SIAM J. SCI. COMPUT.
Vol. 18, No. 3, pp. 709–735, May 1997

004

IMMERSED INTERFACE METHODS FOR STOKES FLOW WITH ELASTIC BOUNDARIES OR SURFACE TENSION*

RANDALL J. LEVEQUE† AND ZHILIN LI‡

Abstract. A second-order accurate interface tracking method for the solution of incompressible Stokes flow problems with moving interfaces on a uniform Cartesian grid is presented. The interface may consist of an elastic boundary immersed in the fluid or an interface between two different fluids. The interface is represented by a cubic spline along which the singularly supported elastic or surface tension force can be computed. The Stokes equations are then discretized using the second-order accurate finite difference methods for elliptic equations with singular sources developed in our previous paper [*SIAM J. Numer. Anal.*, 31(1994), pp. 1019–1044]. The resulting velocities are interpolated to the interface to determine the motion of the interface. An implicit quasi-Newton method is developed that allows reasonable time steps to be used.

Key words. Stokes flow, creeping flow, interface tracking, discontinuous coefficients, immersed interface methods, Cartesian grids, bubbles

AMS subject classifications. 76M20, 65M06, 76D07

PII. S1064827595282532

1. Introduction.

In this paper we develop an interface tracking method for the solution of incompressible Stokes flow problems with moving interfaces on a uniform Cartesian grid. The interface may consist of an elastic boundary immersed in the fluid, as in the model problem of Tu and Peskin [49], or an interface between two different fluids, as in the study of bubbles or free surfaces.

The method we use is based on the *immersed interface method* (IIM) for elliptic problems developed in our previous paper [22] and the second author's thesis [25]. This is a second-order accurate Cartesian grid method for solving elliptic equations whose solutions are not smooth across some interface, due to discontinuous coefficients or singular source terms in the equation. The main idea is to incorporate the known jumps in the solution or its derivatives into the finite difference scheme, obtaining a modified scheme whose solution is second-order accurate at all points on the uniform grid, even for quite arbitrary interfaces. This approach has also been applied to parabolic equations [28], [30] and hyperbolic wave equations with discontinuous coefficients [23], [24].

Similar ideas have been used in the context of domain embedding, where an irregular region is embedded into a larger rectangular domain on which a Cartesian grid is used. Methods of this type include the method of fictitious domains [2], [8], [16], [29], capacitance matrix methods [9], [43], and Mayo's method [32], [33].

A variety of Cartesian grid methods have been proposed for fluid dynamics problems with arbitrary boundaries and/or moving interfaces, e.g., [1], [4], [11], [12], [17], [21], [31], [37], [44], [45], [51], [50], [52], and other references below. Such methods are

*Received by the editors March 6, 1995; accepted for publication (in revised form) August 30, 1995. A portion of this paper was adapted from "Simulation of bubbles in creeping flow using the immersed interface method" in *Sixth International Symposium of CFD, Lake Tahoe*, John Wiley, New York, 1995. This work was supported in part by NSF grants DMS-8657319, DMS-9204329, and DMS-9303404, DOE grant DE-FG06-93ER25181, and URI grant N00014092-J-1890.

http://www.siam.org/journals/sisc/18-3/28253.html

†Departments of Mathematics and Applied Mathematics, University of Washington, Seattle, WA 98195 (rjl@amath.washington.edu).

‡Department of Mathematics and Statistics, Mississippi State University, Mississippi State, MS 39762 (zli@math.msstate.edu).

becoming increasingly popular for problems with very complex geometries or moving interfaces where more standard "body-fitted" or unstructured grid approaches can run into difficulties.

Our motivation for the present work was the goal of ultimately developing a second-order accurate version of Peskin's immersed boundary method (IBM), a very robust algorithm for solving the full incompressible Navier–Stokes equations with moving boundaries. This method was originally developed for studying blood flow in a beating heart [38], [39], [40], but has also been used in a wide variety of other problems (e.g., [6], [13], [14], [51]), particularly in biomechanics since these problems often involve complex geometries and are difficult to model with more standard approaches. The physical domain is embedded in a rectangular region and the Navier–Stokes equations are solved on a uniform Cartesian grid. The boundary is modeled by a singular forcing term which, computationally, is then approximated by a set of discrete delta functions which spread the force from the boundary to the nearby Cartesian grid points. This is described further in the context of Stokes flow in [49]. See [39] for complete details. The spreading of forces via discrete delta functions appears to limit Peskin's method to first-order accuracy. Our approach will, we hope, lead to second-order accurate methods for the Navier–Stokes equations.

In this paper we consider the simpler Stokes equations which are still of physical interest in many applications. The Stokes equations model the creeping flow of a highly viscous fluid, in the limit where the Reynolds number goes to zero and both the inertial acceleration and convection terms are dropped from the Navier–Stokes equations. We will concentrate on describing the algorithms in two space dimensions. In section 11 we briefly discuss what is involved in extending them to three space dimensions.

In two dimensions the Stokes equations take the form

$$p_x(x,y,t) = \mu(u_{xx}(x,y,t) + u_{yy}(x,y,t)) + F_1(x,y,t), \tag{1.1a}$$

$$p_y(x,y,t) = \mu(v_{xx}(x,y,t) + v_{yy}(x,y,t)) + F_2(x,y,t), \tag{1.1b}$$

$$u_x(x,y,t) + v_y(x,y,t) = 0. \tag{1.1c}$$

Here $\vec{u} \equiv (u,v)$ is the velocity vector, p is the pressure, μ is the viscosity, and $\vec{F} \equiv (F_1, F_2)$ is the external force. For the time being we assume μ is constant. Some results with discontinuous μ are presented in section 10. We also use the notation $\vec{x} = (x,y)$ below. See, for example, [3], [10], [19], [20], [42] for general discussions of Stokes flow.

The equations (1.1) can be solved as a coupled system (as is done in [49]) or alternatively reduced to a sequence of three Poisson problems, one for each variable. Differentiating (1.1a) with respect to x, (1.1b) with respect to y, and adding the equations together gives

$$\nabla^2 p = \nabla \cdot \vec{F}, \tag{1.2}$$

where ∇^2 is the Laplacian. Since the right-hand side is known, this is a Poisson problem for the pressure. Once p is known, the equations (1.1a) and (1.1b) are independent Poisson problems for u and v, respectively.

Note that the time evolution of the flow is governed entirely by the time dependence of the forces $\vec{F}$. If $\vec{F}$ is known at a given instant in time, then the system (1.1) is elliptic and the solution (u,v,p) is determined independently of the past history of the flow. This is a reflection of the fact that there is no inertia in the system;

i.e., the convective and time-derivative acceleration terms have been dropped from the momentum equations. This allows us to use the techniques developed in [22] directly, once we have determined the appropriate jump conditions on the solutions. We assume that the reader is familiar with that paper.

The jumps in the solution result from the fact that the force $\vec{F}$ is singular and is supported only along the interface (resulting from the elastic force or surface tension). This singular force in (1.1a) and (1.1b) leads to jumps in the first derivatives of u and v across the interface. Since the Poisson problem (1.2) for p involves derivatives of $\vec{F}$, and hence a dipole source, the pressure will be discontinuous, along with its derivatives.

The singular force can be written as

$$\vec{F}(\vec{x},t) = \int_0^{L_0} \vec{f}(s,t)\delta(\vec{x} - \vec{X}(s,t))\, ds, \tag{1.3}$$

where $\vec{X}(s,t)$ gives the location of the interface at time t, parameterized by s for $0 \le s \le L_0$, $\vec{f}(s,t)$ is the force strength at this point, and δ is the two-dimensional delta function. This singular force is best viewed as a distribution whose action on any smooth test function $\phi(\vec{x})$ is

$$\langle F, \phi \rangle = \int_0^{L_0} \vec{f}(s,t)\, \phi(\vec{X}(s,t))\, ds. \tag{1.4}$$

In practice we eliminate the singular source term from the right-hand side of (1.2) and instead solve

$$\nabla^2 p = 0 \tag{1.5}$$

together with specified jump conditions across the interface, using the techniques of [22]. This is described in more detail in section 4.

We first consider the model problem used by Tu and Peskin [49], an immersed elastic band in a fluid with the same fluid properties on each side. In section 10 we extend our approach to study an interface between two different fluids, with surface tension providing the force at the interface rather than an elastic band.

The elastic band problem is the two-dimensional analogue of an elastic balloon in a highly viscous fluid (with the same fluid inside and out). In equilibrium, an ideal balloon would have a spherical shape, with zero velocity everywhere and uniform pressure both inside and out but with a jump in pressure across the elastic membrane. The magnitude of this jump depends on how far the membrane is stretched from its resting configuration. In two space dimensions the analogue is an elastic band, or more physically the cross section of an impermeable cylindrical elastic tube, which contains an incompressible fluid and is stretched to a diameter greater than its resting diameter. The equilibrium configuration is a circle with a jump in pressure that balances the elastic force exerted by the stretched membrane in a manner made clear in section 3.

Note that we make an important distinction between the equilibrium configuration and the resting configuration. By the *equilibrium configuration* we always mean the steady state solution with a given quantity of fluid inside the elastic band, which will typically be stretched like an inflated balloon since the fluid cannot escape. By the *resting configuration* we mean the shape that would ultimately be obtained if a small leak were introduced, allowing the band to deflate until there is no force exerted by the elastic membrane and no pressure difference across it. The length of the elastic band in this unstretched state will be called the *resting length*, L_0, and the

radius of the corresponding circle will be denoted by $r_0 = L_0/2\pi$. An inflated band, at equilibrium, will have some equilibrium length L_e and radius $r_e = L_e/2\pi$ which depend solely on how much fluid is trapped inside.

If we perturb an inflated band from its equilibrium position, analogous to squeezing an inflated balloon, it will return to an equilibrium circular shape. Our goal is to simulate this motion. We specify an initial position of the boundary (some simple closed curve) and simulate its motion to a circle. The area A enclosed by the band must be invariant with time and the equilibrium radius must be $r_e = \sqrt{A/\pi}$.

We will parameterize the location of the band at any time t by $\vec{X}(s,t) = (X(s,t), Y(s,t))$, where s is arclength along the unstretched band, $0 \le s \le L_0$, measured from some arbitrary but fixed origin. The material particle at $(X(s,t),\ Y(s,t))$ is the same particle that would lie a distance s from the origin if the band were cut at the origin and allowed to relax. The force exerted by the band at $\vec{X}(s,t)$ is given by (1.3) with strength

$$\vec{f}(s,t) = \frac{\partial}{\partial s}(T(s,t)\vec{\tau}(s,t)), \tag{1.6}$$

where $T(s,t)$ is the tension at this point, given by

$$T(s,t) = T_0 \left(\left| \frac{\partial \vec{X}}{\partial s} \right| - 1 \right) \tag{1.7}$$

and $\vec{\tau}(s,t)$ is the tangent vector to the band at this point,

$$\vec{\tau}(s,t) = \frac{\partial \vec{X}}{\partial s} \Bigg/ \left| \frac{\partial \vec{X}}{\partial s} \right|. \tag{1.8}$$

Note that in the relaxed state $|\partial \vec{X}/\partial s| \equiv 1$ since s is the arclength. The single scalar T_0 describes the elastic properties of the band, which are assumed to be uniform along its length, though this is not necessary. The larger T_0, the stiffer the elastic band and the greater the force induced by a stretching of the band. The tension given by (1.7) is a linear Hooke's law relation, which could easily be replaced by a more general nonlinear relation in our algorithm.

The force density can be computed directly from the location of the interface $\vec{X}$ at time t. We will see in section 3 that it is possible to determine the jump conditions for $u,\ v$, and p directly from $\vec{f}(s,t)$. This in turn allows us to apply the immersed interface method to solve for $u,\ v$, and p at all points on a uniform Cartesian grid at time t.

The interface at each time is represented by a cubic spline passing through a given set of *control points* along the interface, as described in the next section, and hence the location of the interface is completely determined by the location of the control points. Taking a time step requires also moving the interface, which is accomplished by moving the control points using the additional constraint that the interface must move with the fluid. The velocity $\vec{u}$ will be continuous across the interface and we have the differential equation

$$\frac{\partial}{\partial t}\vec{X}(s,t) = \vec{u}(\vec{X}(s,t),\ t). \tag{1.9}$$

The velocities computed on the Cartesian grid are interpolated to the control points, which are then moved with this velocity over a time step of length Δt. In practice an implicit method must be used in order to take reasonable time steps, and a quasi-Newton method has been developed to accomplish this in an efficient manner, as described in section 7.

This completes an overview of our method. Each of these steps will be described in greater detail in the remainder of the paper.

Our approach differs from that of Tu and Peskin in several ways. The fundamental difference is our use of jump conditions to derive second-order accurate finite difference schemes rather than using discrete delta functions (which appear to limit the accuracy to first order). But there are other differences as well.

Using a relatively small number of control points to specify the interface rather than the denser set typically used with the IBM has great advantages in solving the implicit system of equations. This approach could be carried over to the IBM too, as described in section 8.

We also use a sequence of three Poisson problems as described above, whereas Tu and Peskin discretize the Stokes equations (1.1) directly as a coupled system, which is necessary in using the IBM since the right-hand side of (1.1) involves delta function forces, whereas the right-hand side of (1.2) has dipoles, which are even more difficult to discretize accurately. Our approach, on the other hand, could also be applied to the coupled system directly, incorporating the jump conditions for u, v, and p simultaneously. This has also been implemented and details are presented in [25]. We have found, however, that the approach presented here gives slightly better results. It also yields the pressure p at all grid points, which may be of interest, whereas discretizing the coupled system and solving by fast Fourier transforms, as done in [49], does not yield p due to zero divisors in the transformed equations. Finally, the approach based on three Poisson problems allows the direct use of software already developed for the general Poisson problem with jumps across an interface, a problem that also arises in many other contexts. However, decoupling the problem into three Poisson problems depends on the fact that periodic boundary conditions are used. Some comments are made in section 11 on handling other boundary conditions.

Another approach to solving Stokes flow problems is to solve a biharmonic equation for the stream function instead of the three Poisson problems used here. Mayo [31] has developed a method based on this approach and techniques from [32] that is similar in spirit to our method.

Boundary integral methods are also very popular for Stokes flow, since this linear problem can be reduced to an integral equation along the interface, reducing the dimensionality of the problem. For a description of this approach and some references, see, e.g., [42]. This approach does not appear to extend to nonlinear problems such as the full Navier–Stokes equations, however, whereas the IBM does. We expect that our method can also be extended, and work is now underway to do so.

2. Representation of the interface and forces. The location of the interface at time t_n is represented by a finite set of control points (X_k^n, Y_k^n) for $k = 0,\ 1, \ldots, N_b$. Since the boundary is always a simple closed curve in the model problem, we assume $(X_0^n, Y_0^n) = (X_{N_b}^n, Y_{N_b}^n)$. The kth control point gives an approximation to $(X(s_k, t_n), Y(s_k, t_n))$, where $s_k = kL_0/N_b$. Based on these control points, we determine a continuous curve $(X^n(s),\ Y^n(s))$ by computing some interpolants $X^n(s)$ through the points X_k^n and $Y^n(s)$ through the points Y_k^n.

The computations presented here were all computed using periodic cubic splines for $X^n(s)$ and $Y^n(s)$. The force exerted by the elastic interface, given by (1.6), can also be represented by cubic splines. From the splines $(X^n(s), Y^n(s))$ we can compute $\partial\vec{X}/\partial s$ and hence the tension $T(s, t_n)$. Multiplying by the tangent vector and differentiating again gives an approximation to (1.6). We now evaluate this function at each of the control points to obtain values $\vec{f}_k^n$ and then interpolate these values by a new cubic spline $\vec{f}^n(s)$ to obtain the representation of the force all along

the interface. The reason for introducing a new cubic spline at this point is that the jump conditions discussed in the next section depend on further derivatives of the force along the interface.

Other representations of the interface are possible. We have also experimented with using a Fourier series representation, which is quite convenient for the smooth closed curves considered here since X and Y are both periodic in s. With this approach it is easy to compute the necessary derivatives and also apply filtering to remove high frequency oscillations of the interface. In some problems this has been found to improve stability properties.

A level set representation (e.g., [36], [47]) could also be considered. In this approach the interface is represented by the $\phi(x, y, t) = 0$ contour of some smooth function ϕ that advects with the fluid velocity. For the surface tension problems discussed in section 10, this could be used since the surface tension force depends only on the curvature of the interface, which can be determined directly from ϕ. However, for the elastic band problem the force depends on the manner in which the band is stretched and not just its location, and hence it seems necessary to explicitly track control points on the interface.

3. Jump conditions across the interface. In order to use the IIM, we need to know the jump conditions for each of the three Poisson problems (for p, u, and v). We need to know both the jump in the function and the jump in its normal derivative at each point along the interface. (See [22] for details on how the jump conditions are used to derive second-order accurate difference formulas.)

The velocity components u and v are both continuous across the interface. Certainly the normal velocity must be continuous and a discontinuity in the tangential velocity is also ruled out by the presence of viscosity and a no-slip boundary condition between the elastic band and the fluid on each side. The normal derivative of all three variables will, however, be discontinuous in general, as will the pressure itself.

The jump conditions are easiest to write in terms of normal and tangential components of the force $\vec{f}(s,t)$. We decompose the force as

$$\vec{f}(s,t) = \begin{bmatrix} f_1(s,t) \\ f_2(s,t) \end{bmatrix} = \begin{bmatrix} \hat{f}_1(s,t)\cos\theta - \hat{f}_2(s,t)\sin\theta \\ \hat{f}_1(s,t)\sin\theta + \hat{f}_2(s,t)\cos\theta \end{bmatrix}, \tag{3.10}$$

where θ is the angle between the x-axis and the normal direction pointing outward from the interface at $\vec{X}(s,t)$, and $\hat{f}_1$, $\hat{f}_2$ are the normal and tangential force strengths, respectively, defined as

$$\begin{aligned} \hat{f}_1(s,t) &= f_1(s,t)\cos\theta + f_2(s,t)\sin\theta, \\ \hat{f}_2(s,t) &= -f_1(s,t)\sin\theta + f_2(s,t)\cos\theta. \end{aligned} \tag{3.11}$$

The jump conditions are then given by

$$[p](s) = \hat{f}_1(s,t), \tag{3.12a}$$

$$[p_n](s) = \frac{\partial}{\partial s}\hat{f}_2(s,t), \tag{3.12b}$$

$$[\mu u_n](s) = \hat{f}_2(s,t)\sin\theta, \tag{3.12c}$$

$$[\mu v_n](s) = -\hat{f}_2(s,t)\cos\theta. \tag{3.12d}$$

These are derived in the Appendix.

Note that in the equilibrium case we expect there to be no tangential component to the force, i.e., $\hat{f}_2 = 0$, since in equilibrium the tension $T(s,t)$ will be constant in s and so $\vec{f}(s,t)$ will be the derivative of a constant-length tangent vector, and hence will point in the normal direction. The only nonzero jump will then be that given by (3.12a), as we expect since $u = v = 0$ and p is piecewise constant in this case.

The jump conditions can be determined at any point on the interface by applying (3.12) to the cubic spline function $\vec{f}^n(s)$ representing the force. Alternatively, we could calculate the jump conditions at each control point and then interpolate these by cubic splines.

4. Solving the Poisson problem for *p*. Given the location of the interface at time t_n, and the jump conditions for p across it from (3.12a) and (3.12b), we wish to solve the Poisson problem (1.5) with these specified jump conditions. We solve a difference equation of the form

$$\frac{1}{h^2}(p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} - 4p_{ij}) = C_{ij} \tag{4.13}$$

on a uniform Cartesian grid, where the right-hand side C_{ij} is determined using the techniques of [22] based on the jump conditions. The value C_{ij} will be nonzero only at "irregular" grid points, those for which the five point stencil includes points from both sides of the interface. These values can be thought of as giving a spreading of the dipole source in (1.2) from the interface to the nearby grid points, in the spirit of Peskin's IBM. However, they are determined from the jump conditions rather than the dipole in such a way that the computed values p_{ij} are second-order accurate at all points (provided only that the interface is smooth).

For the model problem of Tu and Peskin, we have periodic boundary conditions, and so we know that a solution exists only if the C_{ij} satisfy

$$\sum_{i,j} C_{ij} = 0.$$

Unfortunately, since the method of [22] is based on requiring only that the local truncation error be $O(h)$ at irregular points, the values of C_{ij} will have $O(h)$ errors which, when summed over the $O(1/h)$ irregular points, will give an $O(1)$ result for $\sum C_{ij} \equiv \bar{C}$. However, if we perturb each C_{ij} to

$$\hat{C}_{ij} = C_{ij} - \frac{\bar{C}}{N^2},$$

we obtain a solvable perturbed system

$$\frac{1}{h^2}(p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} - 4p_{ij}) = \bar{C}_{ij} \tag{4.14}$$

whose solution is the least squares solution to the unperturbed equation (see [48]). Notice that $\bar{C}/N^2$ is of order h^2. This means the order of the local truncation errors, which are $O(h^2)$ away from the interface and $O(h)$ near the interface, have not been changed. We expect the solution to the perturbed equation (4.14) will approximate the true solution of the Poisson equation to second-order accuracy. We use the Fourier method described in [48] to solve the perturbed equation (4.14) and set $p_{00} = 0$ to get a particular solution.

5. Solving for u and v. Next we need to solve the Poisson problems (1.1a) and (1.1b) for u and v. These are essentially identical and we will explain the procedure for the solution of (1.1a) to obtain u. The forcing term F_1 is singular along the interface but is a delta function singularity rather than a dipole, leading to a jump in the normal derivative of u but not in u itself. Also note that, since p is discontinuous across the interface, p_x will contain a delta function singularity. The strength of the jump in the normal derivative u_n depends on both of these effects, and is given by (3.12c). As in our solution of the Poisson problem for p, we do not attempt to model these singular terms directly but instead drop them from the equation and solve the Poisson problem

$$\nabla^2 u = \frac{1}{\mu} p_x$$

away from the interface, where p_x is smooth, coupled with the jump conditions $[u] = 0$ and (3.12c) across the interface. Using the IIM, this gives the discrete system

$$\frac{1}{h^2}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{ij}) = \frac{1}{\mu}\pi_{ij} + D_{ij}, \tag{5.15}$$

where $\pi_{ij} \approx p_x(x_i, y_j)$ and the correction terms D_{ij}, resulting from the jump conditions, are nonzero only at the irregular grid points. These are again derived following [22]. Note that if μ is discontinuous across the interface, then the elliptic equations for u and v have discontinuous coefficients and the techniques of [22] can be used to derive a six-point stencil that yields second-order accuracy. This is used in section 10.

At regular grid points we can approximate p_x by the standard central difference,

$$\pi_{ij} = \frac{1}{2h}(p_{i+1,j} - p_{i-1,j}).$$

At irregular grid points we normally use one-sided differences to obtain first-order accurate approximations. Alternatively, one could use the known jump conditions for p and p_x to correct the centered formula, but this also gives only a first-order accurate formula unless higher-order jump conditions are derived and used. Fortunately, first-order local accuracy at these points is sufficient to maintain second-order accuracy globally [22].

There is one situation where a one-sided approximation cannot be used, if both the points $(i-1, j)$ and $(i+1, j)$ lie on the opposite side of the interface from (i, j). This could happen, for example, at the top or bottom of a circular interface. In this case we interpolate $p_{i-1,j}$, p_{ij} and $p_{i+1,j}$ to get $(p_x)_{ij}$ to first order by using the known jump conditions $[p]$, $[p_x]$, and $[p_y]$ from (3.12a) and the relations

$$[p_x] = [p_n]\cos\theta - [p_s]\sin\theta, \tag{5.16}$$

$$[p_y] = [p_n]\sin\theta + [p_s]\cos\theta. \tag{5.17}$$

We know the jump $[p_n]$ already from (3.12b), and

$$[p_s] = \frac{\partial p^+}{\partial s} - \frac{\partial p^-}{\partial s} = \frac{\partial}{\partial s}[p]; \tag{5.18}$$

i.e., $[p_s]$ is the derivative of jump of pressure p along the interface. So we find

$$[p_x] = \frac{\partial \hat{f}_2}{\partial s}\cos\theta - \frac{\partial \hat{f}_1}{\partial s}\sin\theta,$$

$$[p_y] = \frac{\partial \hat{f}_2}{\partial s}\sin\theta + \frac{\partial \hat{f}_1}{\partial s}\cos\theta.$$

Let (X_k, Y_k) be the control point closest to (x_i, y_j), and $x_l (l = i+1, \text{ or } i-1)$ be the one of x_i and x_{i+1} which is closer to X_k. Then we can use the following interpolation formula:

$$\pi^{\pm}{}_{ij} = \frac{p_{ij} - p_{lj} \mp [p] \mp [p_x]\,(x_l - X_k\,) \mp [p_y]\,(y_j - Y_k\,)}{(x_i - x_l)}, \tag{5.19}$$

where the sign in the expression depends on which side of the interface the point (i, j) is on, and $[p]$, $[p_x]$, and $[p_y]$ are calculated at (X_k, Y_k). In [25] it is shown that (5.19) gives a first-order accurate approximation to p_x.

Solving the system (5.15) gives velocities u_{ij} that are second-order accurate at all grid points. An analogous procedure on (1.1b) gives the y velocities v_{ij}.

6. Moving the interface—an explicit method. The interface should move at the local fluid velocity, which can be accomplished by moving each control point (X_k^n, Y_k^n) at velocity (U_k^n, V_k^n), determined by interpolating the velocity fields u_{ij} and v_{ij} to the kth control point. This interpolation is complicated by the fact that u and v are not smooth across the interface, and so the known jumps in their normal derivatives must be used in the interpolation formulas. There are various ways in which this can be done. Here we give one example based on using three nearby points to perform linear interpolation, modified to incorporate the jump conditions at the interface.

Dropping superscripts and subscripts for simplicity, let (X, Y) be an arbitrary point on the interface and consider the problem of interpolating from the u_{ij} to obtain the x component U of the velocity at (X, Y).

First we choose the first three grid points (x_{i1}, y_{j1}), (x_{i2}, y_{j2}), and (x_{i3}, y_{j3}) closest to (X, Y). Then we form a linear combination of the grid values at these points plus a correction term to approximate U:

$$U = \gamma_1\, u_{i1,j1} + \gamma_2\, u_{i2,j2} + \gamma_3\, u_{i3,j3} - C. \tag{6.20}$$

We use Taylor expansion about (X, Y) to get the equations for the coefficients γ's so that we have a second-order approximation:

$$\begin{aligned} U &= \gamma_1 u(x_{i1}, y_{j1}) + \gamma_2 u(x_{i2}, y_{j2}) + \gamma_3 u(x_{i3}, y_{j3}) - C \\ &= a_1 u^- + a_2 u^+ + a_3 u_x^- + a_4 u_x^+ + a_5 u_y^- + a_6 u_y^+ - C + O(h^2) \\ &= (a_1 + a_2) u^- + (a_3 + a_4) u_x^- + (a_5 + a_6) u_y^- + a_2 [u] \\ &\quad + \ a_4 [u_x] + a_6 [u_y] - C + O(h^2). \end{aligned}$$

Here the a_i are linear combinations of the γ's as in equation (3.26) of [22], obtained from the Taylor series expansion of each (x_i, y_j) about (X, Y). We then desire

$$\begin{aligned} a_1 + a_2 &= 1, \\ a_3 + a_4 &= 0, \\ a_5 + a_6 &= 0. \end{aligned}$$

This gives a linear system of equations for the γ's which can be solved to yield

$$\begin{aligned} \gamma_2 &= \frac{(y_{j1} - Y)\,(x_{i3} - x_{i1}) - (x_{i1} - X)\,(y_{j3} - y_{j1})}{(x_{i2} - x_{i1})\,(y_{j3} - y_{j1}) - (x_{i3} - x_{i2})\,(y_{j2} - y_{j1})}, \\ \gamma_3 &= \frac{(y_{j1} - y_{j1})\,(x_{i1} - X) - (x_{i2} - x_{i1})\,(y_{j1} - Y)}{(x_{i2} - x_{i1})\,(y_{j3} - y_{j1}) - (x_{i3} - x_{i2})\,(y_{j2} - y_{j1})}, \\ \gamma_1 &= -\,(\gamma_2 + \gamma_3)\,. \end{aligned} \tag{6.21}$$

Once we get the coefficients γ_1, γ_2, and γ_3, we are able to compute the correction term C as

$$C = -(a_2[u] + a_4[u_x] + a_6[u_y]).$$

We can use the same coefficients γ_i and a new correction term based on the jumps in v to find the y component V of the velocity at (X, Y). These velocities (U, V) can then be used to move the interface at the control point (X, Y).

Applying this procedure at each control point (X_k^n, Y_k^n) gives the velocities (U_k^n, V_k^n). The simplest explicit method is forward Euler, in which we move the interface by shifting each control point according to

$$\begin{aligned} X_k^{n+1} &= X_k^n + \Delta t U_k^n, \\ Y_k^{n+1} &= Y_k^n + \Delta t V_k^n. \end{aligned}$$

This completes the description of an explicit IIM for the Stokes equations. In the next time step the whole process is repeated. To review, the process consists of the following:

1. Use the location of the interface, as determined by the control points, to determine the forces and jump conditions.
2. Solve three Poisson problems, using these jump conditions, to determine u_{ij}^n and v_{ij}^n on the uniform grid.
3. Interpolate to determine U_k^n and V_k^n at the control points.
4. Move the control points at these velocities for time Δt.

There are two obvious difficulties. One is that Euler's method is only first-order accurate in time. A more serious difficulty is that the system is very stiff (for reasonable values of T_0), and very small time steps must be taken to maintain stability, time steps over which there is barely any discernible motion of the interface. The problem is that small perturbations in the smoothness of the interface can lead to large forces, resulting in large transient velocities that amplify the perturbations catastrophically if used over too large a time step. This difficulty is discussed in detail by Tu and Peskin [49]. (See also [18], [34].) In order to take reasonable time steps, we must use an implicit method, as described in the next section.

7. Moving the interface—an implicit method. Steps 1 through 3 of the procedure described in the previous section can be used to define an operator $\mathcal{U}$ that maps a set of control points $\vec{X} = (\vec{X}_1, \vec{X}_2, \ldots, \vec{X}_{N_b})$ to the resulting velocities $\vec{U} = (\vec{U}_1, \vec{U}_2, \ldots, \vec{U}_{N_b})$ at the control points. We write

$$\vec{U} = \mathcal{U}(\vec{X}).$$

Applying $\mathcal{U}$ to $\vec{X}$ requires computing forces and jumps along the interface, solving three Poisson problems, and interpolating the resulting velocities back to the control points. The forward Euler method of the previous section can now be written succinctly as

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

Instead we wish to use an implicit method, such as the trapezoidal method

$$\vec{X}^{n+1} = \vec{X}^n + \frac{1}{2}\Delta t \left(\mathcal{U}(\vec{X}^n) + \mathcal{U}(\vec{X}^{n+1})\right).$$

This means that the distance each control point moves should be determined by the average of its velocity based on the old interface location and its velocity based on the new interface location. This is second-order accurate and also eliminates most stability problems, but of course it is much more difficult to apply. At time t_n, $\vec{X}^n$ is known and so $\vec{U}^n = \mathcal{U}(\vec{X}^n)$ can be computed as before. But then $\vec{X}^{n+1}$ must be determined from the implicit system $g(\vec{X}^{n+1}) = 0$, where

$$g(\vec{X}) = \vec{X} - \frac{1}{2}\Delta t\,\mathcal{U}(\vec{X}) - \left(\vec{X}^n + \frac{1}{2}\Delta t\,\vec{U}^n\right).$$

Normally a Newton-like method must be used with the trapezoidal method in order to obtain convergence of the iteration with reasonable size time steps. Newton's method requires the Jacobian matrix

$$J(\vec{X}) \equiv g'(\vec{X}) = I - \frac{1}{2}\Delta t\,\mathcal{U}'(\vec{X}). \tag{7.22}$$

Unfortunately the matrix $\mathcal{U}'(\vec{X})$ is impossible to calculate exactly, and even obtaining a finite difference approximation would be prohibitively expensive. To see this, recall that $\vec{X}$ represents the positions of all the control points (a vector with $2N_b$ entries for N_b control points), while $\mathcal{U}(\vec{X})$ represents the $2N_b$ vector of resulting velocities. Hence $\mathcal{U}'(\vec{X})$ is a $2N_b \times 2N_b$ matrix and its approximation by finite differences would require evaluating $\mathcal{U}$ $2N_b$ times. Since each evaluation of $\mathcal{U}$ requires solving three Poisson problems (plus other work), this is out of the question.

Instead we use a quasi-Newton method, in which we maintain an approximation to the Jacobian matrix J that is modified in each iteration by a low-rank update. The update is based on information obtained in each step about the directional derivative of $\mathcal{U}$ in the direction moved in this iteration. A wide variety of quasi-Newton methods are available, see, e.g., [15]. In practice one wants to avoid factoring the new matrix J in each iteration, and so either the LU factorization is updated directly or else the inverse matrix $B = J^{-1}$ is updated. We have implemented the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method with updates to B and have found this to be quite effective.

The efficiency of this method is enhanced by the fact that we have a very similar system to solve in each time step. Since the interface moves only a small amount from one time step to the next, the approximate Jacobian from one time step is still a good approximation in the next time step. In each time step we begin with the final approximate B from the previous time step as our initial B. Of course we also have a good initial guess for the solution $\vec{X}^{n+1}$ to the system $g(\vec{X}) = 0$. We can use $\vec{X}^n$ as our initial guess, or even an extrapolated value such as $2\vec{X}^n - \vec{X}^{n-1}$. We find that we need only two or three iterations of the quasi-Newton method to converge to $\vec{X}^{n+1}$.

The above comments are valid once the process is going. In the very first time step, from $\vec{X}^0$ to $\vec{X}^1$, we do not yet have a previous approximation to the Jacobian. In the first step we initialize B to the identity matrix, which is reasonable since from (7.22) we see that $J = I + O(\Delta t)$. (This seems to work in spite of the fact that, in the stiff case, we want to take Δt for which $\lambda\Delta t$ may be large for some eigenvalues λ of $\mathcal{U}'$.) In the first few time steps, before a good approximate Jacobian has been built up, more iterations are required than indicated above (five or six, typically).

8. Comparison with the IBM. The original IBM can also be applied to Stokes flow, as has been done by Tu and Peskin [49]. We have already explained the primary differences between the two algorithms and, in particular, how we expect to

achieve second-order accuracy rather than first-order accuracy by using jump conditions rather than discrete delta functions. These expectations are confirmed in the numerical results presented in the next section.

There is another fundamental difference in the approach that is worth discussing further since we believe it could also be used effectively in connection with the standard IBM. In Peskin's approach the Lagrangian points defining the boundary are also the points where the discrete delta functions are centered in the process of transferring the singular sources from the interface to the neighboring grid points. Since the diameter of the support of these delta functions is $O(h)$, where h is the uniform grid spacing, this requires a fairly dense set of points along the interface. Typically the distance between marker points on the interface is roughly h. With an implicit method, this can give a very large system of equations to solve for the new position of the interface in each time step.

By contrast, we use a much smaller set of control points to mark the interface, and interpolate by cubic splines to determine intermediate locations or forces. If the interface is smooth it is possible for the distance between these control points to be quite large relative to h with no effect on the accuracy. This leads to a much smaller system of equations to solve and allows us to use a fully implicit method quite efficiently.

Presumably, improvements could be made to a code based on Peskin's approach by using a hybrid method in which the boundary is represented by a smaller set of control points for the purpose of identifying its location (and hence in the implicit system to be solved). One could then use cubic splines to interpolate forces to a denser set of points along the interface where the discrete delta functions are applied for the purpose of spreading forces and interpolating velocities. In essence this is what we do, since we also need to know the forces and jump conditions at a denser set of points in the process of calculating the C_{ij} in the systems (4.13) and (5.15).

There are other advantages to solving for a smaller set of control points besides the obvious efficiency gained by having a smaller system of equations. Reducing the number of degrees of freedom in the interface position also leads to a much better conditioned system in many cases. Beyer [5], [6] used the IBM to model the motion of the basilar membrane in the inner ear and implemented an implicit method to deal with the rigid walls of the cochlea. He found that the system of equations was extremely ill conditioned and had to resort to a singular value decomposition to obtain good results, at considerable expense. We suspect that using a much smaller set of control points would eliminate the bulk of this ill conditioning and lead to a much more efficient method for handling rigid walls. Similar issues are also discussed in [18].

9. Numerical results. In this section we present the results from some sample calculations on the Stokes equations with immersed elastic boundaries. We compare with results obtained using the IBM where the equations (1.1) are discretized directly using a discrete delta function for the singular forces. This is the method used by Tu and Peskin [49] and we have attempted to implement the method exactly as described in that paper.

The main interest in [49] was in studying the issue of explicit versus implicit methods for moving the interface in the context of a relatively simple test problem. While their emphasis was not on obtaining optimal accuracy for this particular problem, we feel that the comparison is appropriate since we hope to extend our method to the full incompressible Navier–Stokes equations. It is important to verify that at least in this special case we are able to obtain better results than the existing IBM.

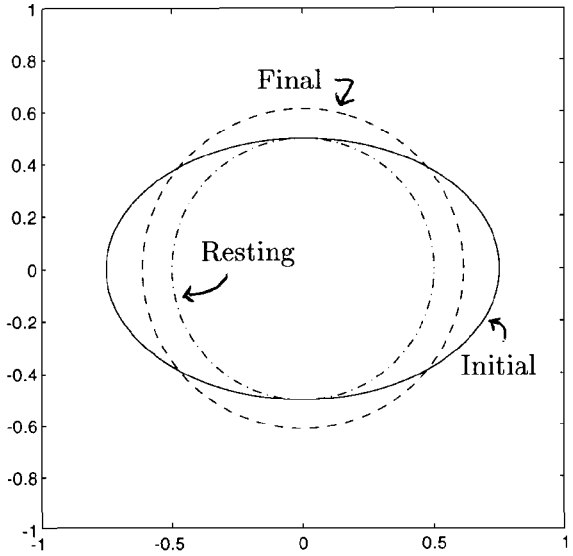
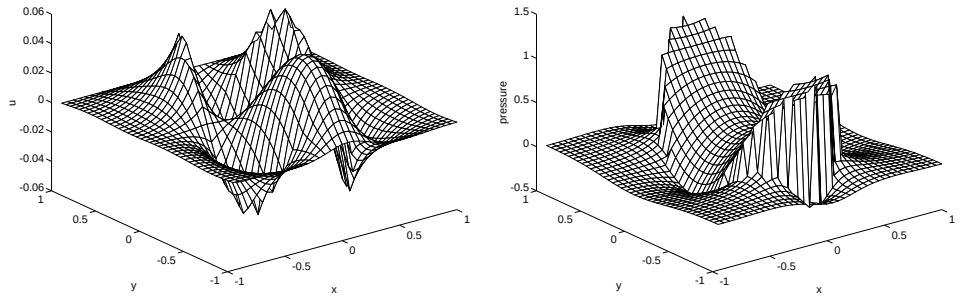


FIG. 1. *The interface at different states: the initial interface is the solid ellipse with* $a = 0.75$, $b = 0.5$. *The equilibrium position is the dashed circle with* $r_e = \sqrt{ab} \approx 0.6123\cdots$. *The resting circle, shown as a dash–dot line, has radius* $r_0 = 0.5$.

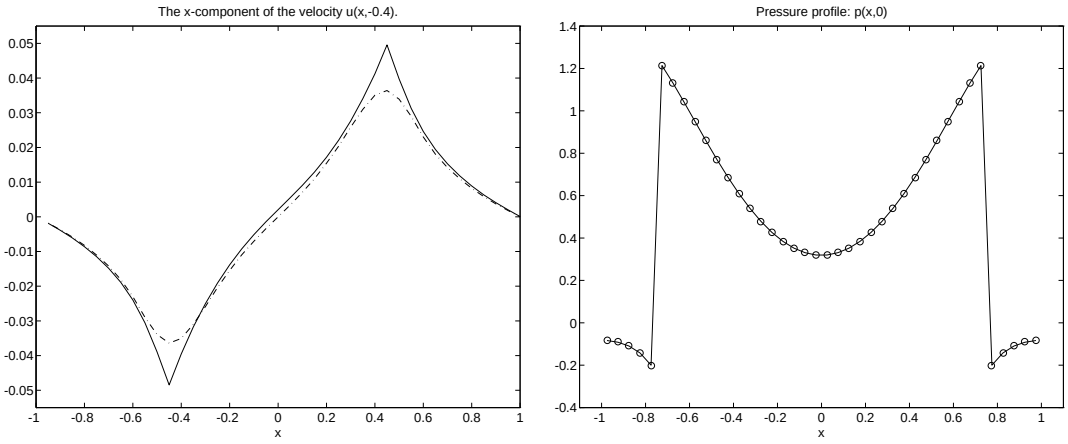


(a) (b)

FIG. 2. (a) *The* x *component of velocity* u *in the Stokes flow at* $t = 0$. *It is continuous but not smooth across the interface.* (b) *The computed pressure distribution of the Stokes flow at* $t = 0$. *The pressure is discontinuous.*

Example 9.1. This example is taken from Tu and Peskin [49]. The initial interface (the solid line in Fig. 1) is an ellipse with major and minor axes $a = 0.75, b = 0.5$, respectively. The unstretched interface (the dash–dot line in Fig. 1) is a circle with radius $r_0 = 0.5$. Due to the restoring force, the ellipse will converge to an equilibrium circle (the dashed line in Fig. 1) with radius $r_e = \sqrt{ab} \approx 0.61237$; this is larger than the unstretched interface because of the incompressibility of the enclosed fluid. So the interface is still stretched at the equilibrium state, and the nonzero boundary force is balanced by a nonzero jump in the pressure (Fig. 2(b)).

We begin by computing the velocities and pressure at time $t = 0$ based on the initial elliptical interface, before the interface has moved at all. Figure 2 shows u and p over the uniform grid. As expected, p is discontinuous across the interface while u is continuous but not smooth. Figure 3 shows this more clearly, displaying cross sections of u and p along the line $y = -0.4$. In Fig. 3(b) we see that the discontinuity in pressure is captured very sharply by the immersed interface approach.



(a) (b)

FIG. 3. (a) *A slice of u, the x component of velocity in the Stokes flow at* $t = 0$ *and* $y = -0.4$. *It is continuous, but* u_x *should be discontinuous across the interface. Solid line: IIM results; dot–dashed line: IBM results.* (b) *A slice of pressure at* $t = 0$ *and* $y = 0$. *The points and solid line both show the computed results with the IIM at the grid points. (Pressure is not available with the IBM.) Note that the large jump in pressure across the interface is computed with no smearing.*

TABLE 1

The errors in the computed p, u, and v at $t = 0$ *using the IIM method via three Poisson equations. Second-order accuracy can be observed.*

N	$\|p_N - p_{320}\|_\infty$	ratio	$\|u_N - u_{320}\|_\infty$	ratio	$\|v_N - v_{320}\|_\infty$	ratio
40	1.9730×10^{-2}		2.6739×10^{-3}		5.0411×10^{-3}	
80	1.5416×10^{-3}	12.7986	6.3611×10^{-4}	4.2035	5.5415×10^{-4}	9.0969
160	2.6087×10^{-4}	5.9094	1.1161×10^{-4}	5.6996	1.0713×10^{-4}	5.1729

In Fig. 3(a) we have also plotted the cross section of the velocity u that was computed using our implementation of the IBM on the same grid. This gives a similar result except near the interface, where it is quite smeared out with the sharp peak in velocity being lost. Since it is the velocity right at the interface that is used to move the interface, this can be expected to have a substantial impact on the overall performance of the algorithm. (We cannot directly compare the accuracy of the pressure from the two algorithms, since this is not available from the method of Tu and Peskin.)

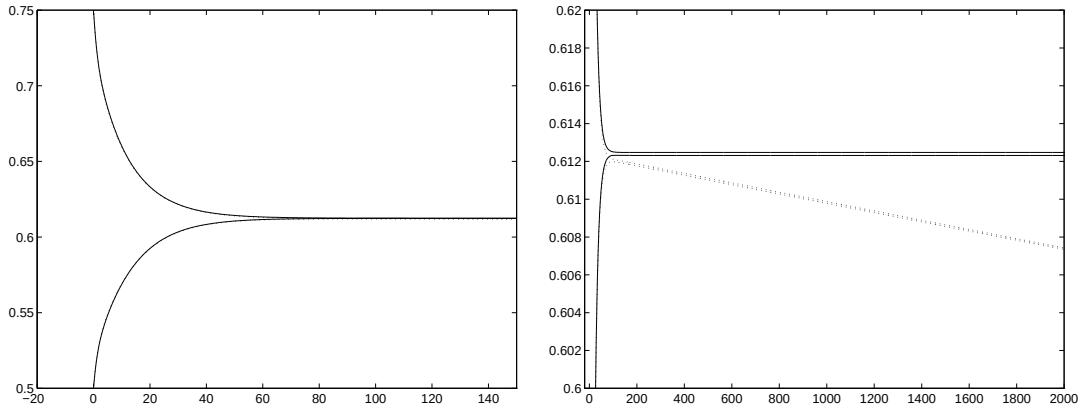
Table 1 shows the results of a grid refinement study on the IIM where the values on three different $N \times N$ grids with $N = 40,\ 80,\ 160$ are compared with a fine grid solution with $N = 320$. The errors in $p,\ u$, and v are measured in the max-norm over all N^2 grid points and displayed along with the ratios of successive errors. Since we are comparing with a computed solution on a grid that is not very much finer, we do not expect the standard error ratios of two for a first-order method or four for a second-order method. In particular, when going from $N = 80$ to $N = 160$ we expect a qth-order accurate method to produce a ratio

$$\frac{\|p_{80} - p_{320}\|}{\|p_{160} - p_{320}\|} \approx \frac{C(1/80)^q - C(1/320)^q}{C(1/160)^q - C(1/320)^q} = \frac{4^q - 1}{2^q - 1}$$

rather than the ratio 2^q. For $q = 1$ this ratio is three while for $q = 2$ it is five. Table 1 shows the final ratio to be between five and six for all three variables, indicating second-order accuracy.

TABLE 2
The errors in the computed u and v at t = 0 using the IBM. First-order accuracy can be observed.

N	$\|u_N - u_{320}\|_\infty$	ratio	$\|v_N - v_{320}\|_\infty$	ratio
40	1.0170×10^{-2}		5.0540×10^{-3}	
80	4.4694×10^{-3}	2.2755	2.0512×10^{-3}	2.4639
160	1.5012×10^{-3}	2.9773	7.4032×10^{-4}	2.7707



(a) (b)

FIG. 4. *Plot of r_{max} (upper curve) and r_{min} (lower curve) showing the distance from the origin to the interface as a function of time t on a 160×160 grid with $N_b = 160$ on the boundary. Solid line: IIM results. Dotted line: IBM results.* (a) *To plotting accuracy the behavior is identical over the time period $0 \le t \le 150$, during which convergence to a near-circle is apparent.* (b) *Over a longer time scale, it is seen that the IBM suffers leaking and the circle shrinks. Here the vertical scale is also expanded, showing that the numerical equilibrium is not exactly circular in either case.*

Table 2 shows results for the IBM, for the velocity components only. Now the final ratios are roughly three, indicating the expected first-order accuracy.

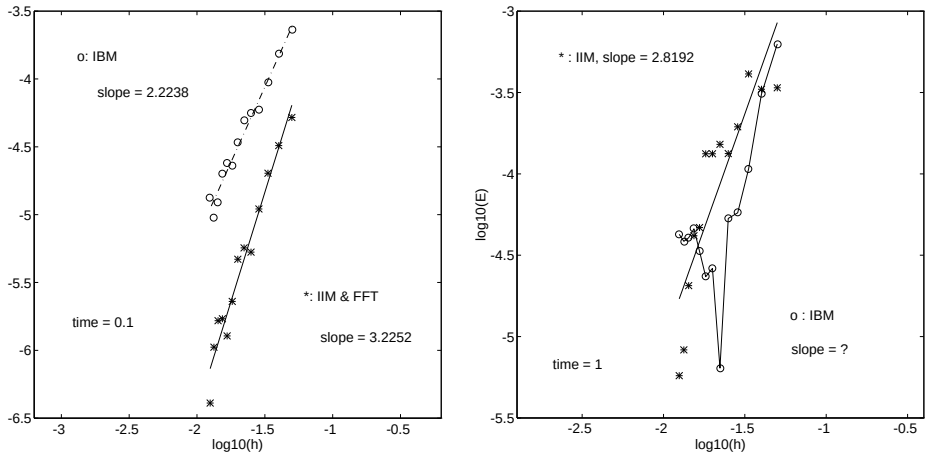
We now consider the error at later times, after the interface has moved. Comparing the solution at all the uniform grid points is difficult, since the interface may lie to one side of a given point in one calculation but slightly to the other side in a different calculation. Instead we will focus on the interface location as a measure of the accuracy, which is appropriate since this is often what we are most interested in.

One simple and effective measure is to study the values r_{min} and r_{max}, defined as

$$r^n_{min} = \min_{1 \le k \le N_b} \sqrt{(X^n_k)^2 + (Y^n_k)^2}, \qquad r^n_{max} = \max_{1 \le k \le N_b} \sqrt{(X^n_k)^2 + (Y^n_k)^2}.$$

These measure the smallest and greatest distances from the origin to the interface. Note that since we expect the interface to converge to a circle centered at the origin (by symmetry and the positioning of the original ellipse), we expect that $r^n_{min} \to r_e$ and $r^n_{max} \to r_e$ as $n \to \infty$.

Figure 4(a) shows how r_{min} and r_{max} behave computationally over a short time scale. The solid line is from our IIM with $N = N_b = 160$ and the dotted line is using the IBM on the same grid. To plot resolution these are indistinguishable on this scale. Figure 4(b) shows what happens over a longer time scale, blown up near the true equilibrium position $r_e \approx 0.61237$. With our method, a numerical equilibrium is reached that agrees well with the true equilibrium, and this equilibrium is then maintained. At $t = 2000$ we have $r_{min} \approx 0.61232$ and $r_{max} \approx 0.61248$ and this is



(a) (b)

FIG. 5. *Grid refinement studies of the error in r_{max} on a log–log scale. The solid line and the stars ($*$): IIM results. The dash–dot line and "o": IBM results.* (a) At $t = 0.01$. (b) At $t = 1$.

maintained at later times. With the IBM, some leaking is apparent which causes the circle to shrink. This continues over longer times as well. At $t = 20000$ the circle has shrunk to a radius $r \approx 0.57$ (not shown) and it still appears to be shrinking linearly with time. This problem is also mentioned by Tu and Peskin [49]. Peskin and Printz [41] have suggested a modification to fix the leaking problem, but we have not tried to implement this modification.

We can compare the accuracy over shorter times by again comparing the computed solutions with a fine grid solution. The results for r_{max} are shown in Fig. 5 at two different times, $t = 0.1$ and $t = 1.0$. At the earlier time we see a clear improvement with our method. At the later time our method still gives smoothly decreasing errors with a good rate. The IBM gives more chaotic results which look better on coarser grids but poorer on fine grids. (The results seen are quite sensitive to the particular time chosen.)

Above we have considered the errors in r_{max}. We could also look at some norm of the error along the entire interface, for example, the 2-norm. Let us take $N^* = N_b^*$ as the finest grid. For the coarser grid with $N \times N$, we take $N_b = N^*/l$, where $l = \text{int}(N^*/N)$. In this way we are guaranteed that each control point $(x_i^{(N)}, y_i^{(N)})$, $i = 1, 2, \ldots, N_b$ on the interface is also a control point for the finest grid $N^* \times N^*$ and N_b^*. Then it is possible to compute the error

$$e_N = \frac{1}{N_b} \sqrt{\sum_{i=1}^{N_b} \left(x_i^{(N)} - x_{i*l}^{(N^*)} \right)^2 + \left(y_i^{(N)} - y_{i*l}^{(N^*)} \right)^2}. \tag{9.23}$$

Our test results show that the error defined above is indeed a monotone decreasing function as N increases. In Fig. 6, we plot the global error at $t = 1$ with the finest grid being $N^* = 320$, and N and N_b being the pairs of $(40, 40)$, $(50, 40)$, $(60, 40)$, $(70, 40)$, $(80, 80)$, $(90, 80), \ldots, (150, 80)$, $(160, 160)$. Note that the number of control points N_b does not decrease smoothly with N. We do this so that N_b always divides 320, allowing direct comparison at control points with the fine grid solution, as required in (9.23).

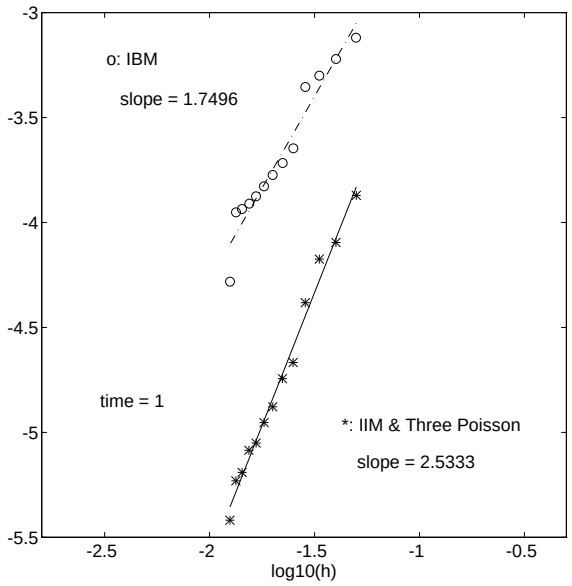


FIG. 6. *The error in the interface location at* $t = 1$ *as measured in the norm* (9.23). *Solid line and the star* ($*$)*: IIM results. Dash–dot line and the "o's": IBM results.*

Figure 6 shows that our method converges with a smaller error and a faster convergence rate than the IBM. (Again the slope is greater than we would expect if we had the exact solution to compare with rather than a fine grid solution.)

Another interesting phenomenon we can observe from Fig. 6 is that the number of control points N_b on the interface plays an important role in the IBM. When we refine the uniform grid but keep the number of control points N_b on the interface fixed, the errors obtained with their approach gradually cease to decline even as we refine the grid because the error in expressing the interface will dominate. Then a refinement of the interface grid will lead to a relatively large fall in the error as we can see in Fig. 6. There is a sharp drop in the error with the IBM when N_b changes from 40 to 80 and from 80 to 160. As expected, we must refine the grid for the domain and the interface simultaneously with this approach. In our approach, as we mentioned in section 2, we can take fewer control points on the interface with little effect on the accuracy with our method, as we can see from Fig. 6, where we have the same sequence of grids. The jumps in the error when N_b is increased are much less pronounced.

Example 9.2. We now change the example slightly and perturb a circular interface that is initially the resting circle, with $r_e = r_0$. There is still a restoring force that should bring the interface back to a circular shape, but this force now vanishes as the equilibrium (= resting) state is reached. Asymptotically there is no force exerted by the interface and no jump in pressure. With both methods, the interface will stop moving before an exact circle is reached, once the error in the discretization dominates the force. The deviation from circularity of the final numerical equilibrium gives an indication of the accuracy of the method. In Fig. 7 we have again plotted r_{min} and r_{max}, the minimum and maximum distances from the origin to the interface, for both methods on a sequence of grids. It is clear that our method is more accurate and it has been confirmed that $r_{max} - r_{min} = O(h^2)$ for large t, whereas with the IBM, $r_{max} - r_{min} = O(h)$ asymptotically.

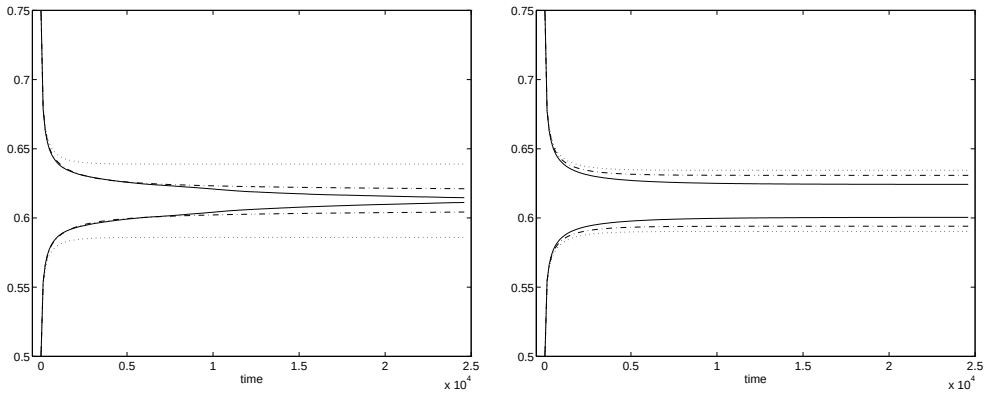


FIG. 7. *Plots of r_{max} (upper curves) and r_{min} (lower curves) for the case when the equilibrium state is the same as the resting circle, $r_0 = r_e$. Solid line, dash–dot line, and dotted line are the results obtained with 160×160, 80×80, and 40×40 grids, respectively.* (a) *IIM results.* (b) *IBM results.*

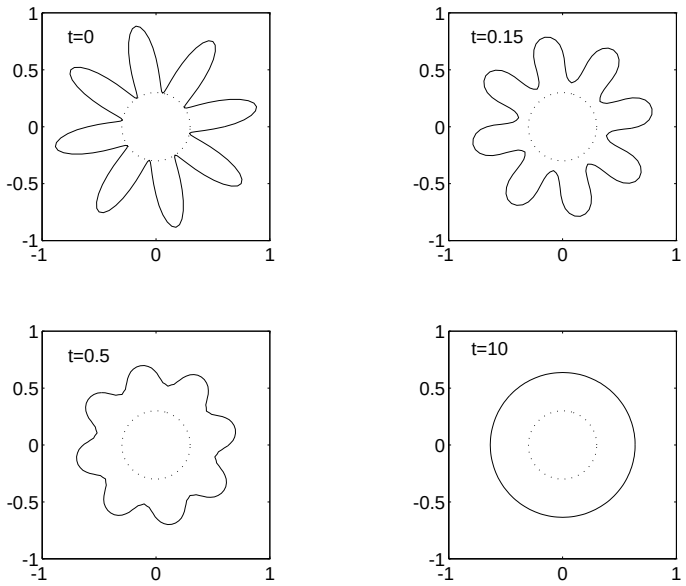


FIG. 8. *The interface at different times with a 160×160 grid. The dotted circle is the unstretched interface with $r = 0.3$. (IIM results only.)*

Example 9.3. This example shows that we can handle more complicated regions. The initial interface in polar coordinates is $\rho = 0.6 + 0.3 \sin 8\theta$. The unstretched interface is the circle with the radius $r_0 = 0.3$. We present results only for $N = 160$ and $N_b = 160$. The problem is very stiff and we need to take a fairly small time step even with the implicit method. We started with $\Delta t = O(h^2)$ but could increase the time step at later times. A comparison with the IBM reveals a similar behavior as in Example 9.1. We will not give detailed numerical results but instead present only the location of the interface at several times in Fig. 8.

10. Multifluid flows and surface tension. The method developed in the previous section can be easily adapted for the interface between two different fluids, with surface tension providing the singular force rather than an elastic membrane. Here we also allow the viscosity μ to be discontinuous.

The force strength $\vec{f}(s,t)$ is now given by

$$\vec{f}(s,t) = \gamma \frac{\partial^2}{\partial s^2} \vec{X}(s,t), \tag{10.24}$$

where the constant γ depends on physical properties of the two fluids, and s is arc-length along the interface. The vector $\partial^2 \vec{X}/\partial s^2$ is normal to the interface with magnitude equal to the curvature.

The main new feature that we need to include is the effect of gravity, which is important in most applications since the two fluids may have different densities. If gravity is directed in the negative y direction, we need only modify (1.1b) to read

$$p_y = \mu(v_{xx} + v_{yy}) + F_2 - g\rho, \tag{10.25}$$

where g is the gravitational constant and ρ is the density, which we assume has the constant value ρ_1 in one fluid and ρ_2 in the other. The Poisson problem for p then becomes

$$\left(\frac{1}{\mu} p_x\right)_x + \left(\frac{1}{\mu} p_y\right)_y = \nabla \cdot \left(\frac{1}{\mu} \vec{F}\right) - \rho g y.$$

Since ρ is piecewise constant, the term $g\rho_y$ gives only an additional delta function source along the interface, which contributes to the jump in the normal derivative p_n across the interface. In fact this is the only contribution to this jump, since the force (10.24) is normal to the interface and so $\hat{f}_2(s,t) = 0$ in (3.12). So we have

$$\left[\frac{1}{\mu} p_n\right] = \frac{1}{2}\left(\frac{1}{\mu^+} + \frac{1}{\mu^-}\right) g[\rho] \sin\theta,$$

where the $\sin\theta$ term arises from the fact that the delta function source is directed vertically, and hence at angle θ to the interface. The jump conditions for p, u_n, and v_n are still given by (3.12) with $\hat{f}_1 = f$ and $\hat{f}_2 = 0$. Note that the velocity is now continuously differentiable across the interface, simplifying the procedure for interpolation to the interface that was presented in section 6.

Even if $\rho_1 = \rho_2$, the addition of gravity will induce a hydrostatic pressure gradient that is linear in y. This means that periodic boundary conditions are no longer reasonable. However, if we are computing on the rectangle $\Omega = [a,b] \times [c,d]$ and we set

$$\rho_0 = \frac{1}{(b-a)(d-c)} \int\int_\Omega \rho(x,y)\, dx\, dy,$$

then we can write p as

$$p(x,y,t) = g\rho_0(d-y) + \tilde{p}(x,y,t),$$

where $\tilde{p}$ is the deviation from the linear profile obtained from the average density ρ_0. If the boundaries are well away from the interface, then we expect $\tilde{p}$ to be roughly constant along the entire boundary of $\partial\Omega$, so the periodic boundary conditions are physically reasonable.

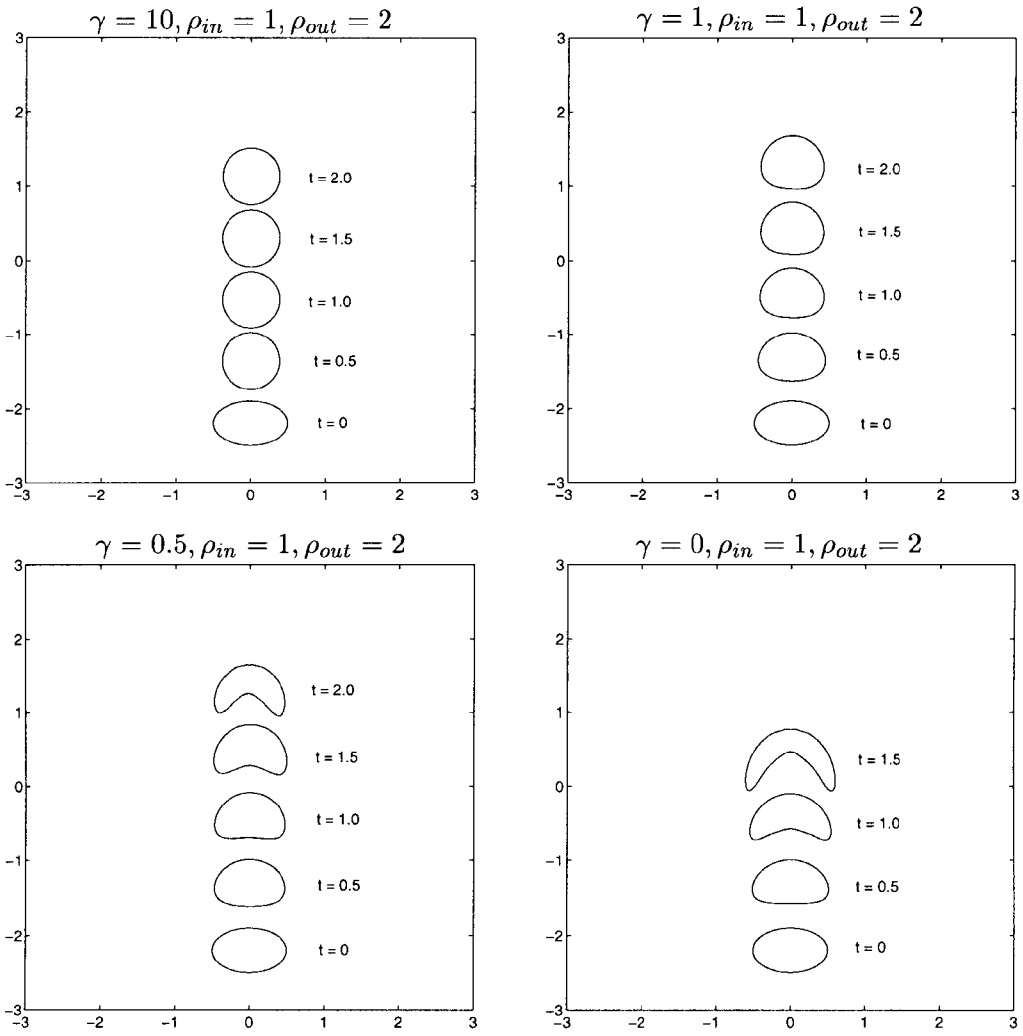


FIG. 9. *Bubble computations with different values of the surface tension parameter γ. In these computations $\rho = 2$ outside and $\rho = 1$ inside the bubble and $\mu = 1$ everywhere. The computations are on a 160×160 grid with $N_b = 80$ points on the boundary. The computation with no surface tension ($\gamma = 0$) breaks down before $t = 2$ when the bubble splits into pieces.*

In terms of the pressure deviation $\tilde{p}$, (10.25) becomes

$$\tilde{p}_y = \mu(v_{xx} + v_{yy}) + F_2 - g(\rho - \rho_0).$$

Example 10.1. We consider a rising bubble of fluid with density $\rho_1 = 1$ inside the bubble and density $\rho_2 = 2$ outside. We take the same viscosity in both fluids, $\mu_1 = \mu_2 = 1$, though this is not necessary (see Example 10.2 below). Figure 9 shows experiments with four different values of the surface tension coefficient $\gamma = 10,\ 1,\ 0.5$, and 0. In each case the bubble was initialized to an elliptical shape, $X = 0.5\cos(\theta),\ Y = 0.3\sin(\theta) - 2.2$.

When γ is large, the surface tension is sufficiently strong to bring the bubble back to a nearly circular shape even as it rises. For smaller values of γ, the bubble is distorted. For sufficiently small values, the bubble would eventually split into pieces.

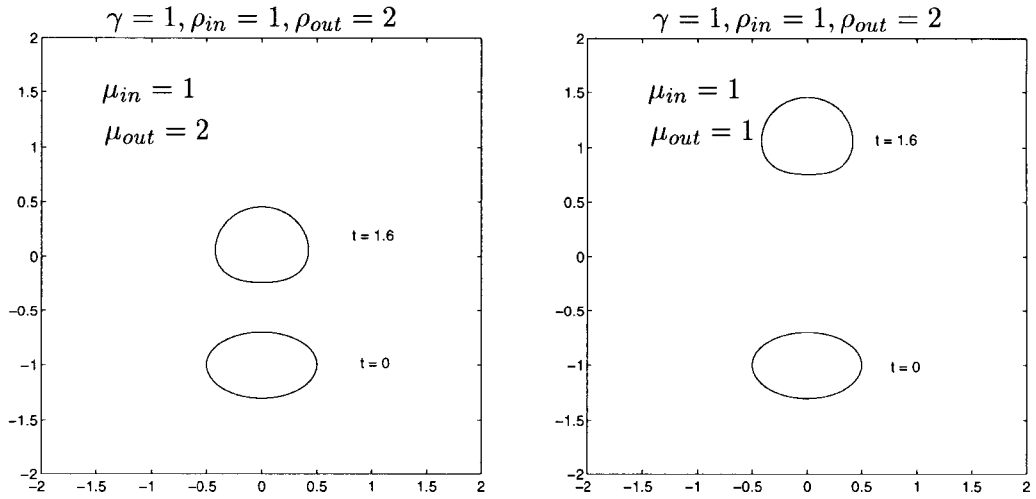


FIG. 10. *Bubble computations with discontinuous viscosity. On the left $\mu = 2$ outside and $\mu = 1$ inside. On the right, $\mu = 1$ everywhere. The surface tension parameter $\gamma = 1$ in both cases.*

Our current code cannot handle this change in topology. (See, e.g., [47] for such a calculation.)

This behavior agrees qualitatively with the known behavior of axisymmetric three-dimensional bubbles, the case most frequently treated in the literature. (See, e.g., [3, 10, 46].) Our computations, however, are purely two-dimensional, which corresponds to the cross section of a cylindrical bubble in three dimensions. The additional curvature effects in the more realistic axisymmetric case are known to have an effect on the shape of bubbles (e.g., [35]) and so a direct comparison is not possible.

We also note that if we start with a circular bubble rather than an ellipse, the bubble remains circular (to reasonable accuracy) for all values of γ. This also agrees with expected behavior [10].

Example 10.2. The next example shows that discontinuities in viscosity μ can also be incorporated. In this case the Poisson problems to be solved for the velocities u and v have discontinuous coefficients and are solved using the techniques of [22]. The velocities are continuous but now have jumps in their normal derivatives arising from the jump conditions

$$[\mu u_n] = [\mu v_n] = 0.$$

Figure 10 shows the results with $\mu = 2$ outside and $\mu = 1$ inside, and, for comparison, also the computation with $\mu = 1$ everywhere. As expected, the bubble moves more slowly in the more viscous fluid. In this example we have also changed the boundary condition relative to the previous example, and impose $u = v = 0$ at the top and bottom, due to limitations in our current code for the discontinuous coefficient Poisson problem. This explains why the bubble with $\mu = 1$ everywhere rises more slowly than the corresponding bubble in Fig. 9.

These examples demonstrate that our technique is capable of dealing with discontinuities in density and viscosity as well as singular forces at the interface. The ability to handle discontinuous coefficients as well as singular forces is another advantage of our approach over the standard IBM, with which the discontinuities are typically smeared over some region of width $O(h)$, e.g., [47], [51].

11. Extensions. Although we have presented our method in the context of two-dimensional flows, the ideas extend quite simply to three space dimensions. The immersed interface Poisson solver for three-dimensional problems is described by Li [27], [25] and can be used directly. The main difficulty is in representing the interface, which is now a surface, by a finite set of control points. Various techniques are currently under consideration and we hope to develop a three-dimensional code in the future.

The special case of axisymmetric flow can be solved as a two-dimensional problem, if the Laplacian operator is suitably modified. In (r, y) coordinates, the Laplacian is $\nabla^2 = \partial_{rr} + \frac{1}{r}\partial_r + \partial_{yy}$ and this must be used in each of the three Poisson problems solved with our approach. The jump conditions must also be modified to incorporate the three-dimensional curvature effects.

As noted in the introduction, decoupling the problem into three Poisson problems depends on the fact that periodic boundary conditions are used. With other boundary conditions, e.g., no-slip conditions at solid walls, we would know u and v but not p along the boundary, and hence would not have a boundary condition for the Poisson problem for pressure. If arbitrary boundary conditions are specified for p and the three Poisson problems solved as described above, then the divergence will satisfy Laplace's equation

$$\nabla^2(u_x + v_y) = 0,$$

but this guarantees that $u_x + v_y \equiv 0$ only if we also impose the boundary condition $u_x + v_y = 0$ along the boundary. This is the correct additional boundary condition which must be used to uniquely define the pressure. This creates a troublesome coupling between the Poisson problems.

One possibility is to introduce the values of p along the boundary as another set of unknowns, with a discrete form of the divergence boundary condition as another set of equations to be solved via the quasi-Newton procedure along with the new locations of the control points. This would give a much larger nonlinear system. Another possibility is to revert to solving the coupled system of equations (1.1) directly as done by Tu and Peskin. Results reported in [25] indicate that other aspects of the immersed interface method presented here can be carried over directly to solving the coupled system.

In this paper we have considered only Stokes flow in the zero Reynolds number limit of creeping flow, where the inertial time derivatives of the velocity can be neglected. We are currently working on a version that includes these inertial terms in the momentum equations (1.1a) and (1.1b). The techniques presented here can be used to compute the time derivative of the velocities at both time t_n and t_{n+1}, and then a Crank–Nicolson time differencing can be used to achieve a second-order accurate method in time. Again the quasi-Newton method can be used to solve for the new interface location as part of this updating process. The primary new difficulty arises from the fact that the time derivative will not be smooth in time at any Cartesian grid point crossed by the interface during the time step. This can be accounted for by including additional terms in the Crank–Nicolson algorithm to correct for these jumps. Such techniques have been successfully used for one-dimensional problems in [7], [26] and should carry over to multidimensional problems. This work will be reported on elsewhere.

12. Appendix. Derivation of jump conditions. Here we present a brief derivation of the jump conditions (3.12). We let $\delta(\vec{x})$ be the two-dimensional delta function as defined in (1.3) and (1.4), and let $\phi(x, y)$ be an arbitrary twice continuously differentiable test function defined on an appropriate region Ω.

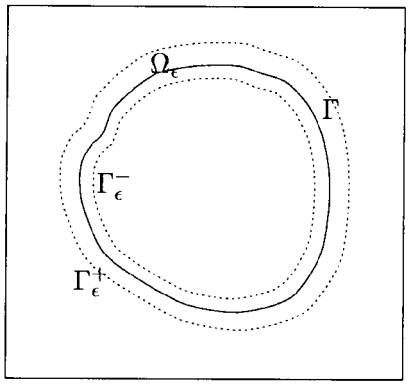


FIG. 11. *Diagram used for the derivation of jump relations.*

For a vector function $\vec{G} = [G_1(x,y), G_2(x,y)]^T$, by using Green's integral theorem, we know that

$$\iint_\Omega (\nabla \cdot \vec{G})\,\phi\,dx\,dy = \int_{\partial\Omega} (\vec{G}\cdot\vec{n})\,\phi\,ds - \iint_\Omega \vec{G}\cdot\nabla\phi\,dx\,dy.$$

We can use this to generalize the one-dimensional result

$$\int \delta'(x-\alpha)\phi(x)\,dx = -\phi'(\alpha)$$

to two dimensions, where $\nabla\cdot\vec{F}$ now behaves like the derivative of a delta function in the direction normal to the interface. Multiplying by ϕ and integrating gives

$$\begin{aligned}\iint_\Omega (\nabla\cdot\vec{F})\,\phi(x,y)\,dx\,dy &= \iint_\Omega \left\{\int_\Gamma \nabla\cdot\vec{f}(s,t)\ \delta\left(\vec{x}-\vec{X}(s,t)\right)\,ds\right\}\phi(x,y)\,dxdy \\ &= -\int_\Gamma \left(f_1(s,t)\,\frac{\partial\phi}{\partial x}\,(X(s),Y(s)) + f_2(s,t)\,\frac{\partial\phi}{\partial y}\,(X(s),Y(s))\right)ds. \qquad (12.26)\end{aligned}$$

Referring to Fig. 11, we take a belt domain Ω_ϵ which encloses the interface Γ. Let Γ_ϵ^+ and Γ_ϵ^- be the outer and inner boundary of Ω_ϵ, respectively. Here ϵ is some measure of the distance from Γ to Γ^+ and Γ^-.

From the Poisson equation (1.2) for pressure we have

$$\begin{aligned}\iint_{\Omega_\epsilon} (\nabla^2 p)\,\phi\,dx\,dy &= \iint_{\Omega_\epsilon} \phi\,\nabla\cdot\vec{F}\,dx\,dy \\ &= -\int_\Gamma \left(f_1\,\frac{\partial\phi}{\partial x} + f_2\,\frac{\partial\phi}{\partial y}\right)ds. \qquad (12.27)\end{aligned}$$

Referring to Fig. 11, let us handle the left-hand side of (12.27) first by using Green's theorem repeatedly:

$$\begin{aligned}\iint_{\Omega_\epsilon} (\nabla^2 p)\,\phi\,dx\,dy &= \int_{\Gamma_\epsilon^+} (\nabla p^+\cdot\vec{n})\phi\,ds + \int_{\Gamma_\epsilon^-} (\nabla p^-\cdot(-\vec{n}))\phi\,ds - \iint_{\Omega_\epsilon} \nabla p\cdot\nabla\phi\,dx\,dy \\ &= \int_{\Gamma_\epsilon^+} p_n^+\,\phi\,ds - \int_{\Gamma_\epsilon^-} p_n^-\,\phi\,ds - \int_{\Gamma_\epsilon^+} (\nabla\phi\cdot\vec{n})\ p^+\,ds \\ &\quad - \int_{\Gamma_\epsilon^-} (\nabla\phi\cdot(-\vec{n}))\ p^-\,ds + \iint_{\Omega_\epsilon} p\,(\nabla^2\phi)\,dx\,dy,\end{aligned}$$

where the superscripts + and − indicate the values taken from outside and inside of the interface Γ, respectively. Notice that ϕ is twice continuously differentiable and p is bounded and discontinuous only along the interface. So as ϵ approaches zero, we have

$$\iint_{\Omega_\epsilon} p\,(\nabla^2\phi)\,dx\,dy \longrightarrow 0 \qquad \text{as } \epsilon \to 0,$$

$$\iint_{\Omega_\epsilon} (\nabla^2 p)\,\phi\,dx\,dy \longrightarrow \int_\Gamma [p_n]\,\phi\,ds - \int_\Gamma [p]\,\phi_n\,ds \qquad \text{as } \epsilon \to 0. \tag{12.28}$$

To deal with the right-hand side of (12.27), we express $\partial\phi/\partial x$ and $\partial\phi/\partial y$ in terms of normal and tangential derivatives along the interface

$$\phi_n = \nabla\phi\cdot\vec{n} = \frac{\partial\phi}{\partial x}\cos\theta + \frac{\partial\phi}{\partial y}\sin\theta, \tag{12.29}$$

$$\phi_s = \nabla\phi\cdot\vec{\tau} = -\frac{\partial\phi}{\partial x}\sin\theta + \frac{\partial\phi}{\partial y}\cos\theta. \tag{12.30}$$

After solving the linear equations above for $\partial\phi/\partial x$ and $\partial\phi/\partial y$, we get

$$\frac{\partial\phi}{\partial x} = \phi_n\cos\theta - \phi_s\sin\theta, \tag{12.31}$$

$$\frac{\partial\phi}{\partial y} = \phi_n\sin\theta + \phi_s\cos\theta. \tag{12.32}$$

Using these two equations in (12.27) and collecting terms, we have

$$\begin{aligned}\int_\Gamma \left(f_1\frac{\partial\phi}{\partial x} + f_2\frac{\partial\phi}{\partial y}\right) ds &= \int_\Gamma \left((f_1\cos\theta + f_2\sin\theta)\frac{\partial\phi}{\partial n} + (f_2\cos\theta - f_1\sin\theta)\frac{\partial\phi}{\partial s}\right) ds \\ &= \int_\Gamma \left(\hat{f}_1\phi_n + \hat{f}_2\phi_s\right) ds.\end{aligned}$$

Integrating by parts in the second term and noting that Γ is closed, we may rewrite the equation above as

$$\int_\Gamma \left(f_1\frac{\partial\phi}{\partial x} + f_2\frac{\partial\phi}{\partial y}\right) ds = \int_\Gamma \left(\hat{f}_1\phi_n - \frac{\partial\hat{f}_2}{\partial s}\,\phi\right) ds.$$

Comparing this with (12.28) and using the fact that ϕ is arbitrary, we must have

$$[p] = \hat{f}_1,$$
$$[p_n] = \frac{\partial\hat{f}_2}{\partial s}.$$

To get the jump for u_n, we multiply by $\phi(x,y)$ on both sides of (1.1a) and integrate

$$\iint_{\Omega_\epsilon} \mu(\nabla^2 u)\,\phi\,dx\,dy - \iint_{\Omega_\epsilon} p_x\,\phi\,dx\,dy = -\int_\Gamma f_1(s)\phi\,ds. \tag{12.33}$$

The first term of the left-hand side of (12.33) is

$$\begin{aligned}\iint_{\Omega_\epsilon} \mu(\nabla^2 u)\,\phi\,dx\,dy &= \int_{\Gamma_\epsilon^+} \mu^+ u_n^+\,\phi\,ds - \int_{\Gamma_\epsilon^-} \mu^- u_n^-\,\phi\,ds - \iint_{\Omega_\epsilon} \mu\,(\nabla u\cdot\nabla\phi)\,dx\,dy \\ &\longrightarrow \int_\Gamma [\mu u_n]\,\phi\,ds, \qquad \text{as } \epsilon \to 0.\end{aligned} \tag{12.34}$$

The second term of the left-hand side of (12.33) is

$$\begin{aligned}\iint_{\Omega_\epsilon} p_x\,\phi\,dx\,dy &= \iint_{\Omega_\epsilon} \phi\,\nabla\cdot\begin{bmatrix} p \\ 0 \end{bmatrix} dx\,dy \\ &= \int_{\Gamma_\epsilon^+} \phi\,\left(\,[p^+,\,0]^T\cdot\vec{n}\right)\,ds - \int_{\Gamma_\epsilon^-} \phi\,\left(\,[p^-,\,0]^T\cdot\vec{n}\right)\,ds \\ &\quad - \iint_{\Omega_\epsilon}\left[\frac{\partial\phi}{\partial x},\,\frac{\partial\phi}{\partial y}\right]^T\cdot\begin{bmatrix} p \\ 0 \end{bmatrix} dx\,dy \\ &\longrightarrow \int_\Gamma \phi\,[p]\,\cos\theta\,ds, \qquad \text{as } \epsilon\to 0, \end{aligned} \tag{12.35}$$

where $\vec{n} = [\,\cos\theta,\,\sin\theta\,]^T$. Since ϕ is arbitrary, from (3.11) and (12.35), we must have

$$\begin{aligned}[\mu u_n] &= [p]\,\cos\theta - f_1 \\ &= \cos\theta\,(f_1\cos\theta + f_2\sin\theta) - f_1 \\ &= \sin\theta\,(-f_1\,\sin\theta + f_2\,\cos\theta) \\ &= \hat{f}_2\,\sin\theta. \end{aligned}$$

Similarly, for v we can get

$$\begin{aligned}[\mu v_n] &= [p]\,\sin\theta - f_2 \\ &= \cos\theta\,(f_1\,\sin\theta - f_2\,\cos\theta) \\ &= -\hat{f}_2\,\cos\theta. \end{aligned}$$

This completes the derivation of (3.12).

Acknowledgments. We have benefited from conversations with many people during the course of this work. In particular, it is a pleasure to acknowledge the encouragement and advice of Charlie Peskin. Much of this work was done while the second author was a graduate student in applied mathematics at the University of Washington. This work was completed while the first author was visiting the Scientific Computing Division of the National Center for Atmospheric Research, supported by the National Science Foundation.

REFERENCES

[1] A. S. Almgren, J. B. Bell, P. Colella, and T. Marthaler, *A Cartesian Grid Projection Method for the Incompressible Euler Equations in Complex Geometries*, Lawrence Livermore National Lab Report UCRL-JC-118091, Livermove, CA, 1994.

[2] G. P. Astrakantsev, *Methods of fictitious domains for a second order elliptic equation with natural boundary conditions*, U.S.S.R. Comput. Math. and Math. Phys., 18 (1978), pp. 114–121.

[3] G. K. Batchelor, *An Introduction to Fluid Dynamics*, Cambridge University Press, London, 1967.

[4] M. Berger and R. J. LeVeque, *A rotated difference scheme for Cartesian grids in complex geometries*, AIAA Conference on Computational Fluid Dynamics, CP-91-1602, Honolulu, HI, 1991.

[5] R. P. Beyer, *A Computational Model of the Cochlea Using the Immersed Boundary Method*, Ph.D. thesis, University of Washington, Seattle, WA, 1989.

[6] R. P. Beyer, *A computational model of the cochlea using the immersed boundary method*, J. Comput. Phys., 98 (1992), pp. 145–162.

[7] R. P. Beyer and R. J. LeVeque, *Analysis of a one-dimensional model for the immersed boundary method*, SIAM J. Numer. Anal., 29 (1992), pp. 332–364.

[8] C. Börgers and O. Widlund, *On finite element domain imbedding methods*, SIAM J. Numer. Anal., 27 (1990), pp. 963–978.
[9] B. L. Buzbee, F. W. Dorr, J. A. George, and G. H. Golub, *The direct solution of the discrete Poisson equation on irregular grids*, SIAM J. Numer. Anal., 8 (1971), pp. 722–736.
[10] R. Clift, J. R. Grace, and M. E. Weber, *Bubbles, Drops and Particles*, Academic Press, New York, 1978.
[11] H. H. D. Clarke and M. Salas, *Euler Calculations for Multielement Airfoils Using Cartesian Grids*, AIAA Paper 85-0291, Washington, DC, 1985.
[12] D. De Zeeuw and K. Powell, *An Adaptively-Refined Cartesian Mesh Solver for the Euler Equations*, AIAA Paper 91-1542, Washington, DC, 1991.
[13] L. J. Fauci, *Interaction of oscillating filaments–a computational study*, J. Comput. Phys., 86 (1990), pp. 294–313.
[14] A. L. Fogelson, *A mathematical model and numerical method for studying platelet adhesion and aggregation during blood clotting*, J. Comput. Phys., 56 (1984), pp. 111–134.
[15] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, New York, 1981.
[16] R. Glowinski, T.-S. Pan, and J. Periaux, *A fictitious domain method for Dirichlet problem and applications*, Comput. Methods Appl. Mech. Engrg., 111 (1994), pp. 283–303.
[17] R. Glowinski, T.-S. Pan, and J. Periaux, *A fictitious domain method for external incompressible viscous flow modeled by Navier–Stokes equations*, Comput. Methods Appl. Mech. Engrg., 112 (1994), pp. 133–148.
[18] T. Y. Hou, J. S. Lowengrub, and M. J. Shelley, *Removing the stiffness from interfacial flows with surface tension*, J. Comput. Phys., 114 (1994), pp. 312–338.
[19] O. A. Ladyzhenskaya, *The Mathematical Theory of Viscous Incompressible Flow*, Gordon and Breach, New York, 1963.
[20] W. E. Langlois, *Slow Viscous Flow*, Macmillan, New York, 1964.
[21] R. J. LeVeque, *High resolution finite volume methods on arbitrary grids via wave propagation*, J. Comput. Phys., 78 (1988), pp. 36–63.
[22] 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.
[23] R. J. LeVeque and C. Zhang, *Immersed interface methods for wave equations with discontinuous coefficients*, Wave Motion, to appear.
[24] R. J. LeVeque and C. Zhang, *Finite difference methods for wave equations with discontinuous coefficients*, in Proc. 1995 ASCE Conference, Boulder, CO, S. Sture, ed., to appear.
[25] Z. Li, *The Immersed Interface Method—A Numerical Approach for Partial Differential Equations with Interfaces*, Ph.D. thesis, University of Washington, Seattle, WA, 1994.
[26] Z. Li, *Immersed Interface Methods for One-Dimensional Moving Interface Problems*, CAM Report #95-01, University of California, Los Angeles, CA, 1995; Numer. Algorithms, to appear.
[27] Z. Li, *A note on immersed interface methods for three dimensional elliptic equations*, Comput. Math. Appl., 31 (1996), pp. 9–17.
[28] Z. Li and A. Mayo, *ADI methods for heat equations with discontinuties along an arbitrary interface*, Proc. Sympos. Appl. Math., 48 (1994), pp. 311–316.
[29] G. I. Marchuk, Y. A. Kuznetsov, and A. M. Matsokin, *Fictitious domain and domain decomposition methods*, Soviet J. Numer. Anal. Math. Modelling, 1 (1986), pp. 3–35.
[30] A. Mayo, *On the Rapid Evaluation of Heat Potentials on General Regions*, IBM Technical Report 14305, Yorktown Heights, NY.
[31] A. Mayo, *Rapid Fourth Order Accurate Methods for the Solution of the Stokes Problem in the Presence of an Immersed Boundary*, IBM report RC19298, Yorktown Heights, NY.
[32] A. Mayo, *The fast solution of Poisson's and the biharmonic equations on irregular regions*, SIAM J. Numer. Anal., 21 (1984), pp. 285–299.
[33] A. Mayo and A. Greenbaum, *Fast parallel iterative solution of Poisson's and the biharmonic equations on irregular regions*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 101–118.
[34] 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.
[35] M. Miksis, J. M. Vanden-Broeck, and J. B. Keller, *Axisymmetric bubbles or drops in a uniform flow*, J. Fluid Mech., 108 (1981), pp. 89–100.
[36] S. Osher and J. A. Sethian, *Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.
[37] 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 complex geometries*, J. Comput. Phys., 120 (1995), pp. 278–304.

[38] C. S. Peskin, *Numerical analysis of blood flow in the heart*, J. Comput. Phys., 25 (1977), pp. 220–252.
[39] C. S. Peskin, *Lectures on mathematical aspects of physiology*, Lectures in Appl. Math., 19 (1981), pp. 69–107.
[40] C. S. Peskin and D. M. McQueen, *Modeling prosthetic heart valves for numerical analysis of blood flow in the heart*, J. Comput. Phys., 37 (1980), pp. 113–132.
[41] 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.
[42] C. Pozrikidis, *Boundary Integral and Singularity Methods for Linearized Viscous Flow*, Cambridge University Press, London, 1992.
[43] W. Proskurowski and O. Widlund, *On the numerical solution of Helmholtz's equation by the capacitance matrix method*, Math. Comp., 30 (1976), pp. 433–468.
[44] J. J. Quirk, *An Alternative to Unstructured Grids for Computing Gas Dynamic Flow Around Arbitrarily Complex Two-Dimensional Bodies*, ICASE Report 92-7, Hampton, VA, 1992.
[45] S. S. Samant, J. E. Bussoletti, F. T. Johnson, R. H. Burkhart, B. L. Everson, and R. G. Melvin, *TRANAIR: A Computer Code for Transonic Analyses of Arbitrary Configurations*, AIAA Paper 87-0034, Washington, DC, 1987.
[46] H. Stone, *Dynamics of drop deformation and breakup in viscous flows*, Ann. Rev. Fluid Mech., 26 (1994), pp. 65–102.
[47] M. Sussman, P. Smereka, and S. Osher, *A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow*, University of California, CAM Report 93-18, Los Angeles, CA, 1993.
[48] P. N. Swarztrauber, *Fast Poisson solvers*, in Studies in Numerical Analysis, G. H. Golub, ed., vol. 24, The Mathematical Association of America, Washington, DC, 1984, pp. 319–370.
[49] 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.
[50] S. O. Unverdi and G. Tryggvason, *Computations of multi-fluid flows*, Physica D, 60 (1992), pp. 70–83.
[51] S. O. Unverdi and G. Tryggvason, *A front-tracking method for viscous, incompressible, multi-fluid flows*, J. Comput. Phys., 100 (1992), pp. 25–37.
[52] B. Wedan and J. South, *A method for solving the transonic full-potential equations for general configurations*, in Proc. AIAA Computational Fluid Dynamics Conference, Danvers, CT, 1983, pp. 515–526.