

A PARTIALLY IMPLICIT HYBRID METHOD FOR COMPUTING INTERFACE MOTION IN STOKES FLOW

ANITA T. LAYTON

Department of Mathematics, Duke University
Durham, NC 27708, USA

J. THOMAS BEALE

Department of Mathematics, Duke University
Durham, NC 27708, USA

(Communicated by the associate editor name)

ABSTRACT. We present a partially implicit hybrid method for simulating the motion of a stiff interface immersed in Stokes flow, in free space or in a rectangular domain with boundary conditions. We assume the interface is a closed curve which remains in the interior of the computational region. The implicit time integration is based on the small-scale decomposition approach and does not require the iterative solution of a system of nonlinear equations. First-order and second-order versions of the time-stepping method are derived systematically, and numerical results indicate that both methods are substantially more stable than explicit methods. At each time level, the Stokes equations are solved using a hybrid approach combining nearly singular integrals on a band of mesh points near the interface and a mesh-based solver. The solutions are second-order accurate in space and preserve the jump discontinuities across the interface. Finally, the hybrid method can be used as an alternative to adaptive mesh refinement to resolve boundary layers that are frequently present around a stiff immersed interface.

1. Introduction. Substantial effort has been directed to developing computational techniques for simulating moving boundaries or interfaces within a viscous fluid domain. The most notable example is perhaps the immersed boundary method, developed by Peskin [25] for solving the full incompressible Navier-Stokes equations with moving boundaries. Moving boundaries are represented by means of Lagrangian markers, at which boundary forces are computed. These forces are then transferred to an underlying Cartesian grid via a regularization using discrete (smoothed) delta functions, and the Navier-Stokes equations are solved on the Cartesian grid. In principle, the Lagrangian representation of the immersed boundary allows complicated boundaries to be incorporated without significant difficulty. The immersed boundary method was originally developed for studying blood flow through a beating heart [24], but has since been applied to a wide variety of problems.

2000 *Mathematics Subject Classification.* Primary: 76N07, 58F17; Secondary: 65N06.

Key words and phrases. Stokes flow, stiff forces, boundary layers, adaptive mesh refinement, finite difference method, interface problems.

This work was supported in part by the National Science Foundation under grants DMS-0701412 (first author) and DMS-0806482 (second author).

In the limit of zero Reynolds number, the inertial acceleration and advection terms in the Navier-Stokes equations can be neglected, and the resulting fluid motion can be described using the simpler Stokes equations. Despite their simplicity relative to the full Navier-Stokes equations, the Stokes equations have many applications. Indeed, the Stokes equations with an immersed membrane have been used to simulate the deformation and cleavage of fluid droplets [17] and to simulate biofilms [7].

One difficulty that researchers of immersed boundary problems have to contend with is that boundary forces frequently impose a severe restriction on time-step size in order to maintain numerical stability. This challenge has been well documented in literature [9, 15, 25, 28]. It is very desirable to treat these forces implicitly to advance the interface in time. Much effort has been invested in developing implicit and semi-implicit versions of the immersed boundary method and related methods, e.g., [6, 9, 11, 12, 14, 15, 20, 21, 22, 30]. Nonetheless, the treatment of stiff forces remains a challenge. The implicit or semi-implicit versions of the immersed boundary-type methods usually require the solution of a large system of coupled nonlinear equations via iterations, and the convergence of those iterations can be a concern. Perhaps owing to that difficulty, the majority (though by no means all) of the implementations of the immersed boundary and immersed interface methods are explicit ones. Recently Newren and co-workers [22] proved that a lagged-operators semi-implicit discretization scheme, originally introduced by Peskin [24], is unconditionally stable in its first- or second-order Crank-Nicolson form when inertia is neglected and the interfacial force is linear and self-adjoint. Mori and Peskin [21] considered a variation of this scheme and proposed a fully implicit method in which the system that requires iterative solves has the same structure as the linearized semi-implicit discretization at each iterate. They used the Krylov subspace methods to solve the linear system. Also recently, Ceniceros and co-workers [6] proposed cost-effective computational strategies for solving the linear systems arising from that semi-implicit discretization.

An alternative to an implicit method is to design an approximation to an implicit step for the most singular part of the velocity, or equivalently the part in the high wave numbers, and to use this to modify an explicit method. This is the small-scale decomposition approach of Hou, Lowengrub, and Shelley in [10]. It has the advantage of not requiring the iterative solution of systems of equations. It has been applied to Stokes flow in [12, 13, 27], and a similar approach was used as a preconditioner in [31]. Hou and Shi developed a version of the immersed boundary method for both Stokes flow [12] and Navier-Stokes flow [11] using this approach with arclength-tangent angle coordinates for the interface. We use the small-scale decomposition approach to develop first-order and second-order accurate versions of the present method that allow much larger time steps than a fully explicit method. We present a systematic derivation that could be useful with more general forces.

Stiff boundary forces frequently yield boundary layers neighboring the immersed boundary, owing to the large (higher-order) derivatives in the fluid solution. When standard finite difference or finite element methods are used, the boundary layers may be difficult to resolve unless a sufficiently refined spatial resolution is used. Instead of using a uniformly fine spatial grid, which may require prohibitively large computational times, one could use adaptive mesh refinement (AMR) [5]. Below we describe an alternative to AMR which is equally cost effective but potentially

easier to implement.

The Stokes equations for the fluid motion are

$$\nabla p = \mu \Delta \mathbf{u} + \mathbf{F}, \quad (1)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (2)$$

where $\mathbf{u} = (u, v)$ denotes the fluid velocity; p is the pressure; μ is the fluid viscosity, assumed to be constant; and $\mathbf{F} = (F_1, F_2)$ is the interfacial force, supported entirely along the interface Γ . We assume that at each time Γ is a closed curve in the computational domain. The fluid density is set to 1. Free space boundary conditions are assumed unless stated otherwise.

The force \mathbf{F} is the elastic tension force that arises from the stretching of Γ . The stretching is naturally expressed in terms of a material coordinate α on Γ , chosen to be arclength in the relaxed state. At time t the material point with label α has current position $\mathbf{X}(\alpha, t)$. We denote the arclength on Γ at the current time t by s and write the force $\mathbf{F} = (F_1, F_2)$ as

$$F_i(\mathbf{x}, t) = \int_0^L f_i(s, t) \delta(\mathbf{x} - \mathbf{X}(\alpha(s), t)) ds, \quad i = 1, 2. \quad (3)$$

where f_i is the force strength at point s and δ is the two-dimensional delta function.

The tension force \mathbf{f} , as derived from force balance arguments, is given by

$$\mathbf{f}(s, t) = \frac{\partial}{\partial s} (\gamma(s, t) \mathbf{t}(s, t)), \quad (4)$$

where we assume the tension $\gamma(s, t)$ is given by

$$\gamma(s, t) = T_0 \left(\left| \frac{\partial \mathbf{X}}{\partial \alpha} \right| - 1 \right) \quad (5)$$

when the material is stretched. The tension coefficient T_0 depends on the elastic properties of the interface and is assumed to be a constant in this model. The unit tangent vector $\mathbf{t}(s, t)$ to Γ is

$$\mathbf{t}(s, t) = \frac{\partial \mathbf{X}}{\partial s} = \frac{\partial \mathbf{X} / \partial \alpha}{|\partial \mathbf{X} / \partial \alpha|}. \quad (6)$$

Thus the force density can be computed directly from the location $\mathbf{X}(\alpha, t)$ of the boundary Γ . Note that in the relaxed state $|\partial \mathbf{X} / \partial \alpha| = 1$, and the tension vanishes.

2. Numerical Method.

2.1. A hybrid approach. Owing to the presence of a singular force \mathbf{f} , the solution of (1) and (2) is not smooth across Γ , with p and the normal derivatives of \mathbf{u} having jump discontinuities across the boundary. The jump discontinuity associated with a quantity q is denoted by

$$[q(\mathbf{X}, t)] = \lim_{\epsilon \rightarrow 0^+} q(\mathbf{X} + \epsilon \mathbf{n}, t) - \lim_{\epsilon \rightarrow 0^+} q(\mathbf{X} - \epsilon \mathbf{n}, t). \quad (7)$$

where \mathbf{n} is the vector outward normal to Γ . With this notation, the jump conditions for p and \mathbf{u} are given by [15, 16, 26]

$$[p] = \mathbf{f} \cdot \mathbf{n}, \quad \left[\frac{\partial p}{\partial n} \right] = \frac{\partial}{\partial s} (\mathbf{f} \cdot \mathbf{t}), \quad (8)$$

$$[\mathbf{u}] = 0, \quad \mu \left[\frac{\partial \mathbf{u}}{\partial \mathbf{n}} \right] = -(\mathbf{f} \cdot \mathbf{t}) \mathbf{t}. \quad (9)$$

The no-slip condition is assumed along Γ ; i.e., the boundary moves with the viscous fluid. Thus, \mathbf{u} is continuous across Γ . However, the normal derivatives of the velocity components have jump discontinuities.

These discontinuities tend to introduce substantial inaccuracy into the computed solution obtained by means of a standard finite difference method. To compute second-order accurate solutions while preserving their sharp jumps, the immersed interface method incorporate the jump conditions (8)–(9) into the finite difference stencil [15]. Below we introduce an alternative second-order approach which combines the boundary integral representation of the Stokes solution with a mesh-based solver.

The solutions to the Stokes equations in free space (1)–(2) can be written as boundary integrals

$$p(\mathbf{x}) = \int_{\Gamma} \nabla G(\mathbf{x} - \mathbf{y}) \cdot \mathbf{f}(\mathbf{y}) ds(\mathbf{y}), \quad \mathbf{u}(\mathbf{x}) = \int_{\Gamma} V(\mathbf{x} - \mathbf{y}) \mathbf{f}(\mathbf{y}) ds(\mathbf{y}) \quad (10)$$

where

$$\nabla G(\mathbf{x} - \mathbf{y}) = \frac{\mathbf{x} - \mathbf{y}}{2\pi|\mathbf{x} - \mathbf{y}|^2}, \quad (11)$$

$$V_{ij}(\mathbf{x}) = -\frac{\delta_{ij}}{4\pi\mu} \log|\mathbf{x}| + \frac{\mathbf{x}_i \mathbf{x}_j}{4\pi\mu|\mathbf{x}|^2}, \quad i, j = 1, 2 \quad (12)$$

The first term in the velocity integral has a logarithmic singularity; the second term has a smooth integrand on the interface but not at neighboring points. In free space the motion of the interface can be found by computing only the velocity on the interface. However, the velocity field in the domain may be desired, and for flow with boundary conditions it will be useful to compute the free space velocity at grid points (see below). For points near the interface, the integrals for the pressure and velocity are nearly singular, and routine quadrature would lead to large errors. We compute these integrals using a method developed in [3] and [8]. As in [8], Sec. 4, we rewrite the integrals in (10) as integrals with normal or tangential components of ∇G , a type treated in [3]. We replace ∇G with a smooth version ∇G_{δ} , using a Gaussian regularization and discretize with the trapezoidal rule. Thus the exact fundamental solution, or Stokeslet, is replaced with a regularized Stokeslet. The resulting sum has errors due to the regularization and discretization. We add corrections for the regularization derived in [3], equations (1.9) and (4.15). We choose the radius of smoothing δ to be sufficiently large so that discretization corrections are not needed. The resulting value is essentially $O(h^3)$ accurate, where h denotes the spatial grid subinterval, even for points near the interface; see [3]. For points away from the interface, the integrand is smooth, and no regularization is needed.

The method of regularized Stokeslets described above can be used to evaluate the free space velocity and pressure at any point. However, if values are needed at grid points over the entire computational domain, this method would be too

expensive. Motivated by this observation, we propose a hybrid approach for accurately evaluating Stokes solutions everywhere by combining the integral method with a mesh-based method. Briefly, we use regularized integrals to compute the solution at grid points near the interface; we then form the discrete Laplacian at grid points whose stencil crosses the interface; we use accurate values for the Laplacian away from the interface (zero in the case of the pressure); and finally we invert the discrete Laplacian to solve the entire grid problem. This approach was suggested in [19] and used in [3] in combination with the technique for nearly singular integrals. The process of forming a discrete Laplacian and inverting was used in [1, 18, 19] and has been widely used since with the immersed interface method, e.g. [15], with the discrete operator found from jump conditions rather than integrals. With $\mathcal{O}(h^3)$ values of the integrals, the discrete Laplacian at the interface is only $\mathcal{O}(h^{-2}h^3) = \mathcal{O}(h)$; nonetheless the solution of the Poisson problem is uniformly $\mathcal{O}(h^2)$ accurate. This gain of accuracy was explained analytically in [4]; see Theorem 2.1 and the discussion of nearly singular integrals in Sec. 4.

We now describe the numerical method in more detail. Let $\Delta t > 0$ denote the time step and $t_n \equiv n\Delta t$ be the n th time-level for $n = 0, 1, \dots$. Let $x_i \equiv ih$ and $y_j \equiv jh$ be the i th and j th spatial grid points along the x - and y -axis, respectively, where $i, j = 0, 1, 2, \dots, N$. For an arbitrary function $\psi(x, y, t)$, we denote $\psi(x_i, y_j, t_n)$ by $\psi_{i,j}^n$. The position of the membrane at t_n is represented by a set of boundary markers $\mathbf{X} = (X_k^n, Y_k^n)$, for $k = 0, 1, 2, \dots, N_B$, where $(X_0^n, Y_0^n) = (X_{N_B}^n, Y_{N_B}^n)$ because the membrane is assumed to be a simple closed curve. We assume that N and N_B are of the same order. If α denotes the arclength in the unstretched or related state, the boundary markers are chosen so that the k th marker approximates $(X(\alpha_k, t_n), Y(\alpha_k, t_n))$, where $\alpha_k = kL_0/N_B$; that is, the markers would be equally spaced in the relaxed state, but will not be equally spaced in a typical moving state.

To solve the Stokes equations, we reduce (1)–(2) to a sequence of Poisson problems, first for pressure, then for velocity, as in [15],

$$\Delta p = \nabla \cdot \mathbf{F}, \quad \mu \Delta \mathbf{u} = \nabla p, \quad (13)$$

The Poisson problems are solved on a rectangular computational domain Ω . For the free space problem, Dirichlet conditions are specified along $\partial\Omega$. The boundary values for p and \mathbf{u} along $\partial\Omega$ are computed using (10). No corrections are needed, since the interface is away from $\partial\Omega$. We assume the problem is in free space for now; below we explain how other boundary conditions can be imposed.

In the hybrid method, to solve for the pressure we first label the *irregular* grid points, i.e. the grid points for which the five-point stencil of the discrete Laplacian crosses the interface (see closed circles in Fig. 1). We then evaluate the free-space p as an integral, using regularized Stokeslets as above, at all grid points needed to form the stencil at the irregular points. Thus p is computed as a boundary integral at irregular grid points as well as their four nearest neighbors. We then form the discrete Laplacian at the irregular grid points,

$$\Delta^h p = (p_{i-1,j} + p_{i,j-1} - 4p_{i,j} + p_{i+1,j} + p_{i,j+1})/h^2 \quad (14)$$

As in [19] we set $\Delta^h p$ to zero at the *regular* points where the stencil does not cross the interface, with $\mathcal{O}(h^2)$ accuracy as usual. Having values of $\Delta^h p$ at all grid points, we then solve the linear system for grid values of p , with boundary conditions imposed from (10), using fast Fourier transforms (FFT). The resulting p is uniformly $\mathcal{O}(h^2)$ accurate; see [4], Sec. 4.

Next we compute the velocity \mathbf{u} . We first approximate ∇p at the regular points by the second-order centered difference $\nabla_0^h p$ and set $\Delta^h \mathbf{u} = \nabla_0^h p / \mu$. As above, we compute \mathbf{u} from the integral (10) near the interface and form the discrete Laplacian $\Delta^h \mathbf{u}$ at the irregular grid points. With $\Delta^h \mathbf{u}$ prescribed, we then solve the Poisson equation by FFT.

Note that the method of Stokeslets is used to evaluate $\mathcal{O}(N)$ solution values, thus incurring a computational cost of $\mathcal{O}(N^2)$. Applying FFT to the two-dimensional problem has a cost of $\mathcal{O}(N^2) \log N$.

Now we describe the modifications needed to solve the problem with a boundary condition other than free space. To be specific we treat the periodic case. Suppose $\mathbf{u}^{\text{free}}, p^{\text{free}}$ and $\mathbf{u}^{\text{per}}, p^{\text{per}}$ are the free-space and periodic solutions of (1) and (2), respectively, where \mathbf{f} is the singular force on the interface. Then $\Delta(p^{\text{per}} - p^{\text{free}}) = 0$. Therefore $p^{\text{per}} - p^{\text{free}}$ is smooth across the interface, so that its discrete Laplacian is $\mathcal{O}(h^2)$ accurate, and thus $\Delta^h p^{\text{per}} = \Delta^h p^{\text{free}} + \mathcal{O}(h^2)$. To solve for the pressure in the periodic case, we form $\Delta^h p^{\text{free}}$ at the irregular points as before. We set $\Delta^h p^{\text{per}} = \Delta^h p^{\text{free}}$ at these points and zero otherwise. We solve with the periodic boundary condition. The result gives p^{per} to $\mathcal{O}(h^2)$ except for the constant term.

To calculate the periodic velocity \mathbf{u}^{per} , we first find $p^{\text{diff}} = p^{\text{per}} - p^{\text{free}}$ by solving the Poisson problem $\Delta^h p^{\text{diff}} = 0$ with known boundary conditions. Next we set $\mu \Delta^h \mathbf{u}^{\text{per}} = \nabla_0^h p^{\text{per}}$ at the regular grid points. As before, we calculate \mathbf{u}^{free} near the interface using integrals and then form $\Delta^h \mathbf{u}^{\text{free}}$ at the irregular grid points. For the irregular points we set

$$\mu \Delta^h \mathbf{u}^{\text{per}} = \mu \Delta^h \mathbf{u}^{\text{free}} + \nabla_0^h (p^{\text{per}} - p^{\text{free}}) \quad (15)$$

This expression is $\mathcal{O}(h^2)$ accurate, since again the singular force term is subtracted out. We can now solve the Poisson problem to obtain \mathbf{u}^{per} . Since we also need \mathbf{u}^{per} on the interface, our final step is to solve for the smooth difference $\mathbf{u}^{\text{diff}} = \mathbf{u}^{\text{per}} - \mathbf{u}^{\text{free}}$ from the equation $\mu \Delta^h \mathbf{u}^{\text{diff}} = \nabla_0^h p^{\text{diff}}$ with imposed boundary condition. We can interpolate \mathbf{u}^{diff} to the interface, and then set $\mathbf{u}^{\text{per}} = \mathbf{u}^{\text{free}} + \mathbf{u}^{\text{diff}}$ on the interface, computing \mathbf{u}^{free} from the integral (10).

2.2. Resolving boundary layers. The hybrid approach can be used to improve solution accuracy when the boundary force is stiff. A stiff boundary frequently introduces steep gradients in pressure and velocity near the boundary, generating boundary layers and reducing the accuracy of a mesh-based discretization method. A more accurate approximation can be obtained, in part, by refining the discretization of the boundary in the integral computations, but not of the underlying fluid grid. A more refined boundary discretization should improve the accuracy of the modified Stokeslet approximations.

Additionally, one observes that the accuracy of the discretization of the immersed boundary is relatively insensitive to the steep solution gradient near the boundary. Thus, we may widen the “band” near the Γ where we compute boundary integral solutions. Technically speaking, to attain second-order accuracy only values at the irregular grid points and their immediate neighbors are needed. But in anticipation of a boundary layer, and to improve accuracy of the approximations, we may use modified Stokeslets to compute fluid variables along a band of $\sim 2mh$ ($m > 0$) in width. In the unrefined case, $m = 1$. As an example, the additional grid points in this thicker band are marked by open circles in Fig. 1. Within the band, where solution gradients may be steep, solution values are computed by boundary integrals

instead of a mesh-based discretization method; thus the overall accuracy should be improved.

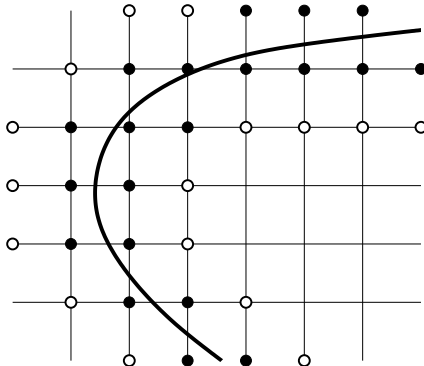


FIGURE 1. A figure that illustrates a widened “band” along a segment of the immersed boundary (thick curve), marked by closed and open circles, where boundary integral solutions are computed. Closed circles denote irregular grid points. Open circles denote additional grid-points at which boundary integral solutions are computed. In this example the boundary-to-grid ratio is $m = 2$.

Suppose the boundary discretization is refined by the same factor m , then the computational costs associated with the integral computations become $\mathcal{O}(m^2 N^2)$, an increase of a factor of m^2 . The cost of computing the solution everywhere using finite differences is unchanged.

This approach has the potential of improving accuracy of the solutions when boundary layers are expected. And because a refined spatial resolution is needed only along the immersed boundary, not in the underlying fluid grid, the method is much easier to implement than AMR, even for an adaptive version where the boundary-to-grid ratio m is allowed to change dynamically.

2.3. An implicit time-stepping method. We derive a partially implicit method of first order and then modify it obtain a second-order version of BDF type. (In the numerical results we refer to these as IM1 and IM2.) The singular part of the velocity is approximated in a way similar to that in [13, 12, 27, 31], leading to an approximation to an implicit step in the high wave numbers, as in the small scale decomposition approach of [10]. The second-order BDF version is analogous to (29) in [29].

For the steady Stokes equations (1,2) with force \mathbf{F} given by a density \mathbf{f} as in (3), the velocity \mathbf{u} is the integral (10) on the interface Γ with the Stokeslet V given by (12). We write the closed curve Γ as $\mathbf{X}(\alpha, t)$, where α is the material coordinate, $-A \leq \alpha \leq A$, so that $\mathbf{X}_t = \mathbf{u}(\mathbf{X}, t)$ ($\partial/\partial t$ with α fixed). We assume that the tension force density \mathbf{f} is given by (4, 5). It will be important that \mathbf{f} has the form $\partial_s \Phi$.

Suppose the curve Γ^n and the velocity \mathbf{u}^n are known at time t^n , and we want to find Γ^{n+1} . With Γ^n given as $\mathbf{X}^n(\alpha) = \mathbf{X}(\alpha, t^n)$, the force $\mathbf{f}^n(\alpha)$ is determined from \mathbf{X}^n from (4, 5), and \mathbf{u}^n is given by (10). We seek an approximation $\mathbf{X}^*(\alpha)$ to \mathbf{X}^{n+1} , the new curve. With time step $\Delta t = \tau$, set $\mathbf{X}^* = \mathbf{X}^n + \tau \mathbf{u}^*$, where we expect $\mathbf{u}^* \approx \mathbf{u}^{n+1}$ as in a backward Euler step. We will determine \mathbf{u}^* from an approximation to \mathbf{f}^* , leading to an equation to solve for \mathbf{u}^* .

We start with the velocity in integral form, with \mathbf{u} and \mathbf{f} replaced by \mathbf{u}^* and \mathbf{f}^* , at time $n + 1$. However, we integrate on Γ^n , the old curve; see [23] for discussion of this point. Then at time $n + 1$ we write the velocity integral in (10) as

$$\mathbf{u}^* = \mathbf{u}^n + (\mathbf{u}^* - \mathbf{u}^n) \approx \mathbf{u}^n + \int_{\Gamma^n} V(\mathbf{x} - \mathbf{y}) (\mathbf{f}^*(\mathbf{y}) - \mathbf{f}^n(\mathbf{y})) ds(\mathbf{y}). \quad (16)$$

We write $\mathbf{f}^n = \partial_s \Phi^n$ where $\Phi^n = T_0(s_\alpha - 1)\mathbf{t}$ at time n , and similarly $\mathbf{f}^* = \partial_s \Phi^*$. We will approximate the last integral, keeping only the log term in \mathbf{u} since the other part has smooth kernel and thus is a smoothing operator,

$$\mathbf{u}^*(\mathbf{x}) \approx \mathbf{u}^n(\mathbf{x}) - \frac{1}{4\pi\mu} \int_{\Gamma^n} \log|\mathbf{y} - \mathbf{x}| \partial_s (\Phi^*(\mathbf{y}) - \Phi^n(\mathbf{y})) ds. \quad (17)$$

Here $\mathbf{x} = \mathbf{X}(\alpha)$ for some α and $\mathbf{y} = \mathbf{X}(\alpha')$. (We will omit the superscript n on \mathbf{X} .) We change the variable of integration to α' . Canceling factors of $s_{\alpha'}$ from ds and ∂_s , we have, with $\mathbf{y} = \mathbf{X}(\alpha')$,

$$\mathbf{u}^*(\mathbf{x}) \approx \mathbf{u}^n(\mathbf{x}) - \frac{1}{4\pi\mu} \int_{-A}^A \log|\mathbf{y} - \mathbf{x}| \partial_{\alpha'} (\Phi^*(\mathbf{y}) - \Phi^n(\mathbf{y})) d\alpha'. \quad (18)$$

We now integrate by parts in α' , obtaining

$$\mathbf{u}^*(\mathbf{x}) \approx \mathbf{u}^n(\mathbf{x}) + \frac{1}{4\pi\mu} \int_{-A}^A \frac{\partial_{\alpha'} \mathbf{X}(\alpha') \cdot (\mathbf{y} - \mathbf{x})}{|\mathbf{y} - \mathbf{x}|^2} (\Phi^*(\mathbf{y}) - \Phi^n(\mathbf{y})) d\alpha' \quad (19)$$

Next we approximate the kernel near the singularity. For α' near α , $\mathbf{y} = \mathbf{X}(\alpha') \approx \mathbf{x} + \partial_{\alpha'} \mathbf{X}(\alpha')$, and the fraction in (19) simplifies to $1/(\alpha' - \alpha)$. Since we are interested only in the singularity, we insert a cut-off function $\psi(\alpha - \alpha')$ in the integral, with ψ chosen so that $\psi(\xi) = 1$ for ξ near 0 and $\xi = 0$ for $|\xi| > A/2$, and extend $\mathbf{X}(\alpha')$ periodically. We have now replaced (18) with

$$\mathbf{u}^*(\mathbf{x}) - \mathbf{u}^n(\mathbf{x}) \approx -\frac{1}{4\pi\mu} \int_{-\infty}^{\infty} \frac{\psi(\alpha - \alpha')}{\alpha - \alpha'} (\Phi^*(\mathbf{X}(\alpha')) - \Phi^n(\mathbf{X}(\alpha'))) d\alpha'. \quad (20)$$

Except for the cut-off function and a constant factor, this is a Hilbert transform. We write (20) briefly as

$$\mathbf{u}^* - \mathbf{u}^n \approx -\frac{1}{4\mu} H^{(0)} (\Phi^* - \Phi^n) \quad (21)$$

where $H^{(0)}$ is the modified Hilbert transform

$$(H^{(0)}g)(\alpha) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{\psi(\alpha - \alpha')}{\alpha - \alpha'} g(\alpha') d\alpha'. \quad (22)$$

Next we need to approximate $\Phi^* - \Phi^n$. Let $\delta\mathbf{X}(\alpha) = \mathbf{X}^*(\alpha) - \mathbf{X}^n(\alpha)$, representing the change in \mathbf{X} from time n to $n + 1$, and similarly for other quantities. We have $\Phi = T_0(s_\alpha - 1)\mathbf{t} = T_0(\mathbf{X}_\alpha - \mathbf{t})$. The first term in $\delta\Phi$ is $T_0\delta\mathbf{X}_\alpha = T_0\partial_\alpha(\delta\mathbf{X})$. For the second term we calculate the linear change in the unit tangent \mathbf{t} , e.g. as in [2], p.1284,

$$\delta\mathbf{t} = (s_\alpha^{-1}\delta X_\alpha^N - \kappa\delta X^T) \mathbf{n} \quad (23)$$

where superscript N or T means normal or tangential component, \mathbf{n} is the unit normal, and κ is curvature. We will neglect the second term in (23) as less important, since the first has a derivative of the unknown $\delta\mathbf{X}$ whereas the second does not; of course if the curvature is large we might want to include that term. Then

$$\delta\Phi \approx T_0 (\delta\mathbf{X}_\alpha - s_\alpha^{-1}\delta X_\alpha^N \mathbf{n}). \quad (24)$$

It appears the first term in (24) dominates the second, provided $s_\alpha \geq 1$, and for simplicity we discard the second term. (A version of the second term could be retained as in [12], Sec. 5.2.) Putting the simplified approximation to $\delta\Phi$ into (20), we get

$$\mathbf{u}^* - \mathbf{u}^n \approx -\frac{T_0}{4\mu} H^{(0)} (\partial_\alpha \delta\mathbf{X}). \quad (25)$$

Recalling that $\delta\mathbf{X} = \mathbf{X}^* - \mathbf{X}^n = \tau\mathbf{u}^*$, we rewrite (25) as

$$\mathbf{u}^* \approx \mathbf{u}^n - \frac{\tau T_0}{4\mu} H^{(0)} (\partial_\alpha \mathbf{u}^*). \quad (26)$$

Introducing the operator $\Lambda = H^{(0)} \partial_\alpha$, we finally obtain an equation to solve for \mathbf{u}^* ,

$$\mathbf{u}^* + \frac{\tau T_0}{4\mu} \Lambda(\mathbf{u}^*) = \mathbf{u}^n. \quad (27)$$

We now explain how (27) is solved in the discrete Fourier transform. Suppose a function $g(\alpha)$ is given with $-A \leq \alpha \leq A$, $\alpha_j = jh$ with $h = 2A/N$, N even. The discrete transform and its inverse are

$$g(\alpha) = \sum_k g_k e^{i\pi k\alpha/A}, \quad g_k = \frac{1}{N} \sum_j g(\alpha_j) e^{-i\pi k\alpha_j/A}. \quad (28)$$

Here $-N/2 < j \leq N/2$ and similarly for k . Substitute the series for g into (22) to get

$$(\Lambda g)(\alpha) = (H^{(0)} \partial_\alpha g)(\alpha) = \frac{1}{\pi} \sum_k \frac{i\pi k}{A} g_k \int_{-\infty}^{\infty} \frac{\psi(\alpha - \alpha')}{\alpha - \alpha'} e^{i\pi k\alpha'/A} d\alpha'. \quad (29)$$

Since we are interested only in the behavior for large k , we now drop the ψ -factor; the difference would contribute terms rapidly decreasing in k . Thus we have

$$(\Lambda g)(\alpha) = \frac{1}{\pi} \sum_k \frac{i\pi k}{A} g_k e^{i\pi k\alpha/A} \int_{-\infty}^{\infty} \frac{1}{\alpha - \alpha'} e^{i\pi k(\alpha' - \alpha)/A} d\alpha' \quad (30)$$

or, omitting the odd part of the integrand,

$$(\Lambda g)(\alpha) = -\frac{1}{\pi} \sum_k \frac{i\pi k}{A} g_k e^{i\pi k\alpha/A} \int_{-\infty}^{\infty} \frac{i}{\xi} \sin(\pi k\xi/A) d\xi. \quad (31)$$

The last integral, meant as a Fourier transform, is $i\pi \operatorname{sgn}(k)$. Thus

$$(\Lambda g)(\alpha) = \sum_k \frac{\pi|k|}{A} g_k e^{i\pi k\alpha/A} \quad (32)$$

and Λ acts in the discrete transform as multiplication by the wavenumber $\pi|k|/A$. Thus the solution of (27) in the transform is

$$\mathbf{u}_k^* = \left(1 + \frac{\tau T_0}{4\mu} \frac{\pi|k|}{A}\right)^{-1} \mathbf{u}_k^n \quad (33)$$

and our first-order time step is

$$\mathbf{X}^{n+1} = \mathbf{X}^n + \tau\mathbf{u}^*. \quad (34)$$

Next we use the above analysis to derive a second-order partially implicit method of BDF2 type. We can interpret (25) as defining a linear operator \mathcal{A} which approximates $\delta\mathbf{u}$ in terms of $\delta\mathbf{X}$:

$$\delta\mathbf{u} \approx -\frac{T_0}{4\mu} \Lambda(\delta\mathbf{X}) \equiv \mathcal{A}(\delta\mathbf{X}). \quad (35)$$

We expect \mathcal{A} to contain the stiff part of $\delta\mathbf{u}$, the variational or Frechet derivative of \mathbf{u} with respect to \mathbf{X} . We begin with the exact BDF2 version of the ordinary differential equation $d\mathbf{X}/dt = \mathbf{u}(\mathbf{X})$, again with $\Delta t = \tau$,

$$\frac{3}{2}\mathbf{X}^{n+1} - 2\mathbf{X}^n + \frac{1}{2}\mathbf{X}^{n-1} = \tau\mathbf{u}^{n+1}. \quad (36)$$

To approximate $\mathbf{u}^{n+1} = \mathbf{u}^n + (\mathbf{u}^{n+1} - \mathbf{u}^n)$, we replace $\mathbf{u}^{n+1} - \mathbf{u}^n$ by $\mathcal{A}(\mathbf{X}^{n+1} - \mathbf{X}^n)$ plus the remainder to get

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \mathcal{A}(\mathbf{X}^{n+1} - \mathbf{X}^n) + (\mathbf{u}^{n+1} - \mathcal{A}\mathbf{X}^{n+1}) - (\mathbf{u}^n - \mathcal{A}\mathbf{X}^n) \quad (37)$$

or after simplifying

$$\mathbf{u}^{n+1} = \mathcal{A}\mathbf{X}^{n+1} + (\mathbf{u}^{n+1} - \mathcal{A}\mathbf{X}^{n+1}). \quad (38)$$

In this exact expression we use an $O(\tau^2)$ explicit approximation for the remainder,

$$\mathbf{u}^{n+1} \approx \mathcal{A}\mathbf{X}^{n+1} + 2(\mathbf{u}^n - \mathcal{A}\mathbf{X}^n) - (\mathbf{u}^{n-1} - \mathcal{A}\mathbf{X}^{n-1}). \quad (39)$$

Putting this into (36) we get

$$\left(\frac{3}{2} - \tau\mathcal{A}\right)\mathbf{X}^{n+1} = 2(1 - \tau\mathcal{A})\mathbf{X}^n - \left(\frac{1}{2} - \tau\mathcal{A}\right)\mathbf{X}^{n-1} + \tau(2\mathbf{u}^n - \mathbf{u}^{n-1}). \quad (40)$$

Setting $\mathcal{R} = \left(\frac{3}{2} - \tau\mathcal{A}\right)^{-1}$, we obtain after some algebra

$$\mathbf{X}^{n+1} = 2\mathbf{X}^n - \mathbf{X}^{n-1} + \mathcal{R}(-\mathbf{X}^n + \mathbf{X}^{n-1} + \tau(2\mathbf{u}^n - \mathbf{u}^{n-1})). \quad (41)$$

This is our second-order time stepping method. The operator \mathcal{R} can be calculated in the transform as in (33), with 3/2 replacing 1.

3. Numerical Examples. In this section, the stability and accuracy of the methods described in Section 2 are studied. In all the examples, fluid viscosity μ was set to 1. We first assess the accuracy of the hybrid approach using examples for which analytic solutions have been derived. A principal goal is to demonstrate how the hybrid approach can be used to obtain highly accurate approximations with a relatively coarse spatial discretization. To that end, we consider examples in which the solutions exhibit steep gradients near the immersed interface. We then compute Stokes solution for moving boundary problems, using the hybrid approach and the implicit time-stepping method.

3.1. Example 1. The first example is a generalization of the example 4a in [8]. In this example, the immersed interface is a unit circle, parameterized by $\mathbf{x}(\theta) = (\cos(\theta), \sin(\theta))$. The boundary force has only a normal component, and is given by

$$\mathbf{f}(\theta) = 2 \sin(k\theta)\mathbf{x}(\theta) \quad (42)$$

The exact solution is given by

$$p(r, \theta) = \begin{cases} r^{-k} \sin(k\theta), & r \geq 1 \\ -r^k \sin(k\theta), & r < 1 \end{cases} \quad (43)$$

$$u(r, \theta) = \begin{cases} \frac{1}{8}r^{-(k-1)} \sin((k-1)\theta) \\ \quad - \frac{3}{16}r^{-(k+1)} \sin((k+1)\theta) + \frac{1}{4}r^{-(k-1)} \sin((k+1)\theta), & r \geq 1 \\ \frac{3}{8}r^{k-1} \sin((k-1)\theta) \\ \quad + \frac{1}{16}r^{k+1} \sin((k+1)\theta) - \frac{1}{4}r^{k+1} \sin((k-1)\theta), & r < 1 \end{cases} \quad (44)$$

$$v(r, \theta) = \begin{cases} \frac{1}{8}r^{-(k-1)} \cos((k-1)\theta) \\ \quad + \frac{1}{16}r^{-(k+1)} 3 \cos((k+1)\theta) - \frac{1}{4}r^{-(k-1)} \cos((k+1)\theta), & r \geq 1 \\ \frac{3}{8}r^{k-1} \cos((k-1)\theta) \\ \quad - \frac{1}{16}r^{k+1} \cos((k+1)\theta) - \frac{1}{4}r^{k+1} \cos((k-1)\theta), & r < 1 \end{cases} \quad (45)$$

Because the boundary force (42) in this example has only the normal component, u and v are \mathcal{C}^1 , whereas p has a jump discontinuity across the interface.

The computational domain Ω was chosen to be $[-2.9, 2.9] \times [-2.9, 2.9]$. We computed the solution for $k = 3$ and $k = 7$. Larger k values correspond to solutions with steeper gradients near the immersed unit circle. We compare the accuracy of the hybrid approach (see Section 2.1) and the immersed interface method [15]. In the hybrid approach, the integrals in (10) were approximated using the method described in Section 2.1, which provided the sufficiently high $O(h^3)$ accuracy when Γ is a unit circle. Along Γ , $N_B = 2N$ subintervals were used. Figure 2A shows normalized L_2 errors in p obtained for $k = 3$. These results show evidence of the expected $O(h^2)$ convergence. The hybrid approach and the immersed interface method yielded comparable accuracy. Similar results were obtained for u and v .

For $k = 7$, the computed solutions exhibit larger errors owing to the steeper gradients near Γ ; see Fig. 2B. To resolve that boundary layer, we recomputed the solution using the hybrid approach, with a more refined boundary discretization of $N_B = 4N$, and also with a wider band where boundary integral solutions are computed. Both approaches resulted in substantial reduction in solution error (see error curves labelled “Hybrid $_{4N,0}$ ” and “Hybrid $_{4N,2}$ ” in Fig. 2B).

3.2. Example 2. The second example is a generalization of the example 4b in [8]. The boundary force has only a tangential component, and is given by

$$\mathbf{f}(\theta) = 2 \sin(k\theta) \mathbf{x}'(\theta) \quad (46)$$

The exact solution is given by

$$p(r, \theta) = \begin{cases} -r^{-k} \cos(k\theta), & r \geq 1 \\ -r^k \cos(k\theta), & r < 1 \end{cases} \quad (47)$$

$$u(r, \theta) = \begin{cases} -\frac{1}{8}r^{-(k-1)} \cos((k-1)\theta) \\ \quad + \frac{5}{16}r^{-(k+1)} \cos((k+1)\theta) - \frac{1}{4}r^{-(k-1)} \cos((k+1)\theta), & r \geq 1 \\ \frac{1}{8}r^{k-1} \cos((k-1)\theta) \\ \quad - \frac{1}{16}r^{k+1} \cos((k+1)\theta) + \frac{1}{4}r^{k+1} \cos((k-1)\theta), & r < 1 \end{cases} \quad (48)$$

$$v(r, \theta) = \begin{cases} \frac{1}{8}r^{-(k-1)} \sin((k-1)\theta) \\ \quad + \frac{5}{16}r^{-(k+1)} \sin((k+1)\theta) - \frac{1}{4}r^{-(k-1)} \sin((k+1)\theta), & r \geq 1 \\ -\frac{1}{8}r^{k-1} \sin((k-1)\theta) \\ \quad + \frac{1}{16}r^{k+1} \sin((k+1)\theta) + \frac{1}{4}r^{k+1} \sin((k-1)\theta), & r < 1 \end{cases} \quad (49)$$

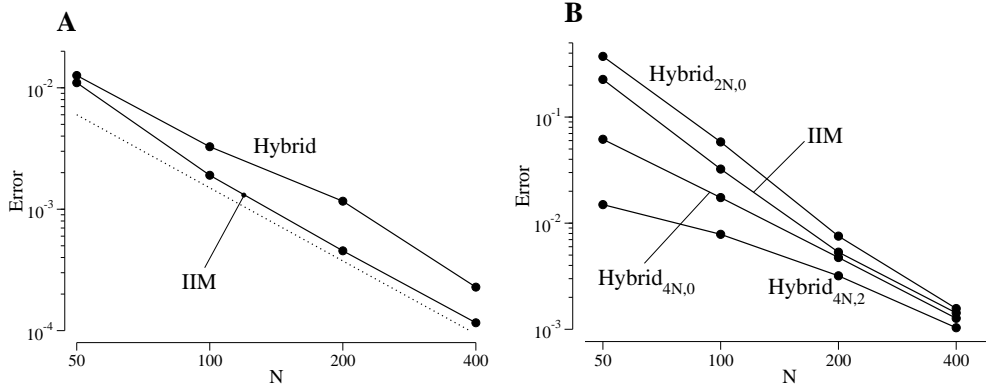


FIGURE 2. Normalized L_2 errors in p computed for Example 1. A, $k = 3$; B, $k = 7$. N denotes the number of subintervals in each dimension. For $k = 3$, $N_B = 2N$. Results show second-order convergence. For $k = 7$, the subscripts associated with ‘Hybrid’ denotes N_B and bandwidth m , respectively. Increasing N_B and m reduces solution errors. Dotted line in Panel A has slope -2 .

The boundary force (46) in this example has only the tangential component,. Thus, p is smooth, whereas the normal derivatives u and v have jump discontinuities across the interface.

We compare the accuracy of the hybrid and immersed interface methods. Results are obtained for $k = 3$ and $k = 7$. The computational domain Ω was chosen to be $[-2.9, 2.9] \times [-2.9, 2.9]$. For $k = 3$, both methods exhibit approximately $\mathcal{O}(h^2)$ convergence; see Fig. 3A, which displays L_2 errors in u . Results for $k = 7$ (Fig. 3B) suggest that the boundary layer can be resolved using the hybrid approach. Similar results were also obtained for p and v .

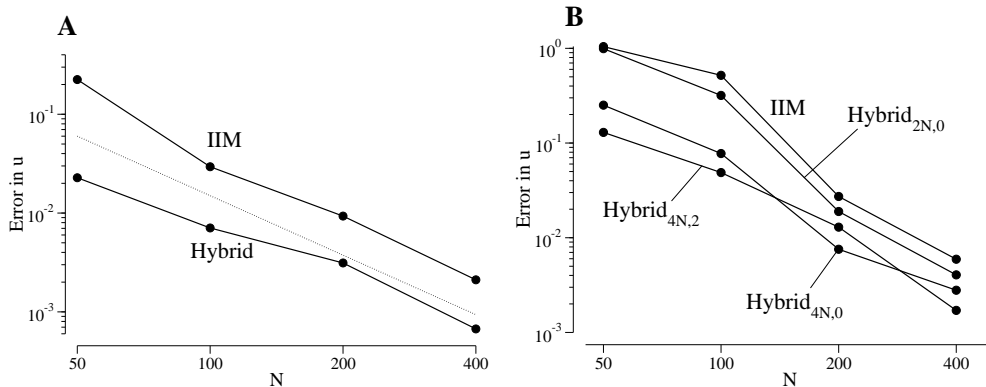


FIGURE 3. Normalized L_2 errors in u computed for Example 2. A, $k = 3$; B, $k = 7$. N denotes the number of subintervals in each dimension. Notations are analogous to Fig. 2.

3.3. Example 3. We then simulate the motion of a relaxing ellipse. This example is frequently used in testing numerical methods for immersed boundary problems

[15, 14, 30]. The interfacial force is given by Eqs. (3) and (4). The initial boundary is an ellipse with major and minor axes set to $a = 0.81$ and $b = 0.61$, respectively. The unstretched boundary was taken to be a circle with radius $r_0 = 0.5$. The computational domain was $[-1.1, 1.1] \times [-1.1, 1.1]$.

We first used this example to assess the convergence of the implicit time-stepping methods described in Section 2.3. The tension coefficient T_0 was set to 1. The boundary was discretized using $N_B = 320$ markers; the computational domain was discretized using $N = 320$ subintervals in each dimension. Equations (1)–(2) were integrated from $t = 0$ to $t = 10^4 h$, using large time-steps $\Delta t = 50h, 100h, 200h$, and $400h$. At the final time $t = 10^4 h$, the system is sufficiently far from equilibrium, that time-discretization errors remain significant. The errors in the final boundary position were estimated, using the boundary position obtained using IM2 with a small $\Delta t = 10h$ as a reference (denoted \mathbf{X}^*). Convergence results are displayed in Fig. 4. Solutions computed by means of IM1 and IM2 exhibit approximately first- and second-order convergence, respectively.

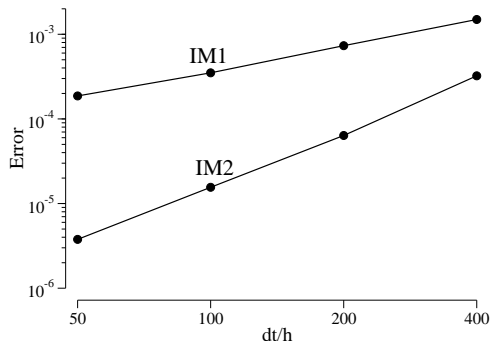


FIGURE 4. Final boundary position errors $|\mathbf{X} - \mathbf{X}^*|_1$, where \mathbf{X}^* denotes the reference boundary position, for the first-order (IM1) and second-order (IM2) implicit time-stepping methods, using $T_0 = 1$.

We then simulated the motion of a stiff boundary to assess the stability of two implicit time-stepping methods. Using a tension coefficient of $T_0 = 10^3$ and integrating to $t = 100h$, we found that both IM1 and IM2 were stable for $\Delta t = 100h$, and forward Euler and two-step Adam-Bashforth were both stable for $\Delta t = h/100$. (A time integration is considered unstable when the predicted boundary configuration sufficiently deviates from its expected course of approaching a circle. In practice, when an integration becomes unstable, model variables frequently become unrealistically large or undefined. Note also that T_0 could be reset to 1 by rescaling time.) When an even stiffer boundary with $T_0 = 10^8$ was used, both IM1 and IM2 were stable for $\Delta t = 10h$ for at least 100 time steps. These results suggest that while the partially implicit methods described in Section 2.3 are not unconditionally stable, they are substantially more stable than some popular explicit time-stepping method.

3.4. Example 4. In a fourth example, we simulate the motion of a stretched ellipse, with a background flow given by $(u_{bg}, v_{bg}) = (0.1 \sin y, 0)$. The background

flow is imposed by adding a term to the boundary force which produces the desired background flow velocity. The initial configuration of the ellipse and the computational domain were the same as in Example 3. Figure 5 shows the position of the boundary at $t = 10^3 h$ for tension coefficients $T_0 = 1$ and $T_0 = 10^5$, and also for the case without background flow for $T_0 = 1$.

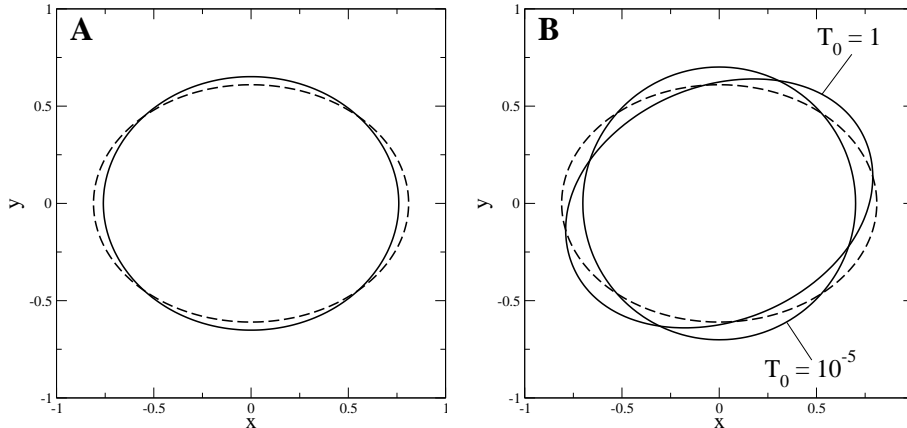


FIGURE 5. Boundary positions at $t = 10^3 h$. Panel A, zero background flow and $T_0 = 1$. Panel B, background flow $(u_{bg}, v_{bg}) = (0.1 \sin y, 0)$ and with $T_0 = 1$ and $T_0 = 10^5$. Dashed line, initial boundary position.

To assess the convergence of the implicit time-stepping methods, we conducted a simulation in which the tension coefficient T_0 was set to 1. The boundary and the computational domain were discretized using $N_B = 320$ markers and $N = 320$ subintervals in each dimension, respectively. Equations (1)–(2) were integrated from $t = 0$ to $t = 200h$, using time-steps of $\Delta t = h, 2h, 4h$, and $8h$. The errors in the final boundary position were estimated, using the boundary position obtained by means of IM2 using a small $\Delta t = 0.25h$ as a reference (denoted \mathbf{X}^*). Convergence results are displayed in Fig. 6. Solutions computed by means of IM1 and IM2 exhibit approximately first- and second-order convergence, respectively.

To assess the stability of implicit time-stepping methods, we simulated the motion of a stiff boundary with $T_0 = 10^5$. Equations (1)–(2) were integrated to $t = 10^3 h$. Our results indicate that IM1 and IM2 were stable for $\Delta t = 10^2 h$.

REFERENCES

- [1] C. R. Anderson. A method of local corrections for computing the velocity field due to a distribution of vortex blobs. *J. Comp. Phys.*, 62:111–123, 1986.
- [2] J. T. Beale, T. Y. Hou, and J. S. Lowengrub. Growth rates for the linearized motion of fluid interfaces away from equilibrium. *Comm. Pure Appl. Math.*, 46:1269–1301, 1993.
- [3] J. T. Beale and M.-C. Lai. A method for computing nearly singular integrals. *SIAM J. Numer. Anal.*, 38:1902–25, 2001.
- [4] J. T. Beale and A. T. Layton. On the accuracy of finite difference methods for elliptic problems with interfaces. *Comm. Appl. Math. Comput. Sci.*, 2006.
- [5] M. J. Berger and J. Olinger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comp. Phys.*, 53:484–512, 1984.

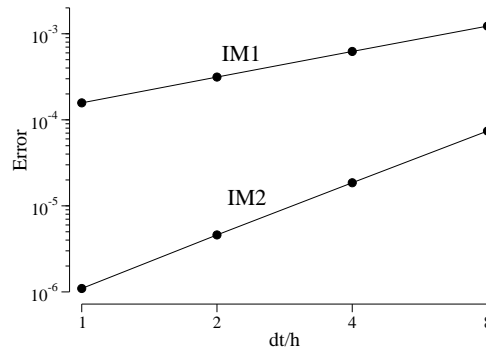


FIGURE 6. Final boundary position errors $|\mathbf{X} - \mathbf{X}^*|_1$ for the first-order (IM1) and second-order (IM2) implicit time-stepping methods, using $T_0 = 1$.

- [6] H. D. Ceniceros, J. E. Fisher, and A. M. Roma. Efficient solutions to robust, semi-implicit discretizations of the immersed boundary method. *J. Comput. Phys.*, 228:7137–7157, 2009.
- [7] N. G. Cogan, R. Cortez, and L. J. Fauci. Modeling physiological resistance in bacterial biofilms. *Bull. Math. Biol.*, 67:831–853, 2005.
- [8] R. Cortez. The method of regularized stokeslets. *SIAM J. Sci. Comput.*, 23:1204–1225, 2001.
- [9] L. J. Fauci and A. L. Fogelson. Truncated Newton methods and the modeling of complex immersed elastic structures. *Comm. Pure Appl. Math.*, 66:787–818, 1993.
- [10] T. Y. Hou, J. S. Lowengrub, and M. J. Shelley. Removing the stiffness from interfacial flows with surface tension. *J. Comput. Phys.*, 114:312–338, 1994.
- [11] T. Y. Hou and Z. Shi. An efficient semi-implicit immersed boundary method for the Navier-Stokes equations. *J. Comput. Phys.*, 227:8968–8991, 2008.
- [12] T. Y. Hou and Z. Shi. Removing the stiffness of elastic force from the immersed boundary method for the 2D Stokes equations. *J. Comput. Phys.*, 227:9138–9169, 2008.
- [13] M. C. A. Kropinski. An efficient numerical method for studying interfacial motion in two-dimensional creeping flows. *J. Comput. Phys.*, 171:479–508, 2001.
- [14] L. Lee and R. J. LeVeque. An immersed interface method for the incompressible Navier-Stokes equations. *SIAM J. Sci. Comp.*, 25:832–856, 2003.
- [15] R. J. LeVeque and Z. Li. Immersed interface methods for Stokes flow with elastic boundaries or surface tension. *SIAM J. Sci. Comput.*, 18(3):709–735, 1997.
- [16] Z. Li and M.-C. Lai. The immersed interface method for the Navier-Stokes equations with singular forces. *J. Comput. Phys.*, 171:822–842, 2001.
- [17] Z. Li and S. R. Lubkin. Numerical analysis of interfacial two-dimensional Stokes flow with discontinuous viscosity and variable surface tension. *Int. J. Numer. Meth. Fluids*, 37:525–540, 2001.
- [18] A. Mayo. The fast solution of Poisson’s and the biharmonic equations on irregular regions. *SIAM J. Numer. Anal.*, 21:285–299, 1984.
- [19] A. Mayo. Fast high order accurate solution of Laplace’s equation on irregular regions. *SIAM J. Sci. Statist. Comput.*, 6:144–157, 1985.
- [20] A. Mayo and C. S. Peskin. An implicit numerical method for fluid dynamics problems with immersed elastic boundaries. In A. Y. Cheer and C. P. von Dam, editors, *Fluid Dynamics in Biology*, pages 261–278, Providence, RI, 1993. AMS.
- [21] Y. Mori and C. S. Peskin. Implicit second order immersed boundary methods with boundary mass. *Comput. Methods Appl. Mech. Engin.*, 197(25–28):2049–2067, 2008.
- [22] E. Newren, A. Fogelson, R. Guy, and M. Kirby. A comparison of implicit solvers for the immersed boundary equations. *Comput. Methods Appl. Mech. Engin.*, 197(25–28):2290–2304, 2008.
- [23] E. P. Newren, A. L. Fogelson, R. D. Guy, and R. M. Kirby. Unconditionally stable discretizations of the immersed boundary equations. *J. Comput. Phys.*, 222:702–19, 2007.

- [24] C. S. Peskin. Numerical analysis of blood flow in the heart. *J. Comput. Phys.*, 25:220–252, 1977.
- [25] C. S. Peskin. The immersed boundary method. *Acta Numerica*, pages 1–39, 2002.
- [26] C. S. Peskin and B. F. Printz. Improved volume conservation in the computation of flows with immersed elastic boundaries. *J. Comput. Phys.*, 105:33–46, 1993.
- [27] J. S. Sohn, Y.-H. Tseng, S. Li, A. Voigt, and J. S. Lowengrub. Dynamics of multicomponent vesicles in a viscous fluid. *J. Comput. Phys.*, 229:119–144, 2010.
- [28] J. M. Stockie and B. R. Wetton. Analysis of stiffness in the immersed boundary method and implications for time-stepping schemes. *J. Comput. Phys.*, 154:41–64, 1999.
- [29] A.-K. Tornberg and M. J. Shelley. Simulating the dynamics and interactions of flexible fibers in Stokes flows. *J. Comput. Phys.*, 196:8–40, 2004.
- [30] 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.*, 13:1361–1376, 1992.
- [31] S. K. Veerapaneni, D. Gueyffier, D. Zorin, and G. Biros. A boundary integral method for simulating the dynamics of inextensible vesicles suspended in a viscous fluid in 2D. *J. Comput. Phys.*, 228:2334–2353, 2009.

Received xxxx 20xx; revised xxxx 20xx.

E-mail address: alayton@math.duke.edu

E-mail address: beale@math.duke.edu