Journal of Computational Physics 228 (2009) 7137–7158

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Efficient solutions to robust, semi-implicit discretizations of the immersed boundary method

Hector D. Ceniceros [a], Jordan E. Fisher [a,*], Alexandre M. Roma [b]

[a] Department of Mathematics, University of California Santa Barbara, CA 93106, United States
[b] Departamento de Matemática Aplicada, Universidade de São Paulo, Caixa Postal 66281, CEP 05311-970, São Paulo-SP, Brazil

A R T I C L E I N F O

Article history:
Received 24 December 2008
Received in revised form 13 April 2009
Accepted 6 May 2009
Available online 27 May 2009

Keywords:
Semi-implicit method
Stokes flow
Navier–Stokes equations
Heart valve
Multigrid

A B S T R A C T

The immersed boundary method is a versatile tool for the investigation of flow-structure interaction. In a large number of applications, the immersed boundaries or structures are very stiff and strong tangential forces on these interfaces induce a well-known, severe time-step restriction for explicit discretizations. This excessive stability constraint can be removed with fully implicit or suitable semi-implicit schemes but at a seemingly prohibitive computational cost. While economical alternatives have been proposed recently for some special cases, there is a practical need for a computationally efficient approach that can be applied more broadly. In this context, we revisit a robust semi-implicit discretization introduced by Peskin in the late 1970s which has received renewed attention recently. This discretization, in which the spreading and interpolation operators are lagged, leads to a linear system of equations for the interface configuration at the future time, when the interfacial force is linear. However, this linear system is large and dense and thus it is challenging to streamline its solution. Moreover, while the same linear system or one of similar structure could potentially be used in Newton-type iterations, nonlinear and highly stiff immersed structures pose additional challenges to iterative methods. In this work, we address these problems and propose cost-effective computational strategies for solving Peskin's lagged-operators type of discretization. We do this by first constructing a sufficiently accurate approximation to the system's matrix and we obtain a rigorous estimate for this approximation. This matrix is expeditiously computed by using a combination of pre-calculated values and interpolation. The availability of a matrix allows for more efficient matrix–vector products and facilitates the design of effective iterative schemes. We propose efficient iterative approaches to deal with both linear and nonlinear interfacial forces and simple or complex immersed structures with tethered or untethered points. One of these iterative approaches employs a splitting in which we first solve a linear problem for the interfacial force and then we use a nonlinear iteration to find the interface configuration corresponding to this force. We demonstrate that the proposed approach is several orders of magnitude more efficient than the standard explicit method. In addition to considering the standard elliptical drop test case, we show both the robustness and efficacy of the proposed methodology with a 2D model of a heart valve.

* Corresponding author.
E-mail addresses: hdc@math.ucsb.edu (H.D. Ceniceros), jordan@math.ucsb.edu (J.E. Fisher), roma@ime.usp.br (A.M. Roma).
URLs: http://www.math.ucsb.edu/~hdc (H.D. Ceniceros), http://www.ime.usp.br/~roma (A.M. Roma).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.05.031

1. Introduction

The immersed boundary (IB) Method introduced by Peskin [1] is a versatile tool for simulating flow-structure interaction in a wide range of applications. The IB method employs a Lagrangian representation of the immersed structures and their interfacial forces and an Eulerian description of the flow variables (velocity and pressure). The Lagrangian description of the immersed boundaries, which does not have to conform to the Eulerian grid, provides a vast structure-building capability while the Eulerian flow description permits the use of efficient flow solvers. The power of the IB Method lies in a seamless connection of the two descriptions by the use of two operations: *spreading* (of interfacial forces) and *interpolation* (of velocity at the immersed boundary), both achieved via mollified delta functions.

In a large number of applications, the immersed boundaries or structures are very stiff and strong *tangential* forces on these interfaces induce severe time-step restrictions for explicit discretization [2,3]. Fully implicit discretizations and some suitable semi-implicit schemes remove this hindering constraint but seemingly at a cost that makes these options impractical [4,5]. Recently, an economical semi-implicit method has been proposed by Hou and Shi [6,7]. This novel approach relies on an ingenious small scale decomposition and becomes explicit in Fourier space. While nearly computationally optimal, the method of Hou and Shi is applicable only to simple periodic interfaces. Thus, there is a practical need for a robust, cost-effective approach that can be applied more broadly; one that can be used for immersed structures of complex geometry with crossed links as well as a mix of tethered and untethered points, which are required in many of the applications of the IB Method. The advancement of such an approach is the main focus of this work.

Our starting point is a semi-implicit scheme introduced by Peskin [1] in the late 1970s, in which the spreading and interpolation operators are lagged, i.e. evaluated at the current interfacial configuration rather than at the future one. Variations of this scheme were also considered by Tu and Peskin [4] and by Mayo and Peskin [5]. This lagged-operators discretization has recently received renewed attention. In particular, Newren, Fogelson, Guy, and Kirby proved that this scheme, in its first order or second order Crank–Nicolson form, is unconditionally stable when inertia is neglected and the interfacial force is linear and self-adjoint [8]. Numerical experiments in [8], as well as our own experiments, suggest the robustness of this discretization extends to the inertial case with nonlinear interfacial force. Thus, it is now established that Peskin's original semi-implicit discretization enjoys great stability properties and robustness, hence the relevant question is whether its solution can be computed at a reasonable cost. Recently, Mori and Peskin [9] took an important step to answer this question. They considered a variation of this scheme, with a linearized tension force discretization which leads to a linear system of equations for the interface configuration at the future time-step. They also proposed a fully implicit method solved iteratively where each of the iterates has the same structure as the linearized semi-implicit discretization. Mori and Peskin opted for Krylov subspace methods to solve the linear system to take advantage of the fact that matrix–vector products can be obtained via standard operations in the IB Method and to avoid the construction of the system's dense matrix.

In this work we demonstrate that the solution to Peskin's operator-lagged discretization can be obtained much more efficiently by working directly with the matrix, and by using suitable linear and nonlinear iterative methods. More precisely, we construct a sufficiently accurate approximation to the matrix which can be expeditiously obtained by using a combination of pre-computed values and interpolation. The availability of a matrix allows for streamlined matrix–vector products and facilitates the design of effective iterative schemes.

Most of the work to date on the investigation and removal of the numerical stiffness of the IB Method has focused on a simple test problem: a relaxing elliptical drop. While this test model contains the characteristic high stiffness of the IB approach in a simple interfacial geometry it does not showcase the additional complications that might arise when the immersed structure is composed of crossed links with both tethered and untethered points. Such complex structures are common in applications, starting with the origins of the method to investigate blood flow in the heart [1]. Thus, in addition to obtaining efficient methods for the standard elliptical drop test case, we also propose cost-effective iterative methods to deal with cases of more complex immersed structures. In our proposed approach we employ a splitting in which we first solve efficiently a linear problem for the interfacial force and then we use a fast-converging nonlinear iteration to find the interface configuration corresponding to this force. We develop this method in the context of a 2D model of a heart valve and demonstrate that the proposed approach is several orders of magnitude more efficient than the standard explicit method.

The rest of the paper is organized as follows. In Section 2, we review the formulation of the IB Method. Section 3 deals with the discretization with the focus on Peskin's semi-implicit scheme with lagged-operators. We devote Section 5 to the construction of the matrix approximation and to a rigorous estimate of that approximation. In Section 6, we consider the case of the relaxing elliptical drop with linear force density while the case of a nonlinear force and the same interfacial configuration is considered in Section 7. Section 8 is devoted to the simple 2D model of the heart valve and the splitting scheme and concluding remarks are presented in Section 9.

2. The immersed boundary method

To describe the method, in its simplest form, we consider a two-dimensional, incompressible, Newtonian fluid occupying a domain $\Omega \subset \mathbb{R}^2$. Inside this domain we assume that there is an immersed, neutrally buoyant, elastic structure (also referred to as boundary or interface). This immersed interface is composed of a system Γ of elastic fibers whose position at any time t

is represented in Lagrangian form by $\mathbf{X}(s,t)$, where $s \in B$ is a Lagrangian parameter. The interface Γ need not be closed or even continuous. The governing equations are:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \mu\nabla^2\mathbf{u} + \mathbf{f}, \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{2}$$

$$\frac{\partial \mathbf{X}}{\partial t} = \mathbf{u}(\mathbf{X},t), \tag{3}$$

where ρ and μ are the density and viscosity, respectively (both assumed to be constant). Here $\mathbf{u}(\mathbf{x},t)$ and $p(\mathbf{x},t)$ are the velocity field and the pressure, respectively, described in terms of the Eulerian, Cartesian coordinate $\mathbf{x}$. The term $\mathbf{f}$ represents the singularly supported interfacial (tension) force of the immersed structure acting onto the fluid. The system (1)–(3) is supplemented with initial and boundary conditions. Throughout this work, we consider only periodic boundary conditions and Ω is a rectangular domain.

The crux of the IB method and much of its versatility is the seamless connection of the Lagrangian representation of the immersed structure with the Eulerian representation of the flow. This is achieved via the identities:

$$\frac{\partial \mathbf{X}}{\partial t} = \int_\Omega \mathbf{u}(\mathbf{x},t)\delta(\mathbf{x}-\mathbf{X}(s,t))d\mathbf{x}, \tag{4}$$

$$\mathbf{f}(\mathbf{x},t) = \int_B \mathbf{F}(\mathbf{X}(\cdot,\cdot),s,t)\delta(\mathbf{x}-\mathbf{X}(s,t))ds, \tag{5}$$

where δ denotes the (two-dimensional) Dirac delta distribution. In (5), $\mathbf{F}$ represents the elastic force density of Γ and is described in Lagrangian coordinates. For example, if the tangent direction $\mathbf{t}$ along the fibers varies smoothly and if the local elastic energy density is assumed to depend only on the tangential strain $|\frac{\partial \mathbf{X}}{\partial s}|$ then

$$\mathbf{F}(\mathbf{X},s,t) = \frac{\partial}{\partial s}\left(T\left(\left|\frac{\partial \mathbf{X}}{\partial s}\right|\right)\mathbf{t}\right). \tag{6}$$

Here $T(|\frac{\partial \mathbf{X}}{\partial s}|)$ is the fiber (interfacial) tension and $\mathbf{t}$ is the unit tangent,

$$\mathbf{t} = \frac{\frac{\partial \mathbf{X}}{\partial s}}{|\frac{\partial \mathbf{X}}{\partial s}|}. \tag{7}$$

Thus $\mathbf{F}$ is in general a nonlinear function of the interfacial configuration. We denote this relation by

$$\mathbf{F} = \mathcal{A}(\mathbf{X}). \tag{8}$$

3. Discretization

We consider uniform Cartesian grids $\mathcal{G}_\Omega$ and $\mathcal{G}_B$ with grid size h and h_B, respectively to discretize Ω and B and employ standard second order finite differences for the spatial derivatives. We write Peskin's original semi-implicit discretization in the form

$$\rho\left(\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t} + \mathbf{u}^n\cdot\mathbf{D}_h\mathbf{u}^n\right) = -\mathbf{D}_h p^{n+1} + \mu L_h\mathbf{u}^{n+1} + \mathcal{S}_n\mathcal{A}_{h_B}(\mathbf{X}^{n+1}), \tag{9}$$

$$\mathbf{D}_h\cdot\mathbf{u}^{n+1} = 0, \tag{10}$$

$$\frac{\mathbf{X}^{n+1}-\mathbf{X}^n}{\Delta t} = \mathcal{S}_n^*\mathbf{u}^{n+1}, \tag{11}$$

where a superscript m denotes a numerical approximation taken at the time $m\Delta t$ and Δt is the time-step. The spatial operators $\mathbf{D}_h$ and L_h are the standard second order approximations to the gradient and the Laplacian, respectively, and $\mathcal{A}_{h_B}$ is a suitable discrete version of $\mathcal{A}$.

$\mathcal{S}_n$ and $\mathcal{S}_n^*$ are the *lagged* spreading and interpolation operators, respectively, given by

$$(\mathcal{S}_n G)(\mathbf{x}) = \sum_{s\in\mathcal{G}_B} G(s)\delta_h(\mathbf{x}-\mathbf{X}^n(s))h_B, \tag{12}$$

$$(\mathcal{S}_n^* w)(s) = \sum_{\mathbf{x}\in\mathcal{G}_\Omega} w(\mathbf{x})\delta_h(\mathbf{x}-\mathbf{X}^n(s))h^2, \tag{13}$$

where $\delta_h(\mathbf{x}) = d_h(x)d_h(y)$ and d_h is an approximation of the one-dimensional delta. These operators are called lagged because the interface configuration $\mathbf{X}^n$ is used instead of the future configuration $\mathbf{X}^{n+1}$.

There is flexibility in the choice of d_h but for concreteness in the presentation we choose Peskin's delta [1]:

$$d_h(r) = \begin{cases} \frac{1}{4h}\left(1+\cos(\frac{\pi r}{2h})\right) & \text{if} \quad |r| \leqslant 2h \\ 0 & \text{otherwise}. \end{cases} \tag{14}$$

Let us rewrite (9) as

$$\mathbf{u}^{n+1} = -\frac{\Delta t}{\rho}\mathbf{D}_h p^{n+1} + \nu\Delta t L_h \mathbf{u}^{n+1} + \mathbf{a}^{n+1}, \tag{15}$$

where $\nu = \mu/\rho$ and

$$\mathbf{a}^{n+1} = \frac{\Delta t}{\rho}\mathcal{S}_n\mathcal{A}_{h_B}(\mathbf{X}^{n+1}) + \mathbf{u}^n - \Delta t\mathbf{u}^n \cdot \nabla_h \mathbf{u}^n. \tag{16}$$

We can eliminate the pressure term in (15) using (10) by introducing a discrete projection P_h defined as

$$\mathbf{v} = P_h\mathbf{v} + \mathbf{D}_h\phi_v, \quad \mathbf{D}_h \cdot P_h\mathbf{v} = 0, \quad P_h\mathbf{D}_h\phi_v = 0. \tag{17}$$

for any smooth vector field $\mathbf{v}$ defined on the grid $\mathcal{G}_\Omega$. Applying P_h to (15), using (10) and that for periodic boundary conditions L_h and P_h commute we get

$$\mathbf{u}^{n+1} = \nu\Delta t L_h\mathbf{u}^{n+1} + P_h\mathbf{a}^{n+1}, \tag{18}$$

that is

$$\mathbf{u}^{n+1} = (I - \nu\Delta t L_h)^{-1}P_h\mathbf{a}^{n+1}. \tag{19}$$

Let us denote

$$\mathcal{L}_h = (I - \nu\Delta t L_h)^{-1}P_h. \tag{20}$$

$\mathcal{L}_h$ is a linear operator which henceforth we will refer to as the fluid solver. Note that with periodic boundary conditions and a standard second order finite difference approximation for the spatial derivatives, $I - \nu\Delta t L_h$ is symmetric and positive definite and as a result so is its inverse. On the other hand, P_h is also symmetric and, being a projection, it is positive semi-definite. Moreover, P_h and $(I - \nu\Delta t L_h)^{-1}$ commute as can be shown using the discrete Fourier transform. Consequently, these two symmetric operators can be diagonalized by the same orthogonal matrix and thus the product, $\mathcal{L}_h$, is positive semi-definite.

Using this notation, Peskin's semi-implicit method can be encoded as

$$\mathbf{u}^{n+1} = \mathcal{L}_h\mathbf{a}^{n+1}, \tag{21}$$

$$\mathbf{X}^{n+1} = \mathbf{X}^n + \Delta t\mathcal{S}_n^*\mathbf{u}^{n+1}, \tag{22}$$

where $\mathbf{a}^{n+1}$ is given by (16). Eliminating $\mathbf{u}^{n+1}$ in (22) we obtain a system of equations for the immersed boundary configuration $\mathbf{X}^{n+1}$:

$$\mathbf{X}^{n+1} = \mathcal{M}_n\mathcal{A}_{h_B}(\mathbf{X}^{n+1}) + \mathbf{b}^n, \tag{23}$$

where

$$\mathcal{M}_n = \alpha\mathcal{S}_n^*\mathcal{L}_h\mathcal{S}_n, \tag{24}$$

with

$$\alpha = \frac{(\Delta t)^2}{\rho} \tag{25}$$

and

$$\mathbf{b}^n = \mathbf{X}^n + \Delta t\mathcal{S}_n^*\mathcal{L}_h[\mathbf{u}^n - \Delta t\mathbf{u}^n \cdot \nabla_h\mathbf{u}^n]. \tag{26}$$

We have thus reduced (9) and (11) to a single system of equations involving only the unknown $\mathbf{X}^{n+1}$. If the number of Lagrangian nodes is N_B then (23) represents a system of $2N_B$ equations for the $2N_B$ values $\mathbf{X}^{n+1} = (X^{n+1}, Y^{n+1})$. If we can solve this system then we can obtain $\mathbf{u}^{n+1}$ via (21).

Note that the positive semi-definiteness of $\mathcal{L}_h$ extends to $\mathcal{M}_n$. Indeed, if we define the inner product on Ω as

$$(u, v)_\Omega = \sum_{\mathbf{x}\in\mathcal{G}_\Omega} u(\mathbf{x})v(\mathbf{x})h^2, \tag{27}$$

$$(\mathbf{u},\mathbf{v})_{\Omega} = (u_1, v_1)_{\Omega} + (u_2, v_2)_{\Omega}, \tag{28}$$

and on B as

$$(F,G)_B = \sum_{\mathbf{x}\in\mathcal{G}_B} F(\mathbf{x})G(\mathbf{x})h_B, \tag{29}$$

$$(\mathbf{F},\mathbf{G})_B = (F_1,G_1)_B + (F_2,G_2)_B, \tag{30}$$

then $\mathcal{S}_n$ and $\mathcal{S}_n^*$ as given in (12) and (13) are adjoints to each other and

$$(\mathbf{F},\mathcal{M}_n\mathbf{F})_B = \alpha(\mathbf{F},\mathcal{S}_n^*\mathcal{L}_h\mathcal{S}_n\mathbf{F})_B = \alpha(\mathcal{S}_n\mathbf{F},\mathcal{L}_h\mathcal{S}_n\mathbf{F})_B \geqslant 0. \tag{31}$$

In fact, $(\mathbf{F},\mathcal{M}_n\mathbf{F})_B > 0$ for $\mathbf{F}$ *ne* 0 unless the Eulerian force $\mathcal{S}_n\mathbf{F}$ is in the kernel of the projection, i.e. if it is a gradient field, or if there are too many Lagrangian points per Eulerian cell and injectivity of $\mathcal{S}_n$ is lost. There are physical situations, such as for example a circular drop under uniform surface tension at equilibrium, where the Eulerian force is a gradient field at the continuous level (to balance the gradient of the pressure). However, as it is well-known, the IB method spreading of the force fails in general to produce a discrete gradient which results in the generation of non-zero velocities referred to as spurious currents [10]. Ironically, this defect of the IB approach has the benefit of rendering $\mathcal{M}_n$ positive definite provided $\mathcal{S}_n$ remains injective. This is something that we will exploit via *multigrid*.

The focus of this paper is to propose efficient methods for solving (23) to be able to remove the severe numerical stiffness of the IB Method in an economical and robust fashion for a wide range of practical flow-structure situations. Of course, to be efficient the specific computational approach has to be dependent on the geometry of the immersed structure and the force operator $\mathcal{A}_{h_B}$. This is something that has been largely overlooked in the literature as the research has concentrated mostly on understanding and removing the stiffness on a simple setting: an elliptical interface with a linear density force $\mathcal{A}_{h_B}$. Here, we consider both linear and nonlinear $\mathcal{A}_{h_B}$ with simple and complex immersed structure geometries to highlight the challenges in producing efficient solvers and to illustrate our proposed approaches.

4. Some comments on computational costs and efficiency

One of the most commonly used schemes with an explicit treatment of the immersed boundary is the so-called Forward Euler/Backward Euler (FE/BE) [3] in which the tension force is explicit (Forward Euler) and the viscous term is implicit (Backward Euler). That is,

$$\mathbf{u}^{n+1} = \mathcal{L}_h\left[\frac{\Delta t}{\rho}\mathcal{S}_n\mathcal{A}_{h_B}(\mathbf{X}^n) + \mathbf{u}^n - \Delta t\mathbf{u}^n\cdot\nabla_h\mathbf{u}^n\right], \tag{32}$$

$$\mathbf{X}^{n+1} = \mathbf{X}^n + \Delta t\mathcal{S}_n^*\mathbf{u}^{n+1}. \tag{33}$$

The main cost of this scheme per time-step is the fluid solver, i.e. the operation involving $\mathcal{L}_h$. On the other hand any solution method for (23) requires the evaluation of $\mathbf{b}^n$, given by (26), and thus at least one fluid solver operation. Hence, it seems appropriate to measure the cost of iterative methods for (23) relative to that of one FE/BE time-step and we will refer to this unit as one FE/BE. Our goal is to present robust solution methods to (23) that have cost of just a few FE/BE's. Naturally, because the semi-implicit scheme allows for time steps several orders of magnitude larger than those permitted by the FE/BE scheme in a stiff problem, the extra computational work per time-step will be more than compensated and a speed-up of several orders of magnitude could be achieved.

In the design of efficient iterative methods for (23) it is crucial to streamline the calculation of quantities of the form $\mathcal{M}_n\mathbf{F}$. These quantities can be computed in two ways. Given an interface-defined vector $\mathbf{F}$ we can apply in sequence the operations of spreading, fluid solver, and interpolation. Alternatively, we can construct a matrix representation of $\mathcal{M}_n$ and use matrix–vector multiplication. Given that a direct computation of the matrix is prohibitively expensive, $Nb \times FE/BE$, all related work to date has avoided the latter approach. However, there are some advantages of having a matrix representation of $\mathcal{M}_n$ and, as we show in the next section, it is possible to construct a sufficiently accurate approximation to the matrix in only $O(FE/BE)$ operations.

One of the advantages of having a matrix representation of the operator $\mathcal{M}_n$ is that we can gain access to a wider range of iterative methods. Multigrid smoothers like Gauss–Seidel or S.O.R., which are inaccessible with the spreading-fluid solver-interpolation approach, could be easily implemented if the matrix is available. Also, multigrid-based methods are greatly simplified when we have a matrix representation of $\mathcal{M}_n$, which allows us to employ a straightforward algebraic multigrid as opposed to a geometric one. One of the more subtle issues with using a geometric multigrid is how to coarsen the Eulerian grid. Uniform coarsening would be ineffective for example in the case of our heart valve model because as we coarsen the valve geometry there remain points close together which would require a fine Eulerian mesh to resolve, see Fig. 4. If a geometric multigrid were to be employed we would need to implement adaptive coarsening.

A second advantage of having a matrix representation of $\mathcal{M}_n$ is that we can obtain direct solutions of *coarse* (small) linear systems of the form $\mathcal{M}_n\mathbf{F} = \mathbf{Z}$ which could be useful in a number of situations; an instance of this will be discussed later in the context of the 2D heart valve model. But perhaps the main advantage is that of cost when computing quantities of the form $\mathcal{M}_n\mathbf{F}$. If this matrix–vector product is computed via the operator sequence spreading-fluid solver-interpolation the cost is about one FE/BE or $O(N^2\log N)+O(N_B)$, when N^2 is the number of Eulerian nodes and the fluid solver is based on the Fast Fourier Transform (FFT). On the other hand, a matrix–vector multiplication involving $\mathcal{M}_n$ requires $O(N_B^2)$ operations. If $N_B\sim N$ as it is typical in 2D applications, then the second approach has lower computational complexity. While the elimination of the factor $\log N$ might seem like a modest gain, we find that in practice (for $N_B=2N$) using the matrix representation to compute matrix–vector products is significantly faster than the spreading-fluid solver-interpolation approach, even at modest resolutions. Table 1 gives the CPU time in units of FE/BE (average CPU time of one FE/BE time-step for a given $N\times N$ Eulerian grid and $N_B=2N$) for matrix–vector multiplications computed using a matrix representation of $\mathcal{M}_n$. For example, at $N_B=1024$, the computation of a matrix–vector product with this approach is 1/71 FE/BE, i.e. 71 times faster than it would be using the spreading-fluid solver-interpolation option. If in addition, we take into the account that many matrix–vector products are needed in the course of an iterative method then the computational savings are significant. We note however that as the ratio N_B/N increases the savings get reduced and the computational advantage of using the matrix to obtain the product could be lost eventually. For example, in our application of the 2D heart valve model where $N_B\sim 4N$, the speedup in using the matrix form drops to roughly 10 at moderate to fine resolutions. While not as dramatic as that in the $N_B=2N$ case, it still leads to substantial savings in the overall algorithm. However, in a fully 3D application with a 2D immersed membrane we would have $N_B\sim N^2$ leading to a cost of $O(N^4)$ for a matrix–vector multiplication using the matrix while obtaining the same product via spreading-fluid solver-interpolation would be $O(N^3\log N)$. Indeed, some 3D applications may use substantially more fiber points. The 3D heart model proposed by McQueen and Peskin [11] employs nearly $N_B\sim N^3$ fiber points. Clearly, alternative methods that do not rely on the full matrix of the fluid operator will be required to handle these applications.

5. An expedited computation of a matrix representation of $\mathcal{M}_n$

5.1. Approximation from translation invariance at the continuum level

Let us consider again the linear fluid solver operator (20) with periodic boundary conditions. Utilizing the Fourier transform we obtain a representation formula of the form

$$\mathcal{L}_h\mathbf{f}(\mathbf{x})=\sum_{\mathbf{y}\in\mathcal{G}_\Omega}G_h(\mathbf{x}-\mathbf{y})\mathbf{f}(\mathbf{y})h^2,\quad \text{for}\quad \mathbf{x}\in\mathcal{G}_\Omega, \tag{34}$$

where the Green function $G_h(\mathbf{x}-\mathbf{y})$ is a 2×2 matrix with vanishing zero Fourier mode.

Recall that $\mathcal{M}_n=\alpha\mathcal{S}_n^*\mathcal{L}_h\mathcal{S}_n$, thus its entries $(\mathcal{M}_n)_{ij}$ correspond to α times the velocity that is obtained by *interpolating* the values produced at a given interfacial node $\mathbf{X}_i$ by *spread* unit horizontal and vertical forces located at another immersed boundary node $\mathbf{X}_j$. For the continuum problem this velocity depends only on the difference $\mathbf{X}_j-\mathbf{X}_i$ but due to the spreading and interpolation operators this translation invariance is not exact at the discrete level. However, as we prove later, we can still obtain sufficiently accurate approximations to $(\mathcal{M}_n)_{ij}$ by assuming translation invariance with the added benefit of a dramatic cost reduction. Specifically, we propose to approximate $(\mathcal{M}_n)_{ij}$ with values obtained by shifting both $\mathbf{X}_i$ and $\mathbf{X}_j$ an equal amount such that $\mathbf{X}_j$ lies exactly on an Eulerian node, which we may take as the origin. That is, we can fix the point (the origin) at which unit horizontal and vertical forces are applied (and spread) and then evaluate the effects everywhere else on the Eulerian grid by applying just *two* fluid solves: one each for a horizontal and vertical force at the origin. These Eulerian grid values can be pre-computed at the beginning of the simulation and then be used as a look-up table to obtain the corresponding values at any interfacial point $\mathbf{X}_i$ via standard interpolation. The generation of the one-time lookup table is only two *FE/BE* and the interpolation to generate the approximation of $\mathcal{M}_n$ is only $O(N_B^2)$.

We proceed now to detail the computation of $\mathcal{M}_n$. Define for each interfacial node $j, 1\leqslant j\leqslant N_B$, the unit forces

$$\mathbf{f}_k^j=\begin{cases}(0,0) & \text{If}\quad k\neq j,\\ \mathbf{e}_1 & \text{If}\quad k=j\end{cases}\quad 1\leqslant k\leqslant N_B, \tag{35}$$

and

$$\mathbf{g}_k^j=\begin{cases}(0,0) & \text{If}\quad k\neq j,\\ \mathbf{e}_2 & \text{If}\quad k=j\end{cases}\quad 1\leqslant k\leqslant N_B, \tag{36}$$

where $\mathbf{e}_1=(1,0)$ and $\mathbf{e}_2=(0,1)$. Then

Table 1
Average CPU time in FE/BE units for a matrix–vector multiplication involving $\mathcal{M}_n$ for given $N_B=2N$.

$N_B=2N$	256	512	1024
Matrix–vector product	1/26.8	1/58	1/71

$$(\mathcal{M}_n \mathbf{f}^j)_i = \left(\alpha \mathcal{S}_n^* \mathcal{L}_h \sum_{\mathbf{X}_k \in \mathcal{G}_B} \mathbf{f}_k^j \delta_h(\mathbf{x} - \mathbf{X}_k) h_B \right)_i, \tag{37}$$

$$= \alpha h_B \left(\mathcal{S}_n^* \mathcal{L}_h \mathbf{e}_1 \delta_h(\mathbf{x} - \mathbf{X}_j) \right)_i, \tag{38}$$

$$= \alpha h_B \sum_{\mathbf{z} \in \mathcal{G}_\Omega} (\mathcal{L}_h \mathbf{e}_1 \delta_h(\mathbf{x} - \mathbf{X}_j))(\mathbf{z}) \delta_h(\mathbf{z} - \mathbf{X}_i) h^2, \tag{39}$$

$$\approx \alpha h_B \sum_{\mathbf{z} \in \mathcal{G}_\Omega} (\mathcal{L}_h \mathbf{e}_1 \delta_h(\mathbf{x}))(\mathbf{z}) \delta_h(\mathbf{z} - (\mathbf{X}_i - \mathbf{X}_j)) h^2, \tag{40}$$

and similarly

$$(\mathcal{M}_n \mathbf{g}^j)_i \approx \alpha h_B \sum_{\mathbf{z} \in \mathcal{G}_\Omega} (\mathcal{L}_h \mathbf{e}_2 \delta_h(\mathbf{x}))(\mathbf{z}) \delta_h(\mathbf{z} - (\mathbf{X}_i - \mathbf{X}_j)) h^2. \tag{41}$$

With these two approximations we define the vector-valued functions

$$\mathbf{T}(\mathbf{y}) = \alpha h_B \sum_{\mathbf{z} \in \mathcal{G}_\Omega} (\mathcal{L}_h \mathbf{e}_1 \delta_h(\mathbf{x}))(\mathbf{z}) \delta_h(\mathbf{z} - \mathbf{y}) h^2, \tag{42}$$

$$\mathbf{U}(\mathbf{y}) = \alpha h_B \sum_{\mathbf{z} \in \mathcal{G}_\Omega} (\mathcal{L}_h \mathbf{e}_2 \delta_h(\mathbf{x}))(\mathbf{z}) \delta_h(\mathbf{z} - \mathbf{y}) h^2, \tag{43}$$

and note that

$$(\mathcal{M}_n \mathbf{f}^j)_i \approx \mathbf{T}(\mathbf{X}_i - \mathbf{X}_j), \tag{44}$$

$$(\mathcal{M}_n \mathbf{g}^j)_i \approx \mathbf{U}(\mathbf{X}_i - \mathbf{X}_j). \tag{45}$$

We can now compute the entries of a matrix representation of the linear operator $\mathcal{M}_n$ from **T** and **U**. To this end, we write the configuration **X** of the discretized immersed structure as the $2N_B$-array

$$\mathbf{X} = \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} X_1 \\ \vdots \\ X_{N_B} \\ Y_1 \\ \vdots \\ Y_{N_B} \end{bmatrix}, \tag{46}$$

and similarly we write $\mathbf{F} = \mathcal{A}_{h_B}(\mathbf{X}) = (F, G)^T$. We seek then four $N_B \times N_B$ matrices A, B, C, and D such that

$$\mathcal{M}_n \mathbf{F} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} F \\ G \end{bmatrix}. \tag{47}$$

Then from (44) and (45) and the definition of the interfacial point forces (35) and (36), it follows that

$$\begin{bmatrix} A_{ij} \\ C_{ij} \end{bmatrix} \approx \mathbf{T}(\mathbf{X}_i - \mathbf{X}_j), \tag{48}$$

and

$$\begin{bmatrix} B_{ij} \\ D_{ij} \end{bmatrix} \approx \mathbf{U}(\mathbf{X}_i - \mathbf{X}_j). \tag{49}$$

The domain of both **T** and **U** is Ω. To expedite the calculation we precompute both functions on the Eulerian grid $\mathcal{G}_\Omega$. The value of **T** or **U** at any other point is obtained via simple linear interpolation from the corresponding grid values. Thus, each evaluation of **T** or **U** costs only $O(1)$ and consequently, we construct in this manner the entire matrix representation of $\mathcal{M}_n$ at an optimal $O(N_B^2)$ cost. Table 2 gives the CPU time to obtain the matrix in units of FE/BE, when $N_B = 2N$.

Table 2
Average CPU time in FE/BE units to construct the matrix approximation to $\mathcal{M}_n$ for given $N_B = 2N$.

$N_B = 2N$	256	512	1024
Matrix construction	1/3.3	1/1.8	1/2.5

5.2. Error estimate for approximation of the matrix representation of $\mathcal{M}_n$

We proceed to obtain an error estimate for the above approximation of $\mathcal{M}_n$. In what follows we assume the periodic domain is $\Omega = [0,1] \times [0,1]$ and consider a uniform grid $\mathcal{G}_\Omega$ with grid size $h = 1/N$. We first require a bound on G_h, the Green's function used in (34). Given the vector fields

$$\mathbf{g}_1(\mathbf{x}) = \begin{cases} (1/h^2, 0) & \text{if } \mathbf{x} = \mathbf{0}, \\ (0,0) & \text{otherwise}, \end{cases} \tag{50}$$

and

$$\mathbf{g}_2(\mathbf{x}) = \begin{cases} (0, 1/h^2) & \text{if } \mathbf{x} = \mathbf{0}, \\ (0,0) & \text{otherwise}, \end{cases} \tag{51}$$

defined for $\mathbf{x} \in \mathcal{G}_\Omega, G_h$ is a 2×2 matrix valued function given by

$$G_h(\mathbf{x}) = \begin{pmatrix} u_1(\mathbf{x}) & u_2(\mathbf{x}) \\ v_1(\mathbf{x}) & v_2(\mathbf{x}) \end{pmatrix}, \tag{52}$$

where $(u_1, v_1) = \mathcal{L}_h(\mathbf{g}_1)$ and $(u_2, v_2) = \mathcal{L}_h(\mathbf{g}_2)$.

We make the following claims about the four components of G_h. In the estimates that follow, C stands for a generic constant, not necessarily the same.

Lemma 5.1. *If Δt is proportional to h then the* sup *norm of each component of G_h over $\mathcal{G}_\Omega$ is bounded asymptotically by $C\Delta t^{-1}\log(h^{-1})$, where C is a constant.*

Proof. By definition of the fluid operator $\mathcal{L}_h$, we have

$$(u_1, v_1) = (I - \nu\Delta t L_h)^{-1} P_h \mathbf{g}_1. \tag{53}$$

The right hand side of (53) can be readily obtained in Fourier space. Let $\widetilde{\mathcal{G}}_\Omega = \{\mathbf{x} = (x,y)^T : \mathbf{x} \in \mathcal{G}_\Omega, x \neq 1, y \neq 1\}$ and $\mathcal{G}_F = \{\mathbf{k} = (k_1,k_2)^T : |k_1|, |k_2| < N/2\}$. Then, we can define the discrete Fourier transform (DFT) of a doubly periodic function F on $\widetilde{\mathcal{G}}_\Omega$ by

$$\widehat{F}(\mathbf{k}) = \sum_{\mathbf{x}\in\widetilde{\mathcal{G}}_\Omega} F(\mathbf{x}) e^{-2\pi i \mathbf{x}\cdot\mathbf{k}} \tag{54}$$

for $\mathbf{k} \in \mathcal{G}_F$ with discrete Fourier inverse

$$F(\mathbf{x}) = \frac{1}{N^2} \sum_{\mathbf{k}\in\mathcal{G}_F} \widehat{F}(\mathbf{k}) e^{2\pi i \mathbf{x}\cdot\mathbf{k}} \tag{55}$$

for $\mathbf{x} \in \widetilde{\mathcal{G}}_\Omega$. Then, for $\mathbf{k} \in \mathcal{G}_F$ and $|\mathbf{k}| \neq 0$, direct calculation gives us:

$$|\widehat{P_h\mathbf{g}_1}(\mathbf{k})| = |\hat{\mathbf{g}}_1(\mathbf{k}) - |\widehat{D}_h|^{-2}\widehat{D}_h\widehat{D}_h \cdot \hat{\mathbf{g}}_1(\mathbf{k})| \leqslant |\hat{\mathbf{g}}_1(\mathbf{k})| = 1/h^2 \tag{56}$$

and we set the zero mode ($\mathbf{k} = \mathbf{0}$) of the discrete projection to zero. Using (53) together with the inverse DFT (55) we have

$$\|u_1\|_\infty = \frac{1}{N^2} \max_{\mathbf{x}\in\mathcal{G}_\Omega} \left| \sum_{\mathbf{k}\in\mathcal{G}_F} e^{2\pi i \mathbf{x}\cdot\mathbf{k}} \hat{u}_1(\mathbf{k}) \right| \leqslant \frac{1}{N^2} \sum_{\mathbf{k}\in\mathcal{G}_F} \left| \frac{1/h^2}{1 - \nu\Delta t \widehat{L_h}(\mathbf{k})} \right|, \tag{57}$$

Here the Fourier symbol of the five-point Laplacian L_h can be written as

$$-\widehat{L}_h(\mathbf{k}) = \frac{1}{h^2}[4 - 2\cos(2\pi k_1 h) - 2\cos(2\pi k_2 h)] = \frac{4}{h^2}\left[\sin^2(\pi k_1 h) + \sin^2(\pi k_2 h)\right]. \tag{58}$$

Furthermore, using the inequality $\sin(x/2) \geqslant x/\pi$ for $0 \leqslant x \leqslant \pi$ and bounding the resulting sum by an integral, we get

$$\|u_1\|_\infty \leqslant \sum_{\mathbf{k}\in\mathcal{G}_F} \frac{1}{1 + 16\nu\Delta t|\mathbf{k}|^2} \leqslant C\int_0^{\frac{\sqrt{2}}{2h}} \frac{1}{1 + 16\nu\Delta t r^2} r\,dr = \frac{C}{16\nu\Delta t}\log[1 + 8\nu\Delta t h^{-2}]. \tag{59}$$

Thus, if $\Delta t \propto h$ then $\|u_1\|_\infty \leqslant C\Delta t^{-1}\log(h^{-1})$ for sufficiently small h. The remaining three components of G_h can be shown to have the same type of bound via similar calculations. □

Consider now two fiber points, say $\mathbf{X}_1$ and $\mathbf{X}_2$, with a point horizontal force on $\mathbf{X}_2$ of unit magnitude. If we spread this force we obtain a vector field $(f, 0)$ where

$$f(\mathbf{x}) = \delta_h(\mathbf{x} - \mathbf{X}_2)h_B. \tag{60}$$

Substituting this vector field into our fluid solver we obtain $(u, v) = \mathcal{L}_h(f, 0)$, where

$$u(\mathbf{x}) \equiv \sum_{\mathbf{x}_2 \in \mathcal{G}_\Omega} u_1(\mathbf{x} - \mathbf{x}_2)\delta_h(\mathbf{x}_2 - \mathbf{X}_2)h^2 h_B, \tag{61}$$

and u_1 is one of the components of the discrete Green's function (52). The interpolated velocity of u at $\mathbf{X}_1$ is

$$u(\mathbf{X}_1) \equiv \sum_{\mathbf{x}_1 \in \mathcal{G}_\Omega} u(\mathbf{x}_1)\delta_h(\mathbf{x}_1 - \mathbf{X}_1)h^2. \tag{62}$$

The value $A_{1,2} \equiv \alpha u(\mathbf{X}_1)$ is exactly the horizontal displacement of $\mathbf{X}_1$ induced by a unit horizontal force on $\mathbf{X}_2$, where $\alpha = (\Delta t)^2/\rho$ as before. $A_{1,2}$ is one of four entries in the matrix representation of $\mathcal{M}_n$ relating forces at $\mathbf{X}_2$ to displacement at $\mathbf{X}_1$. We will focus only on $A_{1,2}$. The remaining three values may be analyzed in a similar manner.

We dub our approximation to $A_{1,2}$ by $\widetilde{A}_{1,2}$. We obtain our approximation by shifting both $\mathbf{X}_1$ and $\mathbf{X}_2$ an equal amount such that $\mathbf{X}_2$ lies exactly on an Eulerian intersection. That is we shift by $\mathbf{r} \in [-h/2, h/2) \times [-h/2, h/2)$ such that $\mathbf{X}_2 + \mathbf{r} \in \mathcal{G}_\Omega$. We then proceed as before, starting with a unit horizontal force at $\mathbf{X}_2 + \mathbf{r}$. Spreading this force results in a vector field with x-component given by

$$\tilde{f}(\mathbf{x}) = \delta_h(\mathbf{x} - \mathbf{X}_2 - \mathbf{r})h_B. \tag{63}$$

Setting $(\tilde{u}, \tilde{v}) = \mathcal{L}_h(\tilde{f}, 0)$ we have

$$\tilde{u}(\mathbf{x}) \equiv \sum_{\mathbf{x}_2 \in \mathcal{G}_\Omega} u_1(\mathbf{x} - \mathbf{x}_2)\delta_h(\mathbf{x}_2 - \mathbf{X}_2 - \mathbf{r})h^2 h_B. \tag{64}$$

Finally we obtain our proposed approximation via $\widetilde{A}_{1,2} \equiv \alpha\tilde{u}(\mathbf{X}_1 + \mathbf{r})$ where

$$\tilde{u}(\mathbf{X}_1 + \mathbf{r}) \equiv \sum_{\mathbf{x}_1 \in \mathcal{G}_\Omega} \tilde{u}(\mathbf{x}_1)\delta_h(\mathbf{x}_1 - \mathbf{X}_1 - \mathbf{r})h^2. \tag{65}$$

Let M_n be the exact matrix representation of $\mathcal{M}_n$ and $\widetilde{M}_n$ be the approximate matrix representation of $\mathcal{M}_n$ arrived at via the methods given above. We are then in a position to state an estimate for the error $\|M_n - \widetilde{M}_n\|_\infty$.

Theorem 5.1. *$\|M_n - \widetilde{M}_n\|_\infty \leqslant Ch^2 \log h^{-1}$ when both Δt and h_B are proportional to h and where C is a constant.*

Proof. Using (61) and (62), the fact that $A_{1,2} \equiv \alpha u(\mathbf{X}_1)$, and (64) and (65) we have

$$A_{1,2} - \widetilde{A}_{1,2} = \alpha \sum_{\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{G}_\Omega} u_1(\mathbf{x}_1 - \mathbf{x}_2)[\delta_h(\mathbf{x}_1 - \mathbf{X}_1)\delta_h(\mathbf{x}_2 - \mathbf{X}_2) - \delta_h(\mathbf{x}_1 - \mathbf{X}_1 - \mathbf{r})\delta_h(\mathbf{x}_2 - \mathbf{X}_2 - \mathbf{r})]h^4 h_B, \tag{66}$$

hence,

$$|A_{1,2} - \widetilde{A}_{1,2}| \leqslant \alpha\|u_1\|_\infty \sum_{\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{G}_\Omega} [\delta_h(\mathbf{x}_1 - \mathbf{X}_1)\delta_h(\mathbf{x}_2 - \mathbf{X}_2) + \delta_h(\mathbf{x}_1 - \mathbf{X}_1 - \mathbf{r})\delta_h(\mathbf{x}_2 - \mathbf{X}_2 - \mathbf{r})]h^4 h_B = \frac{2(\Delta t)^2}{\rho}\|u_1\|_\infty h_B, \tag{67}$$

where we have made use of the identity $\sum_{\mathbf{x} \in \mathcal{G}_\Omega} \delta_h(\mathbf{x} - \mathbf{y})h^2 = 1$ for any shift $\mathbf{y} \in \Omega$ in the last step of the estimate. Suppose that $\Delta t \propto h$ and $h_B \propto h$, then from Lemma 5.1 we get that $|A_{1,2} - \widetilde{A}_{1,2}| \leqslant Ch^2 \log h^{-1}$, for sufficiently small h.

It is straightforward to extend the above estimate to all entries in the matrix $M_n - \widetilde{M}_n$. This gives the desired $\|M_n - \widetilde{M}_n\|_\infty \leqslant Ch^2 \log h^{-1}$. □

This error for the proposed, approximated matrix $\widetilde{M}_n$ is asymptotically smaller than the $O(h)$ error of the IB Method for points within a distance $O(h)$ of the immersed boundary [12]. Thus, the approximated matrix $\widetilde{M}_n$ can be use without any deterioration of the overall accuracy of the IB Method.

We calculate numerically both matrices and then compute the maximum norm between their difference for the standard example of a relaxing elliptical bubble as described in Section 6. A log–log graph of the norm with respect to h is given in Fig. 1. Here, we have fixed $\Delta t = h$ and $N_B = 2N$. We see that the error is approximately $O(h^2)$, a close match to the analytic result.

5.3. Additional considerations and optimizations concerning $\widetilde{M}_n$

There are a few additional optimizations possible when constructing $\widetilde{M}_n$, the matrix representation of $\mathcal{M}_n$. Note first that on a square domain Ω we have via symmetry that $\mathbf{U} = \mathbf{T}$, so only a single lookup table is required. For rectangular or otherwise irregular grids this symmetry does not hold, we do however have other symmetries; most importantly $\mathbf{T}(\mathbf{x}) = \mathbf{T}(-\mathbf{x})$ and $\mathbf{U}(\mathbf{x}) = \mathbf{U}(-\mathbf{x})$. These symmetries imply that the matrix of $\mathcal{M}_n$ is symmetric. This observation reduces the cost of computing the matrix roughly by half. The cost could be reduced further if interpolation were not used between grid points for evaluating values of $\mathbf{T}$ and $\mathbf{U}$. Unfortunately, the resulting error introduced in the simulation would be significant. A simple compromise is to use linear interpolation when calculating $\mathbf{T}(\mathbf{x})$ if $|\mathbf{x}|$ is small and direct lookup otherwise. This is inexpensive, with only $O(N_B)$ interpolations needed, because most fiber points are distant from each other, as well as accurate, be-

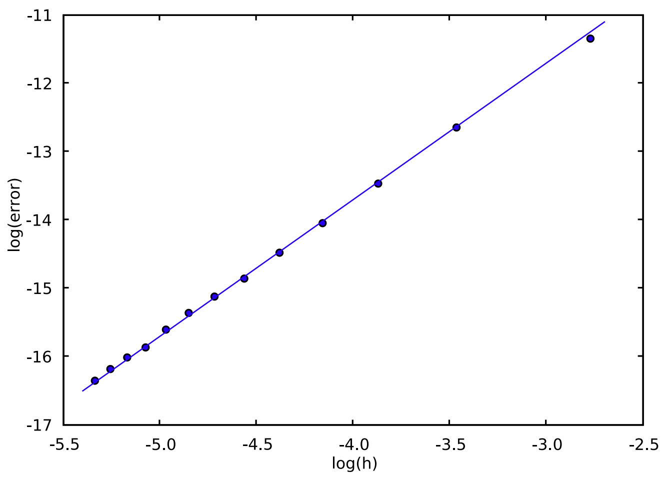

Fig. 1. Log–log plot of $\|M_n - \widetilde{M}_n\|_\infty$ with $\Delta t = h$. The continuous line is a fit with a line of slope 2.

cause **T** and **U** and decay rapidly away from the origin. Alternatively, we could employ an adaptive mesh on which to calculate **T** and **U**, taking a dense grid around the origin where most of the structure lies. This would allow us to use direct look-up for all entries of $\mathcal{M}_n$ without much loss of accuracy. We do not pursue these strategies here and only utilize the symmetry of the matrix of $\mathcal{M}_n$ to expedite the construction as was done in the costs given in Table 2.

A further reduction in cost can be had by only computing part of $\mathcal{M}_n$. Consider the case where $\mathcal{A}_{h_B}$ is linear, represented by the matrix A_{h_B}, and suppose M_n is the matrix of $\mathcal{M}_n$. We approximate M_n with a banded matrix M_n^1 and let $M_n^2 = M_n - M_n^1$. Our linear system then looks like

$$\mathbf{X}^{n+1} = M_n^1 A_{h_B}\mathbf{X}^{n+1} + M_n^2 A_{h_B}\mathbf{X}^{n+1} + \mathbf{b}^n. \tag{68}$$

We approximate $M_n^2 A_{h_B}\mathbf{X}^{n+1} \approx M_n^2 A_{h_B}\mathbf{X}^n$. Utilizing this and rewriting we have

$$\mathbf{X}^{n+1} - \mathbf{X}^n = M_n^1 A_{h_B}(\mathbf{X}^{n+1} - \mathbf{X}^n) + M_n A_{h_B}\mathbf{X}^n + \mathbf{b}^n. \tag{69}$$

We may absorb the term $M_n A_{h_B}\mathbf{X}^n$ into the definition of $\mathbf{b}^n$ leaving the system

$$\mathbf{X}^{n+1} - \mathbf{X}^n = M_n^1 A_{h_B}(\mathbf{X}^{n+1} - \mathbf{X}^n) + \tilde{\mathbf{b}}^n, \tag{70}$$

where

$$\tilde{\mathbf{b}}^n = \Delta t \mathcal{S}_n^* \mathcal{L}_h \left[\frac{\Delta t}{\rho}\mathcal{S}_n \mathcal{A}_{h_B}(\mathbf{X}^n) + \mathbf{u}^n - \Delta t \mathbf{u}^n \cdot \nabla_h \mathbf{u}^n\right]. \tag{71}$$

Solving (70) and (71) yields a stable update provided that the width of the band comprising M_n^1 is large enough. Typically, the width of the band must increase as stiffness increases to maintain stability. This banded matrix approach can lead to a significant speed-up in calculating the system's matrix as well as in matrix–vector multiplication if sparse data structures are used. This is particularly helpful for problems with very large physical domains, where the influence of one immersed fiber onto a distant one is drastically diminished by their large separation. For instance, in the 2D model of a heart valve presented in Section 8, much of the horizontal boundary (walls) has little impact on the fiber comprising the valve. The walls, modeled with tethered nodes, are not the main contributors to the high stiffness of the problem. The corresponding entries of M_n relating these two horizontal parts of the immersed boundary could be held at zero if we employ (70) and (71) to determine the next configuration.

6. Case I: an initially elliptical drop with linear $\mathcal{A}_{h_B}$

We consider as our first example what has been the canonical test for methods intended to remove the severe stiffness of the IB method. This is the case of a closed, continuous membrane $\mathbf{X}(s,t)$ with a force distribution given by $\mathcal{A}(\mathbf{X}) = \sigma\mathbf{X}_{ss}$, where σ is a (large) constant. We take our domain as $\Omega = [0,1]\times[0,1]$ with periodic boundary conditions and we fix $\mu = \rho = 1$. Initially, we have an elliptical drop and zero velocity and the drop relaxes toward the equilibrium configuration. Fig. 2 shows the initial and final configurations of the interface.

We discretize $\mathcal{A}(\mathbf{X})$ as

$$(\mathcal{A}_{h_B}\mathbf{X})_i = \frac{1}{h_B^2}(\mathbf{X}_{i+1} - 2\mathbf{X}_i + \mathbf{X}_{i-1}), \tag{72}$$

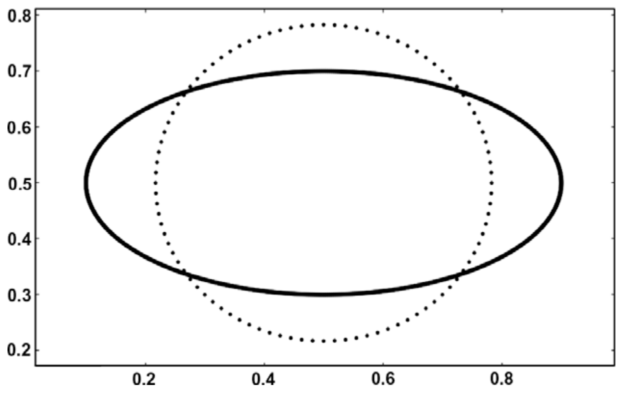

Fig. 2. Initial configuration of fiber in bold and final rest configuration in dotted line.

where we have omitted the parentheses in the discrete force operator $\mathcal{A}_{h_B}$ to emphasize that it is linear. Thus, Peskin's semi-implicit discretization with lagged-operators produces the linear system

$$\mathbf{X}^{n+1} = \mathcal{M}_n \mathcal{A}_{h_B} \mathbf{X}^{n+1} + \mathbf{b}^n, \tag{73}$$

where $\mathbf{b}$ is given by (26), to be solved at each time step. Equivalently, we have

$$(I - \mathcal{M}_n \mathcal{A}_{h_B})\mathbf{X}^{n+1} = \mathbf{b}^n, \tag{74}$$

where I is the identity operator. Note that $I - \mathcal{M}_n \mathcal{A}_{h_B}$ is symmetric and non-singular. Indeed, the argument in [4] for $I - S_n^* P_h S_n \mathcal{A}_{h_B}$ can be applied to this case as well, for suppose

$$(I - \mathcal{M}_n \mathcal{A}_{h_B})\mathbf{X} = 0, \tag{75}$$

then multiplying both sides of (75) by $\mathcal{A}_{h_B}$ and taking the inner product with $\mathbf{X}$ we get

$$0 = -(\mathbf{X}, \mathcal{A}_{h_B}\mathbf{X})_B + (\mathbf{X}, \mathcal{A}_{h_B}\mathcal{M}_n\mathcal{A}_{h_B}\mathbf{X})_B = -(\mathbf{X}, \mathcal{A}_{h_B}\mathbf{X})_B + (\mathcal{A}_{h_B}\mathbf{X}, \mathcal{M}_n\mathcal{A}_{h_B}\mathbf{X})_B. \tag{76}$$

Both terms in the left hand side of (76) are non-negative. Therefore, it follows that each has to be equal to zero and consequently

$$(\mathbf{X}, \mathcal{A}_{h_B}\mathbf{X})_B = 0. \tag{77}$$

This implies that $\mathcal{A}_{h_B}\mathbf{X} = 0$ [4] and hence from (75) we obtain that $\mathbf{X} = 0$. Thus (74) has a unique solution.

If the full matrix representation M_n of $\mathcal{M}$ is available (see the previous section for how to expedite its construction) it is easy to construct simple iterative schemes to solve (74). For example, denoting the diagonal component of M_n by M_n', letting $M_n'' = M_n - M_n'$, and denoting by A_{h_B} the matrix representation of $\mathcal{A}_{h_B}$ we can write an iteration of weighted Jacobi type:

$$\mathbf{X}^{n+1,k+1} = (1-a)\mathbf{X}^{n+1,k} + a(I - M_n' A_{h_B})^{-1}(\mathbf{b}^n + M_n'' A_{h_B}\mathbf{X}^{n+1,k}). \tag{78}$$

When underrelaxed, $0 < a < 1$, this iteration converges well, typically requiring on the order of 10 iterations to obtain adequate residuals (on the order of the truncation error) and to maintain stability. Along the same lines, one can also consider a weighted Gauss–Seidel method of the form:

$$\mathbf{X}_i^{n+1,k+1} = (1-a)\mathbf{X}^{n+1,k} + a\left(\frac{1}{B_{ii}}\left(\mathbf{b}_i^n - \sum_{j<i} B_{ij}\mathbf{X}_j^{n+1,k+1} - \sum_{j>i} B_{ij}\mathbf{X}_j^{n+1,k}\right)\right), \tag{79}$$

for $i = 1, 2, \ldots, 2N_B$, where $B = (I - M_n A_{h_B})$ and $0 < a < 1$. While (79) appears to converge slower than (78) in the numerical experiments it does, however, perform well as a smoother.

The most efficient and robust approach we found to solve (74) for this test problem is a standard algebraic multigrid. Suppose for simplicity that $N_B = 2^m$ for some natural number m. We can then form a collection of Lagrangian grids $\mathcal{G}^l$, $l = 1, \ldots m$, with number of nodes equal to $2^m, 2^{m-1}, \ldots, 2, 1$. Our original grid corresponds to the first level, $l = 1$, with coarser grids coming afterwards. For a given level l of the multigrid hierarchy, the prolongation operator $\mathcal{P}_l$ takes a fiber $\mathbf{X}_{l+1}$ at level $l+1$ and adds a node between each pair of consecutive nodes equidistant between each. As a matrix, $\mathcal{P}_l$ has dimensions $2 \cdot 2^{m-l+1} \times 2 \cdot 2^{m-l}$ and has the form

$$\mathcal{P}_l = \begin{pmatrix} 1 & 0 & 0 & \\ .5 & .5 & 0 & \dots \\ 0 & 1 & 0 & \\ 0 & .5 & .5 & \\ & \vdots & & \ddots \end{pmatrix}. \tag{80}$$

Restriction operators are taken to be the transposes of prolongation operators. Calling $\mathcal{K}_1 = (I - \mathcal{M}_n \mathcal{A}_{h_B})$ we define recursively $\mathcal{K}_l = \mathcal{P}_{l-1}^T \mathcal{K}_{l-1} \mathcal{P}_{l-1}$. The Gauss–Seidel iteration (79) provides a good smoother for the multigrid. All together, a single restriction, prolongation, and smoothing step looks like

- Given a guess $\mathbf{X}_l$ to the linear problem

$$\mathcal{K}_l \mathbf{X}_l = \mathbf{b}_l. \tag{81}$$

calculate the residual and restrict it to the next lowest level

$$\tilde{\mathbf{r}}_{l+1} = \mathcal{P}_l^T (\mathbf{b}_l - \mathcal{K}_l \mathbf{X}_l). \tag{82}$$

- Find the correction on the coarse grid by solving or approximating

$$\mathcal{K}_{l+1} \mathbf{X}_{l+1} = \tilde{\mathbf{r}}_{l+1}. \tag{83}$$

- Correct our initial guess

$$\mathbf{X}_l \leftarrow \mathbf{X}_l + \mathcal{P}_l \mathbf{X}_{l+1}. \tag{84}$$

- Smooth the high frequency errors in $\mathbf{X}_l$ via an underrelaxed Gauss–Seidel.

The number of iterations needed depends more on the desired degree of accuracy than a requirement for stability. Typically a single iteration of a full multigrid step with one V-cycle per level is sufficient to maintain stability. Computationally, it appears to be preferable to use the approximate solution $\mathbf{X}^{n+1}$ we obtain from (74) as the updated fiber position rather than applying (11) directly. This is because small errors in the solution to (74) are amplified when $\mathcal{A}_{h_B}$ acts on it.

6.1. Numerical results

We present now the numerical results for the above test model. The method discussed has five primary costs. First, when constructing the implicit system (74) we must calculate $\mathbf{b}^n$ via a fluid solve, with a cost of roughly one FE/BE. Second, the cost of computing M_n is roughly one half FE/BE. Third is the cost of initializing the multigrid solver, i.e. calculating the coarse representations of M_n on the various levels of our grid. Use of sparse data structures is critical here to maintain $O(N_B^2)$ cost. Fourth, is the cost of our iterations to solve the linear system. Because matrix–vector multiplication is significantly cheaper than FE/BE (see Table 1) a full multigrid step can be much more economical than one FE/BE and this advantage grows as N increases. The final cost is in computing $\mathbf{u}^{n+1}$ once $\mathbf{X}^{n+1}$ is known. This step involves another fluid solve with a cost of roughly one FE/BE. We will see that the sum of these costs for the semi-implicit method results in a single time-step that costs approximately 4 FE/BE. However, the stability restraint on Δt for the FE/BE scheme for a stiff problem like this one is orders of magnitude smaller than that required by the semi-implicit discretization. Thus, our proposed approach yields a much superior computational strategy.

We set the elasticity constant $\sigma = 10^5$ and take $N_B = 2N$. The initial configuration of the fiber is an ellipse given by

$$\mathbf{X}(s,0) = (0.5 + 0.3\cos(2\pi s h_B), 0.5 + 0.2\sin(2\pi s h_B)), \tag{85}$$

for $s \in [0,1]$.

As time elapses, the interfacial tension drives the ellipse toward a circular (cylindrical) configuration. The velocity of the fiber can reach roughly 500 units before it slows down toward equilibrium. Taking a standard length to be the geometric mean of our radii, 0.245, we calculate the Reynolds number of our fluid to be approximately 100.

As a reference for comparison, for fiber-explicit simulations we use the FE/BE scheme with Δt as large as stability permits, which we find to be approximately $\Delta t = 0.00025h$. For our implicit scheme we make use of a linear multigrid with stopping criteria of $\|\mathbf{r}^k\|_\infty < 30\Delta t$, where $\mathbf{r}^k$ is the residual $\mathbf{b}^n - (I - M_n A_{h_B})\mathbf{X}^{n+1,k}$ at the kth iterate and $\|\cdot\|_\infty$ is the sup norm. Here Δt is chosen to comply with the CFL condition as the convection term is treated explicitly. With a conservative estimate for the maximum velocity, the time-step is given approximately by $\Delta t = 0.02h$.

The results of multiple simulations with varying N for the explicit and implicit methods are given in Table 3. The rows in the table correspond to identical simulation runs with different N. The columns under the title Explicit relate data from the FE/BE simulations, whereas the Implicit columns relate data from the implicit simulations. The two columns marked Average give the average CPU time of a single timestep. The columns marked Total give total CPU time for the entire simulation, up to a simulation time of $T = 0.005$.

Table 3
Elliptical drop relaxation for Navier–Stokes. The average CPU time per time-step and the total CPU time up to a simulation time of $T = 0.005$ is given. Δt is the time-step taken and is the maximum allowed while maintaining stability.

N	Explicit			Implicit		
	Δt	Average	Total	Δt	Average	Total
128	1.95×10^{-6}	0.03	73.35	1.17×10^{-4}	0.10	4.34
256	9.76×10^{-7}	0.14	730.21	7.81×10^{-5}	0.55	35.06
384	6.51×10^{-7}	0.34	2613.19	5.21×10^{-5}	1.24	118.98
512	4.87×10^{-7}	0.63	6491.95	3.91×10^{-5}	2.34	299.10

We see in Table 3 that the implicit scheme costs approximately 4 times as much per time-step than FE/BE. However, because Δt can be taken up to the CFL restraint, the scheme is roughly 20 times faster than the explicit scheme in terms of total CPU time. The performance of the semi-implicit approach is limited by the presence of large convection (relative to viscous dissipation) but the method still achieves a significant gain over the explicit alternative. Of course, one could employ a suitable implicit discretization of the convection terms to remove the CFL constraint. Effective options in this context have been proposed by Mori and Peskin [9] and Hou and Shi [7]. We do not pursue this here as our intent is to focus on the treatment of the tension forces. Alternatively, if we consider Stokes flow the semi-implicit method is no longer limited by a CFL restriction and we can take arbitrarily sized time-steps while maintaining stability. We compare the performance of the two methods for Stokes flow in Table 4.

In these simulations we have increased the total simulation time to $T = 0.05$. This simulation time requires unreasonable amounts of CPU time when using the explicit method. We estimate the total CPU time for these longer, explicit simulations by calculating the average CPU time of a single timestep and multiplying by the total number of timesteps. We use an asterisk here and in following tables to denote values based on these extrapolated values.

While Δt for our implicit simulation can be chosen with disregard to stability, care must still be taken to ensure accuracy. Here, we only require that $\Delta t \leqslant h$ to maintain the overall $O(h)$ accuracy of the method.

7. Case II: an initially elliptical drop with nonlinear $\mathcal{A}_{h_B}$

We consider now the case of the simple, initially elliptical membrane with a nonlinear force and detail how to adapt the method from the previous linear case.

We follow Mori and Peskin in [9] for the setup of this nonlinear $\mathcal{A}_{h_B}$ example. At the continuous level, the fiber force distribution is given by

$$\mathbf{F} = \frac{\partial}{\partial s}(T\mathbf{t}), \tag{86}$$

where $\mathbf{t}(s,t)$ is the tangential unit vector to $\mathbf{X}(s,t)$ and $T(s,t)$ is the tension given by

$$T(s,t) = \left|\frac{\partial \mathbf{X}}{\partial s}(s,t)\right| + \left|\frac{\partial \mathbf{X}}{\partial s}(s,t)\right|^2. \tag{87}$$

To discretize this interfacial force we introduce the following difference operators for a function f defined over $\mathcal{G}_B$:

$$D_s^+ f(s) = \frac{f(s+1) - f(s)}{h_B}, \tag{88}$$

$$D_s^- f(s) = \frac{f(s) - f(s-1)}{h_B}. \tag{89}$$

Table 4
Ellipse relaxation for Stokes flows. The average CPU time per timestep and the total CPU time up to a simulation time of $T = 0.05$ is given. Δt is the timestep taken. For the explicit scheme this is the maximum allowed while maintaining stability. The implicit scheme is unconditionally stable; the timestep taken is held constant as N varies.

N	Explicit			Implicit		
	Δt	Average	Total	Δt	Average	Total
128	1.95×10^{-6}	0.03	683.96	0.001	0.09	9.34
256	9.76×10^{-7}	0.15	7410.50	0.001	0.53	53.08
384	6.51×10^{-7}	0.35	26555.30*	0.001	1.24	123.86
512	4.87×10^{-7}	0.71	72669.41*	0.001	2.32	232.05

* Denotes an extrapolated value.

We then define the discrete force distribution as

$$\mathbf{F} = D_s^+\left((|D_s^-\mathbf{X}| + |D_s^-\mathbf{X}|^2)\frac{D_s^-\mathbf{X}}{|D_s^-\mathbf{X}|}\right). \tag{90}$$

Severe stiffness in this model is manifested when the interfacial elastic force is much larger than the viscous forces. Following Mori and Peskin [9] we set $\rho = 1$ and either $\mu = 0.05$ or $\mu = 0.005$. As before the domain is $\Omega = [0,1] \times [0,1]$ which we discretize as Ω_h, an $N \times N$ uniform Eulerian grid. As in the linear case, the initial velocity of the fluid is zero everywhere. The fiber's initial configuration is given by

$$\mathbf{X}(s,0) = \left(\frac{1}{2} + \frac{1}{3}\cos(2\pi s h_B), \frac{1}{2} + \frac{1}{4}\sin(2\pi s h_B)\right). \tag{91}$$

With our implicit strategy we attempt to solve at each time-step the *nonlinear* system (23)

$$\mathbf{X}^{n+1} = \mathcal{M}_n \mathcal{A}_{h_B}(\mathbf{X}^{n+1}) + \mathbf{b}^n,$$

where $\mathcal{M}_n$ and $\mathbf{b}^n$ are as in the linear case. We could simply approximate $\mathcal{A}_{h_B}$ linearly and solve the resulting linear system. This is easily accomplished because the Jacobian J of $\mathcal{A}_{h_B}$ is negative semidefinite, hence $I - \mathcal{M}_n J$ is positive definite. Such linear approximation is simple and reasonably robust but not as stable as the implicit scheme with full nonlinear term. This linear approximation is the method employed by Mori and Peskin [9] for their first order scheme. However, with our economical computational approach we can afford the extra stability (while maintaining accuracy) by solving the nonlinear system (23) via Newton iterations. Specifically, at each time-step we perform Newton iterations until the sup norm of the residual is less than 10^{-4}. This typically requires only 2 or 3 iterations. Note that a single iteration per timestep would be identical to the linear approximation used by Mori and Peskin. We expedite further the computation by approximating the solution to the linear system at each one of Newton's iteration with 3 Multigrid cycles.

7.1. Numerical results

We choose this nonlinear test problem following Mori and Peskin [9] to have their results as a reference. However, there is a slight difference in the model in addition to the different discretizations employed; Mori and Peskin use an immersed fiber with finite mass while our own fiber is neutrally buoyant. We present the results from our numerical experiments in a format similar to that presented in [9] to facilitate comparisons.

Let N_T be the number of timesteps taken, with total simulation time fixed at 1. We fix $N_B = 2N$ as before. Table 5, for $\mu = 0.05$, and Table 6, for $\mu = 0.005$, summarize the total computational cost in units of FE/BE's. The total CPU time for the same cases is displayed in Tables 7 and 8, respectively. In all four tables the columns under FE/BE hold the explicit result while the columns under $N_T = 8$ and $N_T = 16$ hold the implicit results, with N_T as specified.

As noted in [9], the semi-implicit method becomes more efficient compared to the explicit (FE/BE) scheme as N increases and μ decreases (more stiffness). For $N_T = 8$ and $N = 512$ the proposed semi-implicit strategy is about 45 times faster than the explicit approach in the case of $\mu = 0.05$ and about 90 times faster for $\mu = 0.005$. In contrast, the fully implicit approach in [9] gives cost ratios in the range 12–16 for the same parameters.

8. Case III: a model of a heart valve

8.1. The model

We turn now to a more challenging application of the IB Method in which there are rigid immersed structures, tethered points, and crossed links. We consider a 2D model of a rigid valve immersed in blood flowing through an artery. The valve is indirectly restricted in motion by two hinges but is allowed to rotate. The valve, artery walls, and hinges will all be modeled as immersed springs. The flow-structure interaction will be captured via the IB Method.

We select the computational domain to be $\Omega = [0,2] \times [0,1]$ with periodic boundary conditions and discretize it with a $2N \times N$ uniform grid. The geometry of the problem is represented in Fig. 3 and a detailed depiction of the valve's linked structure is shown in Fig. 4. Table 9 details the number of fiber points necessary to construct the geometry given the Eulerian resolution N. The top and bottom of the artery walls include cushions in the shape of two hills. We simulate a horizontal

Table 5
Total CPU cost for the nonlinear ellipse model with $\mu = 0.05$. Values given are total CPU time divided by average CPU time of a single FE/BE timestep for the given N.

N	FE/BE	$N_T = 8$	$N_T = 16$
64	166	38.2	75.5
128	500	49.7	89.7
256	1333	53.0	97.2
512	2666	59.3	92.6

Table 6
Total CPU cost for the nonlinear ellipse model with $\mu = 0.005$. Values given are total CPU time divided by average CPU time of a single FE/BE timestep for the given N.

N	FE/BE	$N_T = 8$	$N_T = 16$
64	333	83.5	112.4
128	1000	74.8	113.1
256	2666	71.7	112.6
512	5333	59.2	101.9

Table 7
Total CPU time for the nonlinear ellipse model with $\mu = 0.05$.

N	FE/BE	$N_T = 8$	$N_T = 16$
64	2.94	0.67	1.33
128	20.89	2.08	3.75
256	227.85	9.05	16.59
512	1930.62	42.94	67.02

Table 8
Total CPU time for the nonlinear ellipse model with $\mu = 0.005$.

N	FE/BE	$N_T = 8$	$N_T = 16$
64	5.31	1.33	1.79
128	42.06	3.14	4.75
256	455.11	12.24	19.22
512	3823.40	42.45	73.10

Table 9
The number of Lagrangian nodes N_B and the maximum stable timestep Δt for a FE/BE method are given for increasing values of N.

N	N_B	Δt
128	520	1.15×10^{-7}
256	1053	2.89×10^{-8}
384	1592	1.29×10^{-8}
512	2122	7.24×10^{-9}

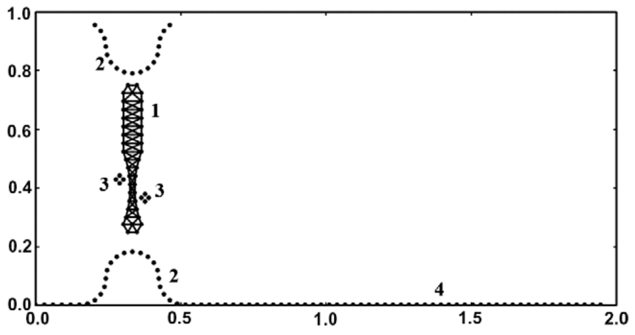

Fig. 3. Configuration of heart valve model. Larger nodes represent tether points. 1: Valve; 2: Cushions; 3: Hinges; 4: Artery wall.

flow through the artery by adding a forcing vector $\mathbf{f}_{j,k} = (v_{flow}, 0)$ to the right hand side of (9), where in general v_{flow} may be time dependent. This changes the explicit term $\mathbf{b}^n$ in our implicit system to

$$\mathbf{b}^n = \mathbf{X}^n + \Delta t \mathcal{S}_n^* \mathcal{L}_h \left[\mathbf{u}^n - \Delta t \mathbf{u}^n \cdot \nabla_h \mathbf{u}^n + \frac{\Delta t}{\rho} \mathbf{f} \right]. \tag{92}$$

All nodes in the wall and the two hinges are modeled as tethers, with one end of the tether fixed for all time. If $\mathbf{X}_i$ is a tethered point with base given by $\overline{\mathbf{X}}_i$ then the force generated by the tether is given by

$$\mathbf{f}_i = -k_i(\mathbf{X}_i - \overline{\mathbf{X}}_i), \tag{93}$$

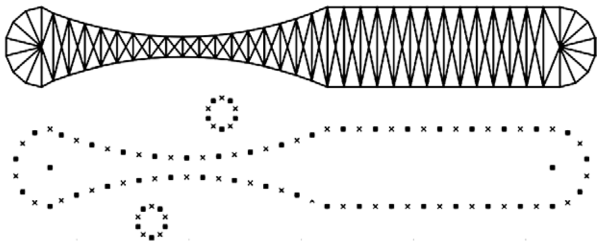

Fig. 4. Top: A fine grid approximation to the valve, showing only linkage. Bottom: 'x's mark the prolongation of a coarse valve marked in 'o'.

where k_i is the spring constant. For the body of the heart valve we will use springs between nodes. Suppose $\mathbf{X}_i$ is connected to multiple other nodes. If $\mathbf{X}_i$ is connected to $\mathbf{X}_j$ then let $k_{i,j}$ and $L_{i,j}$ be respectively the spring constant and resting length of the spring connecting them. If no connection exists between $\mathbf{X}_i$ and $\mathbf{X}_j$ then take $k_{i,j}$ to be zero. The force at $\mathbf{X}_i$ then is given by

$$\mathbf{f}_i = \sum_{j=1}^{N_B} -k_{i,j} \frac{\mathbf{X}_i - \mathbf{X}_j}{|\mathbf{X}_i - \mathbf{X}_j|} \left(|\mathbf{X}_i - \mathbf{X}_j| - L_{i,j} \right). \tag{94}$$

Note that the force operator (72) employed for the elliptical membrane problem is exactly the force given by (94) for a loop of fiber points connected sequentially by springs with suitable spring constants and zero resting length. However, for our valve model we require non-zero resting lengths to preserve the structure and as a consequence we end up with a non-linear force density. Additionally, the problem is severely stiff. To maintain the rigidity of a solid body, we find that the spring constants must be $O(10^9 h^{-2})$. Similarly, for the tether points representing the artery walls, the spring constants must also be very large to portray the tautness of the biological fibers. Furthermore, because the forcing flow acts to bend the valve as it is penned between its hinges, larger values of v_{flow} require larger spring constants to preserve the structure. To illustrate the stiffness, a time integration of the equations of motion with an explicit treatment of the interfacial force and with a modest spatial resolution $(N = 256)$ requires the time-step to be $O(10^{-7})$.

Removing this stiffness for this more prototypal IB Method problem is considerably more challenging than in the case of the simple elliptical fiber. The implicit system we would like to solve is still given by (23):

$$\mathbf{X}^{n+1} = \mathcal{M}_n \mathcal{A}_{h_B}(\mathbf{X}^{n+1}) + \mathbf{b}^n.$$

As noted above, due to the nonzero resting lengths of the springs comprising the valve, $\mathcal{A}_{h_B}$ is nonlinear. Moreover, unlike the case of the elliptical interface with nonlinear force density, the Jacobian J of $\mathcal{A}_{h_B}$ is not semidefinite and the resulting matrix $I - M_n J$ can be shown to lack definiteness as well.

The breakdown of the definiteness of J can be seen in a very simple case. Consider four immersed boundary points at $(1,0)$, $(0,1)$, $(-1,0)$, $(0,-1)$, forming a square with side lengths $\sqrt{2}$ and with links connecting the perimeter of the square. We vary the resting length l of the connecting springs and compute the eight eigenvalues of the resulting force density's Jacobian. The results are plotted in Fig. 5. We see that as the resting length approaches and surpasses the side length of the square the

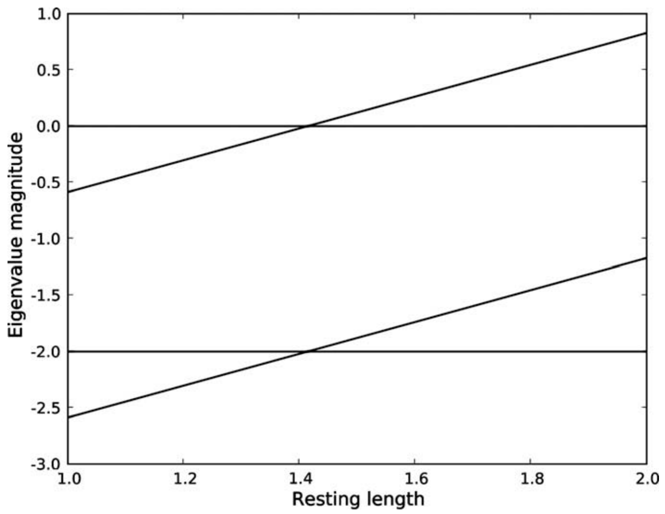

Fig. 5. Eigenvalues for the Jacobian of a fiber forcing function for four points linked as a square with varying resting length.

Jacobian loses its negative semi-definiteness. This seems to be generic for most immersed structures. For our 2D valve model, we take the resting length to be the starting length of the links comprising the valve. Thus, we lose immediately negative semi-definiteness once we perturb the valve structure.

Due to the lack of positive definiteness of $I - M_nJ$ the Conjugate Gradient Method does not converge. The Biconjugate Gradient Method converges but may take in excess of 100 iterations to attain a satisfactory residual. A major obstacle in accelerating convergence is that the natural preconditioner $I - M'_nJ$ (recall M'_n is the diagonal part of M_n) does not capture well the dynamics of this system. Indeed, a Jacobi iteration analogous to (78) has very poor convergence, requiring strong underrelaxation and thousands of iterations. Without adequate smoothers, an efficient multigrid for the linear system in Newton's iteration is not a viable option. Smoothers for the nonlinear system (23) were likewise difficult to uncover, seemingly ruling out a nonlinear multigrid. We propose next a very different approach, which consists of splitting the problem to take simultaneous advantage of the fast convergence of Newton's method and the efficiency of multigrid.

8.2. Solving the implicit system

Perhaps the simplest iterator for (23) is the fixed point iteration given by

$$\mathbf{X}^{n+1,k+1} = M_n\mathcal{A}_{h_B}(\mathbf{X}^{n+1,k}) + \mathbf{b}^n. \tag{95}$$

However, this method fails spectacularly. This is not entirely surprising. If our initial guess is $\mathbf{X}^{n+1,0} = \mathbf{X}^n$ then $\mathbf{X}^{n+1,1}$ is the explicit update provided by the FE/BE scheme which is unstable for practical Δt. Additional iterations of (95) exacerbate the instability: small errors in $\mathbf{X}^{n+1,k}$ are amplified enormously by $\mathcal{A}_{h_B}$ resulting in large errors in $\mathbf{X}^{n+1,k+1}$. As remarked in the context of the elliptical interface case, this is why it is preferable to use the approximate solution to (74) as the updated configuration $\mathbf{X}^{n+1}$ rather than employing (21).

The eigenvalues of J are at least on the order of the spring constants comprising our valve. It should seem natural to take advantage of this by reversing the fixed point iteration (95). Consider the alternative fixed point iteration

$$\begin{cases} M_n\mathbf{F}^{k+1} = \mathbf{X}^{n+1,k} - \mathbf{b}^n, \\ \mathcal{A}_{h_B}(\mathbf{X}^{n+1,k+1}) = \mathbf{F}^{k+1}. \end{cases} \tag{96}$$

First, we solve for $\mathbf{F}^{k+1}$, the force distribution that would move the fiber from $\mathbf{X}^n$ to $\mathbf{X}^{n+1,k}$ after a single timestep. The linear system we must solve only involves the linear positive definite operator $\mathcal{M}_n$, thus we can efficiently solve it using (linear)multigrid. Second, we determine what configuration $\mathbf{X}^{n+1,k+1}$ gives rise to this force through $\mathcal{A}_{h_B}$. In many applications there is a unique $\mathbf{X}^{n+1,k+1}$ such that $\mathcal{A}_{h_B}(\mathbf{X}^{n+1,k+1}) = \mathbf{F}^{k+1}$. For the current problem this is not the case.

To see this note that valve is modeled as a neutrally buoyant object, detached from any fixed points, and the internal forces generated by perturbations in the valve's structure are unaffected by translation. More precisely let $\mathcal{G}_V$ be all indices l such that $\mathbf{X}_l$ is a fiber point belonging to the valve. We consider then two translation vectors $\mathbf{V}^1$ and $\mathbf{V}^2$ given by

$$\mathbf{V}^1_l = \begin{cases} (1,0) & \text{if } l \in \mathcal{G}_V, \\ (0,0) & \text{otherwise}, \end{cases} \tag{97}$$

$$\mathbf{V}^2_l = \begin{cases} (0,1) & \text{if } l \in \mathcal{G}_V, \\ (0,0) & \text{otherwise}. \end{cases} \tag{98}$$

$\mathbf{V}^1$ and $\mathbf{V}^2$ are horizontal and vertical translation vectors, $\mathbf{X} + \mathbf{V}^1$ representing the same configuration as $\mathbf{X}$ with the valve shifted right by one unit. Shifting has no influence on the force density,

$$\mathcal{A}_{h_B}(\mathbf{X} + a_1\mathbf{V}^1) = \mathcal{A}_{h_B}(\mathbf{X} + a_2\mathbf{V}^2) = \mathcal{A}_{h_B}\mathbf{X}, \tag{99}$$

where a_1 and a_2 are arbitrary scalars. Thus, $\mathcal{A}_{h_B}$ is not injective and we may not have a unique solution to (96). In fact, $\mathcal{A}_{h_B}$ is not surjective either so (96) may have no solution at all. This follows physically from conservation of linear momentum and angular momentum: $\mathcal{A}_{h_B}$ cannot directly generate forces that move or rotate the valve. Translation and rotation can only be introduced via the fluid interaction.

To analyze rotation we introduce the operator R_θ which takes in a configuration $\mathbf{X}$ and returns the configuration obtained by rotating the valve by θ about some fixed point. This point of rotation is arbitrary and we take it to be the center of the valve at the previous time-step:

$$\mathbf{x}_c = \left(\frac{(\mathbf{X}^n, \mathbf{V}^1)_B}{(\mathbf{V}^1, \mathbf{V}^1)_B}, \frac{(\mathbf{X}^n, \mathbf{V}^2)_B}{(\mathbf{V}^2, \mathbf{V}^2)_B} \right). \tag{100}$$

We define also

$$\mathbf{V}^3(\mathbf{X}) = \frac{\partial}{\partial\theta} R_\theta(\mathbf{X}). \tag{101}$$

We may equivalently define $\mathbf{V}^3$ via

$$\mathbf{V}_l^3 = \begin{cases} (-Y_l + y_c, X_l - x_c) & \text{if} \quad l \in \mathcal{G}_V, \\ (0,0) & \text{otherwise}, \end{cases} \tag{102}$$

where $\mathbf{X}_k = (X_k, Y_k)$ and $\mathbf{x}_c = (x_c, y_c)$. $\mathbf{V}^3$ is the Jacobian of R_θ. Note that $\mathbf{V}^3$ depends on a configuration $\mathbf{X}$. We use the notation $\mathbf{V}^3(\mathbf{X})$ to emphasize this dependence.

The vectors $\mathbf{V}^1, \mathbf{V}^2, \mathbf{V}^3$ represent points outside the image of $\mathcal{A}_{h_B}$ associated with translation and rotation. Then, we have that there exist solutions to (96) only when $\mathbf{F}^{k+1}$ does not act to translate or rotate the valve. That is, a vector $\mathbf{F}$ lies in the image of $\mathcal{A}_{h_B}$ only when

$$(\mathbf{F}, \mathbf{V}^j)_B = 0 \quad \text{for} \quad j = 1,2,3. \tag{103}$$

Indeed, consider first the simplest case of only two immersed fiber points

$$\mathbf{X} = \begin{pmatrix} (x_1, y_1) \\ (x_2, y_2) \end{pmatrix}, \tag{104}$$

connected with a single link. Then, the force they generate is

$$\mathbf{F} = T \begin{bmatrix} (x_2 - x_1, y_2 - y_1) \\ (x_1 - x_2, y_1 - y_2) \end{bmatrix}, \tag{105}$$

for some tension scalar T. The rotation vector about the origin is simply

$$\mathbf{V}^3 = \begin{bmatrix} (-y_1, x_1) \\ (-y_2, x_2) \end{bmatrix}. \tag{106}$$

We have then that

$$(\mathbf{F}, \mathbf{V}^3)_B = -y_1(x_2 - x_1) + x_1(y_2 - y_1) - y_2(x_2 - x_1) + x_2(y_1 - y_2) = 0. \tag{107}$$

A similar calculation shows that $(\mathbf{V}^1, \mathbf{F})_B = (\mathbf{V}^2, \mathbf{F})_B = 0$. For our more complicated valve structure we simply have a summation of such forces, thus the corresponding force density must satisfy (103) where $\mathbf{F} = \mathcal{A}_{h_B}(\mathbf{X})$ for any configuration $\mathbf{X}$. In general, an arbitrary vector $\mathbf{F}$ may not satisfy (103) and we would be unable to find a solution to (96). To remedy this we must factor out those components outside the image of $\mathcal{A}_{h_B}$. This is easier to do with the linearized $\mathcal{A}_{h_B}$, its Jacobian, where we may simply project onto the vector space free of the problematic vectors $\mathbf{V}^1, \mathbf{V}^2, \mathbf{V}^3$. These observations suggest the modified Newton iteration to approximately solve for $\mathbf{X}^{n+1,k+1}$ in (96):

$$\left\{ \mathbf{X}^{n+1,k+1,0} = \mathbf{X}^{n+1,k}, J(\mathbf{X}^{n+1,k+1,l})(\mathbf{X}^{n+1,k+1,l+1} - \mathbf{X}^{n+1,k+1,l}) = P(\mathbf{F}^{k+1} - \mathcal{A}_{h_B}(\mathbf{X}^{n+1,k+1,l})), \right. \tag{108}$$

where

$$P(\mathbf{F}) = \mathbf{F} - \sum_{j=1}^{3} \mathbf{V}^j(\mathbf{X}^{n+1,k})(\mathbf{F}, \mathbf{V}^j(\mathbf{X}^{n+1,k}))_B, \tag{109}$$

is the projection operator onto rotation and translation free force distributions. Note that because each node in the valve is linked to only a few other nodes, J is $O(N_B)$ sparse. We can solve systems involving J efficiently with various sparse solvers. Additionally, the structure of J never changes throughout a simulation, though its values do, so we may make additional optimizations in the sparse solver if desired.

Linear systems of the form $J\mathbf{X} = P\mathbf{F}$ are, strictly speaking, over determined by three degrees of freedom. To rectify this we simply isolate three degrees of freedom in $\mathbf{X}$ and fix them. These variables must be associated with the valve but are otherwise arbitrary. We could, for instance, fix the x and y component of a single node in the valve as well as the x component of an additional node. Once fixed, we may proceed to solve for $\mathbf{X}$ such that $J\mathbf{X} = P\mathbf{F}$ is satisfied at all other free points. What is important, however, is that iterations of (108) in such a manner will converge to an $\mathbf{X}$ that satisfies $\mathcal{A}_{h_B}\mathbf{X} = P\mathbf{F}$ at all points, even at those we fixed. In this sense we arrive at an updated configuration $\mathbf{X}^{n+1,k+1}$ which satisfies (96) up to components outside the image of $\mathcal{A}_{h_B}$.

The sequence $\mathbf{X}^{n+1,0}, \mathbf{X}^{n+1,1}, \mathbf{X}^{n+1,2}, \ldots$ formed by iterations of (96) typically converges quadratically in the l^2 norm. As an aside, it is important to note that the iterative method (96) converges precisely because the standard fixed point iteration (95) diverges. As we modify the parameters of our simulation, for instance, if we were to drastically decrease the spring constants, then (95) may converge whereas (96) would diverge. There are situations with mixed large and small spring constants where neither iteration converges. The crosslinks of our valve model is one such example. In the case of large spring constants everywhere (96) converges rapidly and increasing the spring constants aids in the convergence.

The limit of our iterations is a stable update of the immersed boundary configuration but is not in general a solution to (23). We need to reintroduce translation and rotation to correct for this discrepancy. We can achieve this in many different ways. Perhaps the simplest method is to use the location and angle of the valve configuration obtained from an explicit update. To accomplish this we may simply take our initial guess $\mathbf{X}^{n+1,0}$ for iterations of (108) to be

$$\mathbf{X}^{n+1,0} = R_\theta \mathbf{X}^n + \mathbf{V}^1(\widetilde{\mathbf{X}} - \mathbf{X}^n, \mathbf{V}^1)_B + \mathbf{V}^2(\widetilde{\mathbf{X}} - \mathbf{X}^n, \mathbf{V}^2)_B, \tag{110}$$

where

$$\theta = (\widetilde{\mathbf{X}} - \mathbf{X}^n - \mathbf{V}^1(\widetilde{\mathbf{X}} - \mathbf{X}^n, \mathbf{V}^1)_B - \mathbf{V}^2(\widetilde{\mathbf{X}} - \mathbf{X}^n, \mathbf{V}^2)_B, \mathbf{V}^3(\mathbf{X}^n))_B, \tag{111}$$

and

$$\widetilde{\mathbf{X}} = \mathbf{X}^n + \mathcal{M}_n \mathcal{A}_{h_B} \mathbf{X}^n. \tag{112}$$

Here, we are taking the FE/BE predictor $\widetilde{\mathbf{X}}$ and extracting its location and angle. We then take the current configuration $\mathbf{X}^n$ and modify the location and angle of the valve to match that in $\widetilde{\mathbf{X}}$. After this, we may proceed to apply iterations of (108) which will preserve the location and angle we extracted from the FE/BE predictor.

Other predictors may be employed as well. For instance we may rediscretize our problem on a coarser grid, solve for the updated time-step, prolong the solution to the original fine grid and extract its gross properties.

Other predictors may involve higher order explicit updates $\widetilde{\mathbf{X}}$ or treating the valve as a true rigid body and predicting its motion through rigid body motion.

8.3. Reintroducing translation and rotation iteratively

Many of the explicit predictors, such as those discussed above, behave quite satisfactorily and in conjunction with (96) provide accurate and stable solutions to (23). In a number of applications this is sufficient. If we employ (108) after using the initial guess (110) we can obtain efficiently an approximate solution to (23) within truncation error with just a few iterations.

In the current application, however, extra care must be taken. The close proximity of the valve to its hinges can quickly lead to unphysical collisions if small errors in translation propagate throughout the simulation. This proximity can also seriously affect the behavior of an explicit predictor, leading to highly inaccurate translation and rotation. Of course, even exact solutions to (23) with Δt large may still produce collisions due to inaccuracies and hence this problem is not intrinsic to the proposed methodology.

We seek a means to reintroduce more accurately the three degrees of freedom we have removed in applying (96). Suppose that we have a solution $\mathbf{X}^{n+1}$ to the original nonlinear system (23). We must have then

$$(\mathbf{F}^{n+1}, \mathbf{V}^j(\mathbf{X}^{n+1}))_B = 0 \quad \text{for} \quad j = 1, 2, 3, \tag{113}$$

where $\mathbf{F}^{n+1}$ is such that

$$\mathcal{M}_n \mathbf{F}^{n+1} = \mathbf{X}^{n+1} - \mathbf{b}^n. \tag{114}$$

This is because if $\mathbf{X}^{n+1}$ solves (23) then $\mathbf{F}^{n+1} = \mathcal{A}_{h_B}\mathbf{X}^{n+1}$ and $\mathcal{A}_{h_B}$ cannot generate forces with nonzero components along $\mathbf{V}_j, j = 1, 2, 3$. Condition (113) suggests that before each iteration of (96) we simply shift and rotate $\mathbf{X}^{n+1,k}$ such that $(\mathbf{F}^{k+1}, \mathbf{V}_j(\mathbf{X}^{n+1,k}))_B = 0$ for $j = 1, 2, 3$, where $\mathcal{M}_n\mathbf{F}^{k+1} = \mathbf{X}^{n+1,k} - \mathbf{b}^n$ as before. This can be approximately accomplished by finding (a_1, a_2) and θ, a translation vector and an angle, such that for $j = 1, 2, 3$

$$(\mathcal{M}_n^{-1}(\mathbf{X}^{n+1,k} + a_1\mathbf{V}^1 + a_2\mathbf{V}^2 + \theta\mathbf{V}^3(\mathbf{X}^{n+1,k}) - \mathbf{b}^n), \mathbf{V}^j(\mathbf{X}^{n+1,k}))_B = 0. \tag{115}$$

Or, equivalently,

$$\begin{bmatrix} (\mathbf{W}^1, \mathbf{V}^1)_B & (\mathbf{W}^2, \mathbf{V}^1)_B & (\mathbf{W}^3, \mathbf{V}^1)_B \\ (\mathbf{W}^1, \mathbf{V}^2)_B & (\mathbf{W}^2, \mathbf{V}^2)_B & (\mathbf{W}^3, \mathbf{V}^2)_B \\ (\mathbf{W}^1, \mathbf{V}^3)_B & (\mathbf{W}^2, \mathbf{V}^3)_B & (\mathbf{W}^3, \mathbf{V}^3)_B \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \theta \end{bmatrix} = - \begin{bmatrix} (\mathbf{F}^{k+1}, \mathbf{V}^1)_B \\ (\mathbf{F}^{k+1}, \mathbf{V}^2)_B \\ (\mathbf{F}^{k+1}, \mathbf{V}^3)_B \end{bmatrix}, \tag{116}$$

where $\mathcal{M}_n\mathbf{W}^j = \mathbf{V}^j$ for $j = 1, 2, 3$. Then, we construct an updated configuration

$$\mathbf{X}^{n+1,k} \leftarrow R_\theta \mathbf{X}^{n+1,k} + a_1\mathbf{V}^1 + a_2\mathbf{V}^2, \tag{117}$$

after which we proceed to calculate $\mathbf{X}^{n+1,k+1}$ via (108) using an unmodified $\mathbf{F}^{k+1}$. In this manner we obtain an iteration which converges rapidly to a solution of (23). This iteration converges nearly as quickly as (96) does for a fixed valve with no rotation or translation. There is an additional cost per iteration though. The predominant cost comes from solving the linear systems involving $\mathcal{M}_n$, which we must now do four times to calculate $\mathbf{W}^j$ for $j = 1, 2, 3$ as well as $\mathbf{F}^{k+1}$.

Fortunately, these added costs are not significant in practice. The vectors $\mathbf{W}^1$ and $\mathbf{W}^2$ only need to be calculated once per timestep. For our simulations, we calculate $\mathbf{W}^3$ only once per timestep as well, even though $\mathbf{V}^3$ depends on the current guess $\mathbf{X}^{n+1,k}$. We find that this does not degrade the final residual beyond the limits of accuracy provided by our multigrid solver. Moreover, $\mathbf{W}^j$ for $j = 1, 2, 3$ will change only incrementally from one time-step to the next and hence these same vectors can be reused as good initial guesses in the following several time-steps. At the end, the predominant cost of the proposed iterators is that of initializing the linear system and multigrid itself.

8.4. Near boundary–boundary interactions

There is an additional subtlety that we wish to point out. The prolongation and restriction operators we use in our multigrid for solving $\mathcal{M}_n\mathbf{W}^j = \mathbf{V}^j$ are geometrically inspired, relying on the underlying structure of the immersed boundary configuration. However, this approach fails to give adequate weight to the fluid interactions *between* immersed boundaries, in particular the interaction between the ends of the valve and the cushions as well as the interaction between the middle of the valve and the hinges.

The linear multigrid we employ still converges in the current situation but not sufficiently fast at key points. To capture accurately all the required dynamics between the valve and hinges to prevent collision and protrusion it is necessary to increase significantly the number of multigrid cycles. While a robust algebraic multigrid algorithm with more appropriate prolongation and restriction operators might remedy this problem we opt here for a simple yet effective local alteration to our smoother.

We maintain the standard Gauss–Seidel relaxation but in addition we also perform a direct solve, via Gaussian elimination, of a small subset of the problem. Because we are only concerned with the high accuracy needed to prevent collision between the valve and hinge we isolate those points close to the hinge (see Fig. 6). Holding all other points outside this region fixed we then proceed to solve the resulting reduced system. Afterward we perform the standard Gauss–Seidel sweep to smooth the interface between the points inside and outside the solved region. Because of the small size of the selected local region, this procedure is quite inexpensive and does not contribute in a significant manner to the cost of the multigrid iteration.

8.5. Numerical results

The initial condition is a rest configuration of all links and tethers. The valve rests between the two hinges and is parallel to the y-axis. The imposed flow is $v_{flow} = 100$ and all spring constants are taken to be $k = 10^9 l^{-2}$, where $l = h/2$ is the

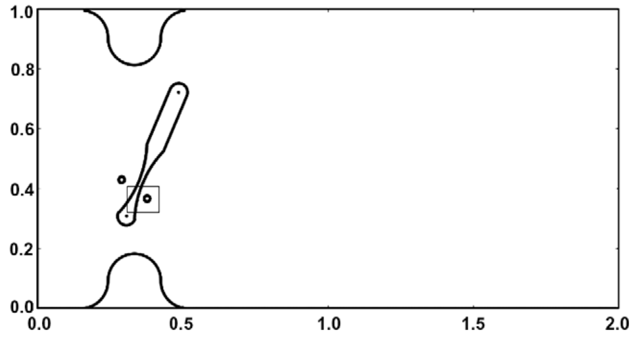

Fig. 6. Configuration of our valve system after some period of time. The small box contains the subsystem of fiber points we solve exactly.

Table 10
Heart valve simulation with Forward Euler/Backward Euler scheme. The total CPU time taken to run the simulation up to time T is given, with the average CPU time per timestep given.

N	Average	$T = 0.01$	$T = 0.05$	$T = 0.1$	$T = 0.5$
128	0.024	2110.49	10508.95	21017.90*	105089.54*
256	0.109	37687.16*	188435.83*	376871.66*	1884358.32*
384	0.249	193168.57*	965842.85*	1931685.70*	9658428.53*
512	0.503	694291.69*	3471458.49*	6942916.98*	34714584.93*

* Denotes an extrapolated value.

Table 11
Heart valve simulation with implicit scheme. The total CPU time taken to run the simulation up to time T is given, with the average CPU time per timestep given. $\Delta t = 0.0025$.

N	Average	$T = 0.01$	$T = 0.05$	$T = 0.1$	$T = 0.5$
128	0.588	2.001	10.436	21.155	117.767
256	2.014	7.640	38.171	76.297	402.749
384	4.747	18.735	93.266	186.270	949.409
512	9.538	34.984	174.702	348.531	1907.635

length between nodes in our valve. The magnitude of the velocities here reaches approximately 10 units, much smaller than that in the elliptical membrane case. Taking a characteristic length to be the length of the heart valve we obtain a Reynolds number of about 5. The smaller speeds lead to a mild CFL constraint and this allows our semi-implicit approach to show its power, even though now $N_B \approx 4N$ and the cost of an implicit time-step is roughly 20 times that of an explicit one.

For the FE/BE simulation Δt is again taken to be the maximum allowable while retaining stability, with $\Delta t = 30hk^{-1/2}$. For the proposed semi-implicit approach we fix $\Delta t = 0.0025$, well below the CFL restraint. With the constant imposed flow, the valve will simply open. The CPU times for the explicit simulations are given in Table 10 with the results from the implicit simulations following in Table 11. With a less restrictive CFL constraint the semi-implicit approach becomes $O(1000)$ faster than FE/BE. Stability-wise, Δt may be taken even larger for the semi-implicit method, up to the limit of the CFL restraint, leading to even greater savings in CPU time. However, this will typically increase leakage through the immersed boundary and eventually allow the valve to collide and pass through the hinges.

To open and close the valve we consider a time dependent imposed flow $v_{flow}(t)$. We define

$$v_{flow}(t) = \begin{cases} 60000t(0.1 - t) & t < 0.1, \\ -60000(t - 0.1)(0.2 - t) & t \geqslant 0.1, \end{cases} \tag{118}$$

A sequence of snap-shots of the motion of the valve as it opens and closes is depicted in Fig. 7 where also flooded contours of the vorticity are shown. Initially, there is a noticeable generation of (positive) vorticity at the upper and lower tips of the valve as the valve opens as well as in the upper cushion. As the valve reverses its motion the vorticity in those same areas evolves toward positive values and becomes large and localized as the valve closes.

9. Conclusion

Implicit methods for alleviating the stiffness of the IB Method have been around for some time. Their implementation involves solving systems of equations typically thought to be prohibitively expensive. One of the main objectives of this work is to illuminate paths to solving the systems arising from certain robust semi-implicit discretizations of the IB Method. We have provided avenues to this end for three particular applications but there will likely remain challenges when extending the techniques detailed here to the incredibly varied body of applications for which the IB Method can be employed.

A key benefit of the proposed methods is that they have no dependence on the specifics of the fluid solver $\mathcal{L}_h$. If we use an adaptive mesh to achieve fluid solves in $O(N_B)$ time then this will carry over to our implicit solvers, allowing us to take large time steps with the minimal order cost possible. This is a particularly exciting prospect for extending the methodology to 3D which is a current aim of our own ongoing research.

Acknowledgments

The authors would like to thank Charles Peskin, David McQueen, Thomas Hou, and Zuoqiang Shi for insightful discussions. Partial support for this work was provided by the National Science Foundation under Grant No. DMS 0609996 (HDC and JEF) and by FAPESP Grant number 00/02801-0 (AMR).

References

[1] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys 25 (1977) 220–252.
[2] J.M. Stockie, B.T.R. Wetton, Stability analysis for the immersed fiber problem, SIAM J. Appl. Math. 55 (6) (1995) 1577–1591.
[3] 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.
[4] C. Tu, C.S. Peskin, Stability and instability in the computations of flows with moving immersed boundaries: a comparison of three methods, SIAM J. Sci. Stat. Comput. 13 (6) (1992) 1361–1376.
[5] A.A. Mayo, C.S. Peskin, An implicit numerical method for fluid dynamics problems with immersed elastic boundaries, in: A.Y. Cheer, C.P.V. Dam (Eds.),