Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Physics 222 (2007) 702–719

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

Unconditionally stable discretizations of the immersed boundary equations

Elijah P. Newren [a,*], Aaron L. Fogelson [a,b], Robert D. Guy [a], Robert M. Kirby [c]

[a] *Department of Mathematics, University of Utah, Salt Lake City, UT 84112, United States*
[b] *Department of Bioengineering, University of Utah, Salt Lake City, UT 84112, United States*
[c] *School of Computing, University of Utah, Salt Lake City, UT 84112, United States*

Received 28 March 2006; received in revised form 8 August 2006; accepted 16 August 2006
Available online 2 October 2006

Abstract

The immersed boundary (IB) method is known to require small timesteps to maintain stability when solved with an explicit or approximately implicit method. Many implicit methods have been proposed to try to mitigate this timestep restriction, but none are known to be unconditionally stable, and the observed instability of even some of the fully implicit methods is not well understood. In this paper, we prove that particular backward Euler and Crank–Nicolson-like discretizations of the nonlinear immersed boundary terms of the IB equations in conjunction with unsteady Stokes Flow can yield unconditionally stable methods. We also show that the position at which the spreading and interpolation operators are evaluated is not relevant to stability so as long as both operators are evaluated at the same location in time and space. We further demonstrate through computational tests that approximate projection methods (which do not provide a discretely divergence-free velocity field) appear to have a stabilizing influence for these problems; and that the implicit methods of this paper, when used with the full Navier–Stokes equations, are no longer subject to such a strict timestep restriction and can be run up to the CFL constraint of the advection terms.

MSC: 65M06; 65M12; 74F10; 76D05; 76D07; 76M20

Keywords: Stability; Fluid–structure interaction; Immersed boundary method; Immersed interface method; Projection methods; Navier–Stokes equations

1. Introduction

The immersed boundary (IB) method was introduced by Peskin in the early 1970's to solve the coupled equations of motion of a viscous, incompressible fluid and one or more massless, elastic surfaces or objects immersed in the fluid [20]. Rather than generating a surface-fitting grid for both exterior and interior regions

[*] Corresponding author. Tel.: +1 801 585 1641; fax: +1 801 585 1640.
E-mail addresses: newren@math.utah.edu (E.P. Newren), fogelson@math.utah.edu (A.L. Fogelson), guy@math.utah.edu (R.D. Guy), kirby@cs.utah.edu (R.M. Kirby).

0021-9991/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2006.08.004

of each surface at each timestep and using these to determine the fluid motion, Peskin instead employed a uniform Cartesian grid over the entire domain and discretized the immersed boundaries by a set of points that are *not* constrained to lie on the grid. The key idea that permits this simplified discretization is the replacement of each suspended object by a suitable contribution to a force density term in the fluid dynamics equations in order to allow a single set of fluid dynamics equations to hold in the entire domain with no internal boundary conditions.

The IB method was originally developed to model blood flow in the heart and through heart valves [20,22,23], but has since been used in a wide variety of other applications, particularly in biofluid dynamics problems where complex geometries and immersed elastic membranes or structures are present and make traditional computational approaches difficult. Examples include platelet aggregation in blood clotting [7,9], swimming of organisms [7,8], biofilm processes [6], mechanical properties of cells [1], cochlear dynamics [3], and insect flight [17,18].

The immersed interface method (IIM) was developed by Leveque and Li to address the lower order accuracy found in the IB method when applied to problems with sharp interfaces [14]. The IIM differs from the IB method in the spatial discretization method used to handle the singular forces appearing in the continuous equations of motion. While we do not address the spatial discretizations involved in the IIM and instead focus on the IB method in this paper, we do present some discussion of that method since the two are closely related (hybrids of the two even exist, such as [13]).

The immersed boundary and immersed interface methods suffer from a severe timestep restriction needed to maintain stability, as has been well documented in the literature [7,14,21,26]. This time step restriction is typically much more stringent than the one that would be imposed from using explicit differencing of the advective or diffusive terms [5]. Much effort has been expended attempting to alleviate this severe restriction. For example in some problems, the fluid viscosity has been artificially increased by a couple orders of magnitude [24]. In others, authors filter out high frequency oscillations of the interface [14,29]. Much effort has been put into developing implicit and semi-implicit methods [7,14,16,27].

Despite these efforts the severe timestep restriction has remained. The instability of these methods is known not to be a problem related to the advection terms in the Navier–Stokes equations, and is known to arise in the parameter regime corresponding to large boundary force and small viscosity [26], but there is little understanding of why this parameter regime is problematic. Despite the effort put into implicit methods which couple the equations of motion for the fluid and immersed boundary, the lack of stability has been puzzling. Conjectures as to causes of instability in these methods turn out to be misleading. It is a common belief in the community that using time-lagged spreading and interpolation operators (i.e. time-lagged discretizations of the delta functions in Eqs. (2) and (5) of Section 2) will cause instability, and that therefore a fully-implicit scheme (i.e. one without time-lagged spreading and interpolation operators) is necessary for unconditional stability [13,16,19,25]. It has also been conjectured that the lack of stability and corresponding timestep restriction in fully-implicit schemes such as [16] is due to accumulation of error in the incompressibility condition [26].

We will present some discretizations that we prove to be unconditionally stable in conjunction with unsteady Stokes flow, and in so doing, show (i) that methods need not be fully-implicit in order to achieve unconditional stability and (ii) that accumulation of error in the incompressibility condition is not the cause of the observed instability of previous implicit methods. We cover the continuous equations of motion and common choices for temporal discretizations in Sections 2 and 3, provide proofs of unconditional stability of various discretization schemes in Section 4, show by computational tests how the schemes are affected by the presence of an approximate projection and advection terms in Section 5, and discuss remaining outstanding questions about these methods in Section 6.

2. Continuous equations of motion

In the IB method, an Eulerian description based on the Navier–Stokes equations is used for the fluid dynamics, and a Lagrangian description is used for each object immersed in the fluid. The boundary is assumed to be massless, so that all of the force generated by distortions of the boundary is transmitted directly to the fluid. An example setup in 2D with a single immersed boundary curve is shown in Fig. 1. Lowercase

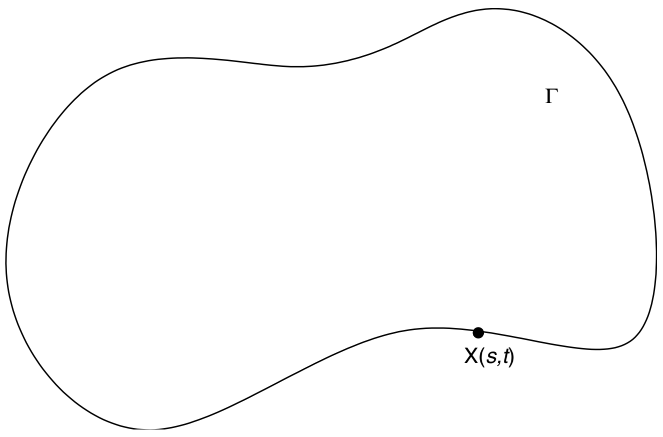

Fig. 1. Example immersed boundary curve, Γ, described by the function $\mathbf{X}(s,t)$.

letters are used for Eulerian state variables, while uppercase letters are used for Lagrangian variables. Thus, $\mathbf{X}(s,t)$ is a vector function giving the location of points on Γ as a function of arclength (in some reference configuration), s, and time, t. The boundary is modeled by a singular forcing term, which is incorporated into the forcing term, $\mathbf{f}$, in the Navier–Stokes equations. The Navier–Stokes equations are then solved to determine the fluid velocity throughout the domain, Ω. Since the immersed boundary is in contact with the surrounding fluid, its velocity must be consistent with the no-slip boundary condition. Thus the immersed boundary moves at the local fluid velocity. This results in the following set of equations:

$$\mathbf{F}(s,t) = \mathcal{A}\mathbf{X}(s,t), \tag{1}$$

$$\mathbf{f}(\mathbf{x},t) = \int_\Gamma \mathbf{F}(s,t)\delta(\mathbf{x}-\mathbf{X}(s,t))\,\mathrm{d}s \tag{2}$$

$$\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p + \nu\Delta\mathbf{u} + \mathbf{f}, \tag{3}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{4}$$

$$\frac{\mathrm{d}\mathbf{X}(s,t)}{\mathrm{d}t} = \mathbf{u}(\mathbf{X}(s,t),t) = \int_\Omega \mathbf{u}(\mathbf{x},t)\delta(\mathbf{x}-\mathbf{X}(s,t))\,\mathrm{d}\mathbf{x}. \tag{5}$$

The force generation operator, $\mathcal{A}$, in Eq. (1) is problem dependent. A commonly used operator is $\gamma\frac{\partial^2}{\partial s^2}$ (where γ is an elastic tension parameter), which results from assuming the boundary is linearly elastic with zero resting length. Note that the terminology "force generation operator" might be misleading to those not familiar with the immersed boundary method, as the force used in the fluid dynamics equations is not the one defined in Eq. (1) but the one in Eq. (2). The latter is nonlinear as well as singular due to the lower-dimensional line integral. The Lagrangian and Eulerian variables in the problem are related through Eqs. (2) and (5). In Eq. (3) we have assumed a constant density, ρ, of 1.0 g/cm^3 and divided it through both sides.

3. Temporal discretization

In departure from most papers on the immersed boundary and immersed interface methods, we first discretize the continuous equations of motion in time, yielding a spatially continuous, temporally discrete system. We do this because the choice of temporal discretization appears to be much more important to stability than the spatial discretization – we will show stability for a wide class of spatial discretizations but only for two types of temporal discretizations. We analyze unsteady Stokes flow because as mentioned above, the advection terms of the Navier–Stokes equations are not the cause of instability in these methods. We will, however, include the advection terms in some computational tests in Section 5.

There are various temporal discretizations possible. Most immersed boundary and immersed interface implementations tend to be of the following form:

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}+\nabla p^{n+\frac{1}{2}}=\frac{\nu}{2}\Delta(\mathbf{u}^{n+1}+\mathbf{u}^n)+\mathbf{f}, \tag{6}$$

$$\nabla\cdot\mathbf{u}^{n+1}=0, \tag{7}$$

$$\frac{\mathbf{X}^{n+1}-\mathbf{X}^n}{\Delta t}=\mathbf{U}, \tag{8}$$

where different discretization possibilities for **f** and **U** will be discussed below. Let us define $\mathcal{S}_m$ and $\mathcal{S}_m^*$ through the formulas

$$\mathcal{S}_m(\mathbf{F})=\int_\Gamma \mathbf{F}(s,t)\delta(\mathbf{x}-\mathbf{X}^m(s,t))\,\mathrm{d}s, \tag{9}$$

and

$$\mathcal{S}_m^*(\mathbf{u})=\int_\Omega \mathbf{u}(\mathbf{x},t)\delta(\mathbf{x}-\mathbf{X}^m(s,t))\,\mathrm{d}\mathbf{x}. \tag{10}$$

The most common temporal discretizations of **f** and **U** are $\mathbf{f}=\mathcal{S}_n\mathcal{A}\mathbf{X}^n$ and $\mathbf{U}=\mathcal{S}_n^*\mathbf{u}^{n+1}$. This results in an explicit system (more precisely, a mixed explicit–implicit system that is implicit in the handling of the viscous terms and explicit in the handling of the immersed boundary terms). But this common explicit discretization requires a small timestep for stability. In an effort to devise more stable schemes, various implicit methods have been presented in the literature. Among these implicit methods, common discretization choices for **f** are

(1) $\mathcal{S}_{n+1}\mathcal{A}\mathbf{X}^{n+1}$,
(2) $\frac{1}{2}\mathcal{S}_n\mathcal{A}\mathbf{X}^n+\frac{1}{2}\mathcal{S}_{n+1}\mathcal{A}\mathbf{X}^{n+1}$,
(3) $\mathcal{S}_{n+\frac{1}{2}}\mathcal{A}\mathbf{X}^{n+\frac{1}{2}}$,

where $\mathbf{X}^{n+\frac{1}{2}}$ is approximated by $\frac{1}{2}(\mathbf{X}^n+\mathbf{X}^{n+1})$, while common discretization choices for **U** are

(A) $\mathcal{S}_{n+1}^*\mathbf{u}^{n+1}$,
(B) $\frac{1}{2}\mathcal{S}_n^*\mathbf{u}^n+\frac{1}{2}\mathcal{S}_{n+1}^*\mathbf{u}^{n+1}$,
(C) $\mathcal{S}_{n+\frac{1}{2}}^*(\frac{1}{2}(\mathbf{u}^n+\mathbf{u}^{n+1}))$.

Some implicit methods also lag the spreading and interpolation operators in time, meaning that $\mathcal{S}_n$ and $\mathcal{S}_n^*$ are used to replace occurrences of $\mathcal{S}$ and $\mathcal{S}^*$ evaluated at other times in the above choices for **f** and **U**.

Where the discretization choices for **f** and **U** can be found in the literature, many variations are used. Mayo and Peskin [16] use method 1A (i.e. take choice 1 for **f** and A for **U** from the lists above). Tu and Peskin [27] use a modified version of 1A with time-lagged spreading and interpolation operators (i.e. $\mathcal{S}_n$ and $\mathcal{S}_n^*$ instead of $\mathcal{S}_{n+1}$ and $\mathcal{S}_{n+1}^*$) with steady Stokes flow. Roma et al. [25] and Lee and Leveque [13] both use method 2B, while Griffith and Peskin [10] approximate 2B using an explicitly calculated $\mathbf{X}^{n+1}$. Mori and Peskin [19] use method 3C with lagged spreading and interpolation operators though with the addition of boundary mass, Peskin [21] and Lai and Peskin [12] approximate method 3C with an explicitly calculated $\mathbf{X}^{n+1}$, and the first author of this work experimented with method 3B. We will prove that both 1A and 3C are unconditionally stable and that the stability is not affected by the location at which $\mathcal{S}$ and $\mathcal{S}^*$ are evaluated, provided both are evaluated at the same location.

4. Unconditionally stable schemes

In this section, we prove that 1A and 3C are unconditionally stable. First we consider the spatially continuous case for method 3C and then extend the proof to the discrete case for both 3C and 1A. In each case, the method we use is to define an appropriate energy for the system and show that it is a non-increasing, and hence bounded, function in time.

4.1. Crank–Nicolson for the spatially continuous problem

We begin with method 3C for the spatially continuous problem, showing that the energy of the system

$$E[\mathbf{u},\mathbf{X}] = \langle \mathbf{u},\mathbf{u}\rangle_\Omega + \langle -\mathcal{A}\mathbf{X},\mathbf{X}\rangle_\Gamma \tag{11}$$

is non-increasing. Here the inner products are on $L^2(\Omega)$ and $L^2(\Gamma)$, respectively. This energy represents the sum of the kinetic energy of the fluid and the potential energy in the elasticity of the boundary. Note that $\mathcal{A}$ (the force generation operator on the immersed boundary) must be negative-definite for this definition of energy to make sense. We further require in the proof that $\mathcal{A}$ be self-adjoint and linear (conditions which are satisfied by the common choice $\mathcal{A} = \gamma\frac{\partial^2}{\partial s^2}$). We also assume that divergence-free velocity fields are orthogonal to gradient fields (which may put limitations on the boundary conditions but is satisfied for example by periodic boundary conditions). We also make use of the fact that $\mathcal{S}$ and $\mathcal{S}^*$ are adjoints, which can be seen from the following calculation with $\mathbf{F}\in L^2(\Gamma)$ and $\mathbf{w}\in L^2(\Omega)$

$$\begin{aligned}\langle \mathcal{S}(\mathbf{F}(s,t)),\mathbf{w}(\mathbf{x},t)\rangle_\Omega &= \int_\Omega \mathcal{S}(\mathbf{F}(s,t))(\mathbf{x},t)\cdot\mathbf{w}(\mathbf{x},t)\,\mathrm{d}\mathbf{x} = \int_\Omega\int_\Gamma \mathbf{F}(s,t)\delta(\mathbf{x}-\mathbf{X}(s,t))\,\mathrm{d}s\cdot\mathbf{w}(\mathbf{x},t)\,\mathrm{d}\mathbf{x}\\ &= \int_\Gamma \mathbf{F}(s,t)\cdot\int_\Omega \mathbf{w}(\mathbf{x},t)\delta(\mathbf{x}-\mathbf{X}(s,t))\,\mathrm{d}\mathbf{x}\,\mathrm{d}s = \langle \mathbf{F}(s,t),\mathcal{S}^*(\mathbf{w}(\mathbf{x},t))\rangle_\Gamma.\end{aligned} \tag{12}$$

Recall that the IB method using 3C is

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t} + \nabla p^{n+\frac12} = \frac{\nu}{2}\Delta(\mathbf{u}^{n+1}+\mathbf{u}^n) + \mathcal{S}_{n+\frac12}\mathcal{A}\left(\frac12(\mathbf{X}^n+\mathbf{X}^{n+1})\right), \tag{13}$$

$$\nabla\cdot\mathbf{u}^{n+1} = 0, \tag{14}$$

$$\frac{\mathbf{X}^{n+1}-\mathbf{X}^n}{\Delta t} = \mathcal{S}^*_{n+\frac12}\left(\frac12(\mathbf{u}^n+\mathbf{u}^{n+1})\right). \tag{15}$$

Multiplying through by Δt, taking the inner product of Eq. (13) with $\mathbf{u}^{n+1}+\mathbf{u}^n$ and the inner product of (15) with $-\mathcal{A}(\mathbf{X}^{n+1}+\mathbf{X}^n)$ yields

$$\begin{aligned}\langle \mathbf{u}^{n+1}+\mathbf{u}^n,\mathbf{u}^{n+1}-\mathbf{u}^n\rangle_\Omega &= -\Delta t\left\langle \mathbf{u}^{n+1}+\mathbf{u}^n,\nabla p^{n+\frac12}\right\rangle_\Omega + \frac{\nu\Delta t}{2}\langle \mathbf{u}^{n+1}+\mathbf{u}^n,\Delta(\mathbf{u}^{n+1}+\mathbf{u}^n)\rangle_\Omega\\ &\quad + \frac{\Delta t}{2}\left\langle \mathbf{u}^{n+1}+\mathbf{u}^n,\mathcal{S}_{n+\frac12}\mathcal{A}(\mathbf{X}^n+\mathbf{X}^{n+1})\right\rangle_\Omega,\end{aligned} \tag{16}$$

$$\langle -\mathcal{A}(\mathbf{X}^{n+1}+\mathbf{X}^n),\mathbf{X}^{n+1}-\mathbf{X}^n\rangle_\Gamma = \frac{\Delta t}{2}\left\langle -\mathcal{A}(\mathbf{X}^{n+1}+\mathbf{X}^n),\mathcal{S}^*_{n+\frac12}(\mathbf{u}^n+\mathbf{u}^{n+1})\right\rangle_\Gamma. \tag{17}$$

Adding these two equations gives us

$$\begin{aligned}&\langle \mathbf{u}^{n+1}+\mathbf{u}^n,\mathbf{u}^{n+1}-\mathbf{u}^n\rangle_\Omega + \langle -\mathcal{A}(\mathbf{X}^{n+1}+\mathbf{X}^n),\mathbf{X}^{n+1}-\mathbf{X}^n\rangle_\Gamma\\ &\quad = -\Delta t\left\langle \mathbf{u}^{n+1}+\mathbf{u}^n,\nabla p^{n+\frac12}\right\rangle_\Omega + \frac{\nu\Delta t}{2}\langle \mathbf{u}^{n+1}+\mathbf{u}^n,\Delta(\mathbf{u}^{n+1}+\mathbf{u}^n)\rangle_\Omega\\ &\qquad + \frac{\Delta t}{2}\left\langle \mathbf{u}^{n+1}+\mathbf{u}^n,\mathcal{S}_{n+\frac12}\mathcal{A}(\mathbf{X}^n+\mathbf{X}^{n+1})\right\rangle_\Omega + \frac{\Delta t}{2}\left\langle -\mathcal{A}(\mathbf{X}^n+\mathbf{X}^{n+1}),\mathcal{S}^*_{n+\frac12}(\mathbf{u}^{n+1}+\mathbf{u}^n)\right\rangle_\Gamma.\end{aligned} \tag{18}$$

The last two terms on the right-hand-side cancel by the adjointness of $\mathcal{S}_{n+\frac12}$ and $\mathcal{S}^*_{n+\frac12}$, the first term on the right-hand side is zero because $\mathbf{u}^{n+1}$ and $\mathbf{u}^n$ are divergence-free while $\nabla p^{n+\frac12}$ is a gradient field, and the left-hand side of the equation can be simplified using the linearity and self-adjointness of $\mathcal{A}$. This leaves us with

$$\langle \mathbf{u}^{n+1},\mathbf{u}^{n+1}\rangle_\Omega - \langle \mathbf{u}^n,\mathbf{u}^n\rangle_\Omega + \langle -\mathcal{A}\mathbf{X}^{n+1},\mathbf{X}^{n+1}\rangle_\Gamma - \langle -\mathcal{A}\mathbf{X}^n,\mathbf{X}^n\rangle_\Gamma = \frac{\nu\Delta t}{2}\langle \mathbf{u}^{n+1}+\mathbf{u}^n,\Delta(\mathbf{u}^{n+1}+\mathbf{u}^n)\rangle_\Omega. \tag{19}$$

Using the negative definiteness of the Laplacian operator, the above equation implies

$$E[\mathbf{u}^{n+1},\mathbf{X}^{n+1}] - E[\mathbf{u}^n,\mathbf{X}^n] \leqslant 0. \tag{20}$$

In other words, the energy of the system is bounded for all time, and thus the system remains stable. The key to the proof is that the energy terms representing the work done by the fluid on the boundary and the work done by the boundary on the fluid (the terms involving $\mathcal{S}_{n+\frac{1}{2}}$ and $\mathcal{S}^*_{n+\frac{1}{2}}$) exactly cancel. This is a property not shared by any of the other methods, though in Section 4.3 we show that the remaining terms for 1A result in a decrease of energy in addition to the energy loss from viscosity, and so scheme 1A is also stable.

It is useful to note that the proof is still valid even if $\mathcal{S}_n$ and $\mathcal{S}^*_n$ are used in place of $\mathcal{S}_{n+\frac{1}{2}}$ and $\mathcal{S}^*_{n+\frac{1}{2}}$; i.e. the system need not be fully implicit in order to achieve unconditional stability. This is contrary to what the community expected (as pointed out in Section 1), and it can be exploited to simplify the set of implicit equations that need to be solved, as we do in Section 5, without sacrificing stability.

4.2. Projection methods and discrete delta functions

We now show that the unconditional stability of method 3C extends to the usage of discrete delta functions and (exact) discrete projection methods. We show computationally in Section 5 that an approximate projection method appears to be sufficient for unconditional stability, but use an exact projection for purposes of the proof. Let ∇^h, $\nabla^h\cdot$, and Δ^h be discrete analogs of ∇, $\nabla\cdot$, and Δ satisfying $\Delta^h = \nabla^h \cdot \nabla^h$ (so that the projection is exact). Let A be a discrete analog of $\mathcal{A}$ maintaining linearity, negative-definiteness, and self-adjointness. Let $\mathcal{S}_{n+\frac{1}{2}}$ and $\mathcal{S}^*_{n+\frac{1}{2}}$ be discrete analogs of $\mathcal{S}_{n+\frac{1}{2}}$ and $\mathcal{S}^*_{n+\frac{1}{2}}$ preserving adjointness. We assume boundary conditions for which discretely divergence-free fields and discrete gradient fields are orthogonal and for which ∇^h and Δ^h commute. We employ a pressure-free projection method in the proof. Some other projection methods, such as pressure update projection methods also work, but others might not; in particular, it appears that it was the projection employed by Mayo and Peskin [16] that caused the instability they saw, as we will discuss in Section 4.3.

We note that many, if not most, existing IB implementations satisfy these conditions. For example, an implementation with periodic boundary conditions, employing a (exact) projection method, using the force generation operator $A = \gamma D_+ D_-$, and using any of the standard discrete delta functions [21] will satisfy all of these conditions. All of these particular choices are quite common. Thus, as we will prove below, such implementations could be made unconditionally stable by switching to discretization method 3C.

The adjointness condition on $S_{n+\frac{1}{2}}$ and $S^*_{n+\frac{1}{2}}$ is

$$\left\langle S_{n+\frac{1}{2}}(\mathbf{F}), \mathbf{w}\right\rangle_{\Omega_h} = \left\langle \mathbf{F}, S^*_{n+\frac{1}{2}}(\mathbf{w})\right\rangle_{\Gamma_h}, \tag{21}$$

where $\mathbf{F} \in \ell^2(\Gamma^h)$, $\mathbf{w} \in \ell^2(\Omega^h)$ and the ℓ^2 inner products are defined by

$$\langle \mathbf{v}, \mathbf{w}\rangle_{\Omega_h} = \sum_{ij} \mathbf{v}_{ij} \cdot \mathbf{w}_{ij} \Delta x \Delta y \tag{22}$$

and

$$\langle \mathbf{Y}, \mathbf{Z}\rangle_{\Gamma_h} = \sum_{k} \mathbf{Y}_k \cdot \mathbf{Z}_k \Delta s. \tag{23}$$

The calculation to show this adjointness property for the standard tensor product discrete delta functions [21] is nearly identical to (12) and so is omitted here. With these definitions and assumptions, using the pressure update needed for second order accuracy [4], and utilizing method 3C, our discrete system becomes

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = \frac{\nu}{2}\Delta^h(\mathbf{u}^* + \mathbf{u}^n) + \frac{1}{2} S_{n+\frac{1}{2}} A(\mathbf{X}^n + \mathbf{X}^{n+1}), \tag{24}$$

$$\Delta^h \phi = \frac{1}{\Delta t}\nabla^h \cdot \mathbf{u}^*, \tag{25}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \Delta t \nabla^h \phi, \tag{26}$$

$$p^{n+\frac{1}{2}} = \phi - \frac{\nu}{2}\nabla^h \cdot \mathbf{u}^*, \tag{27}$$

$$\frac{\mathbf{X}^{n+1} - \mathbf{X}^n}{\Delta t} = \frac{1}{2} S^*_{n+\frac{1}{2}}(\mathbf{u}^n + \mathbf{u}^{n+1}). \tag{28}$$

We proceed analogously to the time-discrete spatially-continuous case, and show that the energy of the system

$$E[\mathbf{u},\mathbf{X}] = \langle \mathbf{u},\mathbf{u}\rangle_{\Omega_h} + \langle -A\mathbf{X},\mathbf{X}\rangle_{\Gamma_h} \tag{29}$$

is non-increasing. Taking the inner product of (24) with $\Delta t(\mathbf{u}^{n+1}+\mathbf{u}^n)$, taking the inner product of (26) with $\mathbf{u}^{n+1}+\mathbf{u}^n$ and taking the inner product of (28) with $-\Delta t A(\mathbf{X}^{n+1}+\mathbf{X}^n)$ yields

$$\langle \mathbf{u}^{n+1}+\mathbf{u}^n, \mathbf{u}^*-\mathbf{u}^n\rangle_{\Omega_h} = \frac{\nu\Delta t}{2}\langle \mathbf{u}^{n+1}+\mathbf{u}^n, \Delta^h(\mathbf{u}^*+\mathbf{u}^n)\rangle_{\Omega_h} + \frac{\Delta t}{2}\left\langle \mathbf{u}^{n+1}+\mathbf{u}^n, S_{n+\frac{1}{2}}A(\mathbf{X}^n+\mathbf{X}^{n+1})\right\rangle_{\Omega_h}, \tag{30}$$

$$\langle \mathbf{u}^{n+1}+\mathbf{u}^n, \mathbf{u}^{n+1}-\mathbf{u}^*\rangle_{\Omega_h} = -\Delta t\langle \mathbf{u}^{n+1}+\mathbf{u}^n, \nabla^h\phi\rangle_{\Omega_h}, \tag{31}$$

$$\langle -A(\mathbf{X}^{n+1}+\mathbf{X}^n), \mathbf{X}^{n+1}-\mathbf{X}^n\rangle_{\Gamma_h} = \frac{\Delta t}{2}\left\langle -A(\mathbf{X}^{n+1}+\mathbf{X}^n), S^*_{n+\frac{1}{2}}(\mathbf{u}^n+\mathbf{u}^{n+1})\right\rangle_{\Gamma_h}. \tag{32}$$

Adding all three of these equations and noting the cancellation of the energy terms from the immersed boundary due to Eq. (21) gives us

$$\begin{aligned}&\langle \mathbf{u}^{n+1}+\mathbf{u}^n, \mathbf{u}^{n+1}-\mathbf{u}^n\rangle_{\Omega_h} + \langle -A(\mathbf{X}^{n+1}+\mathbf{X}^n), \mathbf{X}^{n+1}-\mathbf{X}^n\rangle_{\Gamma_h}\\ &\quad = -\Delta t\langle \mathbf{u}^{n+1}+\mathbf{u}^n, \nabla^h\phi\rangle_{\Omega_h} + \frac{\nu\Delta t}{2}\langle \mathbf{u}^{n+1}+\mathbf{u}^n, \Delta^h(\mathbf{u}^*+\mathbf{u}^n)\rangle_{\Omega_h}.\end{aligned} \tag{33}$$

Now, we can employ (26) to eliminate $\mathbf{u}^*$ from this equation. Simultaneously simplifying the left-hand side of the equation using the assumption that A is linear and self-adjoint, we obtain

$$\begin{aligned}&\langle \mathbf{u}^{n+1},\mathbf{u}^{n+1}\rangle_{\Omega_h} - \langle \mathbf{u}^n,\mathbf{u}^n\rangle_{\Omega_h} + \langle -A\mathbf{X}^{n+1},\mathbf{X}^{n+1}\rangle_{\Gamma_h} - \langle -A\mathbf{X}^n,\mathbf{X}^n\rangle_{\Gamma_h}\\ &\quad = -\Delta t\langle \mathbf{u}^{n+1}+\mathbf{u}^n, \nabla^h\phi\rangle_{\Omega_h} + \frac{\nu\Delta t}{2}\langle \mathbf{u}^{n+1}+\mathbf{u}^n, \Delta^h((\mathbf{u}^{n+1}+\Delta t\nabla^h\phi)+\mathbf{u}^n)\rangle_{\Omega_h}.\end{aligned} \tag{34}$$

Making use of commutativity of ∇^h and Δ^h and orthogonality of discretely divergence-free vector fields and discrete gradient fields, we find

$$\langle \mathbf{u}^{n+1},\mathbf{u}^{n+1}\rangle_{\Omega_h} - \langle \mathbf{u}^n,\mathbf{u}^n\rangle_{\Omega_h} + \langle -A\mathbf{X}^{n+1},\mathbf{X}^{n+1}\rangle_{\Gamma_h} - \langle -A\mathbf{X}^n,\mathbf{X}^n\rangle_{\Gamma_h} = \frac{\nu\Delta t}{2}\langle \mathbf{u}^{n+1}+\mathbf{u}^n, \Delta^h(\mathbf{u}^{n+1}+\mathbf{u}^n)\rangle_{\Omega_h}. \tag{35}$$

Using the negative definiteness of the discrete Laplacian operator, the above equation implies

$$E[\mathbf{u}^{n+1},\mathbf{X}^{n+1}] - E[\mathbf{u}^n,\mathbf{X}^n] \leqslant 0, \tag{36}$$

showing that the energy of the system is non-increasing and thus implying that the system is stable.

Just as with the spatially continuous case, nothing in this proof required the system to be fully implicit to achieve unconditional stability; the proof is still valid if S_n and S^*_n (or indeed spreading and interpolation operators evaluated at any location so long as they are adjoints) are used in place of $S_{n+\frac{1}{2}}$ and $S^*_{n+\frac{1}{2}}$. Again, this is a fact we demonstrate in our computational experiments in Section 5.

4.3. Unconditional stability of backward Euler

We now demonstrate that temporal discretization 1A is unconditionally stable, and in fact, that it dissipates more energy from the system than one would find from the effect of viscosity alone. The method and assumptions are the same as in Section 4.2, the only difference is that more work is required to show that extra non-cancelling terms are in fact negative.

Papers that employ a backward Euler discretization of the immersed boundary terms invariably also use a backward Euler discretization of the viscous terms. So, we modify our system for this case to use a backward Euler discretization of the viscous terms instead of a Crank–Nicolson one. With that change, the relevant equations for method 1A are

$$\frac{\mathbf{u}^*-\mathbf{u}^n}{\Delta t} = \nu\Delta^h\mathbf{u}^* + S_{n+1}A\mathbf{X}^{n+1}, \tag{37}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \Delta t\nabla^h\phi, \tag{38}$$

$$\frac{\mathbf{X}^{n+1}-\mathbf{X}^n}{\Delta t} = S^*_{n+1}\mathbf{u}^{n+1}. \tag{39}$$

Taking the inner product of (37) with $\Delta t(\mathbf{u}^{n+1}+\mathbf{u}^n)$, taking the inner product of (38) with $\mathbf{u}^{n+1}+\mathbf{u}^n$ and taking the inner product of (39) with $-\Delta tA(\mathbf{X}^{n+1}+\mathbf{X}^n)$ yields

$$\langle\mathbf{u}^{n+1}+\mathbf{u}^n,\mathbf{u}^*-\mathbf{u}^n\rangle_{\Omega_h}=\Delta t\langle\mathbf{u}^{n+1}+\mathbf{u}^n,\nu\Delta^h\mathbf{u}^*+S_{n+1}A\mathbf{X}^{n+1}\rangle_{\Omega_h},\tag{40}$$

$$\langle\mathbf{u}^{n+1}+\mathbf{u}^n,\mathbf{u}^{n+1}-\mathbf{u}^*\rangle_{\Omega_h}=-\Delta t\langle\mathbf{u}^{n+1}+\mathbf{u}^n,\nabla^h\phi\rangle_{\Omega_h},\tag{41}$$

$$\langle-A(\mathbf{X}^{n+1}+\mathbf{X}^n),\mathbf{X}^{n+1}-\mathbf{X}^n\rangle_{\Gamma_h}=\Delta t\langle-A(\mathbf{X}^{n+1}+\mathbf{X}^n),S^*_{n+1}\mathbf{u}^{n+1}\rangle_{\Gamma_h}.\tag{42}$$

We can simplify (42) by making use of the assumption that A is linear and self-adjoint and by using Eq. (39) to eliminate the appearance of $\mathbf{X}^n$ on the right-hand side. Making these simplifications and adding all three equations, we obtain

$$\begin{aligned}&\langle\mathbf{u}^{n+1},\mathbf{u}^{n+1}\rangle_{\Omega_h}-\langle\mathbf{u}^n,\mathbf{u}^n\rangle_{\Omega_h}+\langle-A\mathbf{X}^{n+1},\mathbf{X}^{n+1}\rangle_{\Gamma_h}-\langle-A\mathbf{X}^n,\mathbf{X}^n\rangle_{\Gamma_h}\\&\quad=\Delta t\langle\mathbf{u}^{n+1}+\mathbf{u}^n,\nu\Delta^h\mathbf{u}^*+S_{n+1}A\mathbf{X}^{n+1}-\nabla^h\phi\rangle_{\Omega_h}+\Delta t\langle-2A\mathbf{X}^{n+1}+A\Delta tS^*_{n+1}\mathbf{u}^{n+1},S^*_{n+1}\mathbf{u}^{n+1}\rangle_{\Gamma_h}.\end{aligned}\tag{43}$$

Making use of our definition of energy and using the identity $\mathbf{u}^{n+1}+\mathbf{u}^n=(\mathbf{u}^n-\mathbf{u}^{n+1})+2\mathbf{u}^{n+1}$, we obtain

$$\begin{aligned}E[\mathbf{u}^{n+1},\mathbf{X}^{n+1}]-E[\mathbf{u}^n,\mathbf{X}^n]&=\Delta t\langle\mathbf{u}^n-\mathbf{u}^{n+1},\nu\Delta^h\mathbf{u}^*+S_{n+1}A\mathbf{X}^{n+1}-\nabla^h\phi\rangle_{\Omega_h}\\&\quad+\Delta t\langle2\mathbf{u}^{n+1},\nu\Delta^h\mathbf{u}^*+S_{n+1}A\mathbf{X}^{n+1}-\nabla^h\phi\rangle_{\Omega_h}\\&\quad+\Delta t\langle-2A\mathbf{X}^{n+1}+A\Delta tS^*_{n+1}\mathbf{u}^{n+1},S^*_{n+1}\mathbf{u}^{n+1}\rangle_{\Gamma_h}.\end{aligned}\tag{44}$$

We can use Eqs. (37) and (38) to eliminate the first occurrence of $\nu\Delta^h\mathbf{u}^*+S_{n+1}A\mathbf{X}^{n+1}-\nabla^h\phi$ to replace it with $\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}$. Simultaneously using Eq. (38) to eliminate the second appearance of $\mathbf{u}^*$ we obtain

$$\begin{aligned}E[\mathbf{u}^{n+1},\mathbf{X}^{n+1}]-E[\mathbf{u}^n,\mathbf{X}^n]&=\Delta t\left\langle\mathbf{u}^n-\mathbf{u}^{n+1},\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}\right\rangle_{\Omega_h}+\Delta t\langle2\mathbf{u}^{n+1},\nu\Delta^h\mathbf{u}^{n+1}\rangle_{\Omega_h}\\&\quad+\Delta t\langle2\mathbf{u}^{n+1},\nu\Delta t\Delta^h\nabla^h\phi-\nabla^h\phi\rangle_{\Omega_h}+\Delta t\langle2\mathbf{u}^{n+1},S_{n+1}A\mathbf{X}^{n+1}\rangle_{\Omega_h}\\&\quad+\Delta t\langle-2A\mathbf{X}^{n+1},S^*_{n+1}\mathbf{u}^{n+1}\rangle_{\Gamma_h}-\Delta t^2\langle-AS^*_{n+1}\mathbf{u}^{n+1},S^*_{n+1}\mathbf{u}^{n+1}\rangle_{\Gamma_h}.\end{aligned}\tag{45}$$

The fourth and fifth terms on the right-hand side cancel by the adjoint property of S_{n+1} and S^*_{n+1}. The third term vanishes due to the commutativity of Δ^h and ∇^h and due to the orthogonality of discretely divergence-free vector fields and discrete gradients. Hence we are left with

$$\begin{aligned}E[\mathbf{u}^{n+1},\mathbf{X}^{n+1}]-E[\mathbf{u}^n,\mathbf{X}^n]&=-\langle\mathbf{u}^{n+1}-\mathbf{u}^n,\mathbf{u}^{n+1}-\mathbf{u}^n\rangle_{\Omega_h}+2\nu\Delta t\langle\mathbf{u}^{n+1},\Delta^h\mathbf{u}^{n+1}\rangle_{\Omega_h}\\&\quad-\Delta t^2\langle-AS^*_{n+1}\mathbf{u}^{n+1},S^*_{n+1}\mathbf{u}^{n+1}\rangle_{\Gamma_h}.\end{aligned}\tag{46}$$

The second term on the right-hand side is non-positive by the negative definiteness of the Δ^h operator and represents the dissipation of energy due to viscosity. From this computation, we see that method 1A is unconditionally stable, and that it dissipates more energy than one would get just from the effect of viscosity. As with method 3C, the proof did not rely on the time level at which S and S^* are evaluated, other than requiring them to be evaluated at the same time level so that they are indeed adjoints.

We note that Mayo and Peskin [16] also used a backward Euler discretization of both the viscous and immersed boundary terms, but reported a lack of stability in their method that was also confirmed by Stockie and Wetton [26]. However, they did not solve the system of equations (37)–(39). The salient difference between method 1A and their method is that they moved the forcing terms from the momentum equation into the projection step to obtain a system of equations of the form

$$\frac{\mathbf{u}^*-\mathbf{u}^n}{\Delta t}=\nu\Delta^h\mathbf{u}^*,\tag{47}$$

$$\Delta^h\phi=\frac{1}{\Delta t}\nabla^h\cdot\mathbf{u}^*+\nabla^h\cdot(SA\mathbf{X}^{n+1}),\tag{48}$$

$$\mathbf{u}^{n+1}=\mathbf{u}^*-\Delta t\nabla^h\phi+\Delta t(SA\mathbf{X}^{n+1}),\tag{49}$$

$$\frac{\mathbf{X}^{n+1}-\mathbf{X}^n}{\Delta t}=S^*\mathbf{u}^{n+1}.\tag{50}$$

This modification was important to Mayo and Peskin's proof of convergence of the iterative method they used to solve their implicit equations. We believe that this change was the cause of the instability they observed. There were also other minor differences between Mayo and Peskin's method and method 1A, notably the inclusion of advection terms and the use of ADI splitting for the advection and viscous terms. Stockie and Wetton, however, observed the same instability while only considering Stokes equations and without employing an ADI splitting.

5. Computational results

In this section, we demonstrate the unconditional stability of our discretization computationally and indicate how stability is affected by some modifications not covered in the proofs of the preceding section. Since approximate projections are an increasingly commonly used method in the community that provides a velocity field that is not quite discretely divergence-free, it is the first modification that we test. We use an approximate projection that has an $O(h^2)$ error in the divergence-free condition for the velocity. The other modification that we test is the addition of advection terms from the Navier–Stokes equations.

The test problem we use is one typically seen in the literature, in which the immersed boundary is a closed loop initially in the shape of an ellipse [13,14,16,25–27]. We choose an ellipse initially aligned in the coordinate directions with horizontal semi-axis $a = 0.28125$ cm and vertical semi-axis $b = 0.75a$ cm. The fluid is initially at rest in a periodic domain, Ω, with $\Omega = [0,1]\times[0,1]$. For this test problem, the boundary should perform damped oscillations around a circular equilibrium state with the same area as that of the original ellipse. The configuration of the boundary at different times can be seen in Fig. 2.

We employ a cell centered grid with uniform grid spacing $h = 1/64$ cm in both x and y, and discrete gradient, divergence, and Laplacian operators given by the formulas

$$(\nabla^h \cdot \mathbf{u})_{ij} = \frac{u_{i+1,j} - u_{i-1,j}}{2h} + \frac{v_{i,j+1} - v_{i,j-1}}{2h}, \tag{51}$$

$$(\nabla^h p)_{ij} = \left(\frac{p_{i+1,j} - p_{i-1,j}}{2h}, \frac{p_{i,j+1} - p_{i,j-1}}{2h}\right), \tag{52}$$

$$(\Delta^h_{\text{wide}} p)_{ij} = \frac{p_{i+2,j} - 2p_{i,j} + p_{i-2,j}}{4h^2} + \frac{p_{i,j+2} - 2p_{i,j} + p_{i,j-2}}{4h^2}, \tag{53}$$

$$(\Delta^h_{\text{tight}} p)_{ij} = \frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{h^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{h^2}, \tag{54}$$

where $\Delta^h = \Delta^h_{\text{wide}}$ will be used for our Poisson solve in all tests other than those from Section 5.2, and $\Delta^h = \Delta^h_{\text{tight}}$ will be used for the viscous terms as well as the Poisson solve in Section 5.2. We define $\mathcal{A}$ as $\gamma\frac{\partial^2}{\partial s^2}$ and choose N_B, the number of immersed boundary points, to approximately satisfy $N_\text{B} = \frac{2L_\text{B}}{h}$, where L_B is the arclength of the immersed boundary. In all tests, except where otherwise noted, we use (spatially discrete) method 3C with lagged spreading and interpolation operators. We also use the common four point delta function

$$\delta_h(x,y) = \delta_h(x)\delta_h(y), \tag{55}$$

$$\delta_h(x) = \begin{cases} \frac{1}{4h}\left(1 + \cos(\frac{\pi x}{2h})\right), & |x| \leqslant 2h, \\ 0, & |x| \geqslant 2h. \end{cases} \tag{56}$$

The immersed boundary update equation,

$$\frac{\mathbf{X}^{n+1} - \mathbf{X}^n}{\Delta t} = \frac{1}{2} S_n^*(\mathbf{u}^n + \mathbf{u}^{n+1}), \tag{57}$$

can be re-written as

$$g(\mathbf{X}) = \mathbf{X} - \mathbf{X}^n - \frac{\Delta t}{2} S_n^*(\mathbf{u}^n + \mathbf{u}^{n+1}) = 0 \tag{58}$$

so that we can write the implicit system that we must solve as $g(\mathbf{X}^{n+1}) = 0$ (note that $\mathbf{u}^{n+1}$ depends on $\mathbf{X}^{n+1}$ too). To solve this implicit system of equations, we use an approximate Newton solver (by employing finite

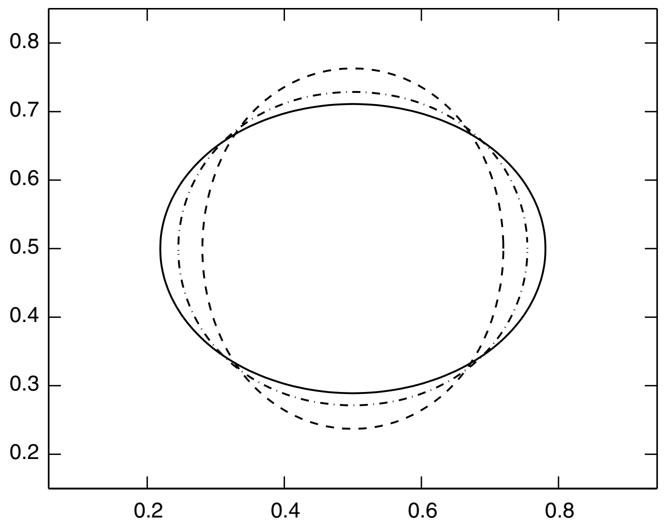

Fig. 2. Computed immersed boundary positions at successive times; the solid line is the initial shape, and the dashed-dotted and dashed lines show the configuration of the boundary later in the simulation.

difference approximations to obtain the Jacobian g', which requires $O(N_B)$ fluid solves per implicit iteration to compute and is thus extremely slow). We discuss the use of the approximate Newton solver further in Section 6.

5.1. Computational verification of the method

5.1.1. Refinement study

We begin with a simple convergence study to verify that the implicit discretization results in a consistent method. We expect only first order accuracy for two reasons: the immersed boundary method has been shown to exhibit first order accuracy on problems with a sharp interface, and our lagging of the spreading and interpolation operators makes the discretization formally first order. The immersed boundary method has three numerical parameters affecting the accuracy – the Eulerian mesh width, h, the average Lagrangian mesh width, Δs, and the timestep, Δt. The values of these numerical parameters that we use in our refinement study are given in Table 1. However, since we always choose N_B so that Δs is approximately $h/2$, we report the size

Table 1
Parameters used in numerical refinement study

Level #	h	N_B	Δt
1	1/16	50	0.05
2	1/32	100	0.025
3	1/64	200	0.0125
4	1/128	400	0.00625

Table 2
Results of a refinement study showing first order convergence of method 3C

q	$E_1(q)$	$E_2(q)$	$E_3(q)$	Rate
u	1.06e − 03	4.17e − 04	1.76e − 04	2.4485
X	5.25e − 04	2.96e − 04	1.58e − 04	1.8239

The convergence rate is defined as sqrt(E_1/E_3). $E_i(q)$ measures the error in variable q at level i, defined by $\|q_i - \text{coarsen}(q_{i+1})\|_2$. Here, q_i is the value of variable q at time $t = 0.2$ s computed with the numerical parameters for level i from Table 1. The coarsen operator is simple averaging of nearest values for Eulerian quantities, and injection from identical gridpoints for Lagrangian quantities.

of N_B instead. For our convergence problem, we set $\gamma = 1$ (cm^4/s^2; note that γ includes a factor of $1/\rho$ where $\rho = 1.0$ g/cm^3 is the density), and $\nu = 0.01$ cm^2/s.

The results of the refinement study are displayed in Table 2. Because an analytic solution is not available, we estimate the convergence rate by comparing the differences of the numerical solutions between successive grid levels. We use the ℓ^2 norms induced by the inner products in Eqs. (22) and (23).

5.1.2. Comparison to explicit method

For sufficiently small timesteps, the explicit and implicit methods produce nearly identical results, as expected. For larger timesteps, the energy measure defined by Eq. (29) gives us a useful way to compare the two methods. In Fig. 3, we show the energy of the system as a function of time for four different simulations. All four use $\nu = 0.01$ and $\gamma = 1$, but differ on timestep size and whether the explicit or implicit method is used. When $\Delta t = 6 \times 10^{-3}$, both the explicit and the implicit method give nearly identical results until $t = 0.2$ at which point the explicit method becomes unstable. When $\Delta t = 6 \times 10^{-2}$, the implicit method gives results nearly identical to the implicit method with the smaller timestep, but the explicit method goes unstable immediately.

5.1.3. Parameter testing on the inviscid problem

Since the parameter regime corresponding to large elastic tension and small viscosity is where the traditional immersed boundary method has been observed to suffer from stability problems unless a small timestep is used, we sought to test that parameter regime with our implicit method. We set the viscosity to 0 (to provide a more stringent test of the stability of our method) and explored with a wide range of timesteps and stiffnesses. We ran with each combination of $\Delta t = 10^{-2}$, 1, 10^2, 10^5, 10^{10}, and $\gamma = 1$, 10^2, 10^5, 10^{10}, representing 20 different tests. Comparing with the explicit method, the explicit method went unstable during the simulation for $\Delta t = 6.0 \times 10^{-3}$ when $\gamma = 1$ and ν is increased to 0.01; also, it went unstable for $\Delta t = 4.5 \times 10^{-8}$ when $\nu = 0.01$, $\gamma = 10^{10}$. In fact, if we increased Δt further to 4.5×10^0 and 6.4×10^{-6} for these pairs of ν and γ, respectively, we saw the energy increase by more than a factor of 10^4 and saw points on the boundary move more than a few dozen times the length of the computational domain *within the very first timestep* with the explicit method.

Note that inviscid simulations are a particularly good check for the method since, as can be seen from the energy proof of Section 4.2, the energy defined by Eq. (29) should remain constant. We ran all our simulations until time $t \geqslant 0.5$ s, and verified that for all 20 combinations of these parameters, the solution to the implicit system remained stable throughout the simulation and that the energy of the system was indeed constant.

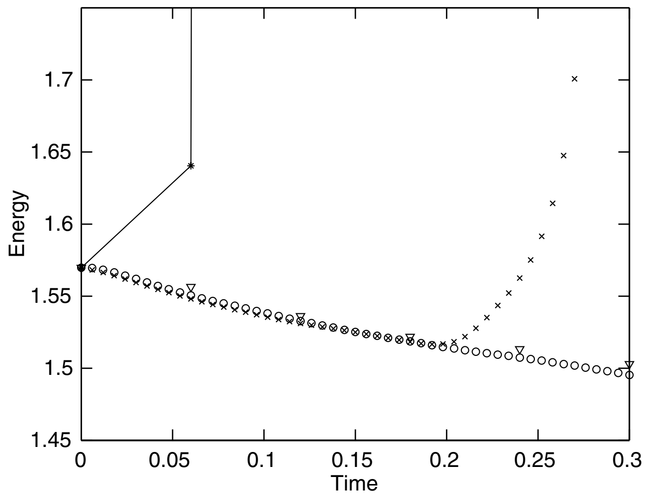

Fig. 3. Energy of the system for four different simulations with $\nu = 0.01$. ○: implicit method, $\Delta t = 6 \times 10^{-3}$; ∇: implicit method, $\Delta t = 6 \times 10^{-2}$; ×: explicit method, $\Delta t = 6 \times 10^{-3}$; –●–: explicit method, $\Delta t = 6 \times 10^{-2}$.

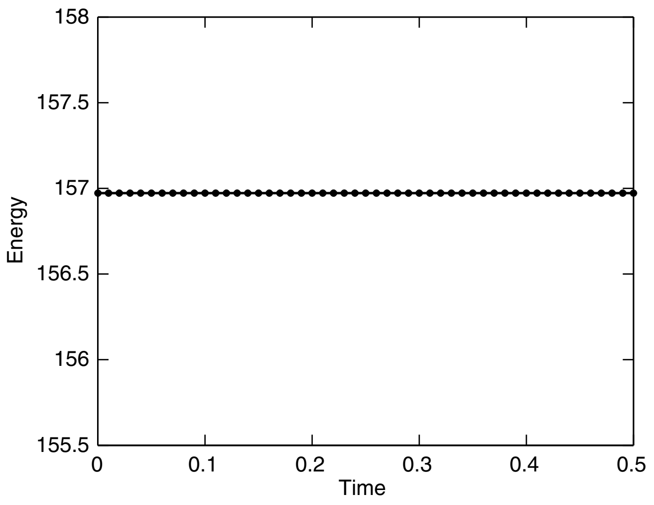

Fig. 4. Energy of the system during the course of an inviscid simulation with the implicit method, showing perfect energy conservation.

Fig. 4 shows the energy of the system for the set of parameters $\gamma = 10^2$, $\Delta t = 10^{-2}$. Computing with large timesteps (e.g. $\Delta t = 10^{10}$) may not yield particularly accurate results (because errors of $O(\Delta t)$ obviously need not be small), but it does illustrate the stability of the method – the boundary configuration was still elliptical at the end of the simulation, points on the immersed boundary moved much less than the length of the computational domain, and there was no change in energy during the simulation.

5.1.4. Comparison to method 1A

The only reason for the stability proof of method 1A given in Section 4.3 was to assist the investigation of why the Mayo and Peskin scheme failed to be unconditionally stable. However, implementing method 1A with lagged spreading and interpolation operators requires only a minor change to the code and provides an additional test of the analytical results. We present three simulations, all with $\Delta t = 10^{-2}$ and $\gamma = 1$. The three simulations were method 3C with no viscosity, method 1A with no viscosity, and method 3C with $\nu = 0.01$. The results are plotted in Fig. 5. Since method 1A is run with no viscosity, the solution to the continuous equations

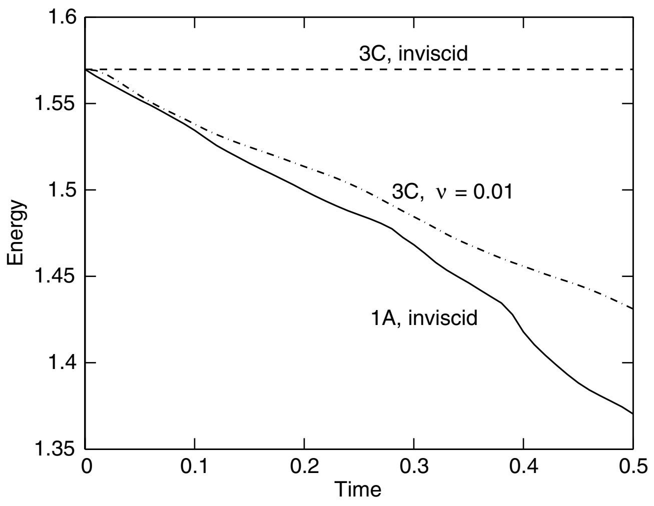

Fig. 5. Energy of the system with $\gamma = 1$. —: method 1A with no viscosity; – –: method 3C with no viscosity, · – ·: method 3C with $\nu = 0.01$.

will have constant energy, but we showed in Section 4.3 that method 1A will have energy dissipation other than from viscosity. From the figure we see indeed that this is the case, and that method 1A loses slightly more energy than method 3C loses with a viscosity of 0.01.

5.2. Unconditional stability with an approximate projection

Calculations with a pressure-free approximate projection version of method 3C were run for the same values of Δt, γ as in Section 5.1.3 and with $v = 0$ or 0.01. In all cases, the solution to the implicit system remained stable throughout the simulation and the energy of the system, as defined by Eq. (29), did not increase. As with the exact-projection calculation, the boundary configuration was elliptical at the end of each simulation. The simulations with $v = 0$ are particularly interesting. For such simulations, the energy remains constant using an exact projection, so this allowed us to more clearly determine the effect of the approximate projection. In all cases, we found that the energy was non-increasing, meaning that the approximate projection appears to have a neutral or stabilizing effect.

The approximate projection, $\tilde{\mathbb{P}}$, has the interesting property that when iterated it will converge to an exact projection, $\mathbb{P}$; i.e. $\tilde{\mathbb{P}}^m \to \mathbb{P}$ as $m \to \infty$ [2]. This means that performing additional (approximate) projections per fluid solve must eventually reduce the additional energy dissipation due to using approximate projections. Fig. 6 shows this effect. This simulation was run with $\gamma = 1$, $v = 0$, $\Delta t = 10^{-2}$, and run until $t = 0.1$. Since approximate projections do not exactly enforce the incompressibility constraint, it is also interesting to note how approximate projections affect the volume conservation of the enclosed membrane. The IB method is well known to exhibit volume loss for closed pressurized membranes [5,14,21,24], even when using an exact projection. Fig. 6 demonstrates how approximate projections also have the effect of increasing the amount of volume loss. For comparison, an exact projection has a volume loss of 3.12% (almost exactly where the dashed line ends up in Fig. 6) and an energy loss of 5.55×10^{-14}%. (The energy loss is not exactly 0 with the exact projection because the implicit equations are only solved to a certain tolerance and because of the presence of round-off errors.)

5.3. Stability for the full Navier–Stokes equations

We solved this problem again including the advection terms from the Navier–Stokes equations. We used a first order upwind discretization of the advection terms in convective form

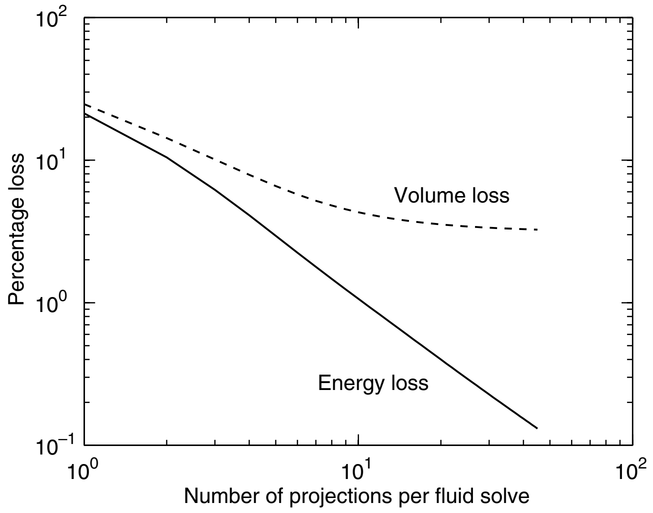

Fig. 6. Energy (—) and volume (- -) loss by time $t = 0.1$ in an inviscid simulation as a function of the number of approximate projections performed. Computations performed with $\gamma = 1$ and $\Delta t = 10^{-2}$.

$$H(x) = \begin{cases} 1, & x > 0, \\ 0, & \text{otherwise}, \end{cases} \tag{59}$$

$$(\mathbf{u}\cdot\nabla\mathbf{u})^n_{ij} = \begin{bmatrix} H(u^n_{ij})u^n_{ij}\frac{u^n_{i+1,j}-u^n_{ij}}{h} + H(-u^n_{ij})u^n_{ij}\frac{u^n_{ij}-u^n_{i-1,j}}{h} + H(v^n_{ij})v^n_{ij}\frac{u^n_{i,j+1}-u^n_{ij}}{h} + H(-v^n_{ij})v^n_{ij}\frac{u^n_{ij}-u^n_{i,j-1}}{h} \\ H(u^n_{ij})u^n_{ij}\frac{v^n_{i+1,j}-v^n_{ij}}{h} + H(-u^n_{ij})u^n_{ij}\frac{v^n_{ij}-v^n_{i-1,j}}{h} + H(v^n_{ij})v^n_{ij}\frac{v^n_{i,j+1}-v^n_{ij}}{h} + H(-v^n_{ij})v^n_{ij}\frac{v^n_{ij}-v^n_{i,j-1}}{h} \end{bmatrix}. \tag{60}$$

We ran the simulation with $\nu = 0$ and at the CFL constraint $\Delta t = \frac{h}{\|\mathbf{u}\|_\infty}$ for each of $\gamma = 1, 10^2, 10^5, 10^{10}$. Since the fluid is initially at rest, we (somewhat arbitrarily) set Δt for the first timestep to be about one-fifth the length of time needed for one oscillation. In each case, we ran until two full oscillations of the boundary had occurred and verified that the energy of the system was decreasing in all cases. Fig. 7 plots the energy as a function of time for the simulation where $\gamma = 10^{10}$.

5.4. Volume loss and stability

Even though the velocity field on the Eulerian grid will be discretely divergence-free when an exact projection is used, this does not guarantee that the interpolated velocity field (in which the immersed boundary moves) is continuously divergence-free. Our implicit method does not fix this problem, revealing an interesting and unexpected view of the relationship of the incompressibility constraint and stability. When running our implicit method with a timestep sufficiently small that the explicit scheme is stable, we observed that the explicit method and our implicit method exhibit nearly identical volume loss. Since the total volume loss by the end of the simulation is an increasing function of the size of the timestep, and because the implicit method is generally used with a larger timestep than the explicit method, use of the implicit method will generally result in a greater volume loss. Stockie and Wetton [26] observed similar volume loss for the Mayo and Peskin scheme and concluded that it was this accumulation of error in the incompressibility condition that caused the stability limitation on the timestep for that scheme. While a reasonable hypothesis, our implicit scheme shows similarly increasing volume loss despite being unconditionally stable. In fact, partly based on correlation of energy and volume loss observed in Section 5.2, we conjecture the opposite – that the loss of volume in the IB method actually aids stability. A loss of volume results in a smaller elastic force due to the shortened distance between points, making the resulting grid forces smaller and thus making the system less likely to overcorrect for any previous errors.

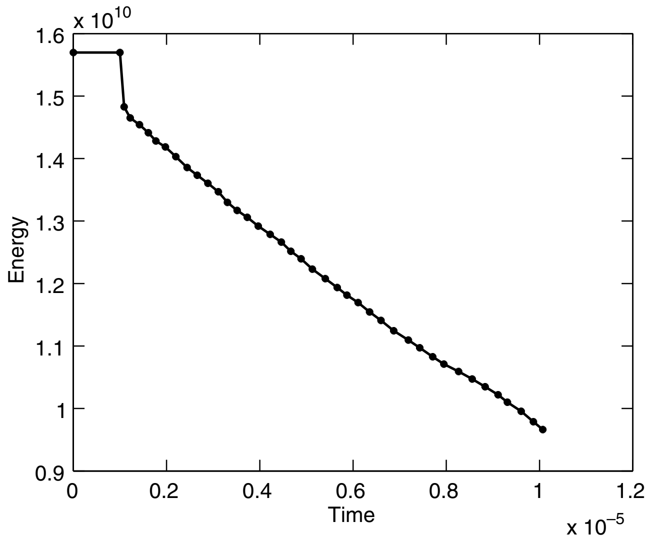

Fig. 7. Energy of the system with advection terms when $\gamma = 10^{10}$.

6. Discussion

In [21], Peskin provides an overview of the current state of the IB method. In his outline of active research directions and outstanding problems, the first issue addressed is the "severe restriction on the time-step duration" to which existing implementations are subject. Pointers to research that has been ongoing in this area for over a decade are provided, and Peskin states that "it would be a huge improvement if this [restriction] could be overcome". The analysis of this paper demonstrates that unconditionally stable IB methods are possible and identifies specific features of the method required to achieve unconditional stability. This paper therefore solves the stability problem posed by Peskin – with some caveats. For example, our results do not apply to all extensions of the IB method, and other outstanding problems of the IB method still exist. The results of this paper raise several interesting questions as well. Can such a method be made efficient? Can it be made more accurate? Can the volume loss be mitigated or removed? Does the stability carry over to common extensions and modifications of the standard IB method? Is there anything that can be done to improve existing explicit codes without switching to an implicit solver? We discuss these questions in this section with potential solutions suggested by results from this paper.

6.1. *Efficiency*

Perhaps the most prominent question is whether the method can be made efficient. Indeed, this question seems to be outstanding in the community for the various implicit methods that have been proposed and used, as evidenced by the fact that Stockie and Wetton [26] is the only paper we are aware of that has concretely compared computational costs of explicit and implicit methods. In some sense, we have returned to where Tu and Peskin [27] left off. There are similarities between Tu and Peskin's paper and ours that many will have noticed: they presented a method that appeared to be unconditionally stable, they solved their implicit system of equations using Newton's method and which was therefore far less efficient than an explicit method – and they then presented a challenge to come up with a more efficient version. There are some important differences: our method applies to more than just steady Stokes flow, we have *proven* that some methods actually are unconditionally stable, we have discovered why the paper written to meet the challenge from Tu and Peskin (namely Mayo and Peskin [16]) failed to remain stable, and we have demonstrated that aspects of the problem that the community suspected to be causing the instability of previous implicit methods (namely, volume loss and lagging of spreading and interpolation operators) were not the actual source of instability.

The ability to time lag spreading and interpolation operators and still have a stable method appears particularly useful in coming up with an efficient scheme. Not lagging those operators results in a set of highly nonlinear equations for the implicit system, which is difficult to solve unless the timestep is small. Lagging the spreading and interpolation operators results in a linear system of equations which opens up a wealth of possibilities and makes a large set of existing linear solvers applicable to the problem.

The authors very recently became aware that Mori and Peskin have created a fully implicit method (including the advection terms from the Navier–Stokes equations and a new way of handling additional boundary mass) and a corresponding implicit system solver that is competitive with the explicit method, particularly when the elastic tension is high. Mori and Peskin have also proven their particular method to be unconditionally stable (Y. Mori, personal communication, July 18, 2006).

6.2. *Accuracy*

By lagging the spreading and interpolation operators, we have a method that is only first order accurate in time. Since the immersed boundary method is only first order accurate for sharp interface problems, and it is to such problems that the method is typically applied, we did not address this issue in this paper. However, the immersed boundary method is beginning to be applied to problems with a thick interface [10,19] where the IB method can achieve higher order accuracy. Also, the immersed interface method can be used in problems with a sharp interface and get second order accuracy.

One possibility for obtaining a second order accurate scheme comes from realizing that our proof for stability did not rely at all on where S and S^* were evaluated – so long as they were evaluated at the same location. This means that we can employ a two step method, where $\mathbf{X}^{n+\frac{1}{2}}$ is computed in the first step to at least first order accuracy, and then the value of $\mathbf{X}^{n+\frac{1}{2}}$ is used as the location of the spreading and interpolation operators in a second step to solve for $\mathbf{X}^{n+1}$ and $\mathbf{u}^{n+1}$. See Lai and Peskin [12] for an example of a very similar explicit two step approach used to obtain formal second order accuracy.

6.3. Volume conservation

Another important issue in some applications concerns the volume loss that occurs with the immersed boundary method. As discussed in Section (5.4), the implicit method will generally exhibit greater volume loss than the explicit method due to the use of a larger timestep. One solution is to use the immersed interface method [14,15] or a hybrid IB/II method [13] – though the stability results of this paper might not hold with the different spatial discretizations used for the "spreading" and interpolation operators in such methods. Another would be to use the modified stencils of Peskin and Printz [24], which satisfy the necessary conditions in the proofs in this paper. Peskin and Printz reported substantially improved volume conservation, but at the cost of wider stencils and more computationally expensive interpolation and spreading.

6.4. Extensions

There are many extensions to the IB method as well as variations on how the fluid equations are solved. One extension that has been mentioned many times already is the immersed interface method and hybrid immersed boundary/immersed interface methods. These methods modify the finite difference stencils of the fluid solver near the immersed boundary instead of utilizing discrete delta functions to spread the force from the Lagrangian to the Eulerian grid. They also typically use a different interpolation scheme, such as bilinear interpolation. For these methods, the necessary adjoint property of the "spreading" and interpolation operators does not hold. Whether the stability results of this paper can be extended for such schemes, or whether those schemes can be modified to be made unconditionally stable, is unknown. Similarly, it is unknown how the stability is affected by more complicated discretizations such as the double projection method proposed by Almgren et al. [2], an L0-stable discretization of the viscous terms from Twizell et al. [28], or some methods of incorporating boundary mass in the IB method [11,30].

6.5. Explicit method stability

Finally, we make one point that might be of use to those with existing explicit codes with nonzero viscosity. Computing the energy of the system can provide a way to monitor the stability of the method and possibly even predict the onset of instability and prevent it. We found that when the explicit method went unstable, the energy at first only slightly increased. This slight increase was followed by a dramatic acceleration of energy increase in ensuing timesteps with the simulation becoming unstable within only a few timesteps. This suggests that such codes could be modified to monitor the energy, and when the energy at the end of any timestep is greater than the energy at the beginning of the timestep, repeat the timestep with a smaller value of Δt.

7. Conclusions

We have shown that both a backward Euler and a Crank–Nicolson-like discretization of the nonlinear immersed boundary terms of the IB equations can yield unconditionally stable methods in conjunction with unsteady Stokes flow. While this might seem unsurprising, there are some subtleties about how the discretization is chosen in order to achieve unconditional stability. In particular, we showed that a backward Euler discretization of the immersed boundary terms is unconditionally stable when the force is included in the momentum equation, while previous authors who included the force in the projection noted an instability. We also discussed the subtleties in how different Crank–Nicholson discretizations of the immersed boundary terms can be obtained, and proved that one particular way of selecting a backward Euler and Crank–Nicholson-like discretization of the full system resulted in unconditionally stable methods.

We also proved (contrary to "common knowledge") that the time discretization need not be fully implicit in order to maintain unconditional stability. That is, the evaluation of force spreading and interpolation operators can be explicit. Because the scheme need not be fully implicit, we have shown that there exists a (consistent) linear numerical scheme approximating the (nonlinear) IB equations which is unconditionally stable.

Another unexpected result was that the error in the incompressibility constraint exhibited by existing implementations apparently does not adversely affect the stability of the computations. We proved that for implementations employing an exact projection and satisfying the necessary conditions from Section 4.2, the method will be unconditionally stable – despite the fact that such methods suffer volume change due to the interpolated velocity field on the Lagrangian grid not being divergence-free. We further demonstrated computationally that a common method (namely, approximate projections) which will result in the divergence-free constraint being satisfied only approximately on both the Eulerian and Lagrangian grids still appears to be unconditionally stable.

Finally, we demonstrated computationally that the advection terms from the Navier–Stokes equations present no additional difficulty beyond the stability (CFL) constraint of advection alone.

Acknowledgments

The work of EPN was supported in part by the Department of Energy Computational Science Graduate Fellowship Program of the Office of Science and National Nuclear Security Administration in the Department of Energy under contract DE-FG02-97ER25308, and in part by NSF Grants DMS-0139926 and DMS-EMSW21-0354259. The work of A.L.F. and R.D.G. was supported in part by NSF Grant DMS-0139926. The work of R.M.K. was supported in part by NSF Career Award CCF0347791.

The authors thank Grady Wright and Samuel Isaacson for helpful discussions while writing this paper.

References

[1] G. Agresar, J.J. Linderman, G. Tryggvason, K.G. Powell, An adaptive, cartesian, front-tracking method for the motion, deformation and adhesion of circulating cells, Journal of Computational Physics 143 (1998) 346–380.
[2] A.S. Almgren, J.B. Bell, W.Y. Crutchfield, Approximate projection methods: Part I. Inviscid analysis, SIAM Journal of Scientific Computing 22 (2000) 1139–1159.
[3] R.P. Beyer, A computational model of the cochlea using the immersed boundary method, Journal of Computational Physics 98 (1992) 145–162.
[4] D.L. Brown, R. Cortez, M. Minion, Accurate projection methods for the incompressible Navier–Stokes equations, Journal of Computational Physics 168 (2001) 464–499.
[5] R. Cortez, M. Minion, The blob projection method for immersed boundary problems, Journal of Computational Physics 161 (2000) 428–453.
[6] R. Dillon, L. Fauci, A. Fogelson, D. Gaver, Modeling biofilm processes using the Immersed Boundary method, Journal of Computational Physics 129 (1996) 85–108.
[7] L.J. Fauci, A.L. Fogelson, Truncated newton methods and the modeling of complex immersed elastic structures, Communications on Pure and Applied Mathematics 66 (1993) 787–818.
[8] L.J. Fauci, C.S. Peskin, A computational model of aquatic animal locomotion, Journal of Computational Physics 77 (1988) 85–108.
[9] A.L. Fogelson, A mathematical model and numerical method for studying platelet adhesion and aggregation during blood clotting, Journal of Computational Physics 1 (1984) 111–134.
[10] B.E. Griffith, C.S. Peskin, On the order of accuracy of the immersed boundary method: higher order convergence rates for sufficiently smooth problems, Journal of Computational Physics 208 (2005) 75–105.
[11] Y. Kim, L. Zhu, X. Wang, C.S. Peskin, On various techniques for computer simulation of boundaries with mass, in: K.J. Bathe (Ed.), Computational Fluid and Solid Mechanics: Proceedings Second M.I.T. Conference on Computational Fluid and Solid Mechanics, June 17–20, 2003, pp. 1746–1750.
[12] M.-C. Lai, C.S. Peskin, An Immersed Boundary method with formal second-order accuracy and reduced numerical viscosity, Journal of Computational Physics 180 (2000) 705–719.
[13] L. Lee, R. Leveque, An Immersed Interface method for incompressible Navier–Stokes equations, SIAM Journal of Scientific Computing 25 (2003) 832–856.
[14] R.J. Leveque, Z. Li, Immersed interface methods for Stokes flow with elastic boundaries or surface tension, SIAM Journal of Scientific Computing 18 (1997) 709–735.
[15] Z. Li, M.-C. Lai, The immersed interface method for the Navier–Stokes equations with singular forces, Journal of Computational Physics 171 (2001) 822–842.

[16] A.A. Mayo, C.S. Peskin, An implicit numerical method for fluid dynamics problems with immersed elastic boundaries, Contemporary Mathematics 141 (1993) 261–277.
[17] L.A. Miller, C.S. Peskin, When vortices stick: an aerodynamic transition in tiny insect flight, The Journal of Experimental Biology 207 (2004) 3073–3088.
[18] L.A. Miller, C.S. Peskin, A computational fluid dynamics of 'clap and fling' in the smallest insects, The Journal of Experimental Biology 208 (2005) 195–212.
[19] Y. Mori, C.S. Peskin, A second order implicit Immersed Boundary method with boundary mass, Journal of Computational Physics (2005) (preprint).
[20] C.S. Peskin, Numerical analysis of blood flow in the heart, Journal of Computational Physics 25 (1977) 220–252.
[21] C.S. Peskin, The immersed boundary method, Acta Numerica (2002) 1–39.
[22] C.S. Peskin, D.M. McQueen, Modeling prosthetic heart valves for numerical analysis of blood flow in the heart, Journal of Computational Physics 37 (1980) 113–132.
[23] C.S. Peskin, D.M. McQueen, A three-dimensional computational method for blood flow in the heart: I. Immersed elastic fibers in a viscous incompressible fluid, Journal of Computational Physics 81 (1989) 372–405.
[24] C.S. Peskin, B.F. Printz, Improved volume conservation in the computation of flows with immersed elastic boundaries, Journal of Computational Physics (1993) 33–46.
[25] A.M. Roma, C.S. Peskin, M.J. Berger, An adaptive version of the Immersed Boundary method, Journal of Computational Physics 153 (1999) 509–534.
[26] J.M. Stockie, B.R. Wetton, Analysis of stiffness in the immersed boundary method and implications for time-stepping schemes, Journal of Computational Physics 154 (1999) 41–64.
[27] C. Tu, C.S. Peskin, Stability and instability in the computation of flows with moving immersed boundaries: a comparison of three methods, SIAM Journal on Scientific and Statistical Computing 13 (1992) 1361–1376.
[28] E.H. Twizell, A.B. Gumel, M.A. Arigu, Second-order, L_0-stable methods for the heat equation with time-dependent boundary conditions, Advances in Computational Mathematics 6 (1996) 333–352.
[29] S. Xu, Z.J. Wang, The Immersed Interface Method for simulating the interaction of a fluid with moving objects, Journal of Computational Physics (preprint).
[30] L. Zhu, C.S. Peskin, Simulation of a flapping flexible filament in a flowing soap film by the immersed boundary method, Journal of Computational Physics 79 (2002) 452–468.