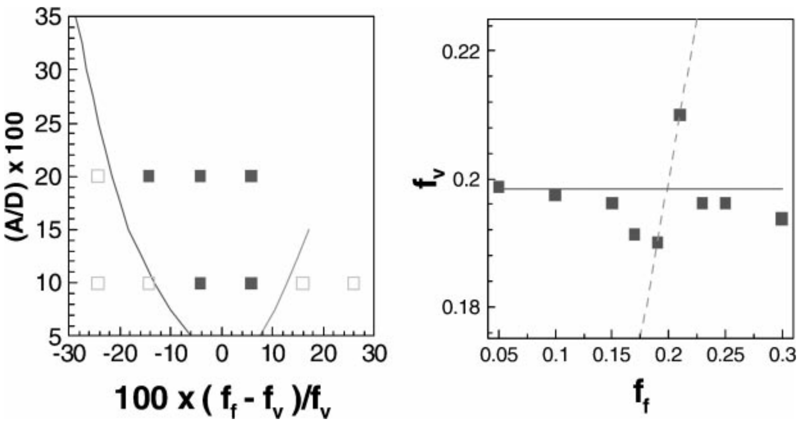

FIG. 13. (a) Plot of the lock-in range determined by Koopman [26] shown by the solid lines. Results from current simulations are also plotted. Filled squares indicate lock-on, while open squares indicate that lock-on did not result. (b) Variation of vortex shedding frequency with forcing frequency for $A = 0.10$. Dashed line indicates lock-on and the horizontal line indicates the natural vortex shedding frequency of 0.198. The filled squares are the results from the calculations.

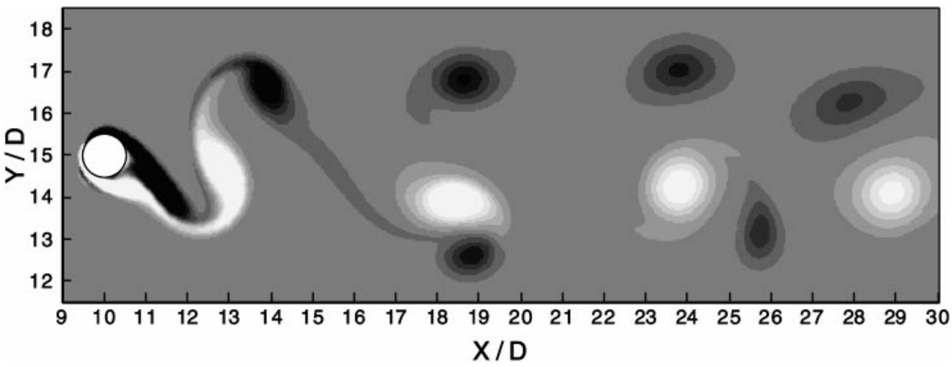

FIG. 14. Contour plot of spanwise vorticity showing the shedding of three vortices per shedding cycle for $A = 0.33$ and a forcing frequency of 0.15872.

of $0.95 f_0$ and $1.05 f_0$. As the oscillation frequency is decreased below $0.95 f_0$, the vortex shedding decreases monotonically and approaches the natural shedding frequency. A somewhat different behavior is observed as the frequency is increased beyond $1.05 f_0$. We observe that as the frequency is increased, the vortex shedding frequency rapidly drops to a value below the natural shedding frequency and then approaches this value as the oscillation frequency is increased further. A similar behavior in the frequency has been observed by Stansby [40]. The results of our simulations are superposed on the plot in Fig. 13a and we find that the lock-on behavior predicted in our simulations is in line with the experiments of Koopmann [26]. Four simulations have also been carried out with $A = 0.2$. The results of these simulations have also been plotted in Fig. 13a and are also found to be in line with the experiments.

Finally, one simulation has been carried out with $A = 0.33$ and a forcing frequency of 0.15872. Meneghini and Bearman [29] have simulated this flow using a vortex method and have found that for these parameters, three vortices are shed in the wake for each shedding cycle. Figure 14 shows a contour plot of the spanwise vorticity obtained from our simulation at one time instant and we also observe two clockwise and one anti-clockwise vortices being shed per cycle of the cylinder oscillation. Thus, the current simulations confirm the experimental observations of Koopmann [26] and also match the computational results of Meneghini and Bearman [29], thereby providing further validation of the current simulation methodology.

3.5. Diaphragm-Driven Micropump

To demonstrate the ability of the method to handle flows with multiple moving boundaries, the final configuration that we have chosen is that of a diaphragm-driven micropump. Figure 15 shows the geometry of the micropump. The pump is driven by the diaphragm, which oscillates sinusoidally in the vertical direction. The directionality of the flow through the pump is controlled by the two valves, which open and close in concert with the oscillating diaphragm. In real applications [60], the diaphragm is usually activated through electrostatic forcing and the valves are driven by the hydrodynamic force produced in the pump chamber due to the diaphragm oscillation. However, in the current computational model, since the objective is to demonstrate the capabilities of the method for complex

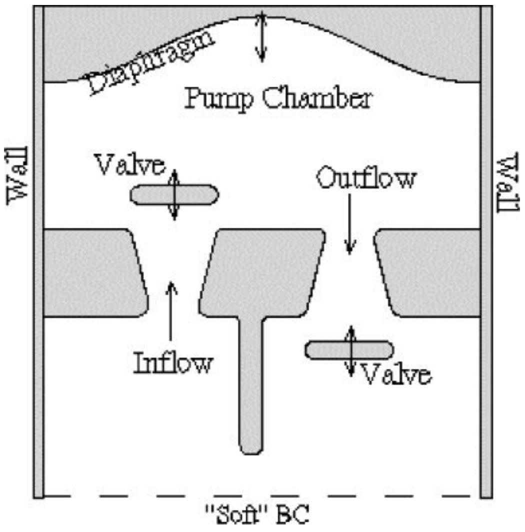

FIG. 15. Schematic of the micropump configuration simulated in the current study.

moving boundary cases, the valves are made to move with a prescribed motion. The sequence of valve motion is chosen so that fluid primarily enters the pump chamber from the left orifice and exits from the right. The inflow and outflow chamber are separated by a wall and a soft boundary condition is applied at the lower computational boundary, which allows inflow or outflow. This flow contains complex stationary as well as multiple moving boundaries. Such a configuration would pose a challenge to any structured or unstructured, body-conformal Lagrangian method and serves to demonstrate the capabilities of the current method. The current simulation of the micropump has been carried out on a uniform 200×224 $(x \times y)$ Cartesian mesh. The length, velocity, and time scale of the flow are chosen to be the diaphragm length, maximum diaphragm velocity, and time period of the diaphragm oscillation, respectively. Based on these, the Strouhal and Reynolds numbers chosen for the current simulation are 1.0 and 100, respectively. The time step size is such that it requires 2000 time steps to complete one cycle. The simulation is carried out until a stationary state is obtained and the results presented here correspond to this stationary state.

Figure 16 shows a sequence of four plots over one pumping cycle. Figure 16a corresponds to the maximum expulsion phase in the cycle. At this phase, the diaphragm is in a neutral position but moving down with its maximum velocity. The valve timing is chosen so that the left valve is closed and the right valve is fully open. This allows most of the flow to exit the pump chamber from the right exit and this is clearly shown by the velocity vectors. The pressure contours also show that a high pressure is created inside the pump chamber due to the motion of the diaphragm. Figure 16b shows the next phase in the pumping cycle where the diaphragm has reached its lowest position and is about to initiate its upward motion. Both the valves are open halfway and this allows flow through both of the orifices. The spanwise vorticity contours are plotted, showing the complex structure of the flow both inside the pump chamber and at the inlet and exit chambers. Figure 16c shows the flow at the maximum ingestion phase of the cycle where the left valve is fully open and right is closed. Finally, Figure 16d corresponds to the phase

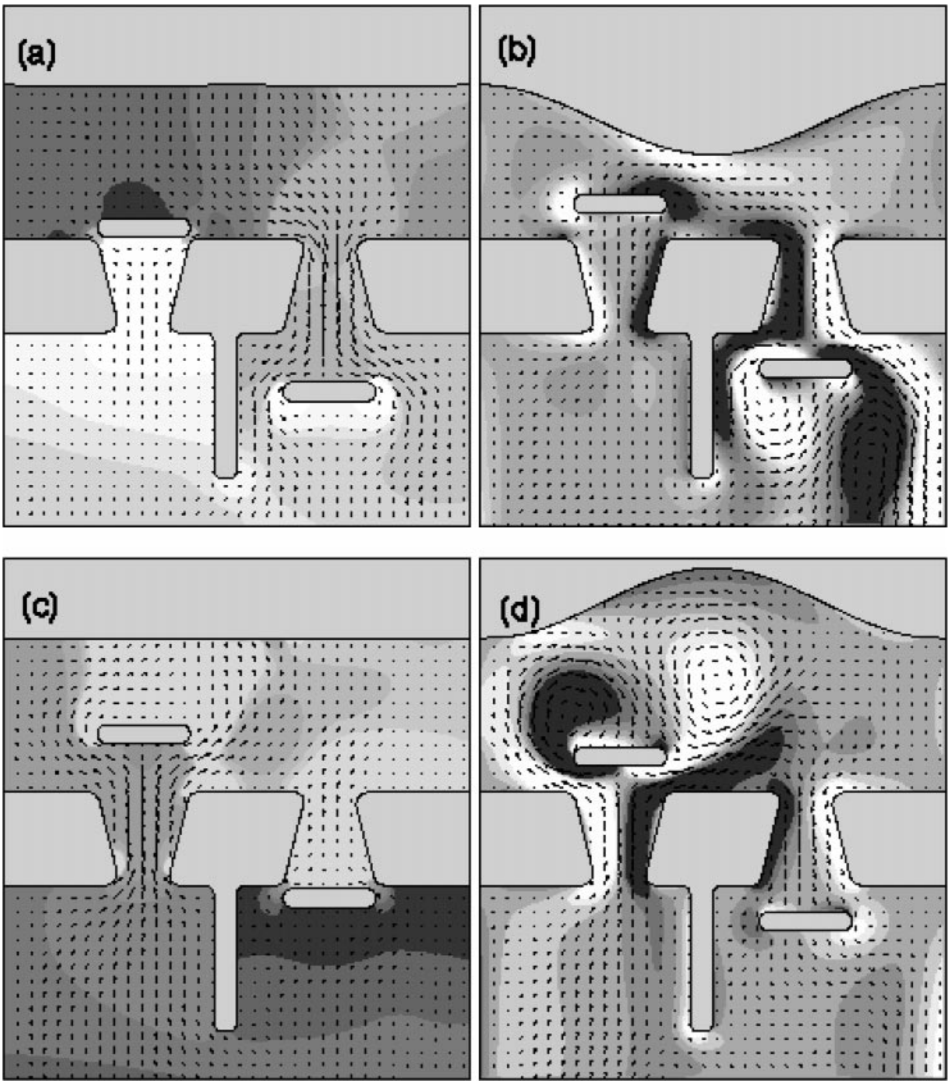

FIG. 16. Flow in a diaphragm-driven micropump. The plots show pressure contours, vorticity contours, and velocity vectors at four phases in the pumping cycle. Dark shades correspond to high pressure and clockwise vorticity, whereas lighter shades correspond to low pressure and counterclockwise vorticity. (a) Maximum expulsion phase. Velocity vectors and pressure contours are shown. (b) Minimum chamber volume phase. Velocity vectors and vorticity contours are shown. (c) Maximum ingestion phase. Velocity vectors and pressure contours are shown. (d) Maximum chamber volume phase. Velocity vectors and vorticity contours are shown.

where the diaphragm is fully deformed upward and the both valves are halfway open. Large-scale vortical structures are formed at this phase due to flow separation on the left valve.

It is worth pointing out that this simulation has been carried out on a desktop DEC Alpha workstation equipped with a 533-MHz, 21164 processor. On average, the simulation requires 8.5 CPU seconds per time step, which translates to 4.7 CPU hours per pumping cycle. Thus, an entire simulation covering about 10 pumping cycles can be completed in about 2 days.

4. SUMMARY

A method for simulating flow around complex moving boundaries on Cartesian grids has been presented. The advantage of the method is that since the body motion is independent of the mesh, problems associated with mesh reconfiguration and motion are circumvented. One key aspect of the current method is that the moving immersed boundary is represented as a sharp interface and this makes the method well suited to convection-dominated problems. The fractional-step method has been reformulated in the context of the current Eulerian–Lagrangian approach; details of this have also been discussed. A unique problem that arises in these fixed grid, sharp interface methods is the temporal discretization of the governing equations in the so-called freshly cleared cells. These are cells which were in the solid at one time step but emerge into the fluid at the next. A consistent and systematic remedy for such cells has been presented.

The convergence of a conventional line-SOR iterative solver has been accelerated by using a modified multigrid algorithm that appropriately accounts for the presence of the sharp, immersed boundary. In this method, a volume-fraction-type approach is applied for restriction, prolongation, and smoothing at the coarser levels and this removes any need for explicit reconstruction of the immersed boundary on the coarse grids levels. The performance of the multigrid vis-à-vis a single grid-level LSOR has been shown in the immersed boundary cases to be comparable to that for the method without immersed boundaries. Furthermore, the multigrid method has been shown to perform well for increasing complexity of the geometry caused by multiple boundaries embedded in the domain. A consequence of this is that the boundary motion does not lead to a significant increase in the required CPU time.

The solver is used to simulate two problems that have reliable experimental data for comparison. The first problem, namely the flow in a channel with a moving indentation in one wall, serves as a benchmark for techniques that simulate the interaction of moving solid boundaries with viscous, incompressible fluids. It is found that our computed results match very well the experimental and numerical data of Pedley and co-workers [34, 38]. In particular, the propagation speed of the eddies and the time and location of their breakup into smaller eddies are predicted correctly. The second case used for validation is flow past a cylinder oscillating transversely in a freestream. Simulations have been carried out over a range of oscillation frequencies and amplitudes and the frequency–amplitude envelope where vortex-shedding lock-on is observed matches well with the experiments of Koopman [26]. As a demonstration of the flexibility and efficiency of the method, flow has also been simulated in a model of a diaphragm-driven micropump containing complex stationary as well as moving immersed boundaries.

REFERENCES

1. R. E. Alcouffe, A. Brandt, J. E. Dendy, and J. W. Painter, The multi-grid method for the diffusion equation with strongly discontinuous coefficients, *SIAM J. Sci. Stat. Comput.* **2**, 430 (1981).
2. A. S. Almgren, J. B. Bell, P. Colella, and T. Marthaler, A Cartesian grid projection method for the incompressible Euler equations in complex geometries, *SIAM J. Sci. Comput.* **18**, 1289 (1997).
3. D. M. Anderson, G. B. McFadden, and A. A. Wheeler, Diffuse interface methods in fluid mechanics, *Ann. Rev. Fluid Mech.* **30**, 139 (1998).

4. S. A. Bayyuk, K. G. Powell, and B. van Leer, A simulation technique for 2-D unsteady inviscid flows around arbitrarily moving and deforming bodies of arbitrary geometry, Technical Paper 93-3391-CP (AIAA Press, Washington DC, 1993).
5. R. P. Beyer and R. J. Leveque, Analysis of a one-dimensional model for the immersed boundary method, *SIAM J. Numer. Anal.* **29**, 332 (1992).
6. J. U. Brackbill, D. B. Kothe, and C. Zemach, A continuum method for modeling surface tension, *J. Comput. Phys.* **100**, 25 (1992).
7. W. L. Briggs, *A Multigrid Tutorial.* (Soc. for Industr. of Appl. Math, Philadelphia, 2000).
8. M. Braza, P. Chassiang, and H. Ha Minh, Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder, *J. Fluid Mech.* **165**, 79 (1986).
9. G. Caginalp, Stefan and Hele–Shaw type models as asymptotic limits of the phase-field equations, *Phys. Rev. A* **39**, 5887 (1989).
10. A. J. Chorin, Numerical solution of the Navier–Stokes equations, *Math. Comput.* **22**, 745 (1968).
11. D. De Zeeuw and K. G. Powell, An adaptively refined Cartesian mesh solver for the Euler equations, Technical Paper 91-1542 (AIAA Press, Washington DC, 1991).
12. C. Dong, R. Skalak, K.-L. P. Sung, G. W. Schmid-Schonbein, and S. Chien, Passive deformation analysis of human leukocytes, *J. Biomech. Eng.* **110**, 27 (1988).
13. J. H. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics* (Springer-Verlag Berlin, New York, 1996).
14. R. Glowinski, T.-S. Pan, and J. Periaux, A fictitious domain method for Dirichlet problems and applications, *Comput. Methods Appl. Mech. Eng.* **111**, 283 (1994).
15. D. Goldstein, R. Handler, and L. Sirovich, Modeling of a no-slip surface with and external flow field, *J. Comput. Phys.* **105**, 354 (1993).
16. O. M. Griffin and M. S. Hall, Vortex shedding lock-on in a circular cylinder wake, in *Flow-induced Vibration* (Balkema, Rotterdam, 1995).
17. T. Y. Hou, Z. Li, S. Osher, and H. Zhao, A hybrid method for moving interface problems with application to the Hele–Shaw flow, *J. Comput. Phys.* **134**, 2, 236 (1997).
18. T. Y. Hou, J. S. Lowengrub, and M. J. Shelley, Removing stiffness from interfacial flows with surface tension, *J. Comput. Phys.* **114**, 312 (1994).
19. C. P. Jackson, A finite element study of the onset of vortex shedding in flow past variously shaped bodies, *J. Fluid Mech.* **182**, 23 (1987).
20. V. Jayaraman, H. S. Udaykumar, and W. Shyy, Adaptive unstructured grid for three-dimensional interface representation, *Numer. Heat Transfer B* **32**, 247 (1997).
21. H. Johansen and P. Collela, A Cartesian grid embedded boundary method for Poisson's equation on irregular domains, *J. Comput. Phys.* **147**, 60 (1998).
22. D. Juric and G. Tryggvasson, A front-tracking method for dendritic solidification, *J. Comput. Phys.* **123**, 127 (1996).
23. H.-C. Kan, H. S. Udaykumar, W. Shyy, and R. Tran-Son-Tay, Hydrodynamics of a compound drop with application to leukocyte modeling, *Phys. Fluids* **10**, 760 (1998).
24. A. Karma and W. J. Rappel, Quantitative phase-field modeling of dendritic growth in two and three dimensions, *Phys. Rev. E* **57**, 4323 (1998).
25. A. Khanna, *Three-dimensional Representation of Evolving Interfaces in Computational Fluid Dynamics*, MS thesis (University of Florida, Gainesville, FL, 2000).
26. G. H. Koopmann, The vortex wakes of vibrating cylinders at low Reynolds numbers, *J. Fluid Mech.* **28**, 501 (1967).
27. 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).
28. Z. Li and B. Soni, Fast and accurate numerical approach for Stefan problems and crystal growth, *Numer. Heat Trans.* **35**, 461 (1999).
29. J. R. Meneghini and P. W. Bearman, Numerical Simulation of High Amplitude Oscillatory Flow About a

Circular Cylinder Using the Discrete Vortex Method, AIAA Shear Flow Conf. Paper 93-3288 (AIAA Press, Washington, DC, 1993).

30. S. Mittal and V. Kumar, Finite element study of vortex-induced cross-flow and in-line oscillations of a circular at low Reynolds numbers, *Int. J. Numer. Meth. Fluids* **31**, 1087 (1999).
31. R. Mittal and S. Balachander, "Direct numerical simulation of flow past elliptic cylinders, *J. Comput. Phys.* **124**, 351 (1996).
32. S. A. Orszag, M. Isreaeli, and M. O. Deville, Boundary conditions for incompressible flows, *J. Sci. Comput.* **1**, 75 (1986).
33. S. Osher and J. A. Sethian, Fronts propagating with curvature dependent speed: Algorithms based in Hamilton–Jacobi formulations, *J. Comput. Phys.* **79**, 12 (1988).
34. T. J. Pedley and K. D. Stephanoff, Flow along a channel with a time-dependent indentation in one wall: The generation of vorticity waves, *J. Fluid Mech.* **160**, 337 (1985).
35. R. B. Pember, J. B. Bell, P. Colella, W. Y. Crutchfield, and M. L. Welcome, An adaptive Cartesian grid method for unsteady compressible flow in irregular regions, *J. Comput. Phys.* **120**, 278 (1995).
36. C. S. Peskin, Numerical analysis of blood flow in the heart, *J. Comput. Phys.* **25**, 220 (1977).
37. J. J. Quirk, An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two-dimensional bodies, *Comput. Fluids* **23**, 125 (1994).
38. M. E. Ralph and T. J. Pedley, Flow in a channel with a moving indentation, *J. Fluid Mech.* **190**, 87 (1988).
39. C. M. Rhie and W. L. Chow, Numerical study of the turbulent flow past an airfoil with trailing edge separation, *AIAA J.* **21**(11), 1525 (1983).
40. P. K. Stansby, The locking-on of vortex shedding due to the cross-stream vibration of circular cylinders in uniform and shear flows, *J. Fluid Mech.* **74**, 641 (1976).
41. J. M. Stockie and S. I. Green, Simulating the motion of flexible pulp fibers using the immersed boundary method, *J. Comput. Phys.* **147**, 147 (1998).
42. M. Sussman, P. Smereka, and S. J. Osher, A level-set method for computing solutions to incompressible two-phase flow, *J. Comput. Phys.* **114**, 146 (1994).
43. R. Temam, Remarks on the pressure boundary condition for the projection method, *Theor. Comput. Fluid Dyn.* **3**, 181 (1991).
44. X. Tong, C. Beckermann, and A. Karma, Velocity and shape selection of dendritic crystals in a forced flow, *Phys. Rev. E* **61**, R49–R52 (2000).
45. D. J. Tritton, Experiments on the flow past a circular cylinder at low Reynolds number, *J. Fluid Mech.* **6**, 547 (1959).
46. C. Tu and C. S. Peskin, Stability and instability in the computation of flows with moving immersed boundaries: A comparison of three methods, *SIAM J. Sci. Stat. Comput.* **13**, 1361 (1992).
47. H. S. Udaykumar, H.-C. Kan, W. Shyy, and R. Tran-Son-Tay, Multiphase dynamics in arbitrary geometries on fixed Cartesian grids, *J. Comput. Phys.* **137**, 366 (1997).
48. H. S. Udaykumar, R. Mittal, and W. Shyy, Solid-fluid phase front computations in the sharp interface limit on fixed grids, *J. Comput. Phys,* **153**, 535 (1999).
49. S. O. Unverdi and G. Tryggvason, A front-tracking method for viscous, incompressible multi-fluid flows, *J. Comput. Phys.*, **100**, 25 (1992).
50. R. Verzicco, J. Mohd-Yusof, P. Orlandi and D. Haworth "Large-eddy simulation in complex geometric configurations using boundary body forces, *AIAA J.* **38**(3), 427 (2000).
51. S.-L. Wang, R. F. Sekerka, A. A. Wheeler, B. T. Murray, S. R. Coriell, R. J. Braun, and G. B. McFadden, Thermodynamically consistent phase-field models for solidification, *Physica D* **69**, 189 (1993).
52. P. Wesseling, *Introduction to Multigrid Methods* (Wiley, New York, 1992).
53. R. Webster, Efficient algebraic multigrid solvers with elementary restriction and prolongation, *Int. J. Numer. Meth. Fluids* **28**, 317 (1998).
54. R. Webster, Algebraic multigrid solver for Navier–Stokes problems, *Int. J. Numer. Meth. Fluids* **18**, 761 (1994).

55. A. A. Wheeler, W. J. Boettinger, and G. B. McFadden, Phase field model for isothermal phase transitions in binary alloys, *Phys. Rev. A* **45**, 7424 (1992).

56. M. W. Williams, D. B. Kothe, and E. G. Puckett, *Convergence and accuracy of Kernel-based Continuum Surface Tension Models*, Los Alamos Report, LA-UR-98-2268 (1998).

57. C. H. K. Williamson, Vortex dynamics in the cylinder wake, *Ann. Rev. Fluid Mech.* **28**, 477 (1996).

58. T. Ye, R. Mittal, H. S. Udaykumar, and W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries, *J. Comput. Phys.* **156**, 209 (1999).

59. Y. Zang, R. L. Street, and J. R. Koseff, A non-staggered grid, fractional step method for time-dependent incompressible Navier–Stokes equations in curvilinear coordinates, *J. Comput. Phys.* **114**, 18 (1994).

60. R. Zengerle, S. Kluge, M. Richter, and A. Richter, A bidirectional silicon micropump, in SENSOR 95, Nurberg, 1995, pp. 727–732.