SIAM J. SCI. COMPUT.
Vol. 25, No. 3, pp. 832–856
© 2003 Society for Industrial and Applied Mathematics

AN IMMERSED INTERFACE METHOD FOR INCOMPRESSIBLE NAVIER–STOKES EQUATIONS*

LONG LEE† AND RANDALL J. LEVEQUE†

Abstract. The method developed in this paper is motivated by Peskin's immersed boundary (IB) method, and allows one to model the motion of flexible membranes or other structures immersed in viscous incompressible fluid using a fluid solver on a fixed Cartesian grid. The IB method uses a set of discrete delta functions to spread the entire singular force exerted by the immersed boundary to the nearby fluid grid points. Our method instead incorporates part of this force into jump conditions for the pressure, avoiding discrete dipole terms that adversely affect the accuracy near the immersed boundary. This has been implemented for the two-dimensional incompressible Navier–Stokes equations using a high-resolution finite-volume method for the advective terms and a projection method to enforce incompressibility. In the projection step, the correct jump in pressure is imposed in the course of solving the Poisson problem. This gives sharp resolution of the pressure across the interface and also gives better volume conservation than the traditional IB method. Comparisons between this method and the IB method are presented for several test problems. Numerical studies of the convergence and order of accuracy are included.

Key words. immersed boundary, immersed interface, incompressible flow, flexible membrane, surface tension, projection method, high-resolution finite-volume method, CLAWPACK

AMS subject classifications. 65M06, 76D05, 76D27

DOI. 10.1137/S1064827502414060

1. Introduction.

In a sequence of papers [19], [20], [21], Peskin and McQueen introduced the *immersed boundary (IB) method* for solving the viscous incompressible Navier–Stokes equations with moving elastic boundaries. This method is developed further in [22], [23], [24], [27]. While originally developed to model blood flow in the heart and through heart valves, it has since been used for many applications, particularly in biofluid dynamics where interesting fluid flow problems with immersed elastic membranes or structures abound. Examples include modeling the swimming of organisms [7], platelet aggregation in blood clotting [8], cochlear dynamics [1], biofilm processes [6], and wood pulp fiber dynamics [25].

The IB method represents the flexible membrane using Lagrangian markers at which force-densities are computed, based on the position of the membrane, and spread to a Cartesian fluid grid by a set of discrete delta functions. The support of the delta function is comparable to the mesh width. The Navier–Stokes equations with this forcing are then advanced one time step on the Cartesian grid using a projection method. The velocities are first updated, ignoring the pressure gradient term in the Navier–Stokes equations in a *transport step*, and then a *projection step* is applied to compute the pressure and impose the incompressibility constraint on the velocity field. The resulting velocities are interpolated back to the Lagrangian markers using the same set of discrete delta functions. The markers are moved based on these velocities and the process is repeated in the next time step. This algorithm is written out in more detail in section 3.

*Received by the editors September 5, 2002; accepted for publication (in revised form) June 24, 2003; published electronically November 21, 2003. This work was supported in part by DOE grants DE-FG03-96ER25292 and DE-FC02-01ER25474 and by NSF grants DMS-9803442 and DMS-0106511.

http://www.siam.org/journals/sisc/25-3/41406.html

†Department of Applied Mathematics, University of Washington, Box 352420, Seattle, WA 98195 (longlee@amath.washington.edu, rjl@amath.washington.edu).

Spreading the singular boundary force to the fluid grid results in large forces at the fluid grid points closest to the boundary. Using this force field to update velocities leads to large nonphysical velocities at these points. The discrete divergence of this velocity field has a discrete dipole behavior near the boundary. This is used as the right-hand side of a Poisson problem in the projection step of the algorithm, which projects to the space of divergence-free velocity fields and computes the pressure in the process. In general the pressure should be discontinuous across the interface and the computed pressure has approximately this behavior due to the discrete dipole nature of the right-hand side. While relatively simple to implement, this smearing of the force gives a method that is generally only first-order accurate.

We have found that the accuracy can be greatly improved by handling part of the singular force in a different manner, imposing the proper jump condition on the pressure directly in the course of solving the Poisson problem, and eliminating the discrete dipole nature of the forcing term. The jump in pressure is determined by the normal component of the force exerted by the interface. We still use discrete delta functions to spread the tangential component of the force. To impose the jump in pressure while solving the Poisson problem, we use an approach based on the *immersed interface method* (IIM) of LeVeque and Li [14], which was applied to Stokes flow problems in [15]. For our application here to a constant-coefficient Poisson problem, the complicated six-point stencil of [14] is not needed and the approach is closely related to Mayo's method for Poisson problems on irregular regions [17].

We refer to our method below as the IIM, though there are other ways in which IIMs could be used in the context of incompressible Navier–Stokes equations. In particular, Li and Lai [16] have recently developed a method that dispenses with discrete delta functions entirely. Their approach is contrasted with ours in sections 3 and 4. Our method also avoids discrete delta functions entirely in the case where the force is purely in the normal direction, as discussed at the end of section 2. This is a special case, but one that often arises in practice, e.g., when the force results from surface tension at a free boundary between different fluids.

In section 2 we review the nature of the solution near an interface where singular forces are applied and present the jump condition needed in our algorithm. The basic explicit form of the algorithm is presented in section 3. The correction terms required to implement the jump conditions are discussed further in section 4, and an implicit version of the algorithm is presented in section 5. In section 6 we show that our approach maintains certain steady states exactly and does not suffer from the leaking across a pressurized membrane that can be a problem with the IB method. The relation between our approach and the blob projection method of Cortez and Minion [4] is briefly explored in section 7. Finally, some numerical results are presented in section 8.

2. Jump conditions across the interface. The two-dimensional incompressible Navier–Stokes equations take the form

$$\begin{aligned} \boldsymbol{u}_t + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p &= \mu\nabla^2\boldsymbol{u} + \boldsymbol{F}, \\ \nabla\cdot\boldsymbol{u} &= 0, \\ \boldsymbol{u} &= \boldsymbol{b} \quad \text{on } \partial\Omega. \end{aligned} \tag{2.1}$$

We assume that the density $\rho \equiv 1$ is constant throughout the fluid, as is the viscosity μ. We impose Dirichlet boundary conditions on the velocity, e.g., $u = 0$ on no-slip boundaries, though periodic or Neumann conditions could also be used. We consider

these equations in two space dimensions and use bold symbols to denote vectors, with the convention $\boldsymbol{u} = (u, v)$ for the velocity field and $\boldsymbol{x} = (x, y)$ for spatial position.

The vector $\boldsymbol{F}$ represents an applied body force. In this paper we consider problems where this is a singular distribution applied along some interface $\Gamma(t)$ immersed in the fluid. This has the form

$$\boldsymbol{F}(\boldsymbol{x}, t) = \int_{\hat{s}_0}^{\hat{s}_1} \boldsymbol{f}(\hat{s}, t)\, \delta(\boldsymbol{x} - \hat{\boldsymbol{X}}(\hat{s}, t))\, d\hat{s}, \tag{2.2}$$

where $\hat{\boldsymbol{X}}(\hat{s}, t)$ is the parameterization of $\Gamma(t)$ in terms of arclength $\hat{s}$, and $\hat{\boldsymbol{f}}(\hat{s}, t)$ is the force per unit length along Γ exerted by the interface at the point $\hat{\boldsymbol{X}}(\hat{s}, t)$. The strength depends on the nature of the interface. Later in this section we present the formulas for an elastic membrane.

The singular force along Γ results in the solution to the equations being nonsmooth across the interface. Since we assume that the interface Γ moves with the fluid and there is no fluid slip at the interface, the velocity is continuous and $[\![\boldsymbol{u}]\!] = 0$ across the interface, where $[\![\cdot]\!]$ denotes the jump in a quantity across the interface. There may, however, be jumps in the pressure and in the normal derivatives of both the pressure and velocity at the interface. Let $\boldsymbol{n}(\hat{s}, t)$ be the unit normal vector to Γ and $\boldsymbol{\tau}(\hat{s}, t)$ the tangent vector so that the normal and tangential components of the force density are

$$\begin{aligned} \hat{\boldsymbol{f}}_n(\hat{s}, t) &= \hat{\boldsymbol{f}}(\hat{s}, t) \cdot \boldsymbol{n}(\hat{s}, t), \\ \hat{\boldsymbol{f}}_\tau(\hat{s}, t) &= \hat{\boldsymbol{f}}(\hat{s}, t) \cdot \boldsymbol{\tau}(\hat{s}, t). \end{aligned} \tag{2.3}$$

Corresponding forces $\boldsymbol{F}_n(x, t)$ and $\boldsymbol{F}_\tau(x, t)$ can be obtained by integrating these as in (2.2). Then at each point on Γ the solution to the incompressible Navier–Stokes equations satisfies the jump conditions

$$\begin{aligned} [\![p]\!](\hat{s}, t) &= \hat{\boldsymbol{f}}_n(\hat{s}, t), \\ \left[\!\!\left[\frac{\partial p}{\partial n} \right]\!\!\right](\hat{s}, t) &= \frac{\partial}{\partial \hat{s}} \hat{\boldsymbol{f}}_{\boldsymbol{\tau}}(\hat{s}, t), \\ [\![\boldsymbol{u}]\!](\hat{s}, t) &= 0, \\ \mu \left[\!\!\left[\frac{\partial \boldsymbol{u}}{\partial n} \right]\!\!\right](\hat{s}, t) &= -\hat{\boldsymbol{f}}_{\boldsymbol{\tau}}(\hat{s}, t) \boldsymbol{\tau}(\hat{s}, t). \end{aligned} \tag{2.4}$$

See, for example, [10] for discussions of these jump conditions.

It is often more convenient to use a parameterization s of the interface that flows with the fluid so that the interface location $\boldsymbol{X}(s, t)$ satisfies

$$\frac{\partial}{\partial t} \boldsymbol{X}(s, t) = \boldsymbol{u}(\boldsymbol{X}(s, t), t), \tag{2.5}$$

where $\boldsymbol{u}(x, t)$ is the fluid velocity. We assume that there is no flow of fluid through the interface and also that there is no slip along the interface so that the velocity $\boldsymbol{u}(\boldsymbol{X}(s, t), t)$ is uniquely defined. In practice we track the interface via a set of marker points $\boldsymbol{X}_k$ that are associated with fixed values s_k and that move with the fluid. In this case s is typically not arclength. Then the force formula (2.2) is replaced by

$$\boldsymbol{F}(\boldsymbol{x}, t) = \int_{s_0}^{s_1} \boldsymbol{f}(s, t)\, \delta(\boldsymbol{x} - \boldsymbol{X}(s, t))\, ds, \tag{2.6}$$

where $\boldsymbol{f}(s,t)$ is now the force density with respect to the measure ds. The force per unit length is given by

$$\frac{\boldsymbol{f}(s,t)}{|\partial \boldsymbol{X}(s,t)/\partial s|}. \tag{2.7}$$

Three of the jump conditions from (2.4) transform easily to expressions in terms of the normal and tangential components of $\boldsymbol{f}(s,t)$,

$$\begin{aligned} [\![p]\!](s,t) &= \boldsymbol{f}_n(s,t)/|\partial \boldsymbol{X}(s,t)/\partial s|, \\ [\![\boldsymbol{u}]\!](s,t) &= 0, \\ \mu \left[\!\!\left[\frac{\partial u}{\partial n} \right]\!\!\right](s,t) &= -\boldsymbol{f}_{\boldsymbol{\tau}}(s,t)\boldsymbol{\tau}(s,t)/|\partial \boldsymbol{X}(s,t)/\partial s|. \end{aligned} \tag{2.8}$$

The transformed version of $[\![\partial p/\partial n]\!]$ is more complicated since the tangential force must be differentiated with respect to arclength, but this can be reformulated in terms of $\boldsymbol{f}(s,t)$. In our formulation we do not explicitly use this jump condition and so we do not present the details.

As a particular form of interface problem, we consider the case of an elastic membrane under tension, with s representing arclength along the unstretched membrane and the force density

$$\boldsymbol{f}(s,t) = \frac{\partial}{\partial s}(T(s,t)\boldsymbol{\tau}(s,t)), \tag{2.9}$$

where $\boldsymbol{\tau}(s,t)$ is the unit tangent vector and the tension is given by

$$T(s,t) = T_0 \left(\left| \frac{\partial \boldsymbol{X}(s,t)}{\partial s} \right| - 1 \right), \tag{2.10}$$

with T_0 constant. See Peskin [20] for a derivation.

Other types of interfaces give rise to forces specified in different ways. In particular, if the interface is simply a free surface separating two different fluids, then the force will be given by surface tension and typically has the form

$$\hat{\boldsymbol{f}}(\hat{s},t) = \gamma \frac{\partial^2}{\partial \hat{s}^2} \boldsymbol{X}(\hat{s},t), \tag{2.11}$$

where γ is some constant. This force points purely in the normal direction. Note that in this case we can dispense with discrete delta functions entirely. Of course in such problems the two fluids typically have different densities and viscosities, which introduces other potential difficulties near the interface. This can be handled with an extension of our algorithm that is currently being developed.

3. The numerical algorithm. A standard approach to solving the Navier–Stokes equations (2.1) is the *projection method*, a fractional step method with the following basic form.

Transport step. Advance $\boldsymbol{U}^n$ to $\boldsymbol{U}^*$ by solving the advection-diffusion equation

$$\boldsymbol{u}_t + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = \mu \nabla^2 \boldsymbol{u} + \boldsymbol{F}_1 \tag{3.1}$$

over time Δt. This equation contains part or all of the force $\boldsymbol{F}$ as discussed below.

Projection step. Advance $\boldsymbol{U}^*$ to $\boldsymbol{U}^{n+1}$ by

$$\boldsymbol{U}^{n+1} = \boldsymbol{U}^* - \Delta t \nabla p^{n+1} + \Delta t \boldsymbol{F}_2, \tag{3.2}$$

where $\boldsymbol{F}_2 = \boldsymbol{F} - \boldsymbol{F}_1$ is the remainder of the force. This is a discretization of

$$\boldsymbol{u}_t + \nabla p = \boldsymbol{F}_2. \tag{3.3}$$

The pressure p^{n+1} is determined by solving the Poisson problem that arises from applying the discrete divergence operator to (3.2) and using the fact that we want $\nabla \cdot \boldsymbol{U}^{n+1} = 0$. This results in

$$\nabla^2 p^{n+1} = \frac{1}{\Delta t} \nabla \cdot \boldsymbol{U}^* + \nabla \cdot \boldsymbol{F}_2. \tag{3.4}$$

The original IB method of Peskin is of this form, with $\boldsymbol{F}_1 = \boldsymbol{F}$ and $\boldsymbol{F}_2 = 0$. It is implemented by spreading the singular force $\boldsymbol{F}$ to the uniform grid points by means of a discrete delta function. The intermediate velocity $\boldsymbol{U}^*$ is typically not divergence-free since both the nonlinear term $(\boldsymbol{u} \cdot \nabla)\boldsymbol{u}$ and the force $\boldsymbol{F}$ may fail to be. The projection step, with $\boldsymbol{F}_2 = 0$, then corresponds to a projection of $\boldsymbol{U}^*$ to $\boldsymbol{U}^{n+1}$, the divergence-free portion.

The main new feature of the algorithm we are proposing is that we split the singular force $\boldsymbol{F}$ into components tangential and normal to the interface via (2.3) and use these components for $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$, respectively. (Actually it is the force density $\boldsymbol{f}$ in (2.6) that is split into components and then $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$ defined by the corresponding integrals.) The tangential component $\boldsymbol{F}_1 = \boldsymbol{F}_\tau$ is spread to the grid using discrete delta functions as in the IB method. The normal component $\boldsymbol{F}_2 = \boldsymbol{F}_n$ is not spread to the grid. Instead, we use the fact that the jump in pressure across the interface should be equal to the normal component of the force. This jump condition is built into the Poisson solver in Step 2 using the immersed interface approach of LeVeque and Li [14]. We solve the equation

$$\nabla^2 p^{n+1} = \frac{1}{\Delta t} \nabla \cdot \boldsymbol{U}^* \tag{3.5}$$

together with the proper jump condition in p so that accurate pointwise values of the pressure can be computed even near the interface.

Note that since $\boldsymbol{F}_2$ has the form of a delta function, $\nabla \cdot \boldsymbol{F}_2$ is a dipole along the interface, which does indeed lead to a jump in p in the solution. In the original IB method, this jump arises in the following manner: The entire force is spread to the grid, giving discrete delta function behavior in the intermediate velocity $\boldsymbol{U}^*$. Applying the discrete divergence to this leads to discrete dipole behavior in $\nabla \cdot \boldsymbol{U}^*$, and using this as the right-hand side in the Poisson problem leads to a smeared jump in the numerical approximation to the pressure. Updating $\boldsymbol{U}^*$ by the discrete version of ∇p then should remove the nonphysical dipole behavior from the velocity. We believe that this discrete dipole in $\boldsymbol{U}^*$ and its use in the Poisson problem are the principal causes of inaccuracy in the IB method and that much better accuracy can be achieved by avoiding these discrete dipoles, imposing the correct jump conditions directly on p.

A more extreme version of this is proposed by Li and Lai [16], who impose all the jump conditions from (2.4) directly on the velocity and pressure and avoid any use of discrete delta functions. Their approach may give somewhat better accuracy than

ours but is more complicated to implement in general (only certain special cases are considered in [16]). We have found that simply handling the pressure jump properly gives a substantial improvement over the original IB method. This is a relatively simple change that could be applied to many existing IB codes. In section 4 we derive the correction term and present further justification for splitting the force into normal and tangential components and handling these by different approaches.

Our implementation also uses a new version of the projection method, based on a combination of high-resolution finite-volume methods for the advection portion of the equations and finite difference methods for the immersed interface solution of the Poisson problem. It is a "pressure-free projection method" in the terminology of [2], since no approximation to the pressure gradient appears in the transport step (3.1). See [2] for a survey of different approaches to designing projection methods and for a discussion of many issues related to the accuracy of these methods. Our algorithm is briefly summarized below. For more details, see [10].

We maintain two different velocity fields, cell-centered values $\boldsymbol{U}^n = (U^n_{ij}, V^n_{ij})$ that represent cell averages over the (i,j) grid cell, and edge values $u^n_{i-1/2,j}$ and $v^n_{i,j-1/2}$ that represent approximations to the normal velocity at the left and bottom edges of the (i,j) grid cell, respectively. At the beginning of each time step, we assume that the edge velocities $\boldsymbol{u}^n$ are discrete divergence-free, $\nabla_h \cdot \boldsymbol{u}^n = 0$, where

$$(\nabla_h \cdot \boldsymbol{u}^n)_{ij} = \frac{u^n_{i+1/2,j} - u^n_{i-1/2,j}}{\Delta x} + \frac{v^n_{i,j+1/2} - v^n_{i,j-1/2}}{\Delta y}. \tag{3.6}$$

At the beginning of the time step we also have a set of points $\boldsymbol{X}^n_k$ for $k = 1,\ 2,\ \ldots,\ N$ that mark the location of the interface. This vector of points is denoted $\boldsymbol{X}^n$.

The algorithm then takes the following form.

Step 1. Solve the advection equation

$$\boldsymbol{U}_t + (\boldsymbol{u}^n \cdot \nabla)\boldsymbol{U} = 0 \tag{3.7}$$

over time Δt to update the cell averages $\boldsymbol{U}^n$ to values $\boldsymbol{U}^\dagger$. We use the high-resolution finite-volume method implemented in CLAWPACK [11], [12], [13]. The edge velocities $\boldsymbol{u}^n$ are used in solving Riemann problems at the cell edges. Limiters are then applied to avoid nonphysical oscillations near steep gradients.

Step 2. Use $\boldsymbol{U}^\dagger$ as data to solve the diffusion equation along with the tangential portion of the force,

$$\boldsymbol{U}_t = \mu \nabla^2 \boldsymbol{U} + \boldsymbol{F}_\tau. \tag{3.8}$$

The force is transferred from the interface to nearby cell centers using the discrete delta function

$$d_h(\boldsymbol{x}) = \delta_h(x)\delta_h(y), \qquad \text{with} \quad \delta_h(r) = \begin{cases} \dfrac{1}{4h}\left(1 + \cos\dfrac{\pi r}{2h}\right), & |r| \le 2h, \\ 0, & |r| \ge 2h. \end{cases} \tag{3.9}$$

The equation (3.8) is then discretized using a Crank–Nicolson method over time Δt to yield new cell-centered values $\boldsymbol{U}^*$.

Step 3. Average $\boldsymbol{U}^*$ from adjacent grid cells to obtain edge values $\boldsymbol{u}^*$,

$$\begin{aligned} u^*_{i-1/2,j} &= \frac{1}{2}(U^*_{i-1,j} + U^*_{ij}), \\ v^*_{i,j-1/2} &= \frac{1}{2}(V^*_{i,j-1} + V^*_{ij}). \end{aligned} \tag{3.10}$$

In general $\nabla_h \cdot \boldsymbol{u}^* \neq 0$.

Step 4. Compute p^{n+1} by solving a discrete version of

$$\Delta t \, \nabla^2 p^{n+1} = \nabla \cdot \boldsymbol{u}^* \tag{3.11}$$

with the jump condition $[\![p^{n+1}]\!] = f_n(s,t)/|\partial \boldsymbol{X}(s,t)/\partial s|$. This is accomplished by solving the discrete system

$$\nabla_h^2 p_{ij}^{n+1} = \frac{1}{\Delta t}(\nabla_h \cdot \boldsymbol{u}^*)_{ij} + C_{ij}, \tag{3.12}$$

where ∇_h^2 is the discrete five-point Laplacian and C_{ij} are correction terms needed to accurately impose the jump conditions on p. These values will only be nonzero in cells near the interface and are specified below.

Step 5. Update the edge velocities by

$$\begin{aligned} u_{i-1/2,j}^{n+1} &= u_{i-1/2,j}^* - \Delta t[(p_{ij} - p_{i-1,j})/\Delta x - B_{i-1/2,j}^1], \\ v_{i,j-1/2}^{n+1} &= v_{i,j-1/2}^* - \Delta t[(p_{ij} - p_{i,j-1})/\Delta y - B_{i,j-1/2}^2]. \end{aligned} \tag{3.13}$$

The correction terms $B_{i-1/2,j}^1$ and $B_{i,j-1/2}^2$ are needed to take into account the jump conditions in p so that accurate pointwise values of the derivative are captured,

$$\begin{aligned} (p_{ij} - p_{i-1,j})/\Delta x - B_{i-1/2,j}^1 &\approx p_x(x_{i-1/2}, y_j), \\ (p_{ij} - p_{i,j-1})/\Delta y - B_{i,j-1/2}^2 &\approx p_y(x_i, y_{j-1/2}). \end{aligned} \tag{3.14}$$

These terms are derived below.

Step 6. Update the cell-centered velocities by

$$\begin{aligned} U_{ij}^{n+1} &= U_{ij}^* - \Delta t[(p_{i+1,j} - p_{i-1,j})/(2\Delta x) - B_{ij}^1], \\ V_{ij}^{n+1} &= V_{ij}^* - \Delta t[(p_{i,j+1} - p_{i,j-1})/(2\Delta y) - B_{ij}^2], \end{aligned} \tag{3.15}$$

where

$$\begin{aligned} B_{ij}^1 &= \frac{1}{2}(B_{i-1/2,j}^1 + B_{i+1/2,j}^1), \\ B_{ij}^2 &= \frac{1}{2}(B_{i,j-1/2}^2 + B_{i,j+1/2}^2). \end{aligned} \tag{3.16}$$

This is consistent with the manner in which the edge velocities are updated and gives the pressure corrections for the cell-centered velocities.

Step 7. Interpolate the velocities to the Lagrangian points $\boldsymbol{X}_k^n$ that mark the interface location. Bilinear interpolation from cell-centered velocities $\boldsymbol{U}_{ij}^{n+1}$ is used to evaluate velocities at the points $\boldsymbol{X}^n$. The resulting vector of velocities is denoted by $\mathcal{U}^{n+1}$. The marker points are then moved with the fluid,

$$\boldsymbol{X}^{n+1} = \boldsymbol{X}^n + \Delta t \, \mathcal{U}^{n+1}(\boldsymbol{X}^n). \tag{3.17}$$

A more stable implicit method is described in section 5 below, in which this is replaced by

$$\boldsymbol{X}^{n+1} = \boldsymbol{X}^n + \frac{1}{2}\Delta t \, (\mathcal{U}^n(\boldsymbol{X}^n) + \mathcal{U}^{n+1}(\boldsymbol{X}^{n+1})). \tag{3.18}$$

This completes a time step, and we return to Step 1.

We should note that the quantity p^{n+1} computed by this algorithm is probably at most a first-order accurate approximation to the pressure. This quantity is called ϕ^{n+1} in [2] and they note, following Kim and Moin [9], that for "pressure-free projection methods" a more accurate approximation to the pressure is obtained as

$$P^{n+1/2} = \phi^{n+1} - \frac{\mu \Delta t}{2} \nabla^2 \phi^{n+1}. \tag{3.19}$$

This does not affect the accuracy of the velocities, however, since our p^{n+1} is the proper term to use in the velocity updates. It is only if the pressure itself is needed to high accuracy that the postprocessing correction (3.19) may be desirable.

4. Correction terms. We now discuss the choice of the correction terms B and C. First we recall that at the end of the time step we need the edge velocities $\boldsymbol{u}^{n+1}$ to be divergence-free in preparation for the next step. Taking the discrete divergence of the expressions (3.13) used in Step 5 yields

$$\nabla_h \cdot \boldsymbol{u}^{n+1} = \nabla_h \cdot \boldsymbol{u}^* - \Delta t \, [\nabla_h^2 p^{n+1} - \nabla_h \cdot \boldsymbol{B}]. \tag{4.1}$$

This will be zero provided that p^{n+1} is defined by solving the discrete Poisson problem (3.12) with a set of correction terms that satisfy

$$\begin{aligned} C_{ij} &= (\nabla_h \cdot \boldsymbol{B})_{ij} \\ &= \frac{B^1_{i+1/2,j} - B^1_{i-1/2,j}}{\Delta x} + \frac{B^2_{i,j+1/2} - B^2_{i,j-1/2}}{\Delta y}. \end{aligned} \tag{4.2}$$

Thus we need only determine the correction terms B^1 and B^2 and the C terms can be computed from (4.2).

The correction terms B^1 and B^2 are determined by the requirements (3.14), together with the jump conditions on p. If the interface lies between cell centers $(i-1, j)$ and (i, j), then let $[\![p]\!]_{i-1/2,j}$ denote the jump in p at the point where the interface cuts the line segment between these cell centers. We assume we can evaluate the force and hence this jump at any point along the interface. (In practice this is done using a cubic spline interpolant through the forces computed at the marker points.) If the interface does not cut between these cell centers, then we set $[\![p]\!]_{i-1/2,j} = 0$. In either case, a Taylor series expansion then gives

$$p_{ij} = p_{i-1,j} + [\![p]\!]_{i-1/2,j} + \Delta x \, p_x(x_{i-1/2,j}, y_j) + O(\Delta x^2), \tag{4.3}$$

provided we assume that there is no jump in p_x at the interface. We will discuss this assumption in a moment. Ignoring the $O(\Delta x^2)$ term in (4.3) and comparing with (3.14) shows that we should take

$$B^1_{i-1/2,j} = \frac{[\![p]\!]_{i-1/2,j}}{\Delta x}. \tag{4.4}$$

Similarly, we find that

$$B^2_{i,j-1/2} = \frac{[\![p]\!]_{i,j-1/2}}{\Delta y}, \tag{4.5}$$

where $[\![p]\!]_{i,j-1/2}$ is defined in an analogous manner as the jump in p across the portion of the interface cutting between cell centers $(i, j-1)$ and (i, j), if it does so, and zero otherwise.

The Taylor series expansion (4.3) is based on the assumption that there is no jump in p_x at the interface. In general there should be a jump in the derivatives of p at the interface according to the jump conditions on $[\![\partial p/\partial n]\!]$ in (2.4). However, this will naturally arise in the numerical solution to the Poisson problem because the right-hand side of (3.12) contains the discrete divergence of $\boldsymbol{u}^*$ as well as the correction term C_{ij}. In computing $\boldsymbol{u}^*$ we spread the tangential component of the force using discrete delta functions, and so the $\nabla_h \cdot \boldsymbol{u}^*$ term provides an approximation to the appropriate singularity on the right-hand side needed to obtain the correct jump in $\partial p/\partial n$, while the correction term C_{ij} provides the singularity needed to obtain the correct jump in p. In the original IB method, the singularities needed for both sets of jumps are provided by the $\nabla_h \cdot \boldsymbol{u}^*$ term, whereas we have eliminated part of this singularity from $\nabla_h \cdot \boldsymbol{u}^*$ and provided a more accurate form of this singularity in C_{ij}.

One could go further and remove all singularity from $\nabla_h \cdot \boldsymbol{u}^*$ by not including the tangential forces $\boldsymbol{F}_\tau$ in (3.8) and then rederive the correction terms C_{ij} to also include the jump conditions on $\partial p/\partial n$. This is the approach used by Li and Lai [16], but it is considerably more complicated and the jumps in derivatives of the velocity (the last jump conditions in (2.4)) must also be explicitly incorporated in a different manner with this approach. We believe it is natural to spread the tangential force to the grid when updating the velocity according to (3.8), because the tangential force does in fact lead to an acceleration of the fluid in the tangential direction, by dragging the membrane through the viscous fluid. Note that this force leads to a jump in $\partial p/\partial n$ only if the magnitude of the force varies along the interface (it is the tangential derivative of the tangential force that appears in the jump condition on $\partial p/\partial n$ in (2.4)). This is because the fluid is incompressible and so differential acceleration along the membrane leads to compensating pressure gradients rather than compression of the fluid. Ideally one might like to include only the divergence-free portion of the tangential force in (3.8) and impose the remainder of the force via additional correction terms in the Poisson solve. However, the divergence-free portion of the singular force is no longer supported only on the interface, and would affect the solution everywhere, so that a local spreading of this force to grid points near the interface would no longer be sufficient. The "blob projection method" is one nonlocal approach that uses this idea, as discussed further in section 8.

We feel that the jump in pressure itself is the dominant source of error in the traditional IB method and that trying to more accurately capture the jump in its normal derivative is of more marginal value. Luckily it is the jump in pressure that is easy to capture with the relatively minor change to the traditional method proposed in this paper.

Recall also that for some problems the interface force is entirely in the normal direction. In particular, if the interface is a free surface between two fluids rather than an elastic membrane, then the surface tension force (2.11) has magnitude proportional to the curvature and points in the normal direction. In this case there is no singular force in (3.8) and $\partial p/\partial n$ is continuous. Also in this case the correction terms derived in this section capture all the singularity of the force.

For the elastic membrane problems considered in this paper the tangential component of the force is nonzero, but by monitoring the force during the calculations we have found that the magnitude of the tangential component is typically less than 1%

of the normal force (for these particular problems). This gives additional justification for spreading this force via discrete delta functions rather than a more expensive treatment.

5. An implicit method. In this section we discuss the manner in which the motion of the interface markers $\boldsymbol{X}_k$ can be made implicit in order to improve the stability of the method and allow reasonable time steps to be taken. We would like the time steps to be determined by the CFL condition needed for the advection step. The high-resolution method used for this step only requires that the Courant number be less than 1,

$$\max_{i,j}\left(\frac{\Delta t}{\Delta x}|u_{i-1/2,j}|,\ \frac{\Delta t}{\Delta y}|v_{i,j-1/2}|\right) \le 1, \tag{5.1}$$

as shown in [12]. To achieve this, the viscous diffusion step must of course be done implicitly, and we use a standard Crank–Nicolson method for this in Step 2. But we must also typically make the interface motion of Step 7 implicit, which is more complicated to implement. The simple explicit approach (3.17) is based on interpolating the updated velocity field $\boldsymbol{U}^{n+1}$ to the old interface locations $\boldsymbol{X}^n$ and moving these points using forward Euler. This can lead to an unstable method when the membrane is stiff unless very small time steps are used, since a small perturbation of the interface can lead to large forces and hence large transient velocities near the interface. With the explicit method, this typically causes the membrane to overshoot its proper location, resulting in even larger forces at the next time step and an exponentially growing instability. This stiffness is well known in IB methods, and various semi-implicit or implicit approaches have been used to overcome it, e.g., [18], [27]. Here we use essentially the same fully implicit quasi-Newton approach as developed for Stokes flow in [15], although it is more complicated to implement for Navier–Stokes.

We use a trapezoidal method of the form (3.18), where $\mathcal{U}^n(\boldsymbol{X}^n)$ is obtained by interpolating $\boldsymbol{U}^n$ to the old locations $\boldsymbol{X}^n$, and $\mathcal{U}^{n+1}(\boldsymbol{X}^{n+1})$ results from interpolating the new velocity field $\boldsymbol{U}^{n+1}$ to the new locations $\boldsymbol{X}^{n+1}$. But a crucial aspect in making this fully implicit is that the new velocity field $\boldsymbol{U}^{n+1}$ is itself computed by using the new interface locations $\boldsymbol{X}^{n+1}$. In Step 2 of the algorithm, the trapezoidal method is applied to (3.8) in such a way that $\boldsymbol{F}_\tau$ is discretized using the average of the forces at t_n based on locations $\boldsymbol{X}^n$ and the forces at t_{n+1} based on $\boldsymbol{X}^{n+1}$. Similarly, the correction terms (4.4) and (4.5) used in the projection step are evaluated by computing the jumps in p at the old interface location and also the jump in p at the new interface location, and then averaging the two. This process is simplified by the fact that the force exerted by the membrane described by marker points $\boldsymbol{X}_k$ depends only on the location of these points, and so for given vectors $\boldsymbol{X}^n$ and $\boldsymbol{X}^{n+1}$ we can compute the forces and jumps needed. The complication is that, although $\boldsymbol{X}^n$ is already known, the correct $\boldsymbol{X}^{n+1}$ is not known when Steps 2–6 are being performed. Instead it is necessary to embed these steps in an iterative method of the following form.

Step I1. Apply Step 1, the advection step, which does not require $\boldsymbol{X}^{n+1}$.

Step I2. Make a guess $\boldsymbol{X}^{[0]}$ at $\boldsymbol{X}^{n+1}$ based on the previous motion of the interface, e.g.,

$$\boldsymbol{X}^{[0]} = 2\boldsymbol{X}^n - \boldsymbol{X}^{n-1}. \tag{5.2}$$

Set $I = 0$.

Step I3. Perform Steps 2–6 using $\boldsymbol{X}^{[I]}$ as $\boldsymbol{X}^{n+1}$. The result is a provisional velocity field $\boldsymbol{U}^{n+1}$.

Step I4. Evaluate

$$g(\boldsymbol{X}^{[I]}) = \boldsymbol{X}^{[I]} - \boldsymbol{X}^n - \frac{1}{2}\Delta t \left(\mathcal{U}^n(\boldsymbol{X}^n) + \mathcal{U}^{n+1}(\boldsymbol{X}^{[I]})\right), \tag{5.3}$$

where $\mathcal{U}^{n+1}(\boldsymbol{X}^{[I]})$ are the interpolated velocities at $\boldsymbol{X}^{[I]}$ based on the provisional velocity field $\boldsymbol{U}^{n+1}$.

Step I5. If $\|g(\boldsymbol{X}^{[I]})\| \le \epsilon$ for some tolerance, then set $\boldsymbol{X}^{n+1} = \boldsymbol{X}^{[I]}$ and we are done with this time step. Otherwise, update $\boldsymbol{X}^{[I]}$ to $\boldsymbol{X}^{[I+1]}$, set $I = I + 1$, and go back to step I3.

Updating $\boldsymbol{X}^{[I]}$ in Step I5 is done using a quasi-Newton method, such as BFGS or SR1 (symmetric rank-one update of Broyden's method) [26]. We have tried both algorithms and used SR1 for the results presented here. Our iteration is designed to find a zero of the function $g(\boldsymbol{X})$ defined by (5.3). The Jacobian cannot be calculated directly since evaluation of this function requires applying Steps 2 through 6 of the algorithm. But in practice we have a good estimate of the Jacobian from the previous time step and typically one or two iterations of the quasi-Newton algorithm gives convergence along with an updated approximation to the Jacobian matrix.

6. Preservation of steady state. Our algorithm has the nice feature that certain steady states are exactly maintained. In particular, consider a circular stretched membrane enclosing some incompressible fluid, a *pressurized membrane*. This is a two-dimensional model for an inflated balloon and is an exact steady state with $\boldsymbol{u} \equiv 0$ and a piecewise constant pressure having two values $p_{\text{in}} > p_{\text{out}}$. The pressure is larger inside the membrane and the jump in pressure equals the force exerted by the membrane, which is purely a normal force with constant magnitude around the membrane. A circular bubble with force exerted by surface tension has the same steady state.

With our IIM, this steady state is exactly maintained numerically. To see this, consider one step of the algorithm with $\boldsymbol{U}^n = 0$ and $\boldsymbol{u}^n = 0$. Since $\boldsymbol{F}_\tau = 0$, we obtain $\boldsymbol{U}^* = 0$ and $\boldsymbol{u}^* = 0$ after applying Steps 1–3. By our choice of correction terms, we find that $B^1_{i-1/2,j}$ and $B^2_{i,j-1/2}$ always take one of the values 0 or $\pm(p_{\text{in}} - p_{\text{out}})$. It can be verified that Step 4 then results in p_{ij} being piecewise constant with value p_{in} or p_{out}, so p is computed exactly at the grid points. Then the update terms in both (3.13) and (3.15) turn out to be exactly zero everywhere, since any jump in p is exactly cancelled by a correction term B, and so $\boldsymbol{u}^{n+1} = \boldsymbol{u}^* = 0$ and $\boldsymbol{U}^{n+1} = \boldsymbol{U}^* = 0$. Hence the steady state is maintained.

By contrast, the original IB method does not exactly maintain a steady state of this form. The incorporation of the normal force in the transport step leads to $\boldsymbol{u}^* \neq 0$, and using its discrete divergence on the right-hand side of the Poisson problem does not lead to a pressure that is exactly correct. While the resulting $\boldsymbol{U}^{n+1}$ may be small on a fine grid, there is typically some inward motion of the interface, a "leaking" of the fluid from the high pressure region. This is often observed with IB methods and is illustrated in some of the examples of section 8. The improvement of Peskin and Printz [24] greatly reduces but does not eliminate this effect.

7. Relation to the blob projection method. Another approach to improving the IB method has recently been studied by Cortez and Minion [4]. It is interesting to compare our approach with theirs, since they also avoid taking the divergence of discrete delta functions. Their *blob projection method* is based on the observation that only the divergence-free portion of the force $\boldsymbol{F}$ contributes to the time derivative of $\boldsymbol{u}$. This is seen by applying $\mathbb{P}$ to the first equation of (2.1), where $\mathbb{P}$ projects to

the divergence-free subspace. Since $\nabla \cdot \boldsymbol{u} = 0$ at all times, $\boldsymbol{u}_t$ is left alone while $\boldsymbol{F}$ is replaced by $\mathbb{P}\boldsymbol{F}$. In the blob projection method, the singular force $\boldsymbol{F}$ is first discretized as a sum of discrete delta functions centered at points on the interface, as in the IB method, using a discrete delta function $d_h(\boldsymbol{x})$ of the form (3.9), for example. But now Cortez and Minion observe that $\mathbb{P}d_h(\boldsymbol{x})$ can be computed analytically, giving a "dipole-field" velocity. Summing the contributions from each boundary force blob at each grid point gives the velocity update without the need to compute a discrete divergence of this force for the right-hand side of a Poisson problem. On the other hand, the dipole-field velocity is nonlocal and nonzero at all grid points, and so a special fast algorithm discussed in [4] must be used to compute these sums efficiently.

There is presumably still an effect of smearing the normal force over discrete blobs, but with the better $d_h(\boldsymbol{x})$ used in [4] and fourth-order accurate algorithms, very good results are reported. In particular, good volume conservation is obtained, although it does not appear that steady states of the form discussed in section 6 can be exactly maintained. Each $\mathbb{P}d_h(\boldsymbol{x})$ based on the normal force gives a velocity update that is nonzero everywhere. Mathematically, integrating the velocity updates that would result from projecting an exact delta function at each point on the boundary should yield complete cancellation in the case where the interface is circular with a normal force of constant magnitude. Numerically, however, when this force is discretized by blobs and the update evaluated at discrete grid points, it is unlikely that exact cancellation will occur.

This method seems more difficult to implement efficiently than our approach. Also, the analytic form of $\mathbb{P}d_h(\boldsymbol{x})$ needed depends on the boundary conditions on $\boldsymbol{u}$ imposed on the computational domain, and so this approach may be limited to particular boundary conditions in simple geometry.

Philosophically, there are connections between our method and the blob projection method that are worth noting. We update $\boldsymbol{u}^*$ using only the tangential component of the force. The normal force goes entirely into the pressure jump. The tangential force fails to be divergence free only if its magnitude varies along the interface, as discussed in section 4, while the normal force is nearly orthogonal to the space of divergence-free vector fields. By using only $\boldsymbol{F}_\tau$ in the transport step, we are essentially performing an approximate projection of the force, but in a way that maintains locality near the boundary. The errors in this projection are fixed in Steps 5 and 6, using the solution p to the nonlocal Poisson problem.

8. Numerical results. In this section we present some numerical results for several test problems with immersed elastic membranes. These problems all involve a pressurized circular membrane whose shape is deformed (the two-dimensional analogue of an inflated balloon). In the first example we compare our results with an asymptotic solution and numerical solutions in the literature. In Examples 2 and 3 we present comparisons of our IIM results with our own implementation of the IB method. The algorithm described in section 3 is used for both the IB method and the IIM, with only one change. For the IB method, the force $\boldsymbol{F}_\tau$ in (3.8) is replaced by the full force $\boldsymbol{F}$. This force is spread via the discrete delta function. In this case no correction terms are added in the projection step or the updates to velocities, i.e., $B^1_{i-1/2,j} = B^2_{i,j-1/2} = C_{ij} = 0$. Otherwise the same advection, diffusion, and projection methods are used for both algorithms. We believe that this demonstrates both that our method is converging to the correct solution (since it agrees asymptotically with the IB results) and that considerable improvement in accuracy is achieved by the more accurate treatment of pressure.

In Examples 2 and 3 we will see that our implementation of IB exhibits "leaking" for large times, with the area interior to the membrane decreasing with time. This is a problem that is often seen with the original IB method when the jump in pressure is sufficiently large. This problem can be addressed by using the modified projection method introduced by Peskin and Printz [24], which was specially designed to improve volume conservation of the original IB method. We have not implemented this correction for two reasons. First, it is developed in [24] only for periodic boundary conditions. It is a more complicated projection operator with a 5×5 stencil and to the best of our knowledge it has not been extended to solid wall boundary conditions. Second, we wished to demonstrate that a significant improvement in results can be achieved simply by handling the pressure jump in the more accurate manner, which we believe is easier to modify in existing codes than the projection method.

Example 1. We first consider an example used by Cortez and Varela [5], who present an asymptotic solution that can be used for comparison purposes. A closed elastic membrane in an inviscid fluid is perturbed slightly from a circular steady state and allowed to oscillate. Because the fluid is inviscid and can slip along the interface, the force exerted by the membrane is in the normal direction only and is assumed to have the form

$$\boldsymbol{f}(s,t) = T(t)\kappa(s,t)\boldsymbol{n}(s,t), \tag{8.1}$$

where $\boldsymbol{n}(s,t)$ is the unit normal and $\kappa(s,t)$ is the curvature. Moreover, the tension $T(t)$ is assumed to be constant along the membrane at each time t with magnitude given by some stiffness factor σ multiplied by the length of the membrane, which we compute using Simpson's rule on the cubic spline through the control points $\boldsymbol{X}_k(t)$. We take $\sigma = 1$.

Note that since the force is in the normal direction, we do not need to use discrete delta functions, and the force is imposed entirely through the jump in pressure in the IIM algorithm.

The initial shape of the membrane is given by

$$r_0(\theta,\, t_0) = (1 - \varepsilon^2/2)^{1/2} + \varepsilon\, \cos(k\theta), \tag{8.2}$$

which specifies the radius of the membrane at each angle θ. For $\varepsilon = 0$ this is just a circle. We take $k = 2$ and $\varepsilon = 0.1$ and use N_b marker points $\boldsymbol{X}_k(t)$ that are initially uniformly spaced on the membrane. The problem is solved with a uniform $N \times N$ grid on the domain $[-2, 2] \times [-2, 2]$ so the mesh spacing is $h = 4/N$. The distance between marker points is roughly $ds \approx 2\pi/N_b$ and we take $N_b \approx N/4$ so that $ds \approx 2\pi h$. Note that the spacing between marker points is about six times greater than the mesh spacing.

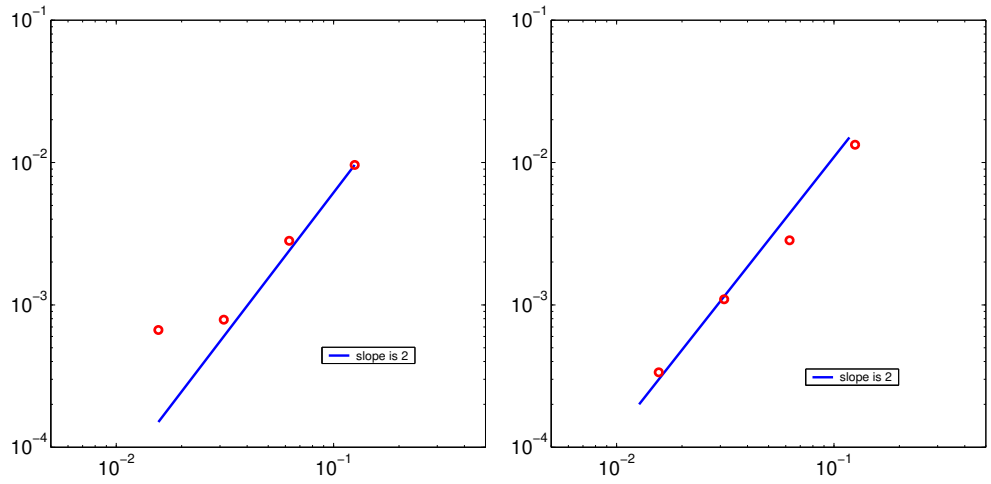
The asymptotic solution derived in [5] for small ε is then given by

$$r(\theta,\, t) = \left[1 - \frac{\varepsilon^2}{4}A^2(t)\right] + \varepsilon A(t)\cos(k\theta) + \varepsilon^2 B(t)\cos(2k\theta), \tag{8.3}$$

where the amplitudes $A(t)$ and $B(t)$ are given by (for $k = 2$)

$$A(t) = \cos\left(\sqrt{6\pi}\left(1 - \frac{223}{480}\varepsilon^2\right)t\right), \tag{8.4}$$

$$B(t) = \frac{3}{20} + \frac{1}{3}\cos(2\sqrt{6\pi}\, t) - \frac{29}{60}\cos(\sqrt{60\pi}\, t). \tag{8.5}$$



(a) (b)

FIG. 8.1. (a) *The error for Example* 1 *in the norm (8.6) at time* $t = 0.3$*, relative to the asymptotic solution* (8.3). (b) *The error relative to a reference solution on a* 512×512 *mesh.*

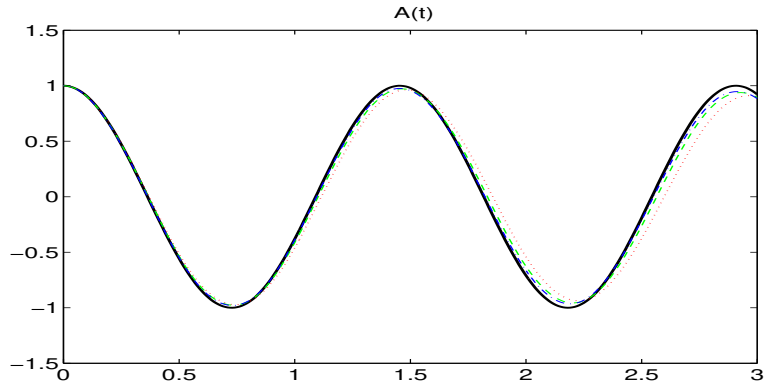
To compare our computed solution to this, we compute $r_k(t)$ and $\theta_k(t)$ from the marker locations $\boldsymbol{X}_k(t)$ and measure the 2-norm of the error in radial position as

$$\sqrt{\frac{1}{N_b}\sum_{k=1}^{N_b}(r_k(t) - r(\theta_k(t), t))^2}. \tag{8.6}$$

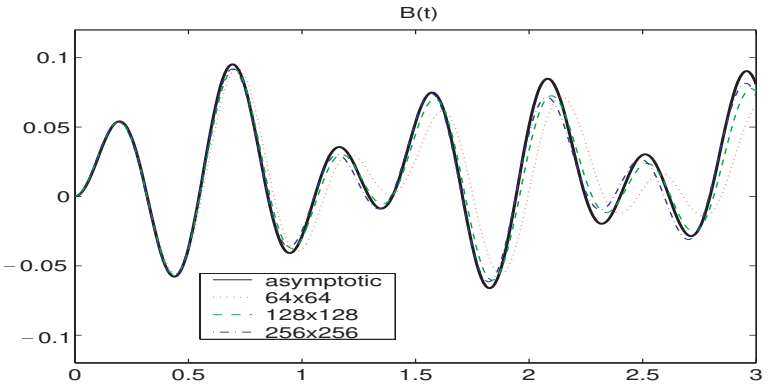
The asymptotic solution is based on the assumption of an infinite domain and is also only accurate up to $O(\varepsilon^3) \sim 10^{-3}$, so we do not expect the numerical solution to converge to the asymptotic solution as the grid is refined. Figure 8.1(a) shows the error (relative to (8.3)) in this norm as the grid is refined, which illustrates that there is good convergence on coarser grids but also that the asymptotic solution evidently has an error of approximately 5×10^{-4} relative to the true solution. Figure 8.1(b) shows the error in this same norm if the numerical solution on the finest (512×512) grid is used as a reference solution. Here we see nearly a slope of two in the log-log plot.

We also compare the amplitudes $A(t)$ and $B(t)$ from (8.4) and (8.5) of the dominant Fourier modes against the computed results, obtained by applying a discrete Fourier transform to the computed function $r(\theta, t)$ determined by the marker points. These comparisons are shown in Figure 8.2. We see that the computed Fourier modes have good agreement with both $A(t)$ and $B(t)$,— somewhat better than the results shown by Cortez and Varela [5]. They use an impulse method with a projected force $\mathbb{P}\boldsymbol{F}$ analogous to that used in the blob projection method. (Cortez [3] reports that better accuracy than shown in [5] has been obtained with better discrete delta functions.)

Example 2. The previous example was for an inviscid fluid with a simplified form of the interface force. In this example, we consider a stretched pressurized membrane immersed in a viscous fluid, with force density given by (2.9). We assume that the



(a)



(b)

FIG. 8.2. *Comparison of the Fourier modes* (8.4) *and* (8.5) *from the asymptotic solution with those computed using the IIM.*

resting unstretched membrane is a circular elastic curve with radius $r_0 = 0.5$. The initial shape of the boundary is an ellipse with major and minor axes $a = 0.75$, $b = 0.5$, respectively. Due to the restoring force of the elastic boundary and the incompressible fluid inside the membrane, the ellipse should converge to an equilibrium circular steady state of the form discussed in section 6, with radius $r_e = \sqrt{ab} \approx 0.61237$. When the Reynolds number is small, $Re = 1$ for instance, the membrane relaxes gradually to the equilibrium state as in Stokes flow, where similar examples have been used as computational tests in [15], [27]. When the Reynolds number is large, $Re \geq 100$, the membrane will oscillate as it converges to the equilibrium state. Figure 8.3 shows three different states of the interface for $Re = 100$, the case considered in our computations. We solve on the domain $[-1, 1] \times [-1, 1]$ with $u = 0$ on all four boundaries, a closed box. Please see http://www.amath.washington.edu/~rjl/iimins.html for animations of these results.

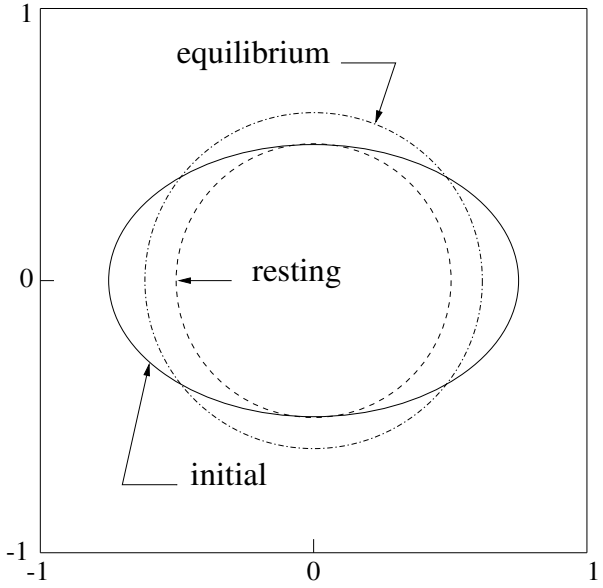


FIG. 8.3. *The interface at different states.*

Figure 8.4 shows nearly one cycle of the oscillation as computed on a 64×64 grid with $N_b = 64$. The solid line is computed by our IIM and the dotted one is computed using our own version of the original IB method. They are almost indistinguishable in a short time period, even though the grid is coarse. For the IB method, we again use the projection algorithm described above, but with $\boldsymbol{F}_\tau$ replaced by the full force in (3.8) and with no correction terms in the projection step, i.e., $B^1_{i-1/2,j} = B^2_{i,j-1/2} = C_{ij} = 0$. We have not implemented the modified projection method introduced by Peskin and Printz [24], which improves the volume conservation of the original IB method (see below) but is more complicated.

To compare results over longer times, let r_x be the radius of the interface measured along the x-axis and r_y the radius of the interface measured along the y-axis. These should both oscillate and converge towards the common value $r_e = \sqrt{ab} \approx 0.61237$ as the oscillation is damped by viscosity and the membrane becomes more circular. We now use a 256×256 grid with $N_b = 256$. Figure 8.5(a) shows the evolution of r_x and r_y computed by both IIM and IB method. The solid line is from the IIM. It clearly shows that both r_x and r_y converge to r_e even when we blow up the figure near the true equilibrium position as shown in Figure 8.5(b). By contrast, the dotted line computed from the IB method indicates that the circle shrinks linearly. The leaking of fluid is wellknown for the original IB method [15], [24].

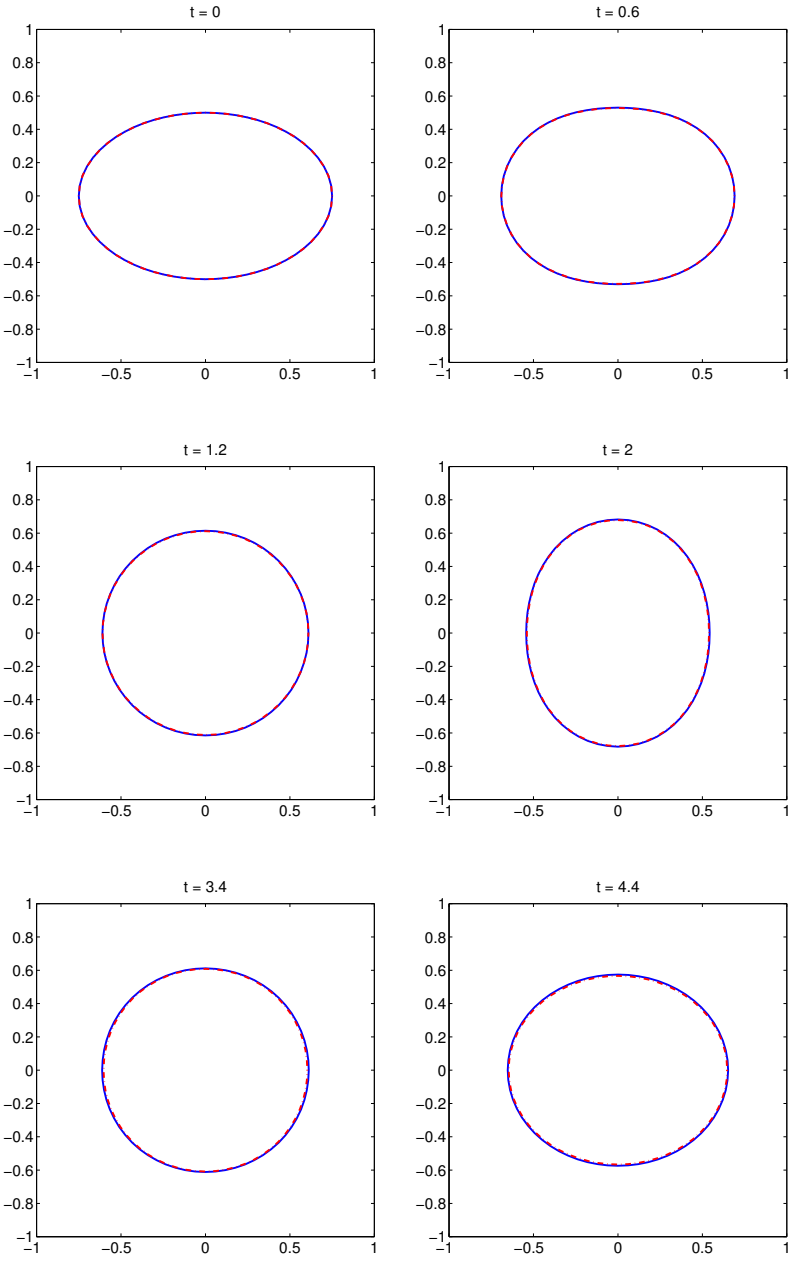
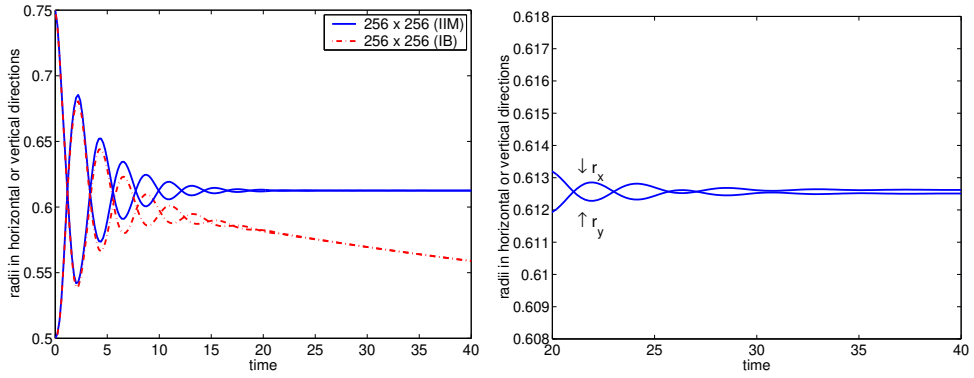


FIG. 8.4. *Nearly one cycle of the oscillation of the pressurized membrane in a box. For each frame, the solid line is the result from the IIM and the dotted line is the one from the IB method. A* 64×64 *mesh with* $N_b = 64$ *is used.*



(a) (b)

FIG. 8.5. (a) *The evolution of r_x and r_y. Solid line is from the IIM. Dotted line is from the IB method.* (b) *Blow-up of IIM results around the equilibrium. It shows that the interface converges to a circle with $r_e \approx 0.61237$.*

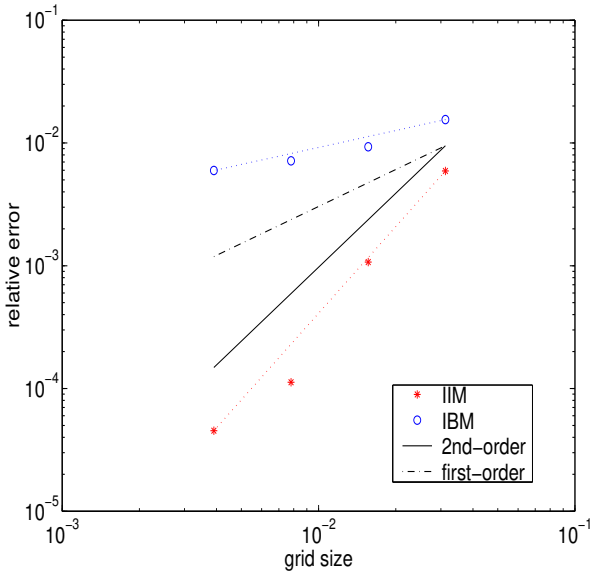


FIG. 8.6. *Grid refinement for the relative error of volume conservation at $t = 1$.*

Since the area enclosed by the membrane should be exactly conserved, we can use the area based on the computed membrane position as one measure of the accuracy of the method. Figure 8.6 shows that the area is conserved with second-order accuracy using our method, and only to first order with our version of the IB method. Similar results were seen for Stokes flow in [15].

Figures 8.7 and 8.8 show slices of the pressure computed at two different times with the IIM and IB method, respectively. These show that the two methods produce consistent results but that the pressure jump is more sharply captured with the IIM.

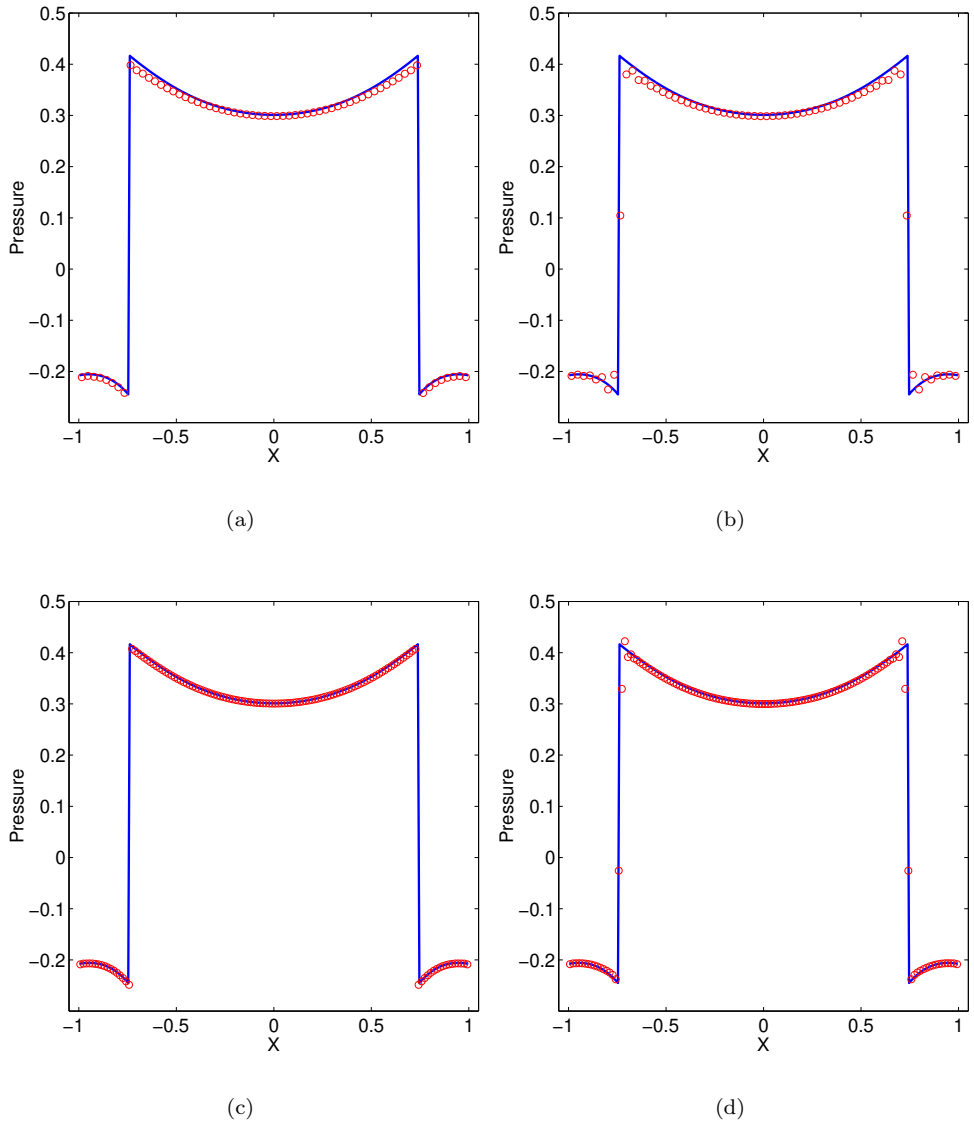


FIG. 8.7. *The pressure profile at* $t = 0.2$. *The slices of data are taken near* $y = 0$. *For all plots, the solid line is the result with IIM using a* 256×256 *grid. In* (a) *and* (b), *the circles are results from the IIM and the IB method using a* 64×64 *grid, respectively.* (c) *and* (d) *are results using a* 128×128 *grid. We can see the smearing near the interface for results using the IB method, i.e., plots* (b) *and* (d).

Figure 8.9 shows slices of the computed velocity with each method, with a grid refinement study. We see that results with the IIM are more accurate and smoother, particularly on the coarser grids.

Example 3. As a final example, we put a pressurized circular membrane in the center of a lid-driven cavity, with zero initial fluid velocity. When the "lid" starts moving, the viscosity will drive the fluid motion. The membrane will also be

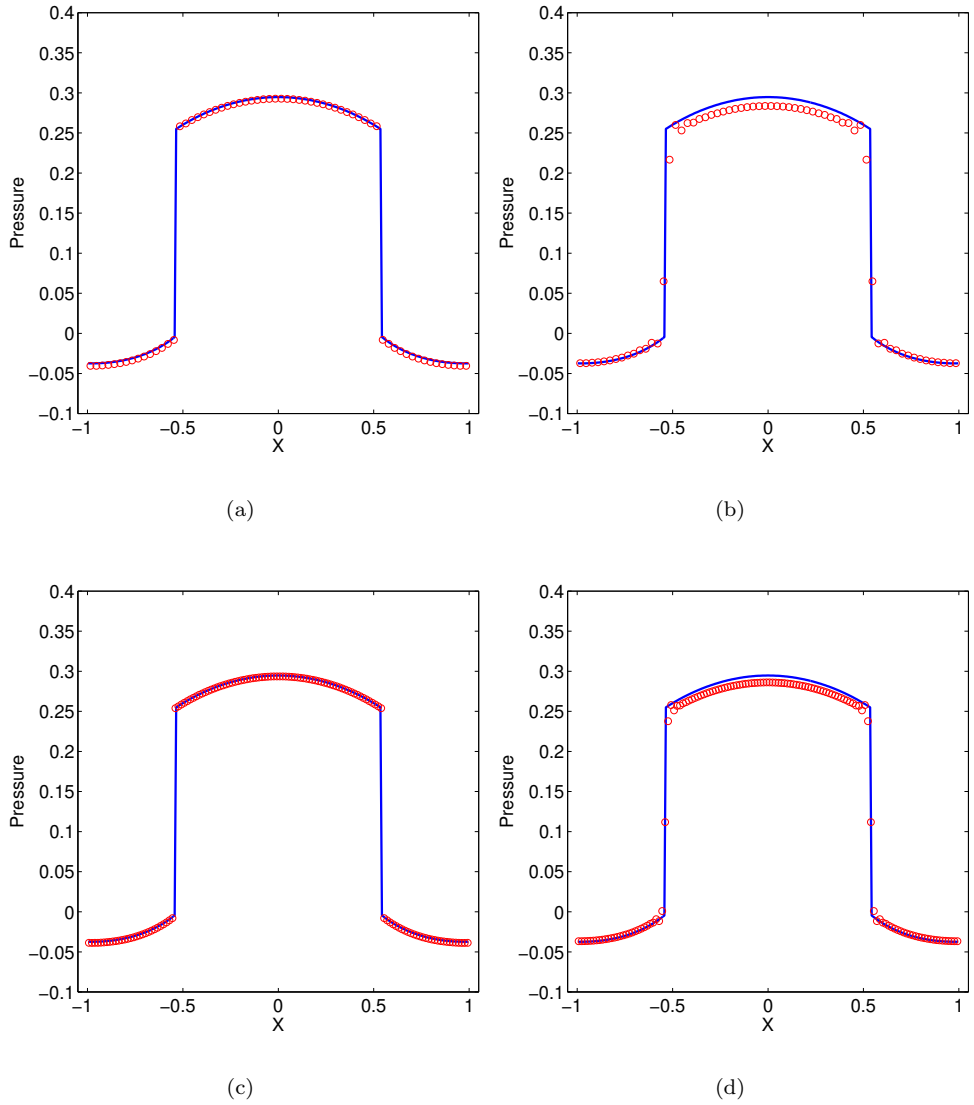
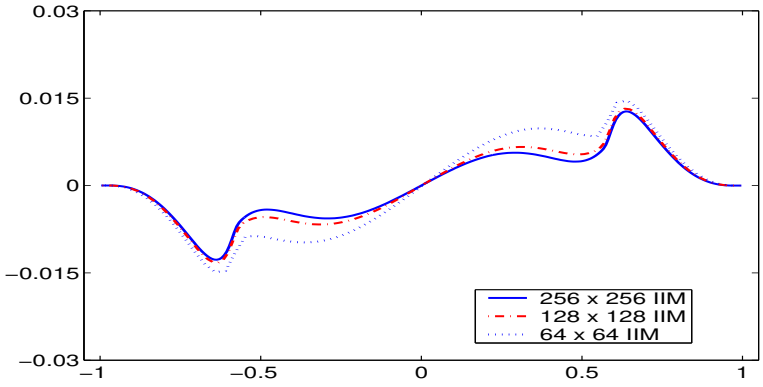


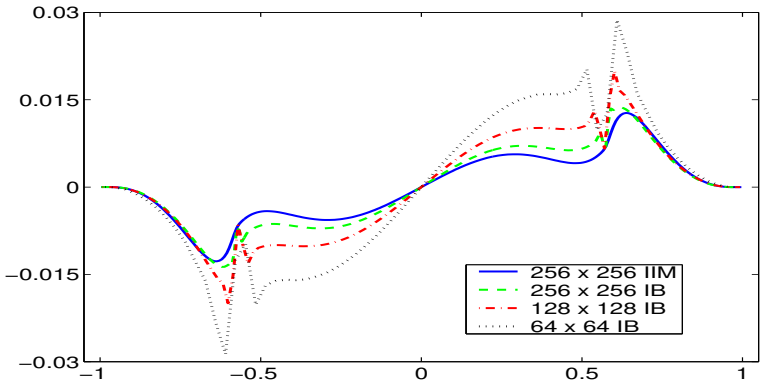
FIG. 8.8. *The same calculation as Figure* 8.7 *at later time* $t = 2.2$. *The IB method delivers weaker pressure inside the membrane. The IIM gives sharp resolution for coarse grids.*

driven and distorted by the flow, although the membrane tension tends to keep the membrane circular. This problem is computed with Reynolds number $Re = 5$ and stiffness constant $T_0 = 1$. The lid is moving from left to right with speed 1 and the other three walls are fixed with no-slip boundary conditions.

Figure 8.10 shows the solution at several times, as computed with the IIM on a fine grid (256×256). In order to show the fluid motion inside and outside the balloon, we also solve a scalar advection equation for some tracer that doesn't affect the flow but helps to visualize it. Three different values are used in the initial data to give the three colors. These values, representing three passive tracers, get smeared out a little bit near the sharp discontinuities by diffusion. The four small circles represent four



(a)



(b)

FIG. 8.9. (a) *The u component velocity profile for the IIM at* $y = -0.2656$ *and* $t = 4.2$*, computed at three grid resolutions.* (b) *The same computation as in* (a) *using the IB method, along with the finest grid IIM result. The number of markers* N_b *used for each* $N \times N$ *mesh grid is* $N_b = N$.

markers on the membrane to indicate the rotation of the membrane.

Figure 8.11 shows the interface location at several times, as computed with each method on two different grids, 128×128 and 256×256. The IIM result on the finer grid is also displayed with the IB method results to make comparison easier. They compare well at early times, but at later times there is more discrepancy. The area enclosed by the membrane is conserved with the IIM, while some shrinkage is apparent with the IB method.

We have shown a case at low Reynolds number, but our method also works well at higher Reynolds numbers. We have also experimented with different membrane stiffness factors and radii. Varying these parameters influences the dynamics and particularly the extent to which the membrane is deformed as it moves with the fluid. Animations of the results shown here, and a few other cases, may be viewed at http://www.amath.washington.edu/~rjl/iimins.html.

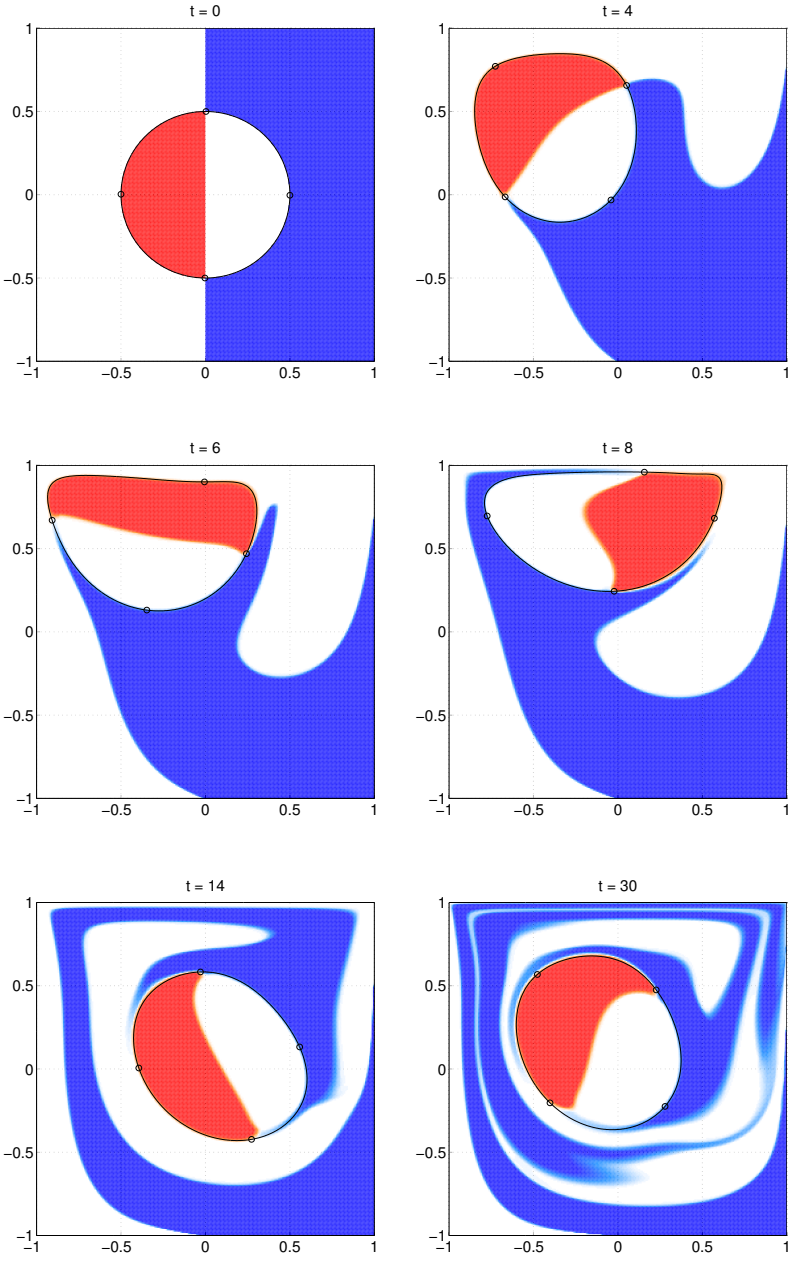


FIG. 8.10. *Stretched membrane in a lid-driven cavity.*

9. Conclusions. We have proposed a relatively simple modification to the classic immersed boundary method and have shown that it gives better resolution and second-order accuracy for some test problems. The proper jump in pressure is easily built into the elliptic equation that must be solved in the projection step. This avoids the need to introduce nonphysical discrete dipoles in intermediate steps. We have

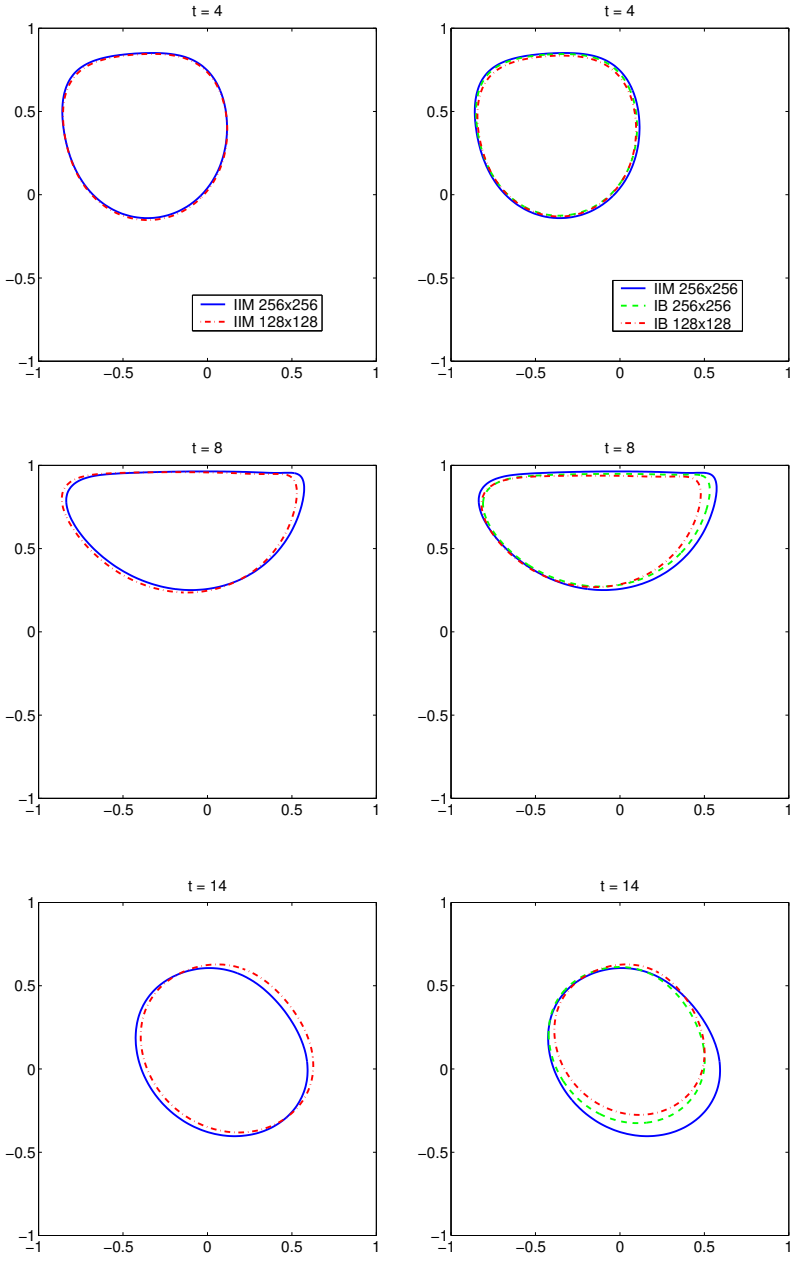


FIG. 8.11. *Membrane location at different times.*

implemented this along with our version of the projection method, which is based on the high-resolution finite-volume methods of CLAWPACK, but the ideas could also be applied to other projection methods as well. Our comparisons with the IB method are based on using our own code also for that method so that the only difference in the two algorithms is the manner in which the normal component of force and jumps in

pressure are handled. We believe that this illustrates the advantages of our suggested approach.

We should note that our approach is applicable only to problems where there is a sharp interface across which a jump in pressure is expected; i.e., our membrane is assumed to be infinitely thin. This assumption is not valid in all applications where the IB method is used. For the three-dimensional heart model currently being studied by Peskin and McQueen [22], for example, the heart wall is not a sharp interface but rather has finite width and is built up from layers of fibers wrapping around. For an application of this sort the original IB method may be best, since the force really should be spread over a finite region. However, for many applications where the IB method is used, a sharp interface is assumed and we expect that the IIM approach would give better results.

In the future we hope to develop more general versions of this algorithm to handle cases where fluids with different densities and viscosities are on each side of the immersed boundary. Incorporating the pressure jump into the elliptic solve may be even more crucial in these cases.

In this paper we have assumed the boundary is massless, a common assumption in problems where these methods are used. When the boundary has mass an additional inertial term is present in the force balance. An extension of the IIM algorithm to this case and some preliminary results are presented in [10] and in a forthcoming paper.

Acknowledgment. We are grateful to Jamal Mohd-Yusof for pointing out an inconsistency in an earlier version of our algorithm.

REFERENCES

[1] R. P. Beyer, *A computational model of the cochlea using the immersed boundary method*, J. Comput. Phys., 98 (1992), pp. 145–162.
[2] D. L. Brown, R. Cortez, and M. L. Minion, *Accurate projection methods for the incompressible Navier–Stokes equations*, J. Comput. Phys., 168 (2001), pp. 464–499.
[3] R. Cortez, *personal communication*, 2001.
[4] R. Cortez and M. Minion, *The blob projection method for immersed boundary problems*, J. Comput. Phys., 161 (2000), pp. 428–453.
[5] R. Cortez and D. A. Varela, *The dynamics of an elastic membrane using the impulse method*, J. Comput. Phys., 138 (1997), pp. 224–247.
[6] R. Dillon, L. Fauci, A. Fogelson, and D. Gaver, *Modeling biofilm processes using the immersed boundary method*, J. Comput. Phys., 129 (1996), pp. 57–73.
[7] L. J. Fauci and C. S. Peskin, *A computational model of aquatic animal locomotion*, J. Comput. Phys., (1988), pp. 85–108.
[8] A. L. Fogelson, *A mathematical model and numerical method for studying platelet adhesion and aggregation during blood clotting*, J. Comput. Phys., 1 (1984), pp. 111–134.
[9] J. Kim and P. Moin, *Application of a fractional-step method to incompressible Navier–Stokes equations*, J. Comput. Phys., 59 (1985), pp. 308–323.
[10] L. Lee, *Immersed Interface Methods for Incompressible Flow with Moving Interfaces*, Ph.D. thesis, University of Washington, Seattle, WA, 2002.
[11] R. J. LeVeque, clawpack *software*, http://www.amath.washington.edu/ claw+.
[12] R. J. LeVeque, *High-resolution conservative algorithms for advection in incompressible flow*, SIAM J. Numer. Anal., 33 (1996), pp. 627–665.
[13] R. J. LeVeque, *Wave propagation algorithms for multi-dimensional hyperbolic systems*, J. Comput. Phys., 131 (1997), pp. 327–353.
[14] R. J. LeVeque and Z. Li, *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources*, SIAM J. Numer. Anal., 31 (1994), pp. 1019–1044.
[15] R. J. LeVeque and Z. Li, *Immersed interface methods for Stokes flow with elastic boundaries or surface tension*, SIAM J. Sci. Comput., 18 (1997), pp. 709–735.
[16] Z. Li and M. C. Lai, *The immersed interface method for the Navier-Stokes equation with singular forces*, J. Comput. Phys., 171 (2001), pp. 822–842.

[17] A. Mayo, *The fast solution of Poisson's and the biharmonic equations on irregular regions*, SIAM J. Numer. Anal., 21 (1984), pp. 285–299.
[18] A. A. Mayo and C. S. Peskin, *An implicit numerical method for fluid dynamics problems with immersed elastic boundaries*, Contemp. Math., 141 (1993), pp. 261–277.
[19] C. S. Peskin, *Numerical analysis of blood flow in the heart*, J. Comput. Phys., 25 (1977), pp. 220–252.
[20] C. S. Peskin, *Lectures on Mathematical Aspects of Physiology*, Lectures in Appl. Math., 19, AMS, Providence, RI, 1981.
[21] C. S. Peskin and D. M. McQueen, *Modeling prosthetic heart valves for numerical analysis of blood flow in the heart*, J. Comput. Phys., 105 (1980), pp. 113–132.
[22] C. S. Peskin and D. M. McQueen, *A three-dimensional computational method for blood in the heart:* I. *Immersed elastic fibers in a viscous incompressible fluid*, J. Comput. Phys., 81 (1989), pp. 372–405.
[23] C. S. Peskin and D. M. McQueen, *A general method for the computer simulation of biological systems interacting with fluids*, in Symposia of the Society for Experimental Biology, Vol. 49, C. P. Ellington and T. J. Pedley, eds., The Company of Biologists Ltd., Cambridge, UK, 1995, pp. 265–276.
[24] C. S. Peskin and B. F. Printz, *Improved volume conservation in the computation of flows with immersed elastic boundaries*, J. Comput. Phys., 105 (1993), pp. 33–46.
[25] J. M. Stockie and S. I. Green, *Simulating the motion of flexible pulp fibres using the immersed boundary method*, J. Comput. Phys., 147 (1998), pp. 147–165.
[26] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, 2nd ed., Springer-Verlag, New York, 1993.
[27] 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. Statist. Comput., 13 (1992), pp. 1361–1376.