Commun. Comput. Phys.
doi: 10.4208/cicp.010211.070711s

Vol. **12**, No. 2, pp. 378-400
August 2012

A Multigrid Method for a Model of the Implicit Immersed Boundary Equations

Robert D. Guy[1,*] and Bobby Philip[2]

[1] *Department of Mathematics, University of California Davis, Davis, CA 95616, USA.*
[2] *Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA.*

Received 1 February 2011; Accepted (in revised version) 7 July 2011

Available online 20 February 2012

Abstract. Explicit time stepping schemes for the immersed boundary method require very small time steps in order to maintain stability. Solving the equations that arise from an implicit discretization is difficult. Recently, several different approaches have been proposed, but a complete understanding of this problem is still emerging. A multigrid method is developed and explored for solving the equations in an implicit-time discretization of a model of the immersed boundary equations. The model problem consists of a scalar Poisson equation with conformation-dependent singular forces on an immersed boundary. This model does not include the inertial terms or the incompressibility constraint. The method is more efficient than an explicit method, but the efficiency gain is limited. The multigrid method alone may not be an effective solver, but when used as a preconditioner for Krylov methods, the speed-up over the explicit-time method is substantial. For example, depending on the constitutive law for the boundary force, with a time step 100 times larger than the explicit method, the implicit method is about 15-100 times more efficient than the explicit method. A very attractive feature of this method is that the efficiency of the multigrid preconditioned Krylov solver is shown to be independent of the number of immersed boundary points.

AMS subject classifications: 65M55, 65F08, 76M20, 76D99

Key words: Preconditioning, implicit methods, fluid-structure interaction.

1 Introduction

The immersed boundary (IB) method was developed by Peskin [18] to solve the coupled equations of motion of viscous fluid with an immersed elastic boundary. The method was

*Corresponding author. *Email addresses:* guy@math.ucdavis.edu (R. D. Guy), philipb@ornl.gov (B. Philip)

developed to simulate blood flow in the heart, and it has since been applied to many different biofluid applications, and it is increasingly being used in other engineering problems [12]. The method involves two coordinate systems and two discrete grids. The fluid variables are represented in Eulerian coordinates which are discretized by a fixed, Cartesian grid. The immersed structures are represented in moving Lagrangian coordinates. The structures move at the local fluid velocity, interpolated from the Eulerian grid to the Lagrangian grid. The forces generated by the deformation of the structures are transferred to the Eulerian grid and appear as a forcing term in the momentum balance equation for the fluid.

Typical implementations of the IB method use a fractional stepping approach to solve the coupled fluid and boundary equations. The fluid velocity and pressure are updated for fixed boundary position, and then the boundary position is updated from the new velocity. Because the fluid and boundary are updated separately, one can use standard methods for solving for the fluid motion. One reason for the popularity of the IB method is that many different applications can be simulated with minor changes to existing codes. However, in many applications the elastic time scales are well below the physical time scales of interest, which means that the IB equations are very numerically stiff. When alternating between updating the fluid velocity and boundary position, this stiffness requires that the time step be very small in order to maintain stability.

Much effort has been devoted to both understanding and alleviating the severe time step restriction of IB methods [5,14,21]. Early attempts at implicit methods were not very efficient and thus not competitive with explicit methods [25], and some semi-implicit methods still presented significant time step restrictions [10, 11]. Newren et al. [14] analyzed the origin of instability in semi-implicit methods using energy arguments, and they gave sufficient conditions for schemes to be unconditionally stable in the sense that the total energy is bounded regardless of the size of the time step. Recently a variety of stable semi-implicit methods have been developed [3, 7, 8, 15], as well as several fully implicit methods [9,13]. Of course, these methods require more sophisticated algorithms in which the velocity and boundary position are solved for simultaneously. These recent methods are generally competitive in efficiency with explicit methods, and in some special cases they can be faster by factors of hundreds. It remains an open question as to whether there is a general, robust implicit method that is easy to use and more efficient than the explicit method for large classes of problems, or whether specialized methods will need to be developed for specific problems.

Many implicit methods reduce the full IB equations (fluid and boundary) to equations on only the boundary [2,3,13]. These methods achieve a substantial speed-up over explicit methods when there are relatively few immersed boundary points [3]. In addition, some methods require that the boundaries be smooth, closed curves [7,8]. Newren et al. [15] explored Kryolv methods for solving the linearized IB equations for different test problems. The relative efficiency of the implicit methods depended on the problem, and unpreconditioned Krylov methods were at least comparable in speed to explicit methods. These results suggest that with appropriate preconditioning, this approach

will offer a significant improvement over explicit methods. One way to achieve generally applicable and robust efficient implicit methods is through the development of good preconditioners for the linearized equations. This is the approach we take in this paper.

The main challenge in solving the implicit IB equations involves the fluid-boundary coupling. In this paper, we explore a model problem related to the implicit immersed boundary equations in which we ignore the inertial terms and the incompressibility constraint. We develop and explore a multigrid method which simultaneously solves for the unknown fluid velocity and boundary position. Multigrid has been used in the past with explicit-time IB methods to solve the fluid equations [6, 19, 28] as well as in model problems of the IB equations [17]. There are two significant complications to applying multigrid to the implicit IB equations which involve the simultaneous solution of both Eulerian and Lagrangian equations. One is the presence of both an Eulerian grid and a Lagrangian grid, and so it is not clear how to coarsen the problem. The second complication is how to smooth the errors in immersed boundary equations. We explore these issues on a model problem which resembles the singularly forced Poisson equation. The model problem involves only one parameter (elastic stiffness) and its simplicity facilitates algorithm exploration.

In the next section we describe the immersed boundary method. In Section 3 we describe the model problem explored in this paper. In Section 4 we present and explore the multigrid method for the solving the model problem. The effectiveness of the multigrid method as a solver for Krylov methods is presented in Section 5. Finally, some discussion of the relevance of the model problem and its relationship to the immersed boundary method are presented in Section 6.

2 Immersed boundary equations

The IB method makes use of two coordinate systems: an Eulerian system for the fluid velocity and pressure and a Lagrangian system for the elastic structure. Let Ω denote the Eulerian domain, and let Γ represent the immersed boundary. The spatial location of the immersed boundary is $\boldsymbol{X}(s,t)$, where s is a parametric coordinate. See Fig. 1. In general we use the convention of lowercase letters for Eulerian variables and capital letters for Lagrangian variables.

The forces generated by the deformation of the boundary drive the motion of the fluid. Generally, it is assumed that the immersed boundary is neutrally buoyant, so that all of the boundary force is transmitted to the fluid. It is also assumed that the boundary moves with the local fluid velocity. The communication between the boundary and the background fluid is handled by convolutions with the Dirac delta function. The equations are

$$Re(\boldsymbol{u}_t+\boldsymbol{u}\cdot\nabla\boldsymbol{u})=\Delta\boldsymbol{u}-\nabla p+\boldsymbol{f}, \tag{2.1}$$

$$\nabla\cdot u=0, \tag{2.2}$$

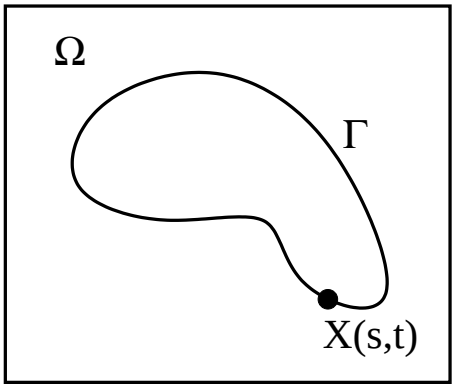


Figure 1: The fluid domain Ω contains the immersed boundary Γ whose position is given by $\boldsymbol{X}(s,t)$.

$$\frac{\partial \boldsymbol{X}(s,t)}{\partial t} = \boldsymbol{U}(s,t) = \int_{\Omega} \boldsymbol{u}(\boldsymbol{x},t)\delta(\boldsymbol{x}-\boldsymbol{X}(s,t))\,d\boldsymbol{x}, \tag{2.3}$$

$$\boldsymbol{f} = \int_{\Gamma} \boldsymbol{F}(s,t)\delta(\boldsymbol{x}-\boldsymbol{X}(s,t))\,ds. \tag{2.4}$$

The first two equations are the incompressible Naiver-Stokes equations to describe the motion of the fluid, where $\boldsymbol{u}$ is the fluid velocity, p is the pressure, and Re is the Reynolds number. The last two equations describe the boundary-fluid communication. The integral operator in (2.4) that transfers boundary forces to the fluid is called the spreading operator, which we denote by S. The operator which transfers the fluid velocity to the boundary is the adjoint of the spreading operator. Eqs. (2.3) and (2.4) can be expressed as

$$\boldsymbol{U} = S^{*}\boldsymbol{u}, \tag{2.5}$$

$$\boldsymbol{f} = S\boldsymbol{F}, \tag{2.6}$$

respectively.

A constitutive law for the boundary forces is needed to complete the description of the system. For simplicity, in this paper, we focus on linear constitutive laws. The three constitutive laws we consider correspond to tethering forces, stretching forces, and bending forces:

$$\text{tether:} \quad F = -\gamma(\boldsymbol{X}-\boldsymbol{X}_0), \tag{2.7}$$

$$\text{stretch:} \quad F = \gamma\frac{\partial^2 \boldsymbol{X}}{\partial s^2}, \tag{2.8}$$

$$\text{bend:} \quad F = -\gamma\frac{\partial^4 \boldsymbol{X}}{\partial s^4}. \tag{2.9}$$

In each case, the constant γ characterizes the stiffness of the elastic material. Tether forces are used to enforce Dirichlet boundary conditions on the immersed boundary. The immersed boundary is connected by linear springs (tethers) to a second boundary located

at position $\boldsymbol{X}_0$ whose motion is prescribed. If the stiffness is taken very large, the tether forces cause the immersed boundary to move at approximately the same velocity as that of the prescribed boundary. These force laws are all of form

$$F = -\gamma A(\boldsymbol{X} - \boldsymbol{X}_0), \tag{2.10}$$

where A is a symmetric positive definite operator, and $\boldsymbol{X}_0 = 0$ for stretching and bending forces.

3 Model problem

In this paper, we explore a model problem rather than the full immersed boundary equations. We ignore the inertial terms and the incompressibility constraint, which gives the scalar problem

$$\Delta u + SF = 0, \tag{3.1}$$
$$X_t = S^* u, \tag{3.2}$$
$$F = -\gamma A(X - X_0). \tag{3.3}$$

The model problem only contains one parameter, the stiffness γ. We refer to u as the velocity and F as the force, even though these quantities do not represent physical quantities. Boundary conditions for u on the Eulerian domain must be specified. For simplicity, we use homogeneous Dirichlet boundary conditions ($u = 0$ on $\partial\Omega$).

By ignoring incompressibility, we reduce the velocity to a scalar and eliminate the pressure as an unknown. The reduced number of unknowns in the model problem facilitates numerical explorations that are computationally intensive such as those that involve computing the inverse or all of the eigenvalues of a matrix. Additionally, many algorithms for solving the equations of viscous incompressible flow involve inverting, or approximately inverting, Laplacian-like operators related to the viscous stress. Extending these algorithms to implicit IB methods involves inverting operators like those that appear in the model problem which include both fluid viscosity and boundary elasticity. Therefore, this model problem is a natural starting point for exploring multigrid algorithms for implicit IB methods.

3.1 Time discretization

We use a backward Euler time discretization in which the positions of the spreading and interpolation operators are lagged in time. As shown in [14], keeping the spreading and interpolation operators fixed over the time step does not affect this stability as long as the spreading and interpolation occur at the same spatial location in the time step. The

discrete-time system is

$$\Delta u^{n+1} + SF^{n+1} = 0, \tag{3.4}$$

$$X^{n+1} = X^n + \Delta t S^* u^{n+1}, \tag{3.5}$$

$$F^{n+1} = -\gamma A\left(X^{n+1} - X_0^{n+1}\right). \tag{3.6}$$

The last two equations can be used to eliminate the force in the first equation, leaving a single linear equation to solve for the velocity:

$$(\Delta - \alpha SAS^*)u^{n+1} = \gamma SA\left(X^n - X_0^{n+1}\right), \tag{3.7}$$

where we define

$$\alpha = \Delta t\gamma. \tag{3.8}$$

Because A is positive definite and the Laplacian is negative definite, the operator $\Delta - \alpha SAS^*$ is negative definite, and therefore, invertible. Once the velocity is known, it is easy to compute the other unknowns.

The advantage of reducing the system (3.4)-(3.6) to the single equation (3.7), is that all of the unknowns are on a single Eulerian grid. This facilitates the development of multigrid methods. We note that many other implicit immersed boundary methods eliminate the velocity using a Schur complement to obtain a single equation for the unknown boundary position [3,7,8,13]. However the Lagrangian grid is often unstructured, which makes it difficult to develop multigrid methods.

3.2 Spatial discretization

For the test problems in this paper, the domain is the unit square $[0,1]^2$. We use an equally spaced, node centered discretization: $(x_i, y_j) = (i\Delta x, j\Delta x)$. The discrete Laplacian is the standard, second-order, five-point operator

$$(Lu)_{i,j} = \frac{u_{i-1,j} + u_{i,j-1} - 4u_{i,j} + u_{i+1,j} + u_{i,j+1}}{\Delta x^2}. \tag{3.9}$$

The two-dimensional discrete delta function is the tensor product of two one-dimensional delta functions:

$$\delta_2(\boldsymbol{x} - \boldsymbol{X}) = \delta(x - X)\delta(y - Y). \tag{3.10}$$

The one-dimensional discrete delta function is

$$\delta(r) = \begin{cases} \dfrac{1}{4\Delta x}\left(1 + \cos\left(\dfrac{\pi r}{2}\right)\right), & \text{if } r < 2\Delta x, \\ 0, & \text{otherwise.} \end{cases} \tag{3.11}$$

The discrete spreading operator is

$$(SF)_{i,j} = \sum_k f_k \delta(x_i - X_k)\delta(y_j - Y_k)\Delta s, \tag{3.12}$$

where Δs represents the grid spacing of the Lagrangian grid.

For second derivative on the boundary that appears in the stretching constitutive law is discretized using the standard three-point, second-order discretization:

$$\left(\frac{\partial^2 \boldsymbol{X}}{\partial s^2}\right)_k \approx \frac{\boldsymbol{X}_{k-1}-2\boldsymbol{X}_k+\boldsymbol{X}_{k+1}}{\Delta s^2}. \tag{3.13}$$

Similarly, the fourth derivative in the bending force law is discretized by the five-point difference that results from two second differences.

4 Multigrid method

In this section, we explore a multigrid method for solving Eq. (3.7). Two challenges to developing an effective multigrid method for implicit immersed boundary methods are (1) how to smooth the error and (2) how to coarsen. We explicitly form the matrix. Note that forming the spreading operator is no more expensive than applying it. We can then use standard smoothers such as Gauss-Seidel. The advantage of using problem (3.7), is that we have eliminated the Lagrangian grid, and coarsening the Eulerian grid is straightforward. We use standard geometric coarsening, with full-weighting for the restriction operator and bilinear interpolation for the prolongation operator. It is not clear what to use for a coarse grid operator. In the next section we compare the rediscretized and Galerkin coarse grid operators.

4.1 Coarse grid operator

For an initial test, the domain is the unit square $[0,1]^2$, and the immersed boundary is a circle of radius 0.15 centered at the point $(0.35,0.45)$. The Lagrangian grid spacing is half that of the Eulerian grid, i.e. $\Delta s = \Delta x/2$. See Fig. 2. Homogeneous Dirichlet boundary conditions on the velocity are enforced on the boundary of the square. We apply the algorithm to the problem

$$(L-\alpha SS^*)u = 0, \tag{4.1}$$

which corresponds to tethering forces (see Eq. (2.7)). The solution to this problem is $u=0$, and so the iterates are the error. The initial guess for u is random and scaled to one in the max-norm. This choice is made so that all spatial frequencies are present in the error.

We begin with a two-grid method in which the fine grid spacing is $\Delta x = 2^{-6}$ and the coarse grid spacing is $\Delta x = 2^{-5}$. On the coarse grid we perform an exact solve. We use only two grids to avoid overlap of the stencil of the spreading operator with the boundary. For a smoother, we use Gauss-Seidel with lexicographic ordering with one pre and one post smooth. We compare the convergence rate of the multigrid algorithm with two different coarse grid operators: one that is rediscretized and the Galerkin coarse grid operator. Note that rediscretization involves coarsening the Lagrangian grid as well as the Eulerian grid in order to define the coarse grid spreading and interpolation operators.

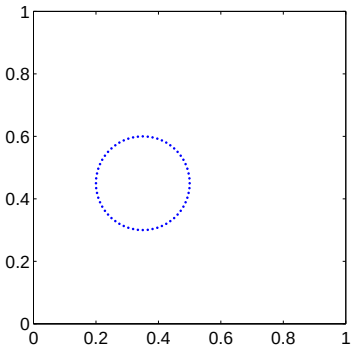


Figure 2: Computational domain for model problem one. The domain is the unit square with a circle of immersed boundary points of radius 0.15 centered around $(0.35,0.45)$.

The Galerkin coarse grid operator is generated by interpolating up to the fine grid, applying the fine grid operator, and then restricting back down to the coarse grid. This can be expressed as

$$(L-\alpha SS^*)_{2h} = I_h^{2h}(L_h - \alpha S_h S_h^*) I_{2h}^h, \tag{4.2}$$

where I_{2h}^h is the prolongation operator and I_h^{2h} is the restriction operator.

In Fig. 3 we show the max-norm of the error for ten iterations of the algorithm for the two different coarse grid operators for stiffnesses, α, in the range $[10^2,10^5]$. For small values of the stiffness, the convergence in the two methods is very similar. The convergence rate for $\alpha < 10^2$ is very similar to $\alpha = 10^2$, and we do not show results for these smaller values of α. Thus, for $\alpha < 10^2$ the convergence rate is like that of the Poisson equation. For the larger values of the stiffness, the errors in the method with the rediscretized operator are amplified by each iteration. There is a dramatic change in convergence behavior between $\alpha = 10^3$ and $\alpha = 10^4$. With the Galerkin coarse grid operator, the convergence slows as the stiffness increases, but the error is always decreasing, albeit slowly. We tested

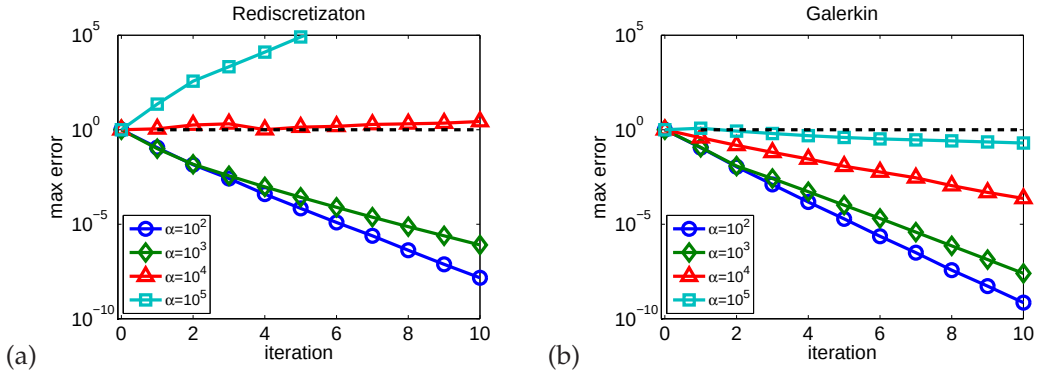


Figure 3: Max norm of the errors from the multigrid method applied to problem (4.1) with random initial data. (a) Coarse grid operator from rediscretization. (b) Galerkin coarse grid operator.

this method for even larger values of α, and it always converged. We conclude that the Galerkin coarse grid operator is necessary for convergence (at least for our choices of coarsening strategy and smoother), and we use this approach in the remainder of the paper.

4.2 Efficiency comparison

The results from the previous section suggest that the multigrid algorithm with the Galerkin coarse grid operator always converges, but the convergence is slow for large values of the stiffness parameter. In this section we compare the efficiency of the multigrid method with the efficiency of an explicit-time method. Consider the explicit-time discretization in which the force is computed based on the current configuration of the boundary

$$Lu^{n+1} - \gamma SAX^n = 0, \tag{4.3}$$

$$\frac{X^{n+1} - X^n}{\Delta t} = S^* u^{n+1}. \tag{4.4}$$

From this system we eliminate u^{n+1} to get

$$\frac{X^{n+1} - X^n}{\Delta t} = \gamma S^* L^{-1} SAX^n. \tag{4.5}$$

Thus, the explicit method is stable when

$$\alpha = \gamma \Delta t < \frac{2}{\rho}, \tag{4.6}$$

where ρ is the spectral radius of $S^* L^{-1} SA$.

We use the same domain as in the previous test. The spectral radii of $S^* L^{-1} SA$ for the three different force operators on four different grids are given in Table 1. For the stretching forces, the spectral radius grows linearly as the grid is refined, and for the bending forces, the spectral radius grows cubically. Also in the same table, we give the maximum value of $\alpha = \gamma \Delta t$ scaled by a power of Δx. Notice that after rescaling, the maximum value of α for stability of the explicit method is around 10 for all constitutive laws. For a given stiffness, γ, we let Δt_{exp} denote the maximum time step for stability and α_{exp} denote the corresponding maximum value of α. In general we define α_{exp} by

$$\alpha_{\text{exp}} = \frac{2}{\rho}. \tag{4.7}$$

For this first test problem, we make the approximation that

$$\alpha_{\text{exp}} = \gamma \Delta t_{\text{exp}} = 10 \Delta x^b, \tag{4.8}$$

Table 1: Spectral radius of the matrix $S^*L^{-1}SA$ as a function of the grid spacing for the different force operators, and the corresponding time step restriction of the explicit method.

	spectral radius			$\max(\gamma\Delta t/\Delta x^b)$		
grid spacing	tether	stretch	bend	tether ($b=0$)	stretch ($b=1$)	bend ($b=3$)
2^{-5}	0.1662	5.482	$6.938\cdot10^3$	12.04	11.68	9.445
2^{-6}	0.1736	10.98	$5.396\cdot10^4$	11.52	11.66	9.716
2^{-7}	0.1759	22.05	$4.369\cdot10^5$	11.37	11.61	9.600
2^{-7}	0.1778	44.06	$3.483\cdot10^6$	11.25	11.62	9.635

where $b=0$ for tether forces, $b=1$ for stretching forces, and $b=3$ for bending forces.

One iteration of the multigrid method takes about the same amount of work as an iteration of multigrid on the Poisson equation. There is some extra work because we use Galerkin coarsening and form the matrices. Let m_α be the number of iterations of multigrid to converge to a given tolerance for a given value of α. The number of iterations needed to solve the Poisson equation in the explicit method is m_0.

We quantify the amount of work for each method as the number of iterations of multigrid per time step times the number of time steps taken. The number of time steps to compute to a fixed point in time is proportional to Δt^{-1}.

In the explicit method, the time step is restricted by the stability constraint, so that $\Delta t_{\text{exp}}\gamma=\alpha_{\text{exp}}=10\Delta x^b$. We assume that time step in the explicit method is chosen to be $\min(\Delta t,\Delta t_{\text{exp}})$ so that the work associated with the explicit method is

$$W_{\text{exp}}=\frac{m_0}{\min\left(\Delta t,\Delta t_{\text{exp}}\right)}=\frac{\gamma m_0}{\min\left(\alpha,\alpha_{\text{exp}}\right)}. \tag{4.9}$$

For the same value of γ, the time step of the implicit method is not restricted by the stability constraint. The amount of work for the implicit method is

$$W_{\text{imp}}=\frac{m_\alpha}{\Delta t}=\frac{\gamma m_\alpha}{\alpha}. \tag{4.10}$$

We define the efficiency factor as

$$\mathcal{E}(\alpha)=\frac{W_{\text{exp}}}{W_{\text{imp}}}=\frac{\alpha m_0}{\min\left(\alpha,\alpha_{\text{exp}}\right)m_\alpha}. \tag{4.11}$$

The value of $\mathcal{E}$ is interpreted as the speed-up one would expect by using the implicit method in place of the explicit method for a given simulation.

To estimate the number of iterations, we initialize the solution with random initial data with max-norm one. We apply V-cycles with one pre and one post smooth until the error is reduced by a factor of 10^{-6}. For this test, the finest grid spacing is 2^{-5} and the grid spacing on the coarsest grid is 2^{-1} (i.e. the coarsest grid has only one grid point). Note that the iteration count we report represents the worst case scenario because we start with a random initial guess. In a time dependent simulation, the solution from the

previous time step would be used as an initial guess, and one would expect a smaller number of iterations.

The number of iterations versus α, scaled by α_{exp}, is shown in Fig. 4(a). The iteration count for stretching and bending forces is about the same. In both of these cases, the number of iterations is fairly constant for $\alpha < \alpha_{\text{exp}}$, and it increases steadily for α above α_{exp}. For tether forces, the iteration count increases very modestly for α between α_{exp} and about $100\alpha_{\text{exp}}$. After that, the iteration count increases more rapidly.

In Fig. 4(b) we show the efficiency factor computed from the iteration count. For stretching and bending forces, the efficiency factor is an increasing function, but it saturates around 6. The efficiency factor for the tether forces increases rapidly for two orders of magnitude in stiffness, and it saturates around a value of 200.

The asymptotic convergence factor is the spectral radius of the iteration matrix [24]. This estimate assumes that all the error is concentrated in the eigenspace with the largest eigenvalue. For a large number of iterations, this is the factor by which the error is reduced per iteration. We estimate the number of iterations needed to reduce the error by a factor of 10^y by

$$m_\alpha = \frac{-y}{\log_{10}\rho(\alpha)}, \tag{4.12}$$

where $\rho(\alpha)$ is the spectral radius of the multigrid iteration matrix. Using this estimate, we express the efficiency factor as

$$\mathcal{E}(\alpha) = \frac{\alpha \log_{10}\rho(\alpha)}{\min(\alpha_{\text{exp}},\alpha)\log_{10}\rho(0)}. \tag{4.13}$$

To compute the asymptotic convergence factor, we explicitly form the multigrid iteration matrix. The j^{th} column of the multigrid iteration matrix is generated by applying one V-cycle to a problem with zero right hand side with initial guess of a unit vector with a one in the j^{th} place and zeros everywhere else.

The asymptotic convergence factors for the three different force laws as a function of α are shown in Fig. 5(a). The results for stretching and bending forces are about the same. The convergence factor is sigmoidal in shape. It increases from around 0.1 to near one for α in the decade above α_{exp}. The convergence factor for the tethering forces increases modestly for the first two decades above α_{exp}, after which it increases rapidly towards one. In Fig. 5(b) we show the efficiency factor computed using the asymptotic convergence factor. These results are similar to those from the previous tests, but the asymptotic efficiency is a little lower than before.

4.3 Smoothing

In this section we explore how the number of smoothing steps per V-cycle affects the convergence of the algorithm. For this test, we use the stretching constitutive law on the same domain as before. We explicitly form the multigrid matrix, as described above, for

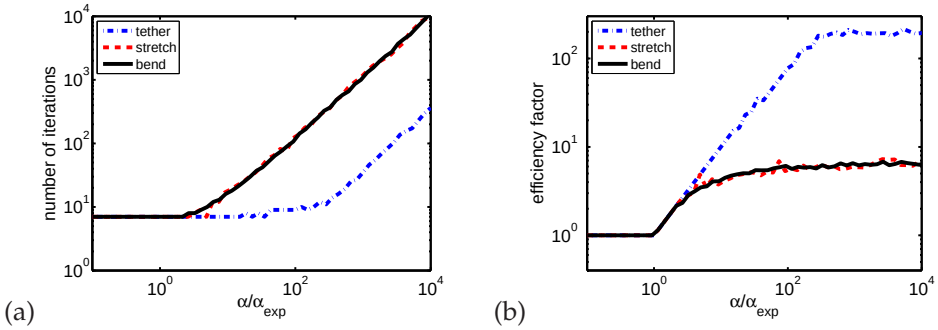


Figure 4: (a) Number of V-cycles needed to reduce the error by a factor of 10^{-6}. (b) Efficiency factor computed from this data.

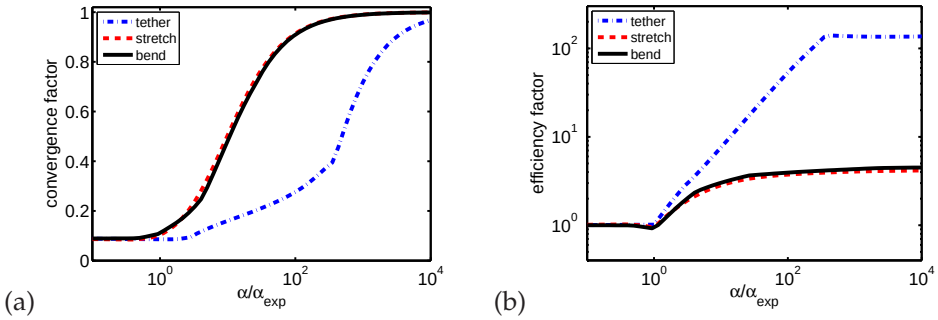


Figure 5: (a) Asymptotic convergence factor (spectral radius) of the multigrid method. (b) Efficiency factor estimated from asymptotic convergence factor.

different numbers of pre and post smoothing steps, and we compute the spectral radius of the multigrid matrix. For even-numbered total smoothing steps, we use equal number of pre and post smoothing steps, an for odd-numbered total smoothing steps we perform an additional pre smoothing step.

In Table 2 we give the spectral radius of the multigrid iteration matrix for the total number of smoothing steps, ν, between 1 and 6 for several orders of magnitude of the stiffness. As expected, the spectral radius decreases as the number of smoothing steps increases. We use the number of smoothing steps plus one as a work unit for the V-cycle. The plus one accounts for the work in computing the residual, and we ignore the work of the transfer operators. The total work to solve a problem is the work unit times the number of cycles needed to converge to a given tolerance. As a measure of total work, we use the quantity

$$W = \frac{\nu+1}{-\log_{10}(\rho)}, \tag{4.14}$$

where ν is the total number of pre and post smooths and ρ is the spectral radius of the multigrid iteration matrix. One can interpret this quantity as the amount of work to reduce the error by one order of magnitude.

Table 2: Spectral radius of the multigrid matrix for different values of the total number of smoothing steps, ν and values of the stiffness, α.

	$\alpha/\alpha_{\text{exp}}$					
ν	0	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
1	0.3105	0.3073	0.3035	0.6505	0.9439	0.9938
2	0.0892	0.0867	0.1024	0.4963	0.9111	0.9900
3	0.0608	0.0592	0.0604	0.4002	0.8830	0.9865
4	0.0451	0.0441	0.0441	0.3259	0.8554	0.9831
5	0.0352	0.0344	0.0358	0.2721	0.8299	0.9797
6	0.0287	0.0280	0.0298	0.2303	0.8055	0.9764

Table 3: Work estimate to reduce the error by one digit of accuracy for different values of the total number of smoothing steps, ν and values of the stiffness, α.

	$\alpha/\alpha_{\text{exp}}$					
ν	0	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
1	3.94	3.90	3.86	10.71	79.71	742.11
2	2.86	2.82	3.03	9.86	74.21	685.83
3	3.29	3.26	3.28	10.06	74.04	679.91
4	3.72	3.69	3.69	10.27	73.73	674.38
5	4.13	4.10	4.15	10.62	74.09	674.04
6	4.54	4.51	4.59	10.98	74.54	674.91

In Table 3, we give the total work as a function of the number of smoothing steps and the stiffness. For a stiffness up to about ten times the stability limit ($\alpha \leq 10\alpha_{\text{exp}}$), the most efficient number of smoothing steps is two. For larger values of the stiffness, the most efficient number of smoothing steps increases to 4 at $\alpha = 100\alpha_{\text{exp}}$ and to 5 at $\alpha = 1000\alpha_{\text{exp}}$. For large values of the stiffness, the efficiency is less sensitive to the number of smoothing steps, and so we use two smoothing steps.

4.4 Grid refinement

In the previous tests, we used a relatively coarse grid. In this section we explore how the performance of the algorithm depends on the grid spacing. In Fig. 6(a) we show the number of iterations to solve the problem described previously to a tolerance of 10^{-6} for a range of stiffness values for four different grid sizes for stretching forces. For very low values of the stiffness, the number of iterations does not depend on the grid spacing, but for large values of the stiffness the number of iterations increases as the mesh is refined. Thus for a fixed value of the stiffness, the convergence of the algorithm is not mesh independent.

In Fig. 6(b) we replot the iteration count against the stiffness scaled by α_{exp}, which for this constitutive law, scales linearly with the grid spacing. With the exception of the coarsest grid, the number of iterations is essentially independent of grid spacing for a fixed relative stiffness. In Fig. 6(c) we show the efficiency factor computed from this data

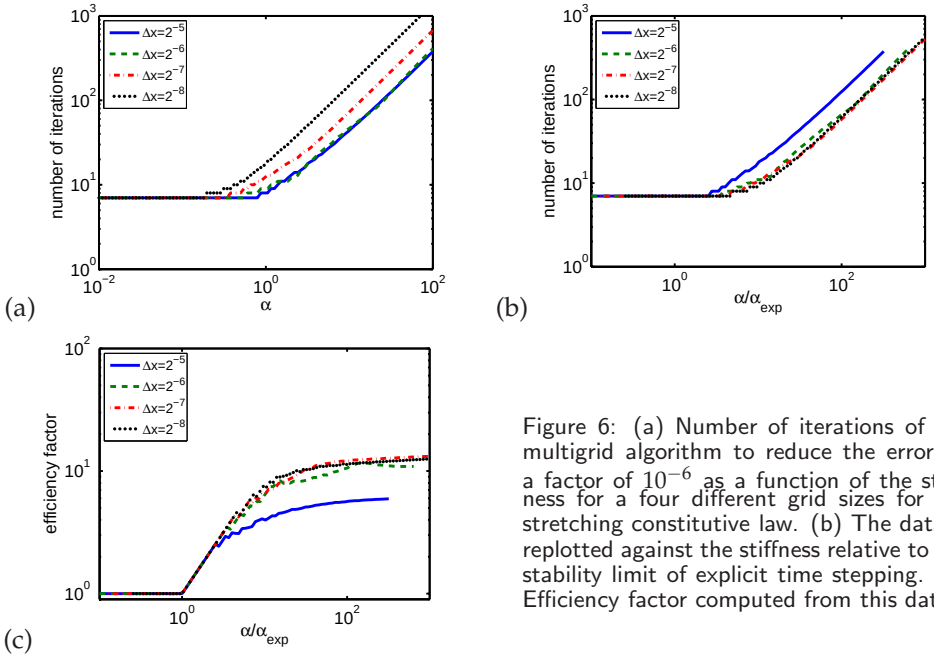


Figure 6: (a) Number of iterations of the multigrid algorithm to reduce the error by a factor of 10^{-6} as a function of the stiffness for a four different grid sizes for the stretching constitutive law. (b) The data is replotted against the stiffness relative to the stability limit of explicit time stepping. (c) Efficiency factor computed from this data.

according to Eq. (4.11). The maximum efficiency factor is around 12, which is twice as large as that of the coarsest grid.

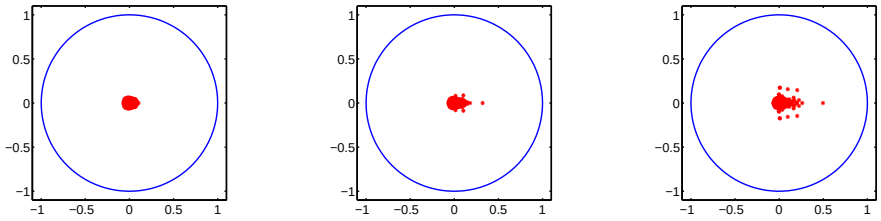
4.5 Spectrum of MG operator

All of the previous tests show that as the stiffness increases above the explicit stability limit, the convergence rate of the multigrid algorithm degrades and the relative efficiency levels off. In this section, we examine the spectrum of the multigrid iteration matrix as a function of the stiffness. For the stretching forces, we explicitly form the multigrid iteration matrix corresponding to one pre and one post smooth as described previously.

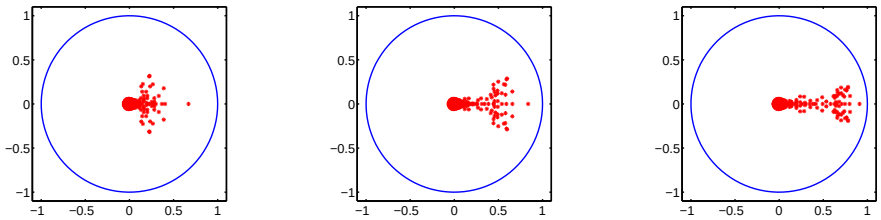
In Fig. 7 we plot the eigenvalues of the matrix in the complex plane for scaled values of the stiffness between 1 and 100 for a grid spacing of $\Delta x = 2^{-5}$. For $\alpha/\alpha_{\text{exp}} = 1$, the convergence rate is similar to that of the Poisson equation ($\alpha = 0$), and the eigenvalues are tightly clustered around the origin. As α increases the convergence rate decays, but as these plots show, it is only a small number of eigenvalues that are responsible for the slow down in convergence. At first only one eigenvalue moves away from the origin, but as the stiffness increases, more eigenvalues move away from the origin and towards the boundary of the unit disc.

5 Multigrid preconditioning

As the plots of the spectrum of the MG iteration matrix from the previous section suggest, even when the spectral radius increases, the convergence is still rapid on a large



(a) $\alpha/\alpha_{\text{exp}}=1$, $\rho\approx 0.1024$ (b) $\alpha/\alpha_{\text{exp}}=5$, $\rho\approx 0.3209$ (c) $\alpha/\alpha_{\text{exp}}=10$, $\rho\approx 0.4963$



(d) $\alpha/\alpha_{\text{exp}}=20$, $\rho\approx 0.6695$ (e) $\alpha/\alpha_{\text{exp}}=50$, $\rho\approx 0.8369$ (f) $\alpha/\alpha_{\text{exp}}=100$, $\rho\approx 0.9111$

Figure 7: Eigenvalues of the multigrid iteration matrix for different values of the stiffness, α, for the model problem. The spectral radius, ρ, is given for each value of α. The finest grid level has a total of $31^2=961$ grid points ($\Delta x=2^{-5}$).

subspace. As discussed in [16], multigrid preconditioned Krylov methods may be very effective in these situations. Multigrid preconditioning has been investigated for CG [23] for symmetric positive definite problems as well as for GMRES and BICGSTAB for more general problems [16, 24]. In multiphase flow applications with sharp variation in material properties, it has been observed that multigrid is a poor solver but very effective preconditioner for Krylov methods [22,27]. In this section we investigate the effectiveness of multigrid as a preconditioner for this model of the immersed boundary equations.

We use right preconditioned GMRES [20]. We also experimented with CG and BICGSTAB, and found that the efficiency results were about the same for all three methods. The application of the preconditioner is accomplished by taking one V-cycle (one pre and one post smooth) with an initial guess of zero. We begin with the same model problem explored previously, and we compute the number of iterations required to reduce the residual by a factor of 10^{-6} as a function of α for the three different force laws. We terminate the iteration after 200 steps if it has failed to converge.

The iteration count as a function of α is shown in Fig. 8(a). For $\alpha<\alpha_{\text{exp}}$ the iteration count is essentially constant. For the stretching and bending force laws, the number of iterations begins to increase around $\alpha\approx\alpha_{\text{exp}}$, and for tether forces the iteration count does not increase until about $\alpha\approx 100\alpha_{\text{exp}}$. This behavior is similar to that of multigrid as a stand-alone solver. For comparison, we overlay the iteration count of the multigrid solver for stretching forces (labeled "mg only") from Fig. 4(a). We see that number of iterations in the multigrid preconditioned solver is increasing much more slowly than in the multigrid solver.

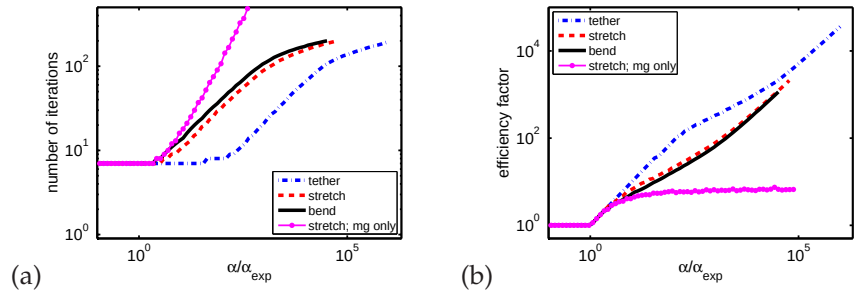


Figure 8: (a) Number of iterations of multigrid preconditioned GMRES to reduce the error by a factor of 10^{-6}. (b) Efficiency factor computed from this data. For comparison, the results from the multigrid solver for stretching forces (labeled "mg only") are replotted from Fig. 4.

In Fig. 8(b) we show the efficiency factor computed from the iteration count. As the stiffness increases, the efficiency factor increases, and it does not level off as it did for multigrid alone. These data show that the multigrid preconditioned GMRES method provides a very substantial speed-up over the explicit method. For example for $\alpha = 100\alpha_{\text{exp}}$ (100 times lager time steps than explicit method), the efficiency gains are 92, 22, and 16 for tether, stretching, and bending forces, respectively. For $\alpha = 1000\alpha_{\text{exp}}$, these efficiency gains are 328, 81, and 61, respectively.

5.1 Grid refinement

We use the stretching constitutive law and explore the effect of grid refinement on performance of the multigrid preconditioned GMRES solver. In Fig. 9(a) we show the number

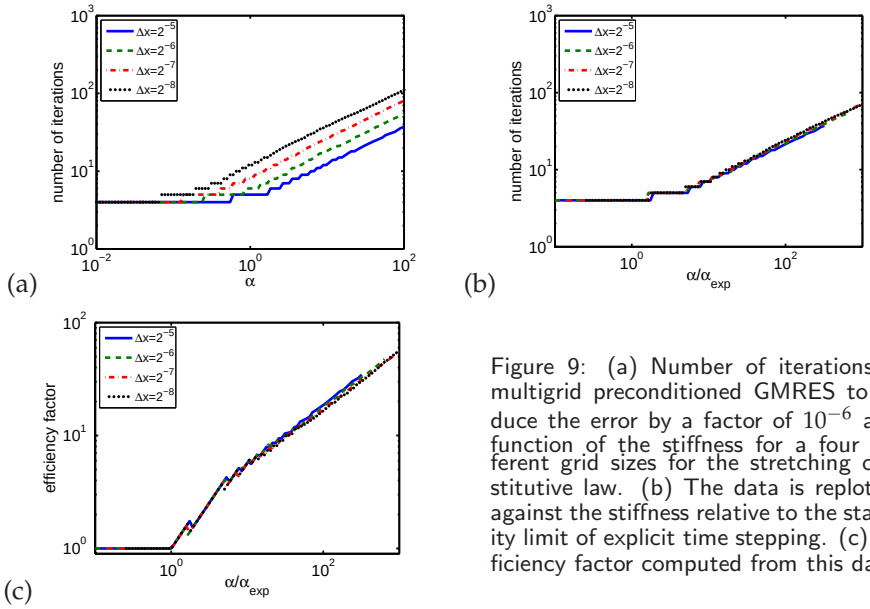


Figure 9: (a) Number of iterations of multigrid preconditioned GMRES to reduce the error by a factor of 10^{-6} as a function of the stiffness for a four different grid sizes for the stretching constitutive law. (b) The data is replotted against the stiffness relative to the stability limit of explicit time stepping. (c) Efficiency factor computed from this data.

of iterations as a function of α (unscaled) to reduce the error by a factor of 10^{-6} for four different grid spacings. For a fixed value of the stiffness, as the grid is refined, the number of iterations increases. This same behavior was observed with the multigrid solver.

The maximum time step allowed by the explicit method decreases as the grid is refined for this constitutive law, and so α_{exp} decreases as the grid is refined. In Fig. 9(b) we plot the iteration count against the scaled stiffness $\alpha/\alpha_{\text{exp}}$. For a fixed value of the scaled stiffness, the iteration count is independent of the grid spacing. In Fig. 9(c) we plot the efficiency factor, which is independent of the grid spacing.

5.2 Tests with more Lagrangian points

All of the tests presented so far have used the same test problem: the immersed boundary was a single circle. The immersed boundary occupies a very small region in the domain, and there are many more Eulerian grid points than Lagrangian grid points. Some other implicit IB methods are very efficient for problems with few Lagrangian points [3]. These methods reduce the problem to the immersed boundary, and they exploit that there are far fewer boundary unknowns than fluid unknowns. The multigrid method presented here reduces the problem to the Eulerian mesh in an effort to avoid increasing complexity when there are large numbers of Lagrangian points. In this section we explore the performance of the multigrid preconditioned GMRES method for problems with different numbers of immersed boundary points.

We use circular immersed boundaries with different radii placed inside the unit square. We compare the performance of the algorithm on problems with 1, 5, 10, and 20 immersed boundaries. The test with one circle is the test used previously in this paper. The domains with 5, 10, and 20 circles are pictured in Fig. 10. The centers and radii of the circles were selected randomly from a uniform distribution. The radii range between 0.05 and 0.15. No circles overlap, and they are all at least two Eulerian mesh points away from the domain boundaries. This second constraint ensures that the support of the discrete delta function does not intersect the domain boundary. Although selected randomly, the same set of circles was used for all numerical tests. The spacing

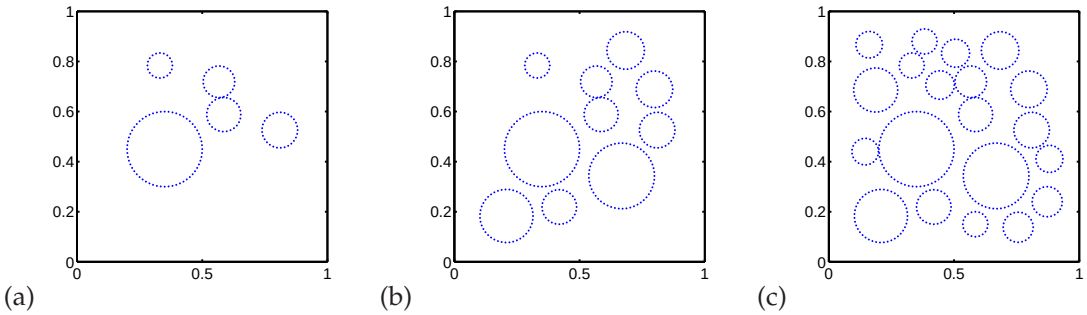


Figure 10: Domains with (a) 5, (b) 10, and (c) 20 circular immersed boundaries.

Table 4: For different numbers of circles, N_{IB} is the number of Lagrangian grid points used when the Eulerian grid has $N_{\text{grid}}=31^2=961$ grid points. The number of rows of the matrix modified by the immersed boundaries is R_{IB}.

circles	N_{IB}	N_{IB}/N_{grid}	R_{IB}	$R_{\text{IB}}/N_{\text{grid}}$
1	60	0.062	147	0.153
5	161	0.168	346	0.360
10	344	0.358	608	0.633
20	577	0.600	831	0.865

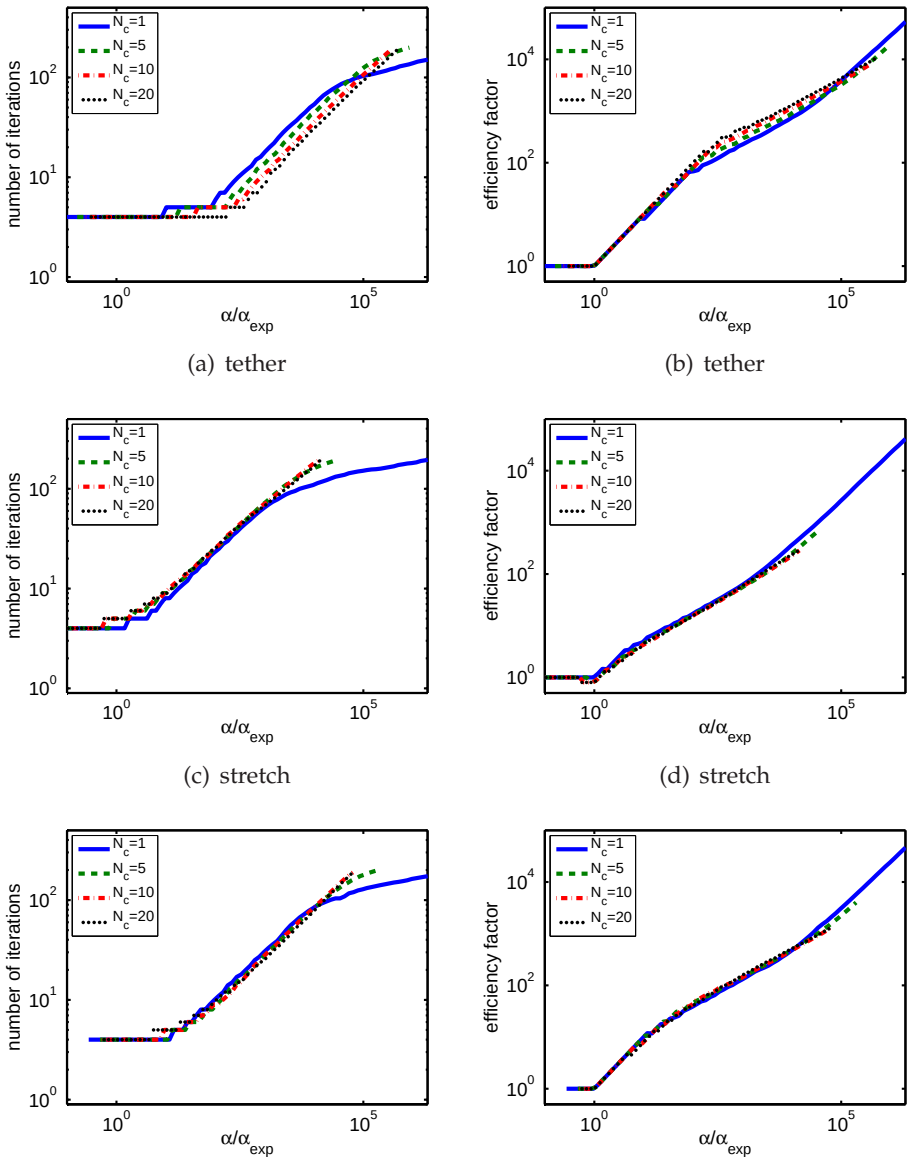
Table 5: Spectral radius of the matrix $S^*L^{-1}SA$ and the corresponding values of α_{exp} for the different test problems for an Eulerian grid spacing of $\Delta x=2^{-5}$.

	spectral radius			α_{exp}		
circles	tether	stretch	bend	tether	stretch	bend
1	0.1662	5.482	$6.938\cdot10^{3}$	12.04	0.3648	$2.883\cdot10^{-4}$
5	0.3208	9.275	$1.073\cdot10^{4}$	6.235	0.2156	$1.864\cdot10^{-4}$
10	0.4777	10.00	$1.087\cdot10^{4}$	4.186	0.1999	$1.840\cdot10^{-4}$
20	0.6174	11.28	$1.398\cdot10^{4}$	3.240	0.1773	$1.431\cdot10^{-4}$

of the immersed boundary points on each circle is approximately one half the Eulerian grid spacing. In Table 4, for an Eulerian grid with spacing $\Delta x=2^{-5}$ and $N_{\text{grid}}=31^2=961$ total grid points, we report the total number of immersed boundary points N_{IB} and the number of rows of the matrix modified by the presence of the immersed boundaries, R_{IB}, for the different test problems. We also report the ratios of these quantities to the total number of Eulerian grid points. For the test problem with only one immersed boundary, the ratio of Lagrangian grid points to Eulerian points is only about 6% and only about 15% of the equations are modified by the immersed boundaries. By contrast, when there are 20 immersed boundaries, the are about 10 times more Lagrangian points, and about 87% of the equations are modified by the immersed boundaries.

The maximum time step allowed by the explicit method is different for the different numbers of immersed boundary points. Recall from (4.5) that the spectral radius of $S^*L^{-1}SA$ determines the stability limit and that α_{exp} is related to the spectral radius by (4.7). In Table 5, we give the spectral radius and corresponding values of α_{exp} for the different test problems for an Eulerian grid with spacing $\Delta x=2^{-5}$.

We solve the model problem on the different domains using the multigrid preconditioned GMRES to a tolerance of 10^{-6}. The number of iterations and corresponding efficiency factors for different constitutive laws are shown in Fig. 11. The results are similar for the different numbers of immersed boundary points. However, in the case of one immersed boundary, the rate of increase slows down for very large values of the stiffness. This feature is not seen with more immersed boundary points. We conjecture that this seeming increase in efficiency is an artifact of having a small number of immersed boundary points. In general, we conclude that the efficiency of the method is indepen-



(a) tether (b) tether

(c) stretch (d) stretch

(e) bend (f) bend

Figure 11: Number of iterations of the multigrid preconditioned GMRES method to reduce the residual by a factor of 10^{-6} and the corresponding efficiency factors for the different test problems on an Eulerian grid with spacing $\Delta x = 2^{-5}$.

dent of the number of immersed boundary points. There is more work per iteration with more immersed boundary points, but the additional points increase the work of both the explicit method and implicit method proportionally.

6 Discussion

The popularity of the immersed boundary method is due, in part, to its simplicity and robustness. Explicit time integration requires a code to solve the fluid equations along with a few routines to compute forces and transfer data between the Eulerian and Lagrangian grids. The price of this simplicity is the severe time step restriction. The implicit methods that have been developed to date require specialized algorithms to achieve a substantial improvement in efficiency over explicit time methods.

The goal of the research presented in this paper is to investigate implicit methods that balance efficiency, robustness, and simplicity. A feature that distinguishes our approach from other successful implicit IB methods is that we formulate the problem only on the Eulerian grid. This formulation facilitates the use of standard tools to solve the equations. The multigrid method uses geometric coarsening and Gauss-Seidel for the smoother – tools which are readily available in existing codes. Another advantage of working only on the Eulerian grid is that the efficiency of the method is independent of the number of Lagrangian points.

The efficiency of the multigrid solver alone is not very impressive, but the multigrid method is a very effective preconditioner for Krylov methods. It may be possible to improve the multigrid algorithm by using a more sophisticated smoother. The goal of this paper is not to find the best possible multigrid algorithm, but rather, to show that respectable efficiency can be achieved with a very simple algorithm. In the future, we will investigate improvements to the multigrid algorithm, but of course, there is a tradeoff between efficiency and simplicity.

Our measurement of efficiency is based on the iteration count, which is admittedly too simple. We explicitly form the discrete force operator, the spreading operator, and the interpolation operator. Forming the matrices is no more expensive than applying them. Similarly, the smoothing step of the implicit method is no more expensive than the application of the forces in the explicit method. However, there is a much larger storage cost of the algorithm compared to the explicit method. The actual efficiency depends not only on the iteration count, but also on the implementation. Because our algorithm uses standard smoothing and coarsening, the implementation is not complex. We chose not to perform timing tests, because we only explored model problems, and such comparisons are not meaningful.

We used a model problem rather than the immersed boundary equations, because this facilitated explorations of different approaches. In addition, because there is only one parameter in the equations, the stiffness, it was simple to assess the results. The model "momentum equation" mimics Stokes equations rather than Navier-Stokes. The time independent momentum equation is more challenging than the time dependent equation. Adding the time derivative to the model is straightforward, and it would result in faster convergence because the operator inverted at each time step is better conditioned and the previous time solution is a good initial guess to the next solution.

Extending the algorithm from the model problem to the immersed boundary equa-

tions requires including the convection terms and incompressibility constraint. The stiffness in the immersed boundary method comes from the elastic forces, not the convection terms. Thus the convection terms can be discretized explicitly without imposing a severe time step restriction. Solving for the velocity and pressure simultaneously is more challenging, and will require further development of the algorithm.

Multigrid can be used to solve for the velocity and pressure simultaneously, but special smoothers must be used. The two main classes of smoothers are coupled smoothers [26] and distributed smoothers [1]. As the name suggests, coupled smoothers locally solve for the velocity and pressure simultaneously. Distributive smoothers are applied to a transformed system in which the velocity components and the pressure are decoupled. For the implicit IB equations, the operators to smooth in the transformed system are of the same form as the operator studied in the model problem.

There are other approaches to extending the algorithm for the model problem to incompressible flow. Applying preconditioners to the discrete Stokes equations that are based on approximate block factorizations involve applying a fast approximate inverse Laplacian [4]. For implicit IB equations, the multigrid algorithm developed in this paper could be used to apply an approximate inverse that includes both the viscous and elastic stresses. Alternatively, for time dependent problems, projection methods are a popular way of handling the incompressibility. In a projection method, the momentum equation is advanced in time, while the pressure is held constant. The intermediate velocity field is then decomposed into a divergence-free field and a gradient field. In the first step of a projection method, the problem solved is identical to our model problem (with time dependence). Therefore the algorithm we propose can be used in conjunction with a projection method without modification.

Acknowledgments

This work was supported in part by UCOP grant 09-LR-03-116724-GUYR to RG and BP, as well as NSF-DMS grant 0540779 to RG. The authors would like to thank Grady Wright and Boyce Griffith for helpful discussions related to this project.

References

[1] A. Brandt and N. Dinar, Multigrid solutions to elliptic flow problems, in Numerical Methods for Partial Differential Equations, S. Parter, ed., Academic Press, New York, 1979, pp. 53–147.
[2] H. D. Ceniceros and J. E. Fisher, A fast, robust, and non-stiff immersed boundary method, J. Comput. Phys., 230 (2011), 5133–5153.
[3] H. D. Ceniceros, J. E. Fisher, and A. M. Roma, Efficient solutions to robust, semi-implicit discretizations of the immersed boundary method, J. Comput. Phys., 228 (2009), 7137–7158.
[4] H. C. Elman, D. J. Silvester, and A. J. Wathen, Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics, Oxford University Press, New York, 2005.

[5] Z. Gong, H. Huang, and C. Lu, Stability analysis of the immersed boundary method for a two-dimensional membrane with bending rigidity, Commun. Comput. Phys., 3 (2008), 704–723.
[6] B. E. Griffith, R. D. Hornung, D. M. McQueen, and C. S. Peskin, An adaptive, formally second order accurate version of the immersed boundary method, J. Comput. Phys., 223 (2007), 10–49.
[7] T. Y. Hou and Z. Shi, An efficient semi-implicit immersed boundary method for the navier-stokes equations, J. Comput. Phys., 227 (2008), 8968–8991.
[8] T. Y. Hou and Z. Shi, Removing the stiffness of elastic force from the immersed boundary method for the 2d stokes equations, J. Comput. Phys., 227 (2008), 9138–9169. Special Issue Celebrating Tony Leonard's 70th Birthday.
[9] D. Le, J. White, J. Peraire, K. Lim, and B. Khoo, An implicit immersed boundary method for three-dimensional fluid-membrane interactions, J. Comput. Phys., 228 (2009), 8427–8445.
[10] L. Lee and R. J. LeVeque, An immersed interface method for incompressible navier–stokes equations, SIAM J. Sci. Comput. 25 (2003), 832–856.
[11] A. A. Mayo and C. S. Peskin, An implicit numerical method for fluid dynamics problems with immersed elastic boundaries, in Fluid Dynamics in Biology (Seattle, WA, 1991), vol. 141 of Contemp. Math., Amer. Math. Soc., Providence, RI, 1993, pp. 261–277.
[12] R. Mittal and G. Iaccarino, Immersed boundary methods, Annual Review of Fluid Mechanics, 37 (2005), 239–261.
[13] Y. Mori and C. S. Peskin, Implicit second-order immersed boundary methods with boundary mass, Computer Methods in Applied Mechanics and Engineering, 197 (2008), 2049–2067. Immersed Boundary Method and Its Extensions.
[14] E. P. Newren, A. L. Fogelson, R. D. Guy, and R. M. Kirby, Unconditionally stable discretizations of the immersed boundary equations, J. Comput. Phys., 222 (2007), 702–719.
[15] E. P. Newren, A. L. Fogelson, R. D. Guy, and R. M. Kirby, A comparison of implicit solvers for the immersed boundary equations, Computer Methods in Applied Mechanics and Engineering, 197 (2008), 2290–2304. Immersed Boundary Method and Its Extensions.
[16] C. W. Oosterlee and T. Washio, An evaluation of parallel multigrid as a solver and a preconditioner for singularly perturbed problems, SIAM J. Sci. Comput., 19 (1998), 87–110.
[17] F. Pacull and M. Garbey, The multigrid/tau-extrapolation technique applied to the immersed boundary method, in Domain Decomposition Methods in Science and Engineering XVI, O. B. Widlund and D. E. Keyes, eds., vol. 55 of Lecture Notes in Computational Science and Engineering, Springer Berlin Heidelberg, 2007, pp. 707–714.
[18] C. S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys., 25 (1977), 220–252.
[19] A. M. Roma, C. S. Peskin, and M. J. Berger, An adaptive version of the immersed boundary method, J. Comput. Phys., 153 (1999), 509–534.
[20] Y. Saad and M. H. Schultz, Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput., 7 (1986), 856–869.
[21] J. M. Stockie and B. R. Wetton, Analysis of stiffness in the immersed boundary method and implications for time-stepping schemes, J. Comput. Phys., 154 (1999), 41–64.
[22] M. Sussman, A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome, An adaptive level set approach for incompressible two-phase flows, J. Comput. Phys., 148 (1999), 81–124.
[23] O. Tatebe, The multigrid preconditioned conjugate gradient method, in Proceedings of the Sixth Copper Mountain Conference on Multigrid Methods, 1993, pp. 621–634.

[24] U. Trottenberg, C. W. Oosterlee, and A. Schuller, Multigrid, Academic Press, London, 2000.
[25] C. Tu and C. S. Peskin, Stability and instability in the computation of flows with moving immersed boundaries: A comparison of three methods, SIAM J. Sci. Stat. Comput., 13 (1992), 1361–1376.
[26] S. P. Vanka, Block-implicit multigrid solution of navier-stokes equations in primitive variables, J. Comput. Phys., 65 (1986), 138–158.
[27] G. B. Wright, R. D. Guy, and A. L. Fogelson, An efficient and robust method for simulating two-phase gel dynamics, SIAM J. Sci. Comput., 30 (2008), 2535–2565.
[28] L. Zhu and C. S. Peskin, Simulation of a flapping flexible filament in a flowing soap film by the immersed boundary method, J. Comput. Phys., 179 (2002), 452–468.