Available online at www.sciencedirect.com

ScienceDirect

Comput. Methods Appl. Mech. Engrg. 197 (2008) 2290–2304

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

A comparison of implicit solvers for the immersed boundary equations

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

[a] Department of Mathematics, University of Utah, Salt Lake City, UT 84112, USA
[b] Department of Bioengineering, University of Utah, Salt Lake City, UT 84112, USA
[c] Department of Mathematics, University of California, Davis, CA 95616, USA
[d] School of Computing, University of Utah, Salt Lake City, UT 84112, USA

Received 20 June 2007; received in revised form 26 November 2007; accepted 30 November 2007
Available online 15 December 2007

Abstract

Explicit time discretizations of the immersed boundary method are known to require small timesteps to maintain stability. A number of implicit methods have been introduced to alleviate this restriction to allow for a more efficient method, but many of these methods still have a stability restriction on the timestep. Furthermore, almost no comparisons have appeared in the literature of the relative computational costs of the implicit methods and the explicit method. A recent paper [E.P. Newren, A.L. Fogelson, R.D. Guy, R.M. Kirby, Unconditionally stable discretizations of the immersed boundary equations, J. Comput. Phys. 222 (2007) 702–719.] addressed the confusion over stability of immersed boundary discretizations. This paper identified the cause of instability in previous immersed boundary discretizations as lack of conservation of energy and introduced a new semi-implicit discretization proven to be unconditionally stable, *i.e., it has bounded discrete energy*. The current paper addresses the issue of the efficiency of the implicit solvers. Existing and new methods to solve implicit immersed boundary equations are described. Systematic comparisons of computational cost are presented for a number of these solution methods for our stable semi-implicit immersed boundary discretization and an explicit discretization for two distinct test problems. These comparisons show that two of the implicit methods are at least competitive with the explicit method on one test problem and outperform it on the other test problem in which the elastic stiffness of the boundary does not dictate the timescale of the fluid motion.

AMS subject classification: 65M06; 65N22; 74F10; 76D05; 76M20

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

1. Introduction

The immersed boundary (IB) method was introduced by Peskin in the early 1970s 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 [24]. Rather than generating a curve-fitting grid for both exterior and interior regions 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 those 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 [24,26,27], but

* Corresponding author. Address: Department of Mathematics, University of Utah, Salt Lake City, UT 84112, USA. Tel.: +1 801 581 8150; fax: +1 801 581 4148.
E-mail addresses: enewren@sandia.gov (E.P. Newren), fogelson@math.utah.edu (A.L. Fogelson), guy@math.ucdavis.edu (R.D. Guy), kirby@cs.utah.edu (R.M. Kirby).

0045-7825/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2007.11.030

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 [9,11], swimming of micro-organisms [9,10], biofilm processes [8], mechanical properties of cells [1], cochlear dynamics [3], and insect flight [18,19]. We refer the reader to [25] for a more extensive list of applications.

The immersed interface (II) method was developed by Leveque and Li to address the low order accuracy found in the IB method when applied to problems with sharp interfaces [16]. The II method differs from the IB method in the spatial discretization method of the singular forces appearing in the continuous equations of motion. While we do not address the spatial discretizations involved in the II method and instead focus on the IB method in this paper, we do present some discussion of that method since the two are closely related, in fact hybrids of the two exist, such as [15].

Explicit timestepping with the IB and II methods leads to a severe timestep restriction in order to maintain stability [9,16,25,29]. This time step restriction is typically much more stringent than one that would be imposed if explicit differencing of the advective or diffusive terms [6] were used. Much effort has been expended attempting to alleviate this severe restriction, including the development of various implicit and semi-implicit methods [4,9,16,17,20,30,31].

The use of implicit methods for solving the IB equations has met with very limited success. Until the recent results in [22], no implicit IB methods were known to be unconditionally stable, and the observed instability even of some of the fully implicit methods was not well understood. There has also been an almost complete lack of computational comparisons of implicit methods with the explicit method. In fact, despite the many papers introducing implicit methods for solving the IB equations, very few of them have done any concrete comparisons of the computational cost of their implicit methods to the explicit method; two of these papers [22,30] have stated that their implicit method was slower than the explicit method, while others have simply overlooked comparing their implicit method with the explicit method in terms of CPU time. The only works of which we know to concretely compare computational cost were that of Stockie and Wetton [29], whose main focus was an analysis of IB stability, and the work of Mori and Peskin [20] found in this issue.

The issue of stability of implicit discretizations of the IB equations was addressed in [22], where the authors showed that previously suspected and asserted causes of numerical instability for the IB method were not the actual sources of instability and identified the cause of instability in previous implicit IB discretizations as a lack of conservation of energy of the numerical discretization. In [22], a new semi-implicit discretization which was proven to be unconditionally stable in the sense that a natural discrete energy was bounded. An intriguing consequence of that work is that linear solvers can be used on stable IB equations. Prior to the results of [22], it was commonly believed that only fully implicit discretizations could produce an unconditionally stable immersed boundary method. Fully implicit discretizations lead to systems of equations that are nonlinear in the IB point locations, and this nonlinearity reduces the range of applicable solvers.

In this paper, we seek to address the efficiency of implicit solvers for the IB method. In particular, we look at methods that take advantage of the ability to use linear solvers afforded to us by the new stable semi-implicit discretization of [22]. Since many implicit solvers have already been introduced in the literature, we begin by cataloguing these methods and discussing their effectiveness and applicability. We also introduce several new methods that exploit the linearity of our stable implicit equations, and then compare their computational cost with that of an explicit method.

In Section 2, we review the immersed boundary equations of motion and their stable discretization. In Section 3, we catalog and discuss existing and new approaches to solving implicit IB equations, and in Section 4, we present detailed comparisons of the relative computational efficiency of some of these implicit solvers and the explicit method.

2. The immersed boundary method

In the IB method, an Eulerian description is used for the fluid variables, 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 letters are used for Eulerian state variables, while uppercase letters are used for Lagrangian variables. Thus, $\mathbf{X}(s,t)$ is a vector

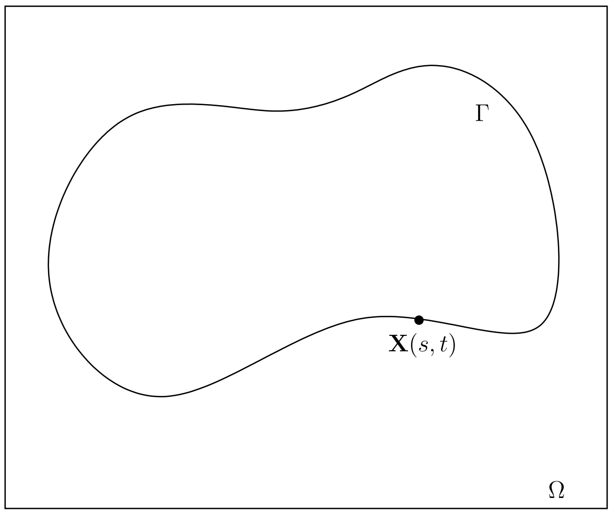

Fig. 1. Example immersed boundary curve, Γ, described by the function $\mathbf{X}(s,t)$, immersed in a fluid-filled region $\mathbf{\Omega}$.

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 force, 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:

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

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

$$\rho(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}) = -\nabla p + \mu\Delta\mathbf{u} + \mathbf{f}, \tag{3}$$

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

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

Eq. (1) is the constitutive law that specifies the force generated by the elastic object in terms of its current configuration. The force generation operator, $\mathcal{A}_{\mathrm{f}}$, is problem dependent. Note that the terminology "force generation operator" might be confusing 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 (in $\mathbf{X}$) as well as singular because the line integral does not remove the singularity of the multi-dimensional δ-function. The Lagrangian and Eulerian force and velocity are related through Eqs. (2) and (5). Eqs. (3) and (4) are the incompressible Navier–Stokes equations. In Eq. (3) we assume that the density, ρ, and viscosity, μ, are constant.

The constitutive law used for the force generation operator in this problem is the most common one in the IB literature. We assume that the material behaves like a fiber under elastic tension, so that the force it generates (per unit s) is given by

$$\mathbf{F}(s,t) = \frac{\partial}{\partial s}(T(s,t)\boldsymbol{\tau}(s,t)), \tag{6}$$

where $T(s,t)$ is the tension and $\boldsymbol{\tau}(s,t)$ is the tangent vector to the boundary at the point $\mathbf{X}(s,t)$ (see [27] for a derivation). The tangent vector is

$$\boldsymbol{\tau}(s,t) = \frac{\partial\mathbf{X}}{\partial s}\Big/\left\|\frac{\partial\mathbf{X}}{\partial s}\right\|. \tag{7}$$

If the reference configuration is assumed to represent an unstressed configuration, then $\left\|\frac{\partial\mathbf{X}}{\partial s}\right\| - 1$ represents the strain. If we assume a Hooke's law material so that the tension is proportional to the strain, then

$$T(s,t) = \gamma\left(\left\|\frac{\partial\mathbf{X}}{\partial s}\right\| - 1\right). \tag{8}$$

If we instead assume that the boundary is linearly elastic with zero resting length then the tension becomes

$$T(s,t) = \gamma\left(\left\|\frac{\partial\mathbf{X}}{\partial s}\right\|\right). \tag{9}$$

In this latter case, $\mathbf{F}(x,t) = \gamma\frac{\partial^2\mathbf{X}}{\partial s^2}$, and

$$\mathcal{A}_{\mathrm{f}} = \gamma\frac{\partial^2}{\partial s^2}. \tag{10}$$

This is the force generation operator we use in our tests.

In discretizing (1)–(5), time is divided into steps of size Δt. We denote by $\mathbf{u}^n(\mathbf{x})$ our approximation to the actual velocity vector $\mathbf{u}(\mathbf{x},n\Delta t)$ at time $n\Delta t$, and similarly denote by $\mathbf{X}^n(s)$ our approximation to $\mathbf{X}(s,n\Delta t)$. We define $\mathcal{S}_m$ and $\mathcal{S}_m^*$ through the formulas

$$\mathcal{S}_m(\mathbf{F}) = \int_\Gamma \mathbf{F}(s,t)\delta(\mathbf{x}-\mathbf{X}^m(s))\,\mathrm{d}s \tag{11}$$

and

$$\mathcal{S}_m^*(\mathbf{u}) = \int_\Omega \mathbf{u}(\mathbf{x},t)\delta(\mathbf{x}-\mathbf{X}^m(s))\mathrm{d}\mathbf{x}, \tag{12}$$

and write the temporal discretization of (1)–(5) derived in [22] as

$$\rho\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t} + \nabla p^{n+\frac{1}{2}} = -\rho[\mathbf{u}\cdot\nabla\mathbf{u}]^{n+\frac{1}{2}} + \frac{\mu}{2}\Delta(\mathbf{u}^{n+1}+\mathbf{u}^n) + \mathcal{S}_n\mathcal{A}_{\mathrm{f}}\left(\frac{1}{2}(\mathbf{X}^n+\mathbf{X}^{n+1})\right), \tag{13}$$

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

$$\frac{\mathbf{X}^{n+1}-\mathbf{X}^n}{\Delta t} = \mathcal{S}_n^*\left(\frac{1}{2}(\mathbf{u}^n+\mathbf{u}^{n+1})\right). \tag{15}$$

There are several choices for the discretization of $[\mathbf{u}\cdot\nabla\mathbf{u}]^{n+\frac{1}{2}}$; we use a simple first order upwind discretization in this paper, which we specify later in this section.

Due to the time lagging of our $\mathcal{S}$ and $\mathcal{S}^*$ operators, as well as the first order upwind discretization of the advection terms, this method is only first order accurate. We could make the temporal discretization formally second order accurate by employing a two-step approach, as outlined in [22]. However, most current immersed boundary implementations use a forward Euler discretization of the $\mathcal{S}$ and $\mathcal{S}^*$ terms, and we are aiming to provide a simple method that other researchers could easily adopt to avoid the "explosive" instabilities found with traditional explicit discretizations of the immersed boundary terms. Our mix of first and second order approximations also conserves energy (with viscosity $\mu = 0$) more accurately than a backward Euler discretization [22].

An attractive feature of the traditional IB method is its modularity, in particular, the freedom to use a standard Navier–Stokes solver code to update the fluid velocities. Such modularity is obtained by using a modified version of Eq. (15) that has an explicit discretization of the immersed boundary forces, such as one would get by replacing $\mathbf{X}^{n+1}$ by $\mathbf{X}^n$ in Eq. (15). With such an explicit discretization of the immersed boundary forces, the Navier–Stokes solver can be provided the fluid force density $\mathbf{f}$ (representing the explicitly evaluated immersed boundary terms) as an input and needs no other information about the immersed boundaries or their properties. In using implicit time-stepping, this nice feature may be lost, depending on how the implicit equations are solved. This issue is discussed further in Section 3.2.

The spatial discretization of these equations uses a cell-centered Cartesian grid for the Eulerian variables, and one or more discrete Lagrangian grids for the IB variables. The expression $\mathbf{u}_{ij}$ denotes the value of the variable $\mathbf{u}$ at the ijth gridpoint of the Eulerian grid. The expression $\mathbf{F}_k$ denotes the value of the Lagrangian variable $\mathbf{F}$ at the kth gridpoint. In particular, the location of the kth IB is $\mathbf{X}_k(t)$.

The interaction between these grids, governed by integration against a delta function in the continuous Eqs. (2) and (5), is handled by introducing a regularized discrete delta function whose support is comparable to the mesh spacing. The discrete delta function is derived from the requirement that a certain set of properties be satisfied; these include, ensuring that the entire force is transmitted to the grid, that the force density on the grid is a continuous function of the IB point locations, and that the communication between Eulerian and Lagrangian grids is very localized. Additional conditions can be imposed as well, yielding different delta functions; see [25] for a more thorough treatment of the possibilities. We use the delta function presented in [24],

$$\delta_h(x,y) = \delta_h(x)\delta_h(y), \tag{16}$$

$$\delta_h(x) = \begin{cases} \frac{1}{4h}\left(1+\cos\left(\frac{\pi x}{2h}\right)\right) & |x| \leqslant 2h, \\ 0 & |x| \geqslant 2h, \end{cases} \tag{17}$$

where h is the grid spacing for the Eulerian grid. This regularized delta function is shown in Fig. 2, and is used to define discrete analogs of $\mathcal{S}$ and $\mathcal{S}^*$ through the equations

$$S_m(\mathbf{F})_{ij} = \sum_k \mathbf{F}_k(t)\delta_h(\mathbf{x}_{ij} - \mathbf{X}_k^m)\Delta s, \tag{18}$$

and

$$S_m^*(\mathbf{u})_k = \sum_{ij} \mathbf{u}_{ij}(t)\delta_h(\mathbf{x}_{ij} - \mathbf{X}_k^m)h^2. \tag{19}$$

We also introduce discrete analogs of ∇, $\nabla\cdot$, Δ, $\mathbf{u}\cdot\nabla$, and $\mathcal{A}_\mathrm{f}$, denoted by ∇^h, $\nabla^h\cdot$, Δ^h, $(\mathbf{u}\cdot\nabla^h)$, and A_f, respectively. In our computations, we use centered difference approximations for the gradient, divergence, and Laplacian operators,

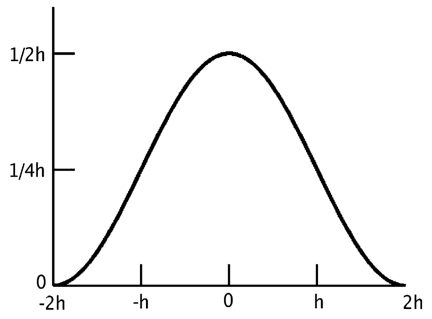

Fig. 2. Discrete delta function.

and an upwind difference approximation for the advection terms:

$$(\nabla^h\cdot\mathbf{u})_{ij} = \frac{u_{i+1,j}-u_{i-1,j}}{2h} + \frac{v_{i,j+1}-v_{i,j-1}}{2h}, \tag{20}$$

$$(\nabla^h p)_{ij} = \left(\frac{p_{i+1,j}-p_{i-1,j}}{2h}, \frac{p_{i,j+1}-p_{i,j-1}}{2h}\right), \tag{21}$$

$$(\Delta^h_{\mathrm{wide}}p)_{ij} = \frac{p_{i+2,j}-2p_{i,j}+p_{i-2,j}}{4h^2} + \frac{p_{i,j+2}-2p_{i,j}+p_{i,j-2}}{4h^2}, \tag{22}$$

$$\left(\Delta^h_{\mathrm{tight}}p\right)_{ij} = \frac{p_{i+1,j}-2p_{i,j}+p_{i-1,j}}{h^2} + \frac{p_{i,j+1}-2p_{i,j}+p_{i,j-1}}{h^2}, \tag{23}$$

$$\begin{aligned}((\mathbf{u}\cdot\nabla^h)c)_{ij} = {} & H(-u_{ij})u_{ij}\frac{c_{i+1,j}-c_{ij}}{h} + H(u_{ij})u_{ij}\frac{c_{ij}-c_{i-1,j}}{h} \\ & + H(-v_{ij})v_{ij}\frac{c_{i,j+1}-c_{ij}}{h} \\ & + H(v_{ij})v_{ij}\frac{c_{ij}-c_{i,j-1}}{h},\end{aligned} \tag{24}$$

where $\Delta^h_{\mathrm{tight}}$ is used for evaluation of the viscous terms, $\Delta^h_{\mathrm{wide}} = \nabla^h\cdot\nabla^h$ is used for Poisson solves in methods employing projections, and $H(x)$ is the Heaviside step function, $H(x)=1$ for $x>0$, $H(x)=0$ for $x\leqslant 0$.

The most straightforward discretization of Eq. (1) is to write the force at an immersed boundary point as a difference in the tensions on either side of that point. Assuming a single closed boundary with no external links, this can be written as

$$\mathbf{F}_k = \frac{(T_{k+1/2}(t)\boldsymbol{\tau}_{k+1/2}(t)) - (T_{k-1/2}(t)\boldsymbol{\tau}_{k-1/2}(t))}{\Delta s} \tag{25}$$

$$\begin{aligned} &= \frac{\left(\frac{\gamma}{\Delta s}(\|\mathbf{X}_{k+1}(t)-\mathbf{X}_k(t)\|-\ell_0)\frac{\mathbf{X}_{k+1}(t)-\mathbf{X}_k(t)}{\|\mathbf{X}_{k+1}(t)-\mathbf{X}_k(t)\|}\right)}{\Delta s} \\ &\quad - \frac{\left(\frac{\gamma}{\Delta s}(\|\mathbf{X}_{k-1}(t)-\mathbf{X}_k(t)\|-\ell_0)\frac{\mathbf{X}_{k-1}(t)-\mathbf{X}_k(t)}{\|\mathbf{X}_{k-1}(t)-\mathbf{X}_k(t)\|}\right)}{\Delta s},\end{aligned} \tag{26}$$

where ℓ_0 is the resting length of the "springs" connecting IB points. The reason for calling the connection between IB points a "spring" in the discrete set of equations is because of the form of (26): $\gamma/\Delta s$ serves as a spring constant, $\|\mathbf{X}_i-\mathbf{X}_k\|-\ell_0$ is the length by which the connection between IB points i and k has been stretched, and $(\mathbf{X}_i-\mathbf{X}_k)/\|\mathbf{X}_i-\mathbf{X}_k\|$ is a unit vector in the direction of the connection, making this look just like a Hookean spring.

Noting the similarity in the two terms of (26), we can instead write the force as a sum over the set $\mathcal{S}_k$ of IB points connected to IB point k:

$$\mathbf{F}_k = \sum_{i\in\mathcal{S}_k}\frac{\gamma}{\Delta s}(\|\mathbf{X}_i(t)-\mathbf{X}_k(t)\|-\ell_0)\frac{1}{\Delta s}\frac{\mathbf{X}_i(t)-\mathbf{X}_k(t)}{\|\mathbf{X}_i(t)-\mathbf{X}_k(t)\|}. \tag{27}$$

An additional advantage of writing in this manner is that it also makes clear how to handle external links connecting objects. The force $\mathbf{F} = A_\mathrm{f}\mathbf{X}$ has k^{th} element given by Eq. (27), and is a linear function of $\mathbf{X}$ if $\ell_0 = 0$.

Combining the temporal and spatial discretizations, the discrete equations of motion are

$$\rho\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\Delta t}+\nabla^{h}p^{n+\frac{1}{2}}=-\rho[(\mathbf{u}\cdot\nabla^{h})\mathbf{u}]^{n+\frac{1}{2}}+\frac{\mu}{2}\Delta^{h}(\mathbf{u}^{n+1}+\mathbf{u}^{n})+SA_{\mathrm{f}}\left(\frac{1}{2}(\mathbf{X}^{n}+\mathbf{X}^{n+1})\right), \tag{28}$$

$$\nabla^{h}\cdot\mathbf{u}^{n+1}=0, \tag{29}$$

$$\frac{\mathbf{X}^{n+1}-\mathbf{X}^{n}}{\Delta t}=S^{*}\left(\frac{1}{2}(\mathbf{u}^{n}+\mathbf{u}^{n+1})\right). \tag{30}$$

Provided that S and S^{*} are evaluated at the same point in time, this discretization was proved in [22] to be unconditionally stable if the advection term $\mathbf{u}\cdot\nabla\mathbf{u}$ is not present, and was demonstrated through computational tests to be stable with the advection terms if the CFL constraint $\|\mathbf{u}\|_{\infty}\Delta t\leqslant h$ was satisfied.

3. Solver possibilities

There are three variables to determine in Eqs. (28)–(30): $\mathbf{u}^{n+1}$, $\nabla^{h}p^{n+\frac{1}{2}}$, and $\mathbf{X}^{n+1}$. If A_{f} is derived from a linear constitutive law (or we lag its evaluation in time), and we lag the evaluation of S and S^{*} in time, then we obtain a linear, semi-implicit system.

Most existing implicit methods for solving the immersed boundary equations use fully implicit discretizations, due to a common belief (before the results of [22]) that time-lagged operators would contribute to the severe instability observed in immersed boundary implementations. With the use of fully implicit discretizations, a number of linear solvers are not directly applicable. In this paper, we exploit the ability to maintain an unconditionally bounded solution with a linear, semi-implicit scheme to explore additional solver possibilities not previously considered.

In this section, we highlight several possible methods for solving the system of Eqs. (28)–(30) derived in Section 2. We briefly compare methods in this section, noting results from the literature and our experience, and perform a more detailed computational comparison of the methods which take advantage of the linearity of our semi-implicit discretization in Section 4.

3.1. Direct application of linear solvers

Perhaps the most obvious strategy for solving our semi-implicit discretization of Eqs. (28)–(30) is to apply linear system solvers directly to the full system. If A_{f} is derived from a linear constitutive law (or we lag its evaluation in time), and we lag the evaluation of S and S^{*} in time, then $SA_{\mathrm{f}}\left(\frac{1}{2}(\mathbf{X}^{n}+\mathbf{X}^{n+1})\right)=\frac{1}{2}SA_{\mathrm{f}}\mathbf{X}^{n}+\frac{1}{2}SA_{\mathrm{f}}\mathbf{X}^{n+1}$. Making use of this and assuming that the advection terms are handled explicitly, we can write the equations in the form of a single matrix equation:

$$\begin{pmatrix}(\rho I-\frac{\mu\Delta t}{2}\Delta^{h}) & \Delta t\nabla^{h} & -\frac{\Delta t}{2}SA_{\mathrm{f}}\\ \nabla^{h}\cdot & 0 & 0\\ -\frac{\Delta t}{2}S^{*} & 0 & I\end{pmatrix}\begin{pmatrix}\mathbf{u}^{n+1}\\ p^{n+\frac{1}{2}}\\ \mathbf{X}^{n+1}\end{pmatrix}=\begin{pmatrix}(\rho I+\frac{\mu\Delta t}{2}\Delta^{h}) & 0 & \frac{\Delta t}{2}SA_{\mathrm{f}}\\ 0 & 0 & 0\\ \frac{\Delta t}{2}S^{*} & 0 & I\end{pmatrix}\begin{pmatrix}\mathbf{u}^{n}\\ p^{n-1/2}\\ \mathbf{X}^{n}\end{pmatrix}+\begin{pmatrix}-\rho[(\mathbf{u}\cdot\nabla^{h})\mathbf{u}]^{n+\frac{1}{2}}\\ 0\\ 0\end{pmatrix}. \tag{31}$$

Alternatively, this equation can be obtained (with an explicit handling of the advection terms) by using $\frac{1}{2}S_{n}A_{\mathrm{f}}\mathbf{X}^{n}+\frac{1}{2}S_{n+1}A_{\mathrm{f}}\mathbf{X}^{n+1}$ as the temporal discretization of $\mathcal{S}A_{\mathrm{f}}\mathbf{X}$ and using a similar discretization of $\mathcal{S}^{*}\mathbf{u}$. However, this modified discretization results in a nonlinear system (because S_{n+1} and S^{*}_{n+1} depend on $\mathbf{X}^{n+1}$), and it does not conserve energy and can lead to instabilities if the timestep is not sufficiently small.

Given the assumptions above, Eq. (31) is a system to which we can apply linear solvers. However, this system poses a number of difficulties. The system is singular since the gradient operator has a nontrivial nullspace. Direct matrix factorization methods such as LU decomposition would result in a huge amount of fill-in for an otherwise very sparse system, and methods like GMRES may fail to converge without a good preconditioner (and it is not clear what would be a good preconditioner for this system). These potential drawbacks make this strategy unpromising, and we are not aware of any work to explore these options.

3.2. Newton-like methods

Before [22], it was commonly believed that fully-implicit discretizations were necessary to overcome the severe stability restrictions found with explicit discretizations of the immersed boundary method. Because of this, many implementations were based on fully implicit discretizations. These discretizations require the use of spreading and interpolation operators, S and S^{*}, that are not lagged in time, and as a result, the equations are nonlinear in $\mathbf{X}^{n+1}$. Thus, for such systems, a nonlinear solver is needed.

Leveque and Li [16] introduced a Newton-like method that handles the nonlinearity in $\mathbf{X}$ and additionally provides for increased modularity by allowing Navier–Stokes codes to be used unmodified as a subsolver. Their method was based on rearranging Eq. (30) to obtain

$$\mathbf{g}(\mathbf{X}^{n+1})=\mathbf{X}^{n+1}-\mathbf{X}^{n}-\frac{\Delta t}{2}S^{*}(\mathbf{u}^{n}+\mathbf{u}^{n+1}(\mathbf{X}^{n+1}))=0, \tag{32}$$

where the notation $\mathbf{u}^{n+1}(\mathbf{X}^{n+1})$ is used to make it clear that each evaluation of $\mathbf{g}$ involves solving the Navier–

Stokes equations. This equation can then be solved with Newton's method or a quasi-Newton method. Tu and Peskin also proposed a similar method [30], though it was restricted to the case of steady Stokes flow and written in a way that was tightly coupled to a specialized Stokes flow solver.

3.2.1. Newton's method

Solving Eq. (32) with Newton's method involves choosing an initial guess for the new immersed boundary point positions, such as $\mathbf{X}^{n+1,0} = \mathbf{X}^n$, and then iterating over the steps

- Find $J = \mathbf{g}'(\mathbf{X}^{n+1,m})$ via formula $J_{ij} = \frac{\mathbf{g}_i(\mathbf{X}^{n+1,m}+\beta\mathbf{e}_j)-\mathbf{g}_i(\mathbf{X}^{n+1,m}-\beta\mathbf{e}_j)}{2\beta}$
- Solve $J\mathbf{s} = -\mathbf{g}(\mathbf{X}^{n+1,m})$ for $\mathbf{s}$
- Set $\mathbf{X}^{n+1,m+1} = \mathbf{X}^{n+1,m} + \mathbf{s}$.

In these steps, $\mathbf{e}_j$ is the vector with ith component equal to $\delta_{ij}^{\text{Kroenecker}}$, and β is some small parameter (e.g., $\sqrt{\epsilon_{\text{machine}}}$). Here, $\mathbf{X}^{n+1,m}$, $\mathbf{s}$, and $\mathbf{e}_j$ are vectors with $2N_B$ components and J is a $2N_B \times 2N_B$ matrix.

Newton's method has been used in both [22,30], and was reported to be very computationally inefficient. The reason for this observed inefficiency is that computing the Jacobian at each iteration involves $4N_B$ fluid solves. By using one-sided differences to compute the Jacobian, this cost could be cut in half, but it will still be orders of magnitude slower than the explicit method.

3.2.2. Quasi-Newton methods

Solving Eq. (32) via a quasi-Newton method is very similar to using Newton's method. It requires having an initial guess for the new immersed boundary point positions, such as $\mathbf{X}^{n+1,0} = \mathbf{X}^n$, as well as an initial guess for the Jacobian, such as $J^{n+1,0} = J^n$ or $J^{n+1,0} = I$. The reason that $J^{n+1,0} = I$ may be a good choice is that

$$J(\mathbf{X}) \equiv g'(\mathbf{X}) = I - \frac{\Delta t}{2}\left(S^*\mathbf{u}^{n+1}\right)'(\mathbf{X}), \tag{33}$$

so $J \to I$ as $\Delta t \to 0$ (for stiff problems, Δt may need to be very small for I to be a good initial guess). Once the initial guesses are selected, quasi-Newton methods proceed by iterating over the steps

(1) Solve $J^{n+1,m}\mathbf{s} = -\mathbf{g}(\mathbf{X}^{n+1,m})$ for $\mathbf{s}$.
(2) Set $\mathbf{X}^{n+1,m+1} = \mathbf{X}^{n+1,m} + \mathbf{s}$.
(3) Set $\mathbf{y} = \mathbf{g}(\mathbf{X}^{n+1,m+1}) - \mathbf{g}(\mathbf{X}^{n+1,m})$.
(4) Use $\mathbf{s}$ and $\mathbf{y}$ to update $J^{n+1,m}$ to $J^{n+1,m+1}$.

There are many different ways to update the Jacobian in the final step of each iteration, and these correspond to different quasi-Newton methods. The only quasi-Newton method that has appeared in the IB or II literature is the BFGS version. Defining $H = (J^{n+1,m})^{-1}$, the BFGS update [7] is

$$(J^{n+1,m+1})^{-1} = H + \frac{(\mathbf{s} - H\mathbf{y})\mathbf{s}^{\mathrm{T}} + \mathbf{s}(\mathbf{s} - H\mathbf{y})^{\mathrm{T}}}{\mathbf{s}^{\mathrm{T}}\mathbf{y}} - \frac{(\mathbf{s} - H\mathbf{y})^{\mathrm{T}}\mathbf{y}\mathbf{s}\mathbf{s}^{\mathrm{T}}}{(\mathbf{s}^{\mathrm{T}}\mathbf{y})^2}. \tag{34}$$

This method is inexpensive in comparison to the Newton method as it only involves one fluid solve per iteration (note that the computation of $\mathbf{g}(\mathbf{X}^{n+1,m+1})$ in Step 3 of each iteration can be reused in Steps 1 and 3 of the following iteration). In fact, the most expensive part of this iteration may be the dense $2N_B \times 2N_B$ matrix multiply in Step 1 (BFGS stores approximations to J^{-1} rather than J), since this step has a computational cost that grows quadratically with the number of immersed boundary points N_B. Further, J becomes increasingly ill-conditioned both as N_B grows and as the immersed boundary points become closer together [16] because of grid refinement. However, for some problems N_B may be small enough relative to the number of Eulerian gridpoints, N, that this method may be reasonable.

The quasi-Newton approach has only been used in II implementations, possibly due to the fact that the representation of the boundary commonly used with the immersed interface method allows use of fewer boundary points, so that N_B can be significantly smaller than in IB calculations. This method was first used by Leveque and Li [16], and has been adopted by others using the II method; see also [14,15].

The BFGS method is designed for problems in which the Jacobian is symmetric and positive definite, and it only produces approximations to the Jacobian with those attributes [7]. We tested a hybrid IB/II method similar to Lee and Leveque's [15] with a value of Δt near the stability limit of the explicit method. We found that the skew-symmetric part of the Jacobian was as large in norm as the symmetric part, and that the Jacobian was in fact indefinite with at least one eigenvalue having very large negative real part. However, it should be noted that BFGS has been used successfully on II problems. Still the observations above suggest that further research into the choice of quasi-Newton method for IB/II calculations may be beneficial. This is not the focus of the current paper. Instead we concentrate on methods which exploit the linearity of our stable semi-implicit discretization.

3.3. Schur complement systems

Another strategy for solving the implicit IB equations involves eliminating one or more of the unknown variables; that is, it involves deriving Schur complement equations for one or more of the unknowns.

In this section we will occasionally assume commutativity of Δ^h and $\mathbb{P}$, where $\mathbb{P} = (I - \nabla^h(\nabla^h \cdot \nabla^h)^{-1}\nabla^h\cdot)$, the operator that projects orthogonally in $L^2(\Omega^h)$ onto the space of discretely divergence-free vector fields. While this assumption only holds for periodic boundary conditions, the assumption is not necessary and is made only for convenience of derivation. The same methods can be derived without assuming commutativity. In particular, rather than starting with the discrete set of equations and applying a number of operators to arrive at our method, we could instead apply some of the corresponding continuous operators to the continuous set of equations (where the projection operator and Laplacian do commute), then discretize, then apply any remaining discrete operators needed.

Recall that the system of equations that we want to solve, Eqs. (28)–(30), are

$$\rho\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t} = -\nabla^h p^{n+\frac{1}{2}} - \rho[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}} + \frac{\mu}{2}\Delta^h(\mathbf{u}^{n+1}+\mathbf{u}^n) + \frac{1}{2}SA_\mathrm{f}(\mathbf{X}^n+\mathbf{X}^{n+1}), \tag{35}$$

$$\nabla^h\cdot\mathbf{u}^{n+1} = 0, \tag{36}$$

$$\frac{\mathbf{X}^{n+1}-\mathbf{X}^n}{\Delta t} = \frac{1}{2}S^*(\mathbf{u}^n+\mathbf{u}^{n+1}). \tag{37}$$

There are multiple ways in which to eliminate one or more of $\mathbf{X}^{n+1}$, $\mathbf{u}^{n+1}$, or $\nabla^h p^{n+\frac{1}{2}}$. We derive a number of these in this section.

We begin by eliminating $\nabla^h p^{n+\frac{1}{2}}$ from Eqs. (35)–(37). We do this by applying $\mathbb{P}$ to each term of Eq. (35). This eliminates $\nabla^h p^{n+\frac{1}{2}}$ as well as the need for the separate incompressibility constraint equation and yields

$$\rho\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t} = -\rho\mathbb{P}[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}} + \frac{\mu}{2}\mathbb{P}\Delta^h(\mathbf{u}^{n+1}+\mathbf{u}^n) + \frac{1}{2}\mathbb{P}SA_\mathrm{f}(\mathbf{X}^n+\mathbf{X}^{n+1}), \tag{38}$$

$$\frac{\mathbf{X}^{n+1}-\mathbf{X}^n}{\Delta t} = \frac{1}{2}S^*(\mathbf{u}^n+\mathbf{u}^{n+1}). \tag{39}$$

Others who have used this particular Schur complement idea [17,20,29] have applied the projection before discretizing spatially. When the equations are still spatially continuous (or if we assume periodic boundary conditions), the operators $\mathbb{P}$ and Δ in Eq. (38) commute. Further, since $\mathbf{u}^{n+1}$ and $\mathbf{u}^n$ are already divergence free, we can use $\mathbf{u}^{n+1} = \mathbb{P}\mathbf{u}^n$ and $\mathbf{u}^n = \mathbb{P}\mathbf{u}^n$ in Eq. (38) to eliminate $\mathbb{P}$ from the viscous term. Doing the aforementioned and solving for $\mathbf{u}^{n+1}$ in (38) and for $\mathbf{X}^{n+1}$ in (39), we obtain

$$\mathbf{u}^{n+1} = M\left(I+\frac{\nu\Delta t}{2}\Delta^h\right)\mathbf{u}^n - \Delta t M\mathbb{P}[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}} + \frac{\Delta t}{2\rho}M\mathbb{P}SA_\mathrm{f}(\mathbf{X}^n+\mathbf{X}^{n+1}), \tag{40}$$

$$\mathbf{X}^{n+1} = \mathbf{X}^n + \frac{\Delta t}{2}S^*(\mathbf{u}^n+\mathbf{u}^{n+1}), \tag{41}$$

where $M = (I-\frac{\nu\Delta t}{2}\Delta^h)^{-1}$.

From here we can eliminate either $\mathbf{u}^{n+1}$ or $\mathbf{X}^{n+1}$. Plugging the value of $\mathbf{u}^{n+1}$ into Eq. (41) and collecting $\mathbf{X}^{n+1}$ terms to the left hand side (again making use of linearity of A_f and time lagging in S and S^*) gives

$$\left(I-\frac{\Delta t^2}{4\rho}S^*M\mathbb{P}SA_\mathrm{f}\right)\mathbf{X}^{n+1} = \left(I+\frac{\Delta t^2}{4\rho}S^*M\mathbb{P}SA_\mathrm{f}\right)\mathbf{X}^n + \frac{\Delta t}{2}S^*\left(\mathbf{u}^n + M\left(I+\frac{\nu\Delta t}{2}\Delta^h\right)\mathbf{u}^n - \Delta t M\mathbb{P}[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}}\right). \tag{42}$$

Eq. (42) is equivalent to (32) given our linearity and lagging assumptions on A_f, S, and S^*; however, the solver suggested by the form of the equations is different. Note that everything on the right hand side of (42) can be calculated explicitly. Similarly, when we substitute the value of $\mathbf{X}^{n+1}$ from (41) into Eq. (40) and collect $\mathbf{u}^{n+1}$ terms to the left hand side, we obtain

$$\left(I-\frac{\Delta t^2}{4\rho}M\mathbb{P}SA_\mathrm{f}S^*\right)\mathbf{u}^{n+1} = \left(M\left(I+\frac{\nu\Delta t}{2}\Delta^h\right)+\frac{\Delta t^2}{4\rho}M\mathbb{P}SA_\mathrm{f}S^*\right)\mathbf{u}^n - \Delta t M\mathbb{P}[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}} + \frac{\Delta t}{\rho}M\mathbb{P}SA_\mathrm{f}\mathbf{X}^n, \tag{43}$$

where again everything on the right hand side of (43) can be calculated explicitly.

Eqs. (42) and (43) give us two possible avenues of attack, which we will discuss later. We can come up with others by beginning again with Eqs. (35)–(37). If we solve Eq. (37) for $\mathbf{X}^{n+1}$, substitute it into Eq. (35) to eliminate $\mathbf{X}^{n+1}$, and then collect terms which include $\mathbf{u}^{n+1}$ (again making use of linearity of A_f and time lagging in S and S^*), we obtain

$$\left(I-\frac{\nu\Delta t}{2}\Delta^h-\frac{\Delta t^2}{4\rho}SA_\mathrm{f}S^*\right)\mathbf{u}^{n+1} = \left(I+\frac{\nu\Delta t}{2}\Delta^h+\frac{\Delta t^2}{4\rho}SA_\mathrm{f}S^*\right)\mathbf{u}^n - \frac{\Delta t}{\rho}\nabla^h p^{n+\frac{1}{2}} - \Delta t[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}} + \frac{\Delta t}{\rho}SA_\mathrm{f}\mathbf{X}^n, \tag{44}$$

$$\nabla^h\cdot\mathbf{u}^{n+1} = 0. \tag{45}$$

Note that multiplying both sides of (44) by $M\mathbb{P}$ yields Equation (43) (modulo one instance of commutativity of $\mathbb{P}$ and Δ^h).

Eq. (44) has some interesting properties. Defining $A_\mathrm{NS} = (I-\frac{\nu\Delta t}{2}\Delta^h)$, $A_\mathrm{elastic} = -\frac{\Delta t^2}{4\rho}SA_\mathrm{f}S^*$, and $A_\mathrm{IB} = A_\mathrm{NS} + A_\mathrm{elastic}$, Eq. (44) simplifies to

$$A_\mathrm{IB}\mathbf{u}^{n+1} = (2I-A_\mathrm{IB})\mathbf{u}^n - \frac{\Delta t}{\rho}\nabla^h p^{n+\frac{1}{2}} - \Delta t[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}} + \frac{\Delta t}{\rho}SA_\mathrm{f}\mathbf{X}^n. \tag{46}$$

If we assume A_f is self-adjoint and negative semi-definite (which was an assumption used in the stability proof of Newren et al. [22], and is true for our choice of A_f), then the adjointness of S and S^* imply that A_elastic is symmetric and positive semi-definite. Combined with the fact that

Δ^h is symmetric and negative semi-definite, we see that A_{IB} is symmetric and positive definite, with all of its eigenvalues greater than or equal to 1. Additionally, $\lim_{\Delta t\to 0} A_{\mathrm{IB}} = I$.

We can proceed further from (45) and (46) by eliminating either $p^{n+\frac{1}{2}}$ or $\mathbf{u}^{n+1}$. By applying $\frac{\rho}{\Delta t}\nabla^h \cdot A_{\mathrm{IB}}^{-1}$ to every term of (46) and making use of (45), we obtain an equation for $p^{n+\frac{1}{2}}$

$$\nabla^h \cdot A_{\mathrm{IB}}^{-1}\nabla^h p^{n+\frac{1}{2}} = \nabla^h \cdot A_{\mathrm{IB}}^{-1}\left[\frac{\rho}{\Delta t}(2I - A_{\mathrm{IB}})\mathbf{u}^n - \rho[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}} + SA_{\mathrm{f}}\mathbf{X}^n\right]. \tag{47}$$

Alternatively, if we apply $\mathbb{P}$ to every term of (46) and substitute $\mathbf{u}^{n+1} = \mathbb{P}\mathbf{u}^{n+1}$ (which is valid due to Eq. (45)) we obtain

$$\mathbb{P}A_{\mathrm{IB}}\mathbb{P}\mathbf{u}^{n+1} = \mathbb{P}\left[(2I - A_{\mathrm{IB}})\mathbf{u}^n - \Delta t[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}} + \frac{\Delta t}{\rho}SA_{\mathrm{f}}\mathbf{X}^n\right]. \tag{48}$$

Since $\mathbb{P}$ is self-adjoint, $\mathbb{P}A_{\mathrm{IB}}\mathbb{P}$ is symmetric positive semi-definite (positive definite on the subspace of divergence-free vector fields). Although Eq. (48) is equivalent to (43) — it can be obtained by applying $\mathbb{P}M^{-1}$ to (43) and utilizing the facts that $\mathbb{P}\mathbf{u}^{n+1} = \mathbf{u}^{n+1}$ and $\mathbb{P}\mathbf{u}^n = \mathbf{u}^n$ — $(I - \frac{\Delta t^2}{4\rho}M\mathbb{P}SA_{\mathrm{f}}S^*)$ is not symmetric or positive definite. Thus, Eq. (48) may be the preferred form for applying linear solvers.

Eqs. (42), (43), (46), (45), (47) and (48) each provide possibilities on which to base solvers. We refer to these methods as the DSX ("Double Schur-complement $\mathbf{X}$"), nonsymmetric-DSU, projection-based, DSP, and DSU methods, respectively. Each will be discussed in subsequent sections.

3.3.1. Fixed point methods

Krylov subspace solvers are always superior to fixed point methods for linear problems, so using fixed point methods does not make sense for linear equations such as ours. However, fully implicit discretizations of the IB equations are nonlinear and fixed point methods have played a prominent role in the IB literature. In particular, the fixed point scheme of Mayo and Peskin [17] was proven to be unconditionally convergent and is the only nonlinear IB equation solver we know of to have such a quality. Other than Mayo and Peskin's fixed point scheme and the work of Mori and Peskin [20] (which we have not yet tested), all other iterative schemes for nonlinear IB equations we know of have either previously been reported in the literature to have occasional convergence issues or we have found them to have occasional convergence issues in our own testing. Because of the important role fixed point methods have played in implicit immersed boundary methods, we discuss them briefly in this section.

Eq. (42) is of the form

$$\left(I - \frac{\Delta t^2}{4\rho}S^*M\mathbb{P}SA_{\mathrm{f}}\right)\mathbf{X}^{n+1} = \mathbf{Z}^n, \tag{49}$$

where $\mathbf{Z}^n$ represents known quantities. We can rewrite this as a fixed point problem

$$\mathbf{X}^{n+1} = \frac{\Delta t^2}{4\rho}S^*M\mathbb{P}SA_{\mathrm{f}}\mathbf{X}^{n+1} + \mathbf{Z}^n. \tag{50}$$

Various iterative methods could be used to try to solve this fixed point problem, among the simplest being

$$\mathbf{X}^{n+1,m+1} = \frac{\Delta t^2}{4\rho}S^*M\mathbb{P}SA_{\mathrm{f}}\mathbf{X}^{n+1,m} + \mathbf{Z}^n \tag{51}$$

or equivalently,

$$\mathbf{X}^{n+1,m+1} - \mathbf{X}^{n+1,m} = \mathbf{Z}^n - \left(I - \frac{\Delta t^2}{4\rho}S^*M\mathbb{P}SA_{\mathrm{f}}\right)\mathbf{X}^{n+1,m}. \tag{52}$$

Modifying the left hand side slightly we get a system of the form

$$(I - \lambda A_{\mathrm{f}})(\mathbf{X}^{n+1,m+1} - \mathbf{X}^{n+1,m}) = \mathbf{Z}^n - \left(I - \frac{\Delta t^2}{4\rho}S^*M\mathbb{P}SA_{\mathrm{f}}\right)\mathbf{X}^{n+1,m}, \tag{53}$$

which solves the same fixed point equation but may converge faster. Here λ is an approximation to $\frac{\Delta t^2}{4\rho}S^*M\mathbb{P}S$.

A method of this form was first introduced by Mayo and Peskin [17]. They used different temporal and spatial discretizations than we have employed, but the form remains the same. In their work, they make λ a diagonal matrix:

$$\lambda = \frac{\Delta t^2}{4\rho}S^*S\mathbf{1}, \tag{54}$$

where $\mathbf{1}(s)$ is a function that takes the value 1 for every s, that is, $\mathbf{1}(s)$ is 1 at each IB point. (Technically, they lump $\Delta t^2/4\rho$ in with A_{f} rather than λ, but the product is equivalent either way.) They report no computational timings other than "[it] often converged slowly". They also explored Aitken extrapolation to try to accelerate convergence of the fixed point iterates, and later tried a Krylov solver directly on (49). Unfortunately, these other methods were not explored in detail, and the comparisons they provide among these methods were only cursory, but they did state that the Krylov solver method was the fastest when it converged. As such, we focus our effort on Krylov methods for solving (49). See Section 3.3.3 for more details on those methods.

The fixed point method of Mayo and Peskin was also used by Stockie and Wetton [29] in their analysis of explicit and implicit IB solvers. It also motivated a very similar fixed point method used by Roma et al. [28]. It should be noted that although Mayo and Peskin proved that their fixed point method would unconditionally converge to the solution of their nonlinear discrete system, their discretization itself was reported to be unstable [17,22,29].

3.3.2. Projection methods

Eqs. 46 and 45,

$$
\begin{aligned}
A_{\mathrm{IB}}\mathbf{u}^{n+1} &= (2I - A_{\mathrm{IB}})\mathbf{u}^n - \frac{\Delta t}{\rho}\nabla^h p^{n+\frac{1}{2}} - \Delta t[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}} \\
&\quad + \frac{\Delta t}{\rho} S A_{\mathrm{f}} \mathbf{X}^n, \\
\nabla^h \cdot \mathbf{u}^{n+1} &= 0,
\end{aligned} \tag{55}
$$

where $A_{\mathrm{IB}} = A_{\mathrm{NS}} + A_{\mathrm{elastic}} = (I - \frac{\nu\Delta t}{2}\Delta^h) + (-\frac{\Delta t^2}{4\rho} S A_{\mathrm{f}} S^*)$, look similar to the discretized Navier–Stokes equations,

$$
\begin{aligned}
A_{\mathrm{NS}}\mathbf{u}^{n+1} &= (2I - A_{\mathrm{NS}})\mathbf{u}^n - \frac{\Delta t}{\rho}\nabla^h p^{n+\frac{1}{2}} - \Delta t[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}}, \\
\nabla^h \cdot \mathbf{u}^{n+1} &= 0.
\end{aligned} \tag{56}
$$

Assuming the advection terms are calculated explicitly, both are of the form

$$
\begin{aligned}
A\mathbf{u} + \frac{\Delta t}{\rho}\nabla^h p &= \mathbf{r}, \\
\nabla^h \cdot \mathbf{u} &= 0,
\end{aligned} \tag{57}
$$

where $\mathbf{r}$ denotes known quantities from the previous timestep. Putting both sets of equations in such a similar form is made possible by our use of a semi-implicit discretization where the evaluation of the spreading and interpolation operators is lagged in time (and due to A_{f} being linear); it would not be possible for a stable fully implicit IB discretization.

Since projection methods are a common method used to solve the incompressible Navier–Stokes equations, and the IB equations can be posed in the nearly identical form (55), it seems natural to apply projection methods to the IB equations as well. This method has not been considered previously in the literature, likely due to the common belief (prior to the results of [22]) that fully implicit discretizations of the IB equations were necessary for unconditional stability.

A projection method approach to solving (57) was explored in depth in [21], and was found to be ineffective. There are four qualities of the Navier–Stokes equations that combine to make projection methods effective for their solution:

- A is relatively small, $\mathrm{O}(1)$, in norm,
- a good initial guess to the pressure is readily available,
- A is a small, $\mathrm{O}(\Delta t)$ perturbation of the identity, and
- A "almost" commutes with ∇^h.

All four properties become important due to the fact that a projection method is an operator splitting method with an associated splitting error. The first and third qualities are related, and ensure that the splitting error is smaller in magnitude than the solution. The second property is what enables incremental projection schemes to achieve an even lower splitting error than non-incremental (or "pressure-free") projection schemes. The fourth property results in the splitting error for "standard" projection schemes predominantly affecting the pressure gradient, with very little error occurring in the velocity field. The fourth property also allows modified projection schemes to be devised that remove most (or in some cases all) of the remaining splitting error. See [13] for more details about the classification of projection methods and analysis of their errors in the case of the Navier–Stokes equations.

All four qualities break down when considering the immersed boundary equations with "thin" interfaces (*i.e.*, for all cases not using the "thick" boundary approach of Griffith and Peskin [12]; three of the four break down when using "thick" boundaries). The appearance of delta functions in the IB equations results in discontinuities in the pressure when using "thin" interfaces. While these discontinuities are "smoothed out" by the use of discrete delta functions, using lagged pressures for initial guesses still results in large errors near the interface. Also, A_{elastic} is typically two or more orders of magnitude larger than A_{NS} in norm, making A_{IB} large in norm (and thus far from a perturbation of the identity). Finally, while A_{NS} "commutes" with ∇^h other than possibly at the boundary, A_{elastic} (the dominant part of A_{IB}) does not.

More details about the use of projection methods to solve the IB equations can be found in [21]. In that work, many different projection methods were studied: standard (incremental) pressure-correction projection methods, a natural extension of the rotational (and incremental) pressure-correction method (i.e. the standard pressure-correction method with the pressure update identified by Brown et al. [5]) to the IB equations, a new projection method based on an improved pressure update formula designed to reduce the splitting error, the generalizations of Perot [23], the different projection choices of Almgren et al. [2], and velocity-correction projection methods. That work investigated many aspects of projection methods: splitting errors, spectral radii of iterative versions, and their connection to fixed-point schemes, and concluded that the splitting error of these methods for the immersed boundary equations is too large for them to be feasible solution techniques.

3.3.3. Krylov solvers

The double Schur complement equations, (42), (43), (47), and (48) are all in the standard form $A\mathbf{x} = \mathbf{b}$ to which we can apply Krylov subspace methods. Once any of those four equations is solved, its solution can be used to solve for the other variables, typically also with a Krylov method. For example, if (47) is solved for $\nabla^h p^{n+\frac{1}{2}}$, then the solution can be substituted into (44) and the resulting equation solved for $\mathbf{u}^{n+1}$. Then, $\mathbf{u}^{n+1}$ can be substituted into (30) and $\mathbf{X}^{n+1}$ determined from the resulting equation.

The DSX ("Double Schur-complement **X**") equation, Eq. (42) or, equivalently (49) is

$$\left(I-\frac{\Delta t^2}{4\rho}S^*M\mathbb{P}SA_{\mathrm{f}}\right)\mathbf{X}^{n+1} = \left(I+\frac{\Delta t^2}{4\rho}S^*M\mathbb{P}SA_{\mathrm{f}}\right)\mathbf{X}^n + \frac{\Delta t}{2}S^*\left(\mathbf{u}^n + M\left(I+\frac{\nu\Delta t}{2}\Delta^h\right)\mathbf{u}^n - \Delta t M\mathbb{P}[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}}\right). \tag{58}$$

This was solved by Mayo and Peskin in [17] with a conjugate gradient squared solver, using $I-\lambda A_{\mathrm{f}}$ as a preconditioner. As with their fixed point scheme (see Section 3.3.1), λ was defined as

$$\lambda = \frac{\Delta t^2}{4\rho}S^*S\mathbf{1}. \tag{59}$$

Unfortunately, only cursory results were given. Mori and Peskin [20] also used a double Schur complement **X** equation as the basis for a solver, using GMRES as their Krylov method. They also tried Biconjugate Gradient Stabilized, Conjugate Gradient Squared, and TFQMR iterations, but found GMRES to be the most efficient. They note that a preconditioner is essential for efficiency for their problem, and present one similar to the Mayo and Peskin preconditioner but which also accounts for the fact that they have thick immersed boundaries and additional boundary mass.

The nonsymmetric-DSU equation, Eq. (43), is

$$\left(I-\frac{\Delta t^2}{4\rho}M\mathbb{P}SA_{\mathrm{f}}S^*\right)\mathbf{u}^{n+1} = \left(M\left(I+\frac{\nu\Delta t}{2}\Delta^h\right)+\frac{\Delta t^2}{4\rho}M\mathbb{P}SA_{\mathrm{f}}S^*\right)\mathbf{u}^n - \Delta t M\mathbb{P}[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}} + \frac{\Delta t}{\rho}M\mathbb{P}SA_{\mathrm{f}}\mathbf{X}^n. \tag{60}$$

This equation was also derived in Mori and Peskin [20]. They could not find a simple way to form an efficient preconditioner for this equation, and so focused on the double Schur complement **X** equation.

The DSP equation, Eq. (47), is

$$\nabla^h\cdot A_{\mathrm{IB}}^{-1}\nabla^h p^{n+\frac{1}{2}} = \nabla^h\cdot A_{\mathrm{IB}}^{-1}\left[\frac{\rho}{\Delta t}(2I-A_{\mathrm{IB}})\mathbf{u}^n - \rho[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}} + SA_{\mathrm{f}}\mathbf{X}^n\right]. \tag{61}$$

We are not aware of anyone having used this equation. It has some interesting properties, however. Since $A_{\mathrm{IB}}\to I$ as $\Delta t\to 0$, this equation looks like a generalized Poisson equation. Because A_{IB} is symmetric and positive definite, A_{IB}^{-1} is as well. Because ∇^h and $\nabla^h\cdot$ are adjoints (at least for uniform Cartesian grids and standard spatial discretizations), $\nabla^h\cdot A_{\mathrm{IB}}^{-1}\nabla^h$ is symmetric and positive definite (excluding the subspace of constants). Hence, (61) is an equation to which conjugate gradient could be applied, though it would require an inner solver for every application of A_{IB}^{-1}, which may make it expensive.

The (symmetric) DSU equation, Eq. (48), is

$$\mathbb{P}A_{\mathrm{IB}}\mathbb{P}\mathbf{u}^{n+1} = \mathbb{P}\left[(2I-A_{\mathrm{IB}})\mathbf{u}^n - \Delta t[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}} + \frac{\Delta t}{\rho}SA_{\mathrm{f}}\mathbf{X}^n\right]. \tag{62}$$

This equation also has not appeared in the literature, but it too has some interesting properties. Because $\mathbb{P}$ is self-adjoint, $\mathbb{P}A_{\mathrm{IB}}\mathbb{P}$ is symmetric and positive definite on the subspace of divergence-free vector fields. Thus, conjugate gradient could be applied to this equation as well. Further, $\mathbb{P}A_{\mathrm{IB}}\mathbb{P}$ need not be applied for every matrix vector product in the conjugate gradient calculations; applying $\mathbb{P}A_{\mathrm{IB}}$ would be sufficient. This follows from the facts that the subspace of divergence-free vector fields is closed under addition and scalar multiplication, and that the CG algorithm only performs additions, subtractions, multiplication by scalars, and applications of $\mathbb{P}A_{\mathrm{IB}}$ on the vectors it is given.

4. Comparison of implicit methods

In this section we compare the computational performance of the double Schur complement implicit methods from Section 3.3 with the explicit method and with each other. The implicit methods are based on the DSX equation,

$$\left(I-\frac{\Delta t^2}{4\rho}S^*M\mathbb{P}SA_{\mathrm{f}}\right)\mathbf{X}^{n+1} = \left(I+\frac{\Delta t^2}{4\rho}S^*M\mathbb{P}SA_{\mathrm{f}}\right)\mathbf{X}^n + \frac{\Delta t}{2}S^*\left(\mathbf{u}^n + M\left(I+\frac{\nu\Delta t}{2}\Delta^h\right)\mathbf{u}^n - \Delta t M\mathbb{P}[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}}\right), \tag{63}$$

the DSP equation,

$$\nabla^h\cdot A_{\mathrm{IB}}^{-1}\nabla^h p^{n+\frac{1}{2}} = \nabla^h\cdot A_{\mathrm{IB}}^{-1}\left(\frac{\rho}{\Delta t}(2I-A_{\mathrm{IB}})\mathbf{u}^n - \rho[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}} + SA_{\mathrm{f}}\mathbf{X}^n\right), \tag{64}$$

and the (symmetric) DSU equation,

$$\mathbb{P}A_{\mathrm{IB}}\mathbb{P}\mathbf{u}^{n+1} = \mathbb{P}\left((2I-A_{\mathrm{IB}})\mathbf{u}^n - \Delta t[(\mathbf{u}\cdot\nabla^h)\mathbf{u}]^{n+\frac{1}{2}} + \frac{\Delta t}{\rho}SA_{\mathrm{f}}\mathbf{X}^n\right). \tag{65}$$

Note that although these implicit methods allow a larger timestep to be taken (while remaining stable) than the explicit method, the resulting linear system that needs to be solved may be very poorly conditioned. It may well be the case that the implicit methods require more iterations to solve at a given timestep than the number of timesteps that would be required by the explicit method to stably compute up to the same simulation time. Since each implicit iteration is roughly equal in work to the cost of carrying out a single timestep with the explicit method, this means that implicit methods may actually be more computationally expensive than the explicit methods. We have seen some implicit methods exhibiting such behavior, as have Mori and Peskin (Y. Mori, personal communication, July 18, 2006).

4.1. Test problems

We ran each of the implicit methods (and the explicit method) on two different test problems, for a range of numerical and problem parameters. The two test problems are described in this section.

4.1.1. Ellipse problem

The first test problem is a standard one from the IB literature, in which the immersed boundary is a closed loop initially in the shape of an ellipse [15–17,28–30]. We choose an ellipse initially aligned in the coordinate directions with horizontal semi-axis $a = 0.28125$ and vertical semi-axis $b = 0.75a$. The fluid is initially at rest in a periodic domain, Ω, with $\Omega = [0,1] \times [0,1]$. The force law used is given by Eq. (27), with $\ell_0 = 0$ (*i.e.*, zero resting length "springs"). 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 is shown in Fig. 3.

4.1.2. Wing problem

The second test problem we consider is a simplified model of a moving wing adapted from Miller and Peskin [18]. The wing is represented by a finite-length string of IB points. The desired or target motion of the wing is set by specifying, as functions of time, the horizontal location $x(t)$ of the center of the wing and the angle of attack $\alpha(t)$ relative to the x-axis:

$$x(t) = \frac{A_0}{2}\cos(2\pi\omega t) \tag{66}$$

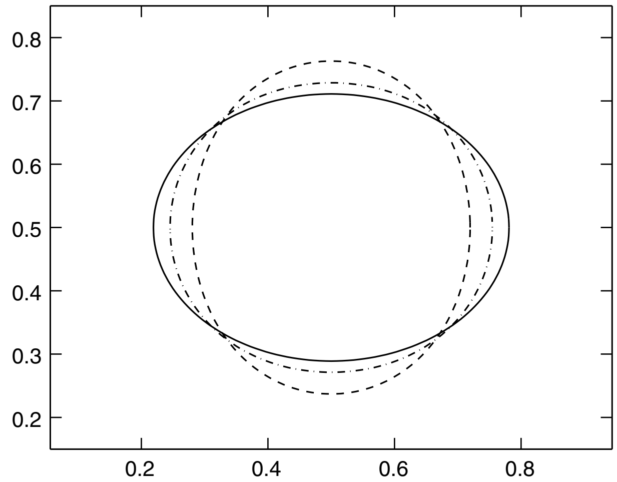

Fig. 3. Computed immersed boundary positions at successive times; the solid line is the initial shape, and the dashed-dotted and dashed lines show the configuration of the boundary later in the simulation.

and

$$\alpha(t) = \alpha_0 + \beta\sin(2\pi\omega t + \phi). \tag{67}$$

Here, A_0 is the stroke amplitude, ϕ is the phase rotation, and β is the change in angle of attack during stroke reversal. The wing flaps back and forth along a horizontal plane with a frequency of ω. In this case, A_0/c was set to 2.8 (where c is the chord length of the wing), ϕ was set to 0, α_0 was set to $\pi/2$, and β was set to $\pi/4$. This provided a symmetric stroke with a minimum angle of attack of 45°. The value $7.5\nu/\pi c A_0$ was used for ω. These equations determine the motion of a set of target points, one such point for each IB point that makes up the wing, plus two target points at the end of the wing to keep it from compressing (we modelled the wing with zero resting-length springs). Each of the actual IB points is linked to the corresponding target point by an elastic spring, and because of the forces generated by these springs at the IB points, the IB points approximately track the desired motion. A schematic view of the wing motion is shown in Fig. 4.

In contrast to Miller and Peskin, we do not include any bending forces in our calculation and we use zero resting length "springs". More concretely, we use Eq. (27), with $\ell_0 = 0$, for the computation of forces at the non-target IB points. Each non-target IB point has exactly three links: one to a target point, and two to the neighboring points within the wing (the endpoints of the wing have an additional target point on the side instead of two neighboring IB points). At target points, we use $\mathbf{F}_k = 0$. (Thus, the only differences between target points and normal IB points in our simulations are that we zero out the immersed boundary force on the target points and the target points move at the externally specified velocity rather than the fluid velocity).

The existence of target points that are unaffected by fluid motion makes our stability proof in [22] inapplicable to this problem. (In fact, the target point motion can be chosen in such a way that the continuous system becomes unstable, since energy is being added to the system through moving target points which have springs attached to them). Therefore, we cannot necessarily expect unconditional stability of our discretization and in particular our time lagging of spreading and interpolation operators may be problematic. However, it may be enlightening to test our methods on this problem since it is another common type of IB problem with very different dynamics than the ellipse problem.

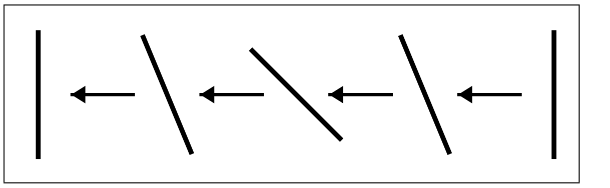

Fig. 4. Schematic showing rotation and translation of target motion of the wing during one half period. The second half period's motion is the mirror image of this.

4.2. Results

In this section, we show computational timings for both the ellipse and wing problems to compare the efficiency of the explicit, DSU, DSX, and DSP methods. In all cases, we use $\rho = 1$, $v = 0.01$, and keep the number of Lagrangian grid points proportional to $1/h$ whenever refining the grid. We use a relative tolerance of 10^{-5} as the stopping criterion for the Krylov subspace iterations.

4.2.1. Ellipse problem

Fig. 5 shows the CPU times required by the explicit, DSU, DSX, and DSP methods for the ellipse problem as a function of the spatial stepsize, h. Two values of elastic stiffness, $\gamma = 1$ and $\gamma = 100$, are compared. The timestep for the explicit method was chosen near its stability limit for each value of γ. The timesteps for the implicit DSU, DSX, and DSP methods were chosen so that the CFL number was 1. The DSU and DSX methods appear roughly comparable to the explicit method for this test problem, but the DSP method took 15–25 times as long as the explicit method. In Table 1, the CPU times relative to the explicit method for the DSU and DSX methods are reported. Times for the DSP method are not reported because it was not competitive.

The "near stability limit" timestep for the explicit method is used when comparing against the implicit methods. This timestep is found by an (expensive) search which involves repeating the simulation with many different values of Δt until a timestep is found which results in a stable simulation and is within 1% of the magnitude of a timestep that results in an unstable simulation. This search needs to be repeated whenever any parameters in the system (e.g., h, γ, initial IB configuration) change. The time required to find such a timestep is not reported, instead only the CPU time the simulation took with the resulting "optimal" timestep is shown.

From Fig. 5 and Table 1, we see that the DSU and DSX methods with the timestep computed from the CFL condition (at CFL number 1) are a bit slower than the explicit method with an optimal timestep when h is coarse. However, both methods rapidly improve relative to the explicit method as h becomes finer, so that both are competitive with the optimal-timestep explicit method. We see that for all methods, the CPU time for $\gamma = 100$ was roughly ten times that for $\gamma = 1$, although the increase in the stiffness decreased the relative effectiveness of the implicit methods for this problem.

Table 1
Ratio of computation times of the DSU and DSX methods to the time of the explicit method for the same parameters, with the ellipse problem

h	$\gamma = 1$		$\gamma = 100$	
	DSU	DSX	DSU	DSX
2^{-5}	1.80	2.53	2.19	3.73
2^{-6}	1.40	1.70	1.67	2.47
2^{-7}	0.95	1.03	1.26	2.14
2^{-8}	0.77	0.55	1.22	1.70
2^{-9}	0.63	0.35	0.96	1.24

Both the DSU and DSX methods were run with a timestep corresponding to CFL number 1, and the explicit method was run with the "optimal" time step.

There are two things to note here. First is that an optimal timestep for the explicit method is not generally available in practice; instead practitioners guess a timestep at which they believe the simulation will remain stable, and use that timestep. If the guess is incorrect, then the simulation must be restarted with a smaller timestep. Therefore, the explicit method in practice is likely to be an order of magnitude slower than the explicit method shown here, which increases the advantage of the implicit methods. Second, the CFL constraint is not significantly larger than the explicit timestep limit for this particular problem, because larger γ results in faster fluid velocities. In the wing problem considered next, the value of the stiffness γ has little affect on the fluid velocity and so the CFL constraint does not become more restrictive as γ is increased. As we see below, the relative effectiveness of the methods is very different in this case.

4.2.2. Wing problem

Fig. 6 shows the CPU times required by the explicit, DSU, and DSX methods for the wing problem as a func-

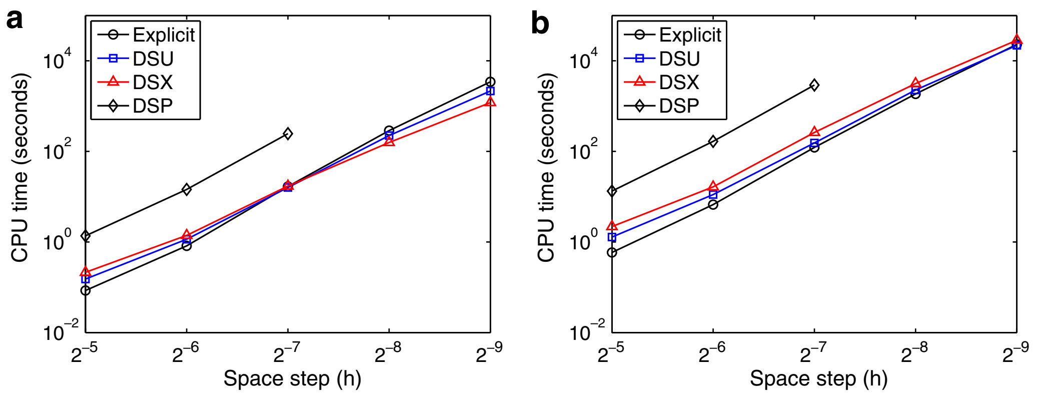

Fig. 5. CPU time for the explicit, DSU, DSX, and DSP methods on the ellipse problem as a function of spatial stepsize when (a) $\gamma = 1$ and (b) $\gamma = 100$.

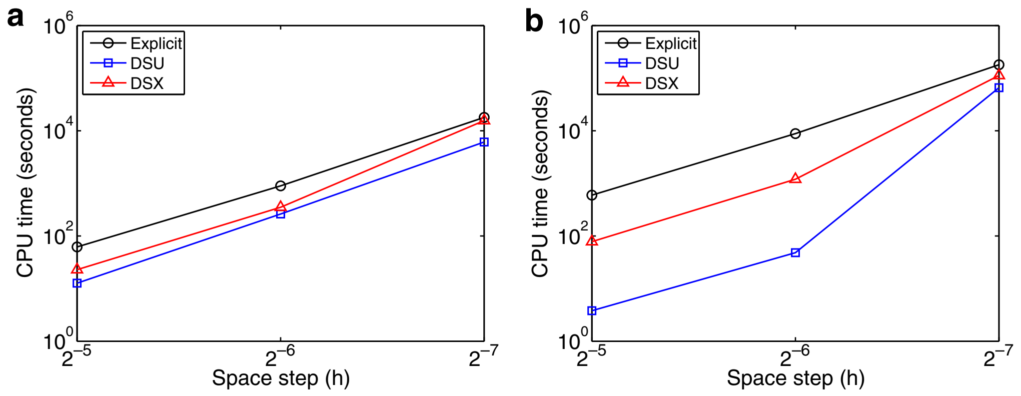

Fig. 6. CPU time for the explicit, DSU, and DSX methods on the wing problem as a function of spatial stepsize when (a) $\gamma = 10^4$ and (b) $\gamma = 10^6$.

tion of the spatial stepsize, h, for elastic stiffnesses $\gamma = 10^4$ and $\gamma = 10^6$. All methods used a "near stability limit" timestep, found via an expensive search outlined below. The timings for the DSP method are again omitted, because we found they were again significantly slower than the DSU and DSX methods and did not want to wait for them to complete on fine grids for this problem.

The "near stability limit" timestep was found by a method very similar to the one used for the explicit solve of the ellipse problem in Section 4.2.1. It involved repeating the simulation with many different values of Δt until one was found that was within 1% of the magnitude of a "bad" timestep. A "bad" timestep was one for which (a) the simulation went unstable (IB points exited the domain during the simulation), (b) *any* IB point was further than h from its target point at the end of *any* timestep, or (c) $\Delta t > h/4$ (a value chosen to ensure Δt remained under the CFL constraint). The latter two constraints were added primarily for the semi-implicit methods; they did not significantly restrict the explicit timestep beyond the restriction needed to satisfy the first constraint. The second constraint was added due to the fact that the semi-implicit methods did not exhibit the explosive instabilities of the explicit method, thus instability or inaccuracy at one timestep would only sometimes result in the simulation "blowing up". Also, there were some cases where the semi-implicit methods could take a timestep up to and even beyond CFL 1 while satisfying the first two constraints, so the third condition was added.

Table 2 shows the CPU time that the DSU and DSX methods took relative to the explicit method for the wing problem described in Section 4.1.2. For these problems, we note that the results are consistently better than the explicit method. In a few cases, we were able to compute up to $\Delta t = h/4$, but in other cases the constraint that the IB points remained near the target points required the use of a timestep that was much closer to the explicit timestep; potentially even within an order of magnitude of it. (Note that the implicit method is more costly per timestep due to requiring multiple iterations to converge to a solution, so a timestep that is 10 times larger will not give an implicit method that is 10 times faster). The ability to sometimes compute up to $\Delta t = h/4$ (our artificially imposed limit to avoid ever going over CFL 1) suggests that an implicit discretization that has better stability properties for the wing problem (see Section 4.1.2) could significantly outperform the explicit method. This is in contrast to the ellipse problem where our method was proven to be unconditionally stable yet only achieved modest gains over the explicit method. We attribute this to the difference in behavior of the two problems: an increase in γ for the ellipse problem will result in greater velocities, whereas a change of γ for the wing problem increases the stiffness of the wing but should have little effect on the velocity.

Table 2
Ratio of computation times of the DSU and DSX methods to the time of the explicit method for the same parameters, with the wing problem

h	$\gamma = 10^4$		$\gamma = 10^6$	
	DSU	DSX	DSU	DSX
2^{-5}	0.2047	0.3693	0.0064	0.1310
2^{-6}	0.2915	0.3923	0.0054	0.1361
2^{-7}	0.3405	0.8689	0.3662	0.6265

Both the DSU and DSX methods were run with a timestep with CFL number less than 1, and the explicit method was run with the "optimal" time step.

4.3. Conclusions

We have reviewed implicit methods which previously were introduced for solving the IB equations, have introduced several new methods for our stable semi-implicit discretization of the IB equations [22], and have compared the computational cost of solvers which take advantage of the linearity of our method to the cost of the explicit method.

For two different test problems, we compared the performance of three implicit methods and an explicit method. The implicit methods are based on Krylov methods applied to different arrangements of the equations. One test problem involves the motion of an initially-elliptical interface under tension. This is a standard test problem in the IB literature. The second test problem involves the motion of a rigid 'wing' as it tracks the prescribed motion of a set of target points. In the ellipse case, larger stiffness causes larger velocity while in the wing case larger stiffness just causes the object to be more rigid without affecting the timescale of the fluid motion. Compared to the explicit method, the efficiency of the implicit methods was very different for the two problems.

For the ellipse problem, we found that two of the implicit methods (DSX and DSU) were competitive with the explicit method using an optimally-tuned timestep. Depending on parameters these implicit methods were between two times slower and two times faster than the explicit method. The third implicit method (DSP) was an order of magnitude or more slower than the (optimal) explicit method. For the ellipse problem, the relative effectiveness of the implicit methods did not improve as the stiffness increased because as the stiffness increased the timescale of the physical problem became correspondingly smaller. In addition, the explicit discretization of the advection terms in the Navier–Stokes equations constrained the timestep by a CFL condition. For large stiffnesses, the CFL constraint was not substantially less strict than the explicit IB method's stability constraint.

The relative effectiveness of the implicit methods is underestimated in this test problem because the optimal timestep of the explicit method to which we are comparing is not generally available in practice. Recall that the optimal timestep for the explicit method was found by running it many times with different timesteps to find one just under the stability limit. In practical computations with the explicit method, it is common for users to simply manually restart simulations with a smaller timestep when a simulation goes unstable. This, of course, can significantly increase the real-world cost of using the explicit method, though this additional cost is not measured or reported in practice. Further, both the DSU and DSX implicit methods were run without any preconditioning. It may be possible to accelerate these methods with the use of an appropriate preconditioner. Mori and Peskin have claimed some success with preconditioning for a similar test problem using a method like our DSX method [20].

For the wing problem, the semi-implicit methods consistently outperformed the explicit method. The timing results on the wing problem reflect the fact that our stability results for the semi-implicit discretization do not hold for this problem. However, there were cases in which we were able to compute up to CFL 1 and finish simulations much faster than with the explicit method, suggesting that an implicit discretization with better stability properties for the wing problem would be able to dramatically outperform the explicit method. This is in contrast to the ellipse problem where our proven unconditionally stable methods were only able to get modest improvements in computational time relative to the explicit method. This difference is due to the nature of the two different problems; in contrast to the ellipse problem, in the wing problem the timescale of fluid and wing motion are not affected by changes in the wing stiffness. Previous studies on implicit IB methods have focussed almost exclusively on the ellipse problem. As our results have demonstrated, it is important to look at test problems with a range of characteristics in order to fully assess the effectiveness of the methods.

In this paper, we have considered solution methods for stable linear implicit schemes for the immersed boundary method equations. Two types of nonlinearities were considered and are handled effectively by these linear schemes. The nonlinear advection term is discretized explicitly in time. The nonlinear position dependence of the immersed boundary force spreading and velocity interpolation operators are 'lagged' by evaluating them only at the beginning of each timestep. We have not yet addressed a third type of nonlinearity, namely, nonlinear constitutive laws for the mechanical behavior of the immersed boundaries (see Eqs. (26) and (27)). A reasonable strategy for handling this type of nonlinearity would be to use an iterative solver which in each iteration would require solving linearized equations similar to those addressed in this paper. This problem remains for future work.

Acknowledgements

The work of E.P.N. 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, Samuel Isaacson, and Laura Miller 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, J. Comput. Phys. 143 (1998) 346–380.

[2] A.S. Almgren, J.B. Bell, W.Y. Crutchfield, Approximate projection methods: Part I. Inviscid analysis, SIAM J. Sci. Comput. 22 (2000) 1139–1159.

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

[4] D. Boffi, L. Gastaldi, L. Heltai, Stability results and algorithmic strategies for the finite element approach to the immersed boundary method, in: Proceeding of the Sixth European Conference on Numerical Mathematics and Advanced Applications, Springer-Verlag, 2005, pp. 557–566.

[5] D.L. Brown, R. Cortez, M. Minion, Accurate projection methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 168 (2001) 464–499.

[6] R. Cortez, M. Minion, The blob projection method for Immersed Boundary problems, J. Comput. Phys. 161 (2000) 428–453.

[7] J. Dennis, R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice Hall, 1996.

[8] R. Dillon, L. Fauci, A. Fogelson, D. Gaver, Modeling biofilm processes using the Immersed Boundary method, J. Comput. Phys. 129 (1996) 85–108.

[9] L.J. Fauci, A.L. Fogelson, Truncated newton methods and the modeling of complex immersed elastic structures, Commun. Pure Appl. Math. 66 (1993) 787–818.

[10] L.J. Fauci, C.S. Peskin, A computational model of aquatic animal locomotion, J. Comput. Phys. 77 (1988) 85–108.

[11] A.L. Fogelson, ;A mathematical model and numerical method for studying platelet adhesion and aggregation during blood clotting, J. Comput. Phys. 1 (1984) 111–134.

[12] B.E. Griffith, C.S. Peskin, On the order of accuracy of the immersed boundary method: higher order convergence rates for sufficiently smooth problems, J. Comput. Phys. 208 (2005) 75–105.

[13] J.L. Guermond, P. Minev, J. Shen, An overview of projection methods for incompressible flows, Comput. Methods Appl. Mech. Engrg. 195 (2006) 6011–6045.

[14] D.-V. Le, B.C. Khoo, J. Peraire, An immersed interface method for the incompressible Navier–Stokes equations, in: Presented at the SMA Symposium, Singapore, 2004.

[15] L. Lee, R. Leveque, An Immersed Interface method for incompressible Navier–Stokes equations, SIAM J. Sci. Comput. 25 (2003) 832–856.

[16] R.J. Leveque, Z. Li, Immersed interface methods for Stokes flow with elastic boundaries or surface tension, SIAM J. Sci. Comput. 18 (1997) 709–735.

[17] A.A. Mayo, C.S. Peskin, An implicit numerical method for fluid dynamics problems with immersed elastic boundaries, Contemp. Math. 141 (1993) 261–277.

[18] L.A. Miller, C.S. Peskin, When vortices stick: an aerodynamic transition in tiny insect flight, J. Exp. Biol. 207 (2004) 3073–3088.

[19] L.A. Miller, C.S. Peskin, A computational fluid dynamics of'clap and fling' in the smallest insects, J. Exp. Biol. 208 (2005) 195–212.

[20] Y. Mori, C.S. Peskin, Implicit second order immersed boundary methods with boundary mass, Comput. Methods Appl. Mech. Engrg., to appear.

[21] E. Newren, Enhancing the Immersed Boundary Method: Stability, Volume Conservation, and Implicit Solvers, PhD Thesis, University of Utah, 2007.

[22] E.P. Newren, A.L. Fogelson, R.D. Guy, R.M. Kirby, Unconditionally stable discretizations of the immersed boundary equations, J. Comput. Phys. 222 (2007) 702–719.

[23] J.B. Perot, An analysis of the fractional step method, J. Comput. Phys. 108 (1993) 51–58.

[24] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220–252.

[25] C.S. Peskin, The immersed boundary method, Acta Numer. (2002) 1–39.

[26] C.S. Peskin, D.M. McQueen, Modeling prosthetic heart valves for numerical analysis of blood flow in the heart, J. Comput. Phys. 37 (1980) 113–132.

[27] 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, J. Comput. Phys. 81 (1989) 372–405.

[28] A.M. Roma, C.S. Peskin, M.J. Berger, An adaptive version of the immersed boundary method, J. Comput. Phys. 153 (1999) 509–534.

[29] J.M. Stockie, B.R. Wetton, Analysis of stiffness in the immersed boundary method and implications for time-stepping schemes, J. Comput. Phys. 154 (1999) 41–64.

[30] C. Tu, C.S. Peskin, Stability and instability in the computation of flows with moving immersed boundaries: a comparison of three methods, SIAM J. Sci. Stat. Comput. 13 (1992) 1361–1376.

[31] X. Wang, An iterative matrix-free method in implicit immersed boundary/continuum methods, Comput. Struct. 85 (2007) 739–748.