ENHANCING THE IMMERSED BOUNDARY METHOD: STABILITY, VOLUME
CONSERVATION, AND IMPLICIT SOLVERS

by

Elijah Paul Newren

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Mathematics
The University of Utah

May 2007
SUPERVISORY COMMITTEE APPROVAL

of a dissertation submitted by

Elijah Paul Newren

This dissertation has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

Chair: Aaron L. Fogelson

Robert D. Guy

Robert M. Kirby

Chris R. Johnson

Richard D. Rabbitt
To the Graduate Council of the University of Utah:

I have read the dissertation of Elijah Paul Newren in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to The Graduate School.

Date

Aaron L. Fogelson
Chair, Supervisory Committee

Approved for the Major Department

Aaron J. Bertram
Chair/Dean

Approved for the Graduate Council

David S. Chapman
Dean of The Graduate School
ABSTRACT

The Immersed Boundary method has long been used to study fluid-structure interaction problems, particularly in biological applications but increasingly with an engineering focus as well. Despite this success, the IB method has presented a number of challenges to practitioners. Due to the large stiffness of the elastic forces involved, the timestep has been severely restricted in explicit discretizations (and even in many implicit discretizations) in order to maintain stability. Because the computed velocity field in which the immersed boundary moves is not continuously divergence-free, fluid can leak across pressurized membranes and this can result in a lack of volume conservation for closed pressurized objects. In addition to the constraint of incompressibility that makes the Navier-Stokes equations difficult to solve, coupling the immersed boundary positions and forces to the fluid velocity increases the difficulty of solving the equations. Solving the fully coupled system of equations can be computationally expensive, but using an explicit discretization to decouple the system can result in a severe timestep restriction that also results in high computational cost.

For some of these challenges, methods have been introduced to fix the problem, but these methods have not been widely adopted due to various drawbacks. For other challenges, solutions have proved elusive. This dissertation explores each of these challenges and presents new solutions for each of them.
To Whom it May Concern
CONTENTS

ABSTRACT ................................................................. iv
ACKNOWLEDGMENTS ................................................. ix

CHAPTERS

1. INTRODUCTION ....................................................... 1
   1.1 Stability ....................................................... 2
   1.2 Volume Conservation ......................................... 3
   1.3 Efficiency .................................................... 4

2. THE IMMERSED BOUNDARY METHOD ................................. 6
   2.1 Spatial Discretization ......................................... 8
   2.2 Explicit Temporal Discretization ............................. 12
   2.3 Navier-Stokes Solver .......................................... 13
      2.3.1 Advection Terms ....................................... 14
      2.3.2 Projection Methods ..................................... 16
      2.3.3 Approximate Projection ................................ 18
      2.3.4 Poisson Solver: Multigrid .............................. 21
      2.3.5 Poisson Solver: Fast Fourier Transform .............. 22
   2.4 Pseudocode ................................................... 23

3. STABILITY ................................................................... 25
   3.1 Temporal Discretization ........................................ 25
   3.2 Unconditionally Stable Schemes .............................. 27
      3.2.1 Crank-Nicolson for the Spatially Continuous Problem ... 27
      3.2.2 Projection Methods and Discrete Delta Functions ....... 30
      3.2.3 Unconditional Stability of Backward Euler ............ 33
   3.3 Computational Results .......................................... 36
      3.3.1 Computational Verification of the Method .............. 39
         3.3.1.1 Refinement Study ..................................... 39
         3.3.1.2 Comparison to Explicit Method ...................... 39
         3.3.1.3 Parameter Testing on the Inviscid Problem ....... 41
         3.3.1.4 Comparison to Method 1A ............................ 42
      3.3.2 Unconditional Stability with an Approximate Projection ...... 43
      3.3.3 Stability for the Full Navier-Stokes Equations ...... 44
      3.3.4 Volume Loss and Stability ............................... 45
   3.4 Discussion ...................................................... 46
3.4.1 Efficiency ................................................. 47  
3.4.2 Accuracy .................................................. 48  
3.4.3 Volume Conservation ...................................... 49  
3.4.4 Extensions ................................................ 49  
3.4.5 Explicit Method Stability ................................. 50  
3.5 Conclusions .................................................. 50  

4. VOLUME CONSERVATION ........................................ 52  
4.1 Continuous vs. Discretely  
  Divergence-free Vector Fields ................................. 52  
4.2 Discrete Incompressibility  
  Constraint on a Surface ...................................... 54  
4.3 Polygonal Area ............................................... 56  
4.4 Effect of Velocity Correction .............................. 58  
4.5 Exact Conservation of Polygonal Area .................... 59  
4.6 Unequally Spaced Surface Points ........................... 60  
4.7 Computational Results ..................................... 61  
  4.7.1 Refinement Study ...................................... 61  
  4.7.2 Volume Conservation Comparison ...................... 62  
  4.7.3 Effect on Energy Conservation ......................... 62  
4.8 Conclusions .................................................. 63  

5. EFFICIENCY — SOLVER POSSIBILITIES ....................... 65  
5.1 Direct Application of Linear Solvers ..................... 66  
5.2 Newton-like Methods ........................................ 66  
  5.2.1 Newton’s Method .................................... 67  
  5.2.2 Quasi-Newton Methods ................................ 68  
5.3 Schur Complement Systems .................................. 70  
  5.3.1 Fixed Point Methods .................................. 73  
  5.3.2 Projection Methods ................................... 74  
  5.3.3 Krylov Solvers ...................................... 74  

6. EFFICIENCY — PROJECTION METHODS ......................... 77  
6.1 Basic Theory ................................................ 78  
  6.1.1 Subtleties in Meaning of Commutativity ............. 79  
  6.1.2 Common Assumptions .................................. 80  
  6.1.3 Velocity-correction Projection Methods .............. 80  
  6.1.4 Pressure-correction Projection Methods .............. 82  
  6.1.5 Alternative Decomposition ............................. 84  
6.2 Important Properties of IB Equations ..................... 87  
6.3 Splitting Error .............................................. 88  
6.4 Improvements: The New Update Operator .................. 95  
  6.4.1 Splitting Error ...................................... 96  
  6.4.2 Relationship to the New Decomposition .............. 97  
  6.4.3 Generalizes Pressure Update Operator ............... 97  
  6.4.4 Consistency of the Pressure .......................... 97
6.4.5 Postprocessing to Reduce the Splitting Error ............... 98
6.4.6 Summary .................................................. 99
6.5 Connection to Fixed Point Schemes ......................... 100
  6.5.1 Iterative Schemes .................................. 100
  6.5.2 Fixed Point on $p$ .................................. 101
  6.5.3 Fixed Point on $u$ .................................. 102
6.6 Computational Tests .................................. 104
6.7 Velocity-correction Projection Methods ................. 108
6.8 Conclusions ........................................... 110

7. EFFICIENCY — COMPARISON OF IMPLICIT METHODS 112
  7.1 Additional Test Problem ................................. 112
  7.2 Results .................................................. 113
    7.2.1 Ellipse Problem ................................. 114
    7.2.2 Wing Problem .................................. 116
  7.3 Conclusions ........................................... 117

APPENDICES
A. GENERALIZED PROJECTION METHOD DETAILS ...... 119
B. ENERGY CORRECTIONS ................................. 141

REFERENCES .................................................. 144
ACKNOWLEDGMENTS

This work was supported in part by the Department of Energy Computational Science Graduate Fellowship Program of the Office of Science and National Nuclear Security Administration in the Department of Energy under contract DE-FG02-97ER25308, and in part by NSF grants DMS-0139926 and DMS-EMSW21-0354259.

I would like to thank all the members of my supervisory committee for the time and effort they put in helping me. I would also like to thank the clotting group for putting up with the pain of listening to me in our Friday meetings — and repeatedly coming back to do it again. I learned a lot from their input at those meetings. In particular from both groups, though, I would like to thank Bob for believing in me, checking on me, and pushing me on when I was at my lowest and was completely convinced I was going to fail. Without his intervention, I am pretty sure that I would have.

Finally, I would like to thank my wonderful wife and daughters. Their patience and support was nothing short of amazing (well, Sarah’s patience left a bit to be desired as a newborn, but the others have been remarkably patient).
CHAPTER 1

INTRODUCTION

The Immersed Boundary (IB) method was introduced by Peskin in the early 1970s to solve the coupled equations of motion of a viscous, incompressible fluid and one or more massless, elastic surfaces or objects immersed in the fluid [31]. Rather than generating a curve-fitting grid for both exterior and interior regions of each surface at each timestep and using these to determine the fluid motion, Peskin instead employed a uniform Cartesian grid over the entire domain and discretized the immersed boundaries by a set of points that are not constrained to lie on the grid. The key idea that permits this simplified discretization is the replacement of each suspended object by a suitable contribution to a force density term in the fluid dynamics equations in order to allow a single set of fluid dynamics equations to hold in the entire domain with no internal boundary conditions.

The IB method was originally developed to model blood flow in the heart and through heart valves [31, 33, 34], but has since been used in a wide variety of other applications, particularly in biofluid dynamics problems where complex geometries and immersed elastic membranes or structures are present and make traditional computational approaches difficult. Examples include platelet aggregation in blood clotting [11, 13], swimming of organisms [11, 12], biofilm processes [10], mechanical properties of cells [1], cochlear dynamics [4], and insect flight [24, 25].

The Immersed Interface Method (IIM) was developed by Leveque and Li to address the low order accuracy found in the IB method when applied to problems with sharp interfaces [21]. The IIM differs from the IB method in the spatial discretization method used to handle the singular forces appearing in the continuous equations of motion. While we do not address the spatial discretizations involved
in the IIM and instead focus on the IB method in this dissertation, we do present some discussion of that method since the two are closely related (hybrids of the two even exist, such as [20]).

The IB method has presented a number of challenges to practitioners. Due to the large stiffness of the elastic forces involved, the timestep has been severely restricted in explicit discretizations (and even in many implicit discretizations) in order to maintain stability. Because the computed velocity field in which the immersed boundary moves is not continuously divergence-free, fluid can leak across pressurized membranes and result in a lack of volume conservation for closed pressurized objects. In addition to the constraint of incompressibility which makes the Navier-Stokes equations difficult to solve, coupling the immersed boundary positions and forces to the fluid velocity increases the difficulty of solving the equations. Solving the fully coupled system of equations can be computationally expensive, but using an explicit discretization to decouple the system can result in a severe timestep restriction that also results in high computational cost.

This dissertation addresses each of these challenges of the IB method. The rest of this chapter discusses each problem in more depth, while Chapter 2 provides background on the idea of the IB method and how the IB equations are typically solved in practice. The following chapters then provide solutions for each of the problems, with Chapter 3 covering the severe timestep restriction, Chapter 4 exploring our new simple solution to the volume leakage problem, and Chapters 5-7 discussing how to obtain an efficient solver for the stable implicit discretization.

1.1 Stability

The IB and IIM methods suffer from a severe timestep restriction needed to maintain stability, as has been well documented in the literature [11, 21, 32, 38]. This time step restriction is typically much more stringent than the one that would be imposed from using explicit differencing of the advective or diffusive terms [8]. Much effort has been expended attempting to alleviate this severe restriction. For example, in some problems, the fluid viscosity has been artificially increased by
a couple of orders of magnitude [35]. In others, authors filter out high frequency oscillations of the interface [21, 43]. Much effort has been put into developing implicit and semi-implicit methods [11, 21, 23, 41].

Despite these efforts the severe timestep restriction has remained. The instability of these methods is known not to be a problem related to the advection terms in the Navier Stokes equations, and is known to arise in the parameter regime corresponding to large boundary force and small viscosity [38], but there is little understanding of why this parameter regime is problematic. Despite the effort put into implicit methods which couple the equations of motion for the fluid and immersed boundary, the lack of stability has been puzzling. Conjectures as to causes of instability in these methods turn out to be misleading. It is a common belief in the community that using time-lagged spreading and interpolation operators (i.e., time-lagged discretizations of the delta functions in equations (2.2) and (2.5) of Chapter 2) will cause instability, and that therefore a fully-implicit scheme (i.e., one without time-lagged spreading and interpolation operators) is necessary for unconditional stability [20, 23, 27, 37]. It has also been conjectured that the lack of stability and corresponding timestep restriction in fully-implicit schemes such as [23] is due to accumulation of error in the incompressibility condition [38].

We will present some discretizations that we prove to be unconditionally stable in conjunction with unsteady Stokes flow, and in so doing, show (i) that methods need not be fully-implicit in order to achieve unconditional stability and (ii) that accumulation of error in the incompressibility condition is not the cause of the observed instability of previous implicit methods.

1.2 Volume Conservation

When an exact projection method is used for solving the Navier-Stokes equations (as a step in solving the IB equations), the velocity field on the Eulerian grid will be discretely divergence-free, but this does not guarantee that the interpolated velocity field (in which the immersed boundary moves) is continuously divergence-free. When the immersed boundary is a closed object, this can result in the volume
of fluid inside that object not being conserved. This is particularly problematic for closed membranes under tension, which have been shown to exhibit volume loss with the IB method [8, 21, 32, 35].

There are multiple ways to fix this volume loss problem. One solution is to use the IIM [21, 22] or a hybrid IB/II method [20]. These methods were designed to provide higher order accurate solutions for problems with a sharp interface, but they also fix the volume loss of the IB method. However, these methods are difficult to implement in comparison to the IB method and are not readily extendable to three dimensions for general problems. Another alternative solution for fixing the volume loss problem is using the modified divergence stencils of Peskin and Printz [35]. Peskin and Printz reported substantially improved volume conservation, but at the cost of wider stencils and more computationally expensive interpolation and spreading.

We will present a method requiring only a small modification to existing codes that substantially improves volume conservation with only a marginal computational cost. This method works by correcting the interpolated velocity field to satisfy a discrete divergence-free constraint defined on the Lagrangian interface.

1.3 Efficiency

In addition to the constraint of incompressibility which makes the Navier-Stokes equations difficult to solve, the IB equations also involve the coupling of the immersed boundary positions and forces to the fluid velocity. This extra coupling (or constraint, depending on your point of view) makes numerically solving the IB equations more challenging and makes it hard to find a method which can efficiently solve the system.

Perhaps the easiest way to solve the IB equations is to use an explicit method in order to decouple the force computation and boundary location update from the solution of the fluid velocity. However, doing so results in a severe timestep restriction. This severe timestep restriction means that much computational effort is wasted solving for variables on a timescale where there is nearly no change.
However, despite allowing a much larger timestep, implicit methods for solving the IB equations have often been more computationally expensive than just using a standard explicit discretization [23, 28, 41]. In other words, iterative methods for solving such implicit discretizations often require more iterations to arrive at the solution at the end of one timestep than the number of timesteps that would be required to compute to the same time with an explicit method.

We will outline some of the implicit methods that have been implemented, including some discussion of those already known to be too slow. We will explain why projection methods in general will also be unsuitable as a solver for the implicit discretization of the IB equations. Finally, we will compare and contrast a few fully coupled implicit solvers, some of which were inspired by some of our generalized projection methods.
CHAPTER 2
THE IMMERSED BOUNDARY METHOD

In this chapter, we briefly introduce the idea behind the IB method and derive the IB equations. We then detail how these equations can be solved numerically. While we include only a few variations on the numerical solution method in this chapter, many of the subsolvers and substeps of each timestep remain the same even when higher level details of the overall discretization or solution method are modified.

In the IB method, an Eulerian description based on the Navier-Stokes equations is used for the fluid dynamics, and a Lagrangian description is used for each object immersed in the fluid. The boundary is assumed to be massless, so that all of the force generated by distortions of the boundary is transmitted directly to the fluid. An example setup in 2D with a single immersed boundary curve is shown in Figure 2.1. Lowercase letters are used for Eulerian state variables, and uppercase letters are used for Lagrangian variables. Thus, \( X(s, t) \) is a vector function giving the location of points on \( \Gamma \) as a function of arclength (in some reference configuration), \( s \), and time, \( t \). The boundary is modeled by a singular force, which is incorporated into the forcing term, \( f \), in the Navier-Stokes equations. The Navier-Stokes equations are then solved to determine the fluid velocity throughout the domain, \( \Omega \). Since the immersed boundary is in contact with the surrounding fluid, its velocity must be consistent with the no-slip boundary condition. Thus the immersed boundary moves at the local fluid velocity. This results in the following set of equations:
Figure 2.1. Example immersed boundary curve, $\Gamma$, described by the function $X(s,t)$.

\[
F(s,t) = \mathcal{A}_f X(s,t) \tag{2.1}
\]

\[
f(x, t) = \int_{\Gamma} F(s, t) \delta(x - X(s,t)) \, ds \tag{2.2}
\]

\[
\rho (u_t + (u \cdot \nabla)u) = -\nabla p + \mu \Delta u + f \tag{2.3}
\]

\[
\nabla \cdot u = 0 \tag{2.4}
\]

\[
\frac{dX(s,t)}{dt} = u(X(s,t), t) = \int_{\Omega} u(x, t) \delta(x - X(s,t)) \, dx. \tag{2.5}
\]

Equation (2.1) is the constitutive law modeling the force generated from the configuration of the immersed object due to its elasticity, meaning that the force generation operator, $\mathcal{A}_f$, is problem dependent. Note that the terminology “force generation operator” might be misleading to those not familiar with the IB method, as the force used in the fluid dynamics equations is not the one defined in equation (2.1) but the one in equation (2.2). The latter is nonlinear (in $X$) as well as singular due to the lower-dimensional line integral. The Lagrangian and Eulerian force and velocity are related through equations (2.2) and (2.5). Equations (2.3) and
(2.4) are the incompressible Navier-Stokes equations. Note that in equation (2.3) we assume that the density, \( \rho \), is constant.

The most common constitutive law used for the force generation operator is to assume that the material behaves like an object under elastic tension. For a boundary under tension, the strength of the force on the boundary is given by

\[
\mathbf{F}(s,t) = \frac{\partial}{\partial s}(T(s,t)\mathbf{\tau}(s,t)),
\]

(2.6)

where \( T(s,t) \) is the tension at the given point and \( \mathbf{\tau}(s,t) \) is the tangent vector to the boundary at that point (see [34] for a derivation). The tangent vector is

\[
\mathbf{\tau}(s,t) = \frac{\partial \mathbf{X}}{\partial s} / \left\| \frac{\partial \mathbf{X}}{\partial s} \right\|.
\]

(2.7)

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

\[
T(s,t) = T_0 \left( \left\| \frac{\partial \mathbf{X}}{\partial s} \right\| - 1 \right).
\]

(2.8)

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

\[
T(s,t) = T_0 \left( \left\| \frac{\partial \mathbf{X}}{\partial s} \right\| \right).
\]

(2.9)

In the latter case \( A_f \) simplifies to \( T_0 \frac{\partial^2}{\partial s^2} \).

### 2.1 Spatial Discretization

These equations are solved on a pair of computational grids: a cell centered Cartesian grid for Eulerian variables, and a discrete set of points for the Lagrangian variables. An example setup in 2D with a single immersed boundary curve is shown in Figure 2.2. Assuming that the lower left corner of the domain is at the origin, the coordinates of the \( ij \)-th Eulerian gridpoint are

\[
x_{ij} = \left( \left( i + \frac{1}{2} \right) \Delta x, \left( j + \frac{1}{2} \right) \Delta y \right).
\]

(2.10)

A pair of subscripts on a variable denotes the location at which the Eulerian variable is being evaluated; thus \( u_{ij} \) denotes the value of the variable \( u \) at the \( ij \)-th
gridpoint. Lagrangian gridpoints are identified by a single index, with variables at such gridpoints identified by the corresponding index appearing as a subscript. Thus $F_k$ denotes the value of the variable $F$ at the $k$-th gridpoint. The location of the $k$th Lagrangian gridpoint is explicitly tracked in $X_k(t)$.

The interaction between these grids, governed by integration against a delta function in the continuous equations (2.2) and (2.5), is handled by introducing a regularized discrete delta function whose support is comparable to the mesh spacing. The discretized forms of equations (2.2) and (2.5) using this discrete delta function are

\[
\begin{align*}
    f_{ij} &= \sum_k F_k \delta_h(x_{ij} - X_k) \Delta s_k \quad (2.11) \\
    \frac{dX_k}{dt} &= U_k = \sum_{ij} u_{ij} \delta_h(x_{ij} - X_k) \Delta x \Delta y. \quad (2.12)
\end{align*}
\]
The discrete delta function appearing in equations (2.11) and (2.12) is derived from the requirement that a certain set of properties be satisfied; these include ensuring that the entire force is transmitted to the grid, that the force density on the grid is a continuous function of the IB point locations, and that the communication between Eulerian and Lagrangian grids is very localized. Additional conditions can be imposed as well, yielding different delta functions; see [32] for a more thorough treatment of the possibilities. We will use the delta function derived in [31],

\[ \delta_h(x, y) = \delta_h(x)\delta_h(y) \]  

\[ \delta_h(x) = \begin{cases} \frac{1}{4h} \left( 1 + \cos \left( \frac{x}{2h} \right) \right) & |x| \leq 2h \\ 0 & |x| \geq 2h \end{cases} \]  

where \( h = \Delta x \) for the delta function in the \( x \) direction and \( h = \Delta y \) for the delta function in the \( y \) direction. This regularized delta function is shown in Figure 2.3.

Figure 2.4 is a graphical representation showing the evaluation of equation (2.11) for a single IB point. Because of the stencil width of \( 4h \) in the definition of the delta function in (2.14), the force at any given IB point affects only the Eulerian force density at the 16 nearest gridpoints. Because the force at an IB point contributes to the Eulerian force density over a nonzero area in the discrete equations, this operation is known as the force spreading operation. Figure 2.4 shows the force from a single IB point being spread to the nearest 16 gridpoints.

Equation (2.12) is a simple interpolation operation and works much like the spreading operator but in reverse; Figure 2.4 could almost be reused as a depiction.
of equation (2.12) as well, but the interpolated field at the IB point would be an average of the surrounding values instead of being much bigger than them all.

The Navier-Stokes equations, equations (2.3) and (2.4), are discretized by introducing $\nabla^h, \nabla^h, \Delta^h$, and $(\mathbf{u} \cdot \nabla^h)$ as discrete analogs of $\nabla, \nabla^\cdot, \Delta,$ and $(\mathbf{u} \cdot \nabla)$, respectively. These will be discussed in Section 2.3.

The most straightforward discretization of equation (2.1) is to write the force at an immersed boundary point as a difference in the tensions on either side of that point. Assuming a single closed boundary with no external links, this can be written as

$$F_k = \frac{(T_{k+1/2}(t)\tau_{k+1/2}(t)) - (T_{k-1/2}(t)\tau_{k-1/2}(t))}{\Delta s}$$  \hspace{1cm} (2.15)

$$= \frac{\left( \frac{T_0}{\Delta s} \left( \|X_{k+1}(t) - X_k(t)\| - \ell_0 \frac{X_{k+1}(t) - X_k(t)}{\|X_{k+1}(t) - X_k(t)\|} \right) \right)}{\Delta s} - \frac{\left( \frac{T_0}{\Delta s} \left( \|X_{k-1}(t) - X_k(t)\| - \ell_0 \frac{X_{k-1}(t) - X_k(t)}{\|X_{k-1}(t) - X_k(t)\|} \right) \right)}{\Delta s}, \hspace{1cm} (2.16)$$

where $\ell_0$ is the resting length of the “springs” connecting IB points. The reason for calling the connection between IB points a “spring” in the discrete set of equations is because of the form of (2.16): $T_0/\Delta s$ serves as a spring constant, $\|X_i - X_k\| - \ell_0$ is the length by which the connection between IB points $i$ and $k$ has been stretched, and $(X_i - X_k)/\|X_i - X_k\|$ is a unit vector in the direction of the connection, making this look just like a Hooke’s Law spring.
Noting the similarity in the two terms of (2.16), we can instead write the force as a sum over IB points connected to IB point $k$:

$$F_k = \sum_i \frac{T_0}{\Delta s}(\|X_i(t) - X_k(t)\| - \ell_0) \frac{1}{\Delta s} \frac{X_i(t) - X_k(t)}{\|X_i(t) - X_k(t)\|}. \tag{2.17}$$

An additional advantage of writing in this manner is that it also makes clear how to handle external links connecting objects. This formula can also be generalized for the case of different reference lengths, $\Delta s_{ik}$, between IB points. Since force spreading requires some $\Delta s$ value defined at an IB point rather than at IB links, one can define $\Delta s_k$ by averaging $\Delta s_{ik}$ from connecting links (excluding “external” links). This changes the force calculation formula to

$$F_k = \sum_i \frac{T_0}{\Delta s_{ik}}(\|X_i(t) - X_k(t)\| - \ell_0) \frac{1}{\Delta s_k} \frac{X_i(t) - X_k(t)}{\|X_i(t) - X_k(t)\|}. \tag{2.18}$$

Although we have implemented (2.18), in hindsight it seems to be an unnecessary generalization; just forcing $\Delta s_{ik} = \Delta s_k = \text{constant}$ in the reference configuration would have simplified the code.

### 2.2 Explicit Temporal Discretization

The most common temporal discretization of the IB equations is a mixed explicit/implicit one, where the forces from the IB and advection terms in the Navier-Stokes equations are treated explicitly while the viscous terms are treated implicitly. The work involved in a timestep of such a method consists of five steps: (1) calculation of forces from the immersed elastic boundary, (2) spreading these elastic forces from the Lagrangian to the Eulerian grid, (3) solving the Navier-Stokes equations with the resulting Eulerian force, (4) interpolating velocities from the Eulerian to the Lagrangian grid, and (5) updating IB points according to the interpolated velocity.

We use a superscript to denote the value of a variable at a given timestep; thus $u^n(x) = u(x, n\Delta t)$ and $X^n(s) = X(s, n\Delta t)$. Using this notation, temporally
discretizing equations (2.1) through (2.5) according to the steps listed above using an explicit handling of all IB terms, we obtain

\[ F^k_n(s) = A_f X^n(s) \] (2.19)

\[ f^n(x) = \int F^n(s) \delta(x - X^n(s)) \, ds \] (2.20)

\[ \frac{u^{n+1} - u^n}{\Delta t} = -\nabla p^{n+\frac{1}{2}} - \rho [(u \cdot \nabla)u]^{n+\frac{1}{2}} + \frac{\mu}{2} \Delta (u^{n+1} + u^n) + f^n \] (2.21)

\[ \nabla \cdot u^{n+1} = 0 \] (2.22)

\[ \frac{X^{n+1} - X^n}{\Delta t} = u^{n+1}(X^n(s)) = \int_\Omega u^{n+1}(x) \delta(x - X^n(s)) \, dx. \] (2.23)

Combining this temporal discretization with the spatial discretizations in Section 2.1 we obtain

\[ F^n_k = \sum_i \frac{T_0}{\Delta s_{ik}} (\|X^n_i - X^n_k\| - \ell_0) \frac{1}{\Delta s_k} \frac{X^n_i - X^n_k}{\|X^n_i - X^n_k\|}. \] (2.24)

\[ f^n_{ij} = \sum_k F^n_k \delta_h(x_{ij} - X^n_k) \Delta s_k \] (2.25)

\[ \frac{u^{n+1}_{ij} - u^n_{ij}}{\Delta t} = -\nabla^h p_{ij}^{n+\frac{1}{2}} - \rho [(u \cdot \nabla^h)u]_{ij}^{n+\frac{1}{2}} + \frac{\mu}{2} \left( \Delta^h (u^{n+1} + u^n) \right)_{ij} + f^n_{ij} \] (2.26)

\[ \nabla^h \cdot u^{n+1} = 0 \] (2.27)

\[ \frac{X^{n+1}_{ij} - X^n_{ij}}{\Delta t} = \sum_{ij} u^{n+1}_{ij} \delta_h(x_{ij} - X^n_{ij}) \Delta x \Delta y. \] (2.28)

Equations (2.24) through (2.28) fully specify the algorithm of each time step for this explicit method with three exceptions: (1) how to evaluate the discrete operators \( \nabla^h, \nabla^h, \Delta^h, \) and \( (u \cdot \nabla^h), \) which have not yet been defined, (2) how the advection terms in equation (2.26) can be evaluated at the \( n + \frac{1}{2} \) time level, and (3) how to solve the Navier-Stokes equations (2.26) and (2.27), which must be solved in a coupled fashion. These issues are addressed in the next section.

### 2.3 Navier-Stokes Solver

To determine the fluid velocity in our model, we will need to solve the incompressible Navier-Stokes equations, (2.26) and (2.27). These equations can be difficult to solve, both because of the nonlinear advection term, \([u \cdot \nabla^h]u]^{n+\frac{1}{2}}, and
because we have an additional constraint on the velocities instead of an evolution
equation for the pressure. In this section, we will list a few ways of handling the
advection terms and outline a class of methods, known as projection methods,
for solving the incompressible Navier-Stokes equations. We also discuss some
subsolvers needed by the projection method.

2.3.1 Advection Terms

There are two tasks in handling the nonlinear advection term, \([ (\mathbf{u} \cdot \nabla^h)\mathbf{u} ]^{n+\frac{1}{2}}\):
deciding which spatial discretization and which temporal discretization to use.
Concentrating on the spatial discretization first, there are a couple of different
discretizations we have used. Given the definition

\[
H(x) = \begin{cases} 
1 & x > 0 \\
0 & \text{otherwise}, 
\end{cases}
\]  

(2.29)

the two choices of \((\mathbf{u} \cdot \nabla^h)\) we have used are

\[
\begin{align*}
((\mathbf{u} \cdot \nabla^h)_{\text{centered}})_{ij} &= u_{i,j} \left( \frac{c_{i+1,j} - c_{i,j-1}}{2\Delta x} \right) + v_{i,j} \left( \frac{c_{i,j+1} - c_{i,j-1}}{2\Delta y} \right) \\
((\mathbf{u} \cdot \nabla^h)_{\text{upwind}})_{ij} &= 
\begin{bmatrix}
H(u_{ij})u_{ij} \frac{c_{i+1,j} - c_{i,j}}{\Delta x} + H(-u_{ij})u_{ij} \frac{c_{i,j} - c_{i,j-1}}{\Delta x} \\
H(v_{ij})v_{ij} \frac{c_{i,j+1} - c_{i,j}}{\Delta y} + H(-v_{ij})v_{ij} \frac{c_{i,j} - c_{i,j-1}}{\Delta y}
\end{bmatrix}.
\end{align*}
\]  

(2.30)

Equations (2.30) and (2.31) only define the advection operator on scalar-valued
arrays but can be applied to vector-valued functions componentwise, i.e., \((\mathbf{u} \cdot \nabla^h)\mathbf{u} = ((\mathbf{u} \cdot \nabla^h)u, (\mathbf{u} \cdot \nabla^h)v)\). \((\mathbf{u} \cdot \nabla^h)_{\text{centered}}\) is second order accurate in space but potentially
problematic at higher Reynolds numbers, whereas \((\mathbf{u} \cdot \nabla^h)_{\text{upwind}}\) is only first order
accurate. The particular reason for choosing \((\mathbf{u} \cdot \nabla^h)_{\text{upwind}}\) is that we understand
its CFL constraint better, which allows us to explore stability more easily.

There are a variety of temporal discretizations possible; we list a few explicit
choices. Perhaps the easiest way of computing \([ (\mathbf{u} \cdot \nabla^h)\mathbf{u} ]^{n+\frac{1}{2}}\) is to use a simple
first-order extrapolation:

\[
[ (\mathbf{u} \cdot \nabla^h)\mathbf{u} ]^{n+\frac{1}{2}} = (\mathbf{u}^n \cdot \nabla^h)\mathbf{u}^n.
\]  

(2.32)

This provides low order accuracy, but when first order discretizations have been
used for other parts of the IB equations then such a replacement makes sense. One
then needs only to decide whether to use \(((u^n \cdot \nabla^h)_\text{centered} u^n)\) or \(((u^n \cdot \nabla^h)_{\text{upwind}} u^n)\) from Section 2.1. One could instead use a second order extrapolation

\[
[(u \cdot \nabla^h)u]^{n+\frac{1}{2}} = \frac{3}{2}[(u^n \cdot \nabla^h)u^n] - \frac{1}{2}[(u^{n-1} \cdot \nabla^h)u^{n-1}].
\] (2.33)

An alternative to extrapolating advective terms is to extrapolate \(u\) to get \(u^{n+\frac{1}{2}}\) and then difference that. In other words, set

\[
[(u \cdot \nabla^h)u]^{n+\frac{1}{2}} = (u^{n+\frac{1}{2}} \cdot \nabla^h)u^{n+\frac{1}{2}}
\] (2.34)
after first computing \(u^{n+\frac{1}{2}}\). A simple extrapolation might be \(u^{n+\frac{1}{2}} = \frac{3}{2}u^n - \frac{1}{2}u^{n-1}\), whereas a more sophisticated extrapolation would use the Taylor series expansion

\[
u^{n+\frac{1}{2}} = u^n + \frac{\Delta t}{2}(u^n) + O(\Delta t^2).
\] (2.35)
similar to the method of Bell et al.[3]. In (2.35), we can use the Navier-Stokes equations to eliminate the time derivative and obtain

\[
u^{n+\frac{1}{2}} = u^n + \frac{\Delta t}{2} \left(-[(u \cdot \nabla^h)u^n + \nu \Delta u^n + \frac{f^n}{\rho} - \nabla p^n] \right).
\] (2.36)

This can also be rewritten in a form that looks like a fully explicit discretization of the momentum equation of the Navier-Stokes equations:

\[ho \frac{u^{n+\frac{1}{2}} - u^n}{\Delta t/2} + \nabla p^n = -\rho[(u \cdot \nabla^h)u^n] + \mu \Delta u^n + f^n.\] (2.37)

This equation is easier to solve than our partially implicit Navier-Stokes equations because everything on the right hand side is known. Since we also expect \(u^{n+\frac{1}{2}}\) to be divergence-free, we can solve this equation in two steps. First, solve for \(p^n\) via

\[
\nabla \cdot \nabla p^n = \nabla \cdot (-\rho[(u \cdot \nabla^h)u^n] + \mu \Delta u^n + f^n),
\] (2.38)
and then obtain \(u^{n+\frac{1}{2}}\) via equation (2.36).

We do not have enough experience to thoroughly compare and contrast these different choices. We initially implemented the combination (2.37) and (2.38), later added (2.32) for testing stability since its stability properties were the most clear, and (2.33) seemed like a natural method to mention. We expect that all three will
provide good results for low to moderate Reynolds number. By way of contrast, we suspect that (2.32) is good at higher Reynolds number but is lower order accurate, that the combination (2.37) and (2.38) will be more expensive (due to the extra Poisson solve) and more accurate than (2.33) for a larger range of Reynolds number, and that those wanting a second order method that handles higher Reynolds number will need to use a method like that of Bell et al.[3].

2.3.2 Projection Methods

Projection methods are based on the Hodge decomposition, which states that any vector field can be decomposed uniquely and orthogonally into a divergence-free vector field and a gradient field, i.e.,

\[ \forall \text{ periodic } \hat{\mathbf{r}} \exists! \text{ periodic } \mathbf{w}, \nabla \psi \]

with \( \nabla \cdot \mathbf{w} = 0 \) and \( \langle \mathbf{w}, \nabla \psi \rangle = 0 \)

such that \( \hat{\mathbf{r}} = \mathbf{w} + \nabla \psi \).

While we have stated the Hodge Decomposition here assuming periodic boundary conditions, it can be restated for other boundary conditions as well. The proof of this decomposition is a simple constructive one. While we omit the details, the idea is to take the divergence of the decomposition we want, \( \hat{\mathbf{r}} = \mathbf{w} + \nabla \psi \), to obtain the Poisson equation

\[ \nabla \cdot \hat{\mathbf{r}} = 0 + \nabla \cdot \nabla \psi \]  \hspace{1cm} (2.39)

for \( \psi \), solve this equation, and set \( \mathbf{w} = \hat{\mathbf{r}} - \nabla \psi \).

The reason that the Hodge Decomposition is useful is that the Navier-Stokes equations can be written in the following alternate form

\[ \rho \mathbf{u}_t + \nabla p = -\rho (\mathbf{u} \cdot \nabla) \mathbf{u} + \mu \Delta \mathbf{u} + \mathbf{f}. \]  \hspace{1cm} (2.40)

Note that since \( \mathbf{u} \) is divergence-free for all \( t \), \( \rho \mathbf{u}_t \) is also. Defining \( \mathbb{P} \) as the operator which produces the divergence-free portion of a vector from the Hodge decomposition, this can also be written as

\[ \rho \mathbf{u}_t = \mathbb{P} (-\rho (\mathbf{u} \cdot \nabla) \mathbf{u} + \mu \Delta \mathbf{u} + \mathbf{f}). \]  \hspace{1cm} (2.41)
This removes the pressure and the extra constraint on the velocities from the equations and leaves us with just an evolution equation for the velocities. Discretizing this directly is difficult, but it does suggest a method to solve these equations; this type of method is known as a projection method.

We discuss projection methods in more detail in Chapter 6, including more variants, some of the theory behind them, and ways to improve them. In this section we give just the basic idea of the method needed to implement it. That basic idea is to solve the Navier-Stokes equations,

\[
\frac{\rho \mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = -\nabla^h p^{n+\frac{1}{2}} - \rho [(\mathbf{u} \cdot \nabla^h) \mathbf{u}]^{n+\frac{1}{2}} + \frac{\mu}{2} \Delta^h (\mathbf{u}^{n+1} + \mathbf{u}^n) + \mathbf{f}^n \\
\nabla^h \cdot \mathbf{u}^{n+1} = 0, \tag{2.42}
\]

by breaking the process into two steps: first, solve the momentum equation with some approximation to the pressure gradient while ignoring the incompressibility constraint, and second, correct the computed velocity to make it divergence-free and update the pressure accordingly.

These two steps can then be embedded into an iterative method, which takes the form

\[
\frac{\rho \mathbf{u}^{*,k} - \mathbf{u}^n}{\Delta t} + \nabla^h p^{n+\frac{1}{2},k} = -\rho [(\mathbf{u} \cdot \nabla^h) \mathbf{u}]^{n+\frac{1}{2}} + \frac{\mu}{2} \Delta^h (\mathbf{u}^{*,k} + \mathbf{u}^n) + \mathbf{f}^{n+\frac{1}{2}} \tag{2.44}
\]

\[
\frac{\rho \mathbf{u}^{n+1,k} - \mathbf{u}^n}{\Delta t} + \nabla^h p^{n+\frac{1}{2},k+1} = \rho \frac{\mathbf{u}^{*,k} - \mathbf{u}^n}{\Delta t} + \nabla^h p^{n+\frac{1}{2},k} \\
\nabla^h \cdot \mathbf{u}^{n+1,k} = 0. \tag{2.45}
\]

If the advection terms and forcing terms are computed explicitly and a good (first order in time) approximation to the pressure is available (such as the pressure from the previous timestep), then a single step of such an iterative method is sufficient. Equation (2.44) can be viewed as an update to the velocity field that ignores the incompressibility constraint, while equations (2.45) and (2.46) together form the projection step that corrects the updated velocity to be divergence-free (or approximately so in the case of approximate projections) while simultaneously giving the corrected pressure.
In equation (2.44), all quantities are treated as known except \( u, k \), which implies that numerically solving it is much like solving a forced diffusion equation. It is well conditioned and can be solved with a variety of techniques. In equation (2.45), there are actually two unknowns to solve for, \( u^{n+1,k} \) and \( \nabla^h p^{n+\frac{1}{2},k+1} \). This itself is broken into two steps. First, a Poisson equation is obtained by applying \( \nabla^h \) to both sides of (2.45) and using (2.46) to simplify. This gives us

\[
\nabla^h \cdot \nabla^h p^{n+\frac{1}{2},k+1} = \frac{\rho}{\Delta t} \nabla^h \cdot u^*,k + \nabla^h \cdot \nabla^h p^{n+\frac{1}{2},k}.
\]  (2.47)

There are a variety of ways that this Poisson equation can be solved, two of which will be discussed in Sections 2.3.4 and 2.3.5. After we solve this Poisson equation for \( p^{n+\frac{1}{2},k+1} \), we can rearrange equation (2.45) to obtain our equation for \( u^{n+1,k} \):

\[
u^{n+1,k} = u^*,k + \frac{\Delta t}{\rho} (\nabla^h p^{n+\frac{1}{2},k} - \nabla^h p^{n+\frac{1}{2},k+1}).
\]  (2.48)

In these equations, \( \nabla^h \), \( \nabla^h \cdot \), and \( \Delta^h \) are defined by the following

\[
(\nabla^h \cdot u)_{ij} = \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y}
\]  (2.49)

\[
(\nabla^h p)_{ij} = \left( \frac{p_{i+1,j} - p_{i-1,j}}{2\Delta x}, \frac{p_{i,j+1} - p_{i,j-1}}{2\Delta y} \right)
\]  (2.50)

\[
(\Delta^h_{\text{wide}} p)_{ij} = \frac{p_{i+2,j} - 2p_{i,j} + p_{i-2,j}}{4\Delta x^2} + \frac{p_{i,j+2} - 2p_{i,j} + p_{i,j-2}}{4\Delta y^2}
\]  (2.51)

\[
(\Delta^h_{\text{tight}} p)_{ij} = \frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{\Delta x^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{\Delta y^2}
\]  (2.52)

Both discrete Laplacian operators are used, with the tight stencil used for the viscous terms and either of the operators potentially being used for the intermediate Poisson equation solves. This is discussed in more detail in the next section.

### 2.3.3 Approximate Projection

In equation (2.47), it is important to note that \( \nabla^h \cdot \nabla^h = \Delta^h_{\text{wide}} \). This wide stencil is problematic because it decouples the Poisson problem into four distinct subgrids. This decoupling introduces a null space containing an oscillatory mode consisting of a different constant on each subgrid, and a special filtering step is needed to remove it. Also, if multigrid is used, then the multigrid operators must
take into account this decoupling. Because of this problem of the wide laplacian stencil, some implementations replace $\nabla^h \cdot \nabla^h$ with $\Delta^h_{\text{tight}}$. By doing so, we obtain a velocity from the algorithm that will not be discretely divergence-free. Instead, it satisfies only the incompressibility constraint (2.46) up to $O(h^2)$. Such a method is referred to as an approximate projection.

It is possible to define different divergence and gradient operators whose composition is $\Delta^h_{\text{tight}}$ so that we can both employ an exact projection and avoid the grid decoupling problem. This can be achieved by employing a MAC staggered grid where scalars are located at cell centers and velocities are located at cell edges as shown in Figure 2.5. We can then define second order MAC divergence and gradient operators through straightforward centered differences

$$\nabla^h_e(u^e)_{i,j} = \frac{u^e_{i,j} - u^e_{i-1,j}}{\Delta x} + \frac{v^e_{i,j} - v^e_{i,j-1}}{\Delta y}$$

$$\nabla^h(p)_{i,j} = \left( \nabla^h_1(p)_{i,j}, \nabla^h_2(p)_{i,j} \right) = \left( \frac{p_{i+1,j} - p_{i,j}}{\Delta x}, \frac{p_{i,j+1} - p_{i,j}}{\Delta y} \right).$$

The reason that these really are centered differences instead of one-sided differences is that the MAC divergence operates on cell-edge quantities and outputs cell-centered quantities, while the MAC gradient operator works on cell-centered quantities and outputs cell-edge quantities. We have added a superscript $e$ to the velocities as a reminder of this.

With a MAC grid, $\nabla^h \cdot \nabla^h = \Delta^h_{\text{tight}}$, so we avoid the grid decoupling problem.

![Figure 2.5. Locations of variables on a MAC grid](image)
that exists when using $\nabla_h^c$ and $\nabla_h^e$. However, others have claimed that adaptive mesh refinement is more straightforward with a cell-centered discretization and cell-centered velocities are needed for higher resolution advection methods such as those introduced by Bell et al. So we will show how the approximate projection method can be derived from a MAC projection; this connects some of the theory of the two methods and facilitates the extension of the approximate projection to refined grids. Our approach is a simplification for a special case of what Minion [26] did.

The basic idea used in constructing an approximate projection, $\tilde{P}$, from a MAC projection, $\hat{P}$, is to interpolate cell-centered quantities to cell edges when they would be required in the MAC projection, and to interpolate quantities back from cell edges to cell centers when needed. Denoting cell-centered quantities with a superscript $c$ and cell-edge quantities with a superscript $e$, the interpolation of the velocity to cell edges is given (in 2D) by the formula

$$
\mathbf{u}^e_{i,j} = \left( \frac{u^c_{i+1,j} + u^c_{i,j}}{2}, \frac{v^c_{i,j+1} + v^c_{i,j}}{2} \right).
$$

(2.55)

This is used in carrying out the first step of the projection, the application of the divergence operator. Since the MAC divergence operator is given by (2.53), the resulting divergence operator is given by

$$
D(\mathbf{u}^e)_{i,j} = \frac{u^e_{i,j} - u^e_{i-1,j}}{\Delta x} + \frac{v^e_{i,j} - v^e_{i,j-1}}{\Delta y} + \frac{(u^e_{i,j} + u^e_{i+1,j})}{2\Delta x} - \frac{(u^e_{i,j} + u^e_{i,j-1})}{2\Delta x} + \frac{(v^e_{i,j} + v^e_{i+1,j})}{2\Delta y} - \frac{(v^e_{i,j} + v^e_{i,j-1})}{2\Delta y}
$$

(2.56)

which is exactly $\nabla_h^e$, defined in (2.49). We can then solve the resulting Poisson problem, which uses $\Delta_{tight}^h$ for the Laplacian operator.

Once we have solved the Poisson problem, we have a scalar, $p$, to which we can apply the MAC gradient operator (2.54) to get a cell-edge gradient. This cell-edge gradient then needs to be subtracted from the cell-centered velocities to complete
the projection. In order to do so, we first interpolate back to cell centers using the interpolant

\[
G^c(\phi)_{i,j} = \left( \frac{\nabla^h_1(\phi)_{i,j} + \nabla^h_1(\phi)_{i-1,j}}{2}, \frac{\nabla^h_1(\phi)_{i,j} + \nabla^h_1(\phi)_{i,j-1}}{2} \right). \tag{2.57}
\]

Thus our effective gradient operator becomes

\[
G^c(p)_{i,j} = \left( \frac{\nabla^h_1(p)_{i,j} + \nabla^h_1(p)_{i-1,j}}{2}, \frac{\nabla^h_1(p)_{i,j} + \nabla^h_1(p)_{i,j-1}}{2} \right)
= \frac{1}{2} \left( \frac{p_{i+1,j} - p_{i,j}}{\Delta x} + \frac{p_{i,j} - p_{i-1,j}}{\Delta x}, \frac{p_{i,j+1} - p_{i,j}}{\Delta y} + \frac{p_{i,j} - p_{i,j-1}}{\Delta y} \right) \tag{2.58}
= \left( \frac{p_{i+1,j} - p_{i-1,j}}{2\Delta x}, \frac{p_{i,j+1} - p_{i,j-1}}{2\Delta y} \right).
\]

This is exactly \( \nabla^h \) defined in (2.50).

### 2.3.4 Poisson Solver: Multigrid

Multigrid methods have emerged as a fast way to solve large systems of equations obtained from the discretization of partial differential equations. The key to multigrid methods is that for many problems an iterative method (known as a “smoother”) can be found that quickly removes or dampens high frequency components of the error, allowing the problem to be transferred to a coarser grid where it is computationally cheaper to solve the problem. After the coarser grid problem is solved, the solution can be interpolated back to the finer grid, where a few more iterations of the smoother can quickly fine tune the solution. Multigrid employs a recursive algorithm, transferring between a hierarchy of grids. To understand the details of multigrid, see [5] for a good tutorial, or [40] for a more thorough treatment. We simply list the components of a multigrid solver we use for solving Poisson and Helmholtz equations.

The “transfer” operations, \( I^{2h}_h \) and \( I^h_{2h} \), we use are simple averaging of nearest neighbors (recall that we are using a cell-centered grid) and bilinear interpolation, respectively. The smoothing operator we use is Red-Black Gauss-Seidel (RBGS). RBGS differs from regular Gauss-Seidel only in the ordering of the unknowns. In RBGS, half the cells are assigned a red color and half the cells are assigned a black
color, with red cells adjacent only to black cells and with black cells adjacent only to red cells (i.e., a checkerboard pattern). The red cells are all numbered before the black cells. The net effect of RBGS is that all red cells are updated first (which can be done in parallel on multiprocessor machines), after which all black cells can be updated (which again can be done in parallel).

2.3.5 Poisson Solver: Fast Fourier Transform

In this section, we show how to use the discrete Fourier transform to solve

$$\Delta^h_{\text{wide}} p = f.$$  \hfill (2.59)

If we assume periodic boundary conditions, then the discrete Fourier transform is given by

$$f(x, t) = \sum_{jk} f_{jk}(t) e^{2\pi i j x / L_x} e^{2\pi i k y / L_y}$$  \hfill (2.60)

where $L_x$, $L_y$ are the dimensions of the domain. By replacing both $p$ and $f$ with their discrete Fourier transform, equation (2.59) becomes

$$\sum_{jk} f_{jk} e^{2\pi i j x / L_x} e^{2\pi i k y / L_y} = \Delta^h_{\text{wide}} \sum_{jk} p_{jk} e^{2\pi i j x / L_x} e^{2\pi i k y / L_y}$$  \hfill (2.61)

$$= \Delta^h_{\text{wide}} \sum_{jk} p_{jk} e^{2\pi i j x / L_x} e^{2\pi i k y / L_y}$$  \hfill (2.62)

$$= \sum_{jk} p_{jk} \left( e^{2\pi i j (x-2\Delta x) / L_x} - 2 e^{2\pi i j x / L_x} + e^{2\pi i j (x+2\Delta x) / L_x} \right) e^{2\pi i k y / L_y}$$

$$+ e^{2\pi i j x / L_x} e^{2\pi i k (y-2\Delta y) / L_y} - 2 e^{2\pi i k y / L_y} + e^{2\pi i k (y+2\Delta y) / L_y} \right) \left( \frac{4\Delta y^2}{(2\Delta y)^2} \right)$$  \hfill (2.63)

which simplifies to

$$= \sum_{jk} p_{jk} e^{2\pi i j x / L_x} e^{2\pi i k y / L_y} \left( -\frac{4 \sin^2(2\pi j \Delta x / L_x)}{4 \Delta x^2} + -\frac{4 \sin^2(2\pi k \Delta y / L_y)}{4 \Delta y^2} \right).$$  \hfill (2.64)

This implies that

$$p_{jk} = f_{jk} / \left( -\frac{4 \sin^2(2\pi j \Delta x / L_x)}{4 \Delta x^2} + -\frac{4 \sin^2(2\pi k \Delta y / L_y)}{4 \Delta y^2} \right).$$  \hfill (2.65)

The denominator of (2.65) is 0 for four combinations of $j$ and $k$, corresponding to the null space of $\Delta^h_{\text{wide}}$. We expect $f_{jk}$ to be 0 for these combinations since
$f$ is defined through taking the discrete divergence of something, and the discrete divergence operator shares the same null space. Computationally, however, it means we need to adjust the denominator for these four cases and make the denominator be 1.

### 2.4 Pseudocode

Putting all of the above together and writing out our approximate projection method for solving the Navier-Stokes equations in algorithmic form yields Algorithm 1. It is important to note that we have not included the additional pressure correction from [6], namely of subtracting $\frac{\nu}{2} \nabla^h \cdot u^{*,k}$ from the final pressure. For that correction to be beneficial, the projection would have to be modified slightly, as discussed in Chapter 6. The linear system solves in this algorithm (two Poisson equations and two Helmholtz equations in 2D or three Helmholtz equations in 3D) that appear in the Navier-Stokes algorithm are solved via either multigrid or FFTs. More details on the multigrid method are in Section 2.3.4, while more details of FFTs are in Section 2.3.5.

The full explicit IB algorithm can be seen in Algorithm 2, which depends upon a Navier-Stokes solver such as Algorithm 1.
Algorithm 1 Navier-Stokes Solver Algorithm

\[(u^{n+1}, p^{n+\frac{1}{2}}) = \text{Navier-Stokes Solve}(u^n, f^n, f^{n+\frac{1}{2}}, \Delta t)\]

Solve \(\Delta^h p^n = \nabla^h \cdot (-\rho[(u \cdot \nabla)u]^n + \mu \Delta^h u^n + f^n)\)

\[u^{n+\frac{1}{2}} = u^n + \frac{\Delta t}{2} \left( -[(u \cdot \nabla)u]^n + \nu \Delta^h u^n + \frac{f^n}{\rho} - \frac{\nabla^h p^n}{\rho} \right)\]

\[\nabla^h p^{n+\frac{1}{2},0} = \nabla^h p^n\]

For \(k=0, \ldots\)

Solve \[\frac{1}{\Delta t} u^{*,k} - \frac{\nu}{2} \Delta^h u^{*,k} = \frac{1}{\Delta t} u^n + \frac{\nu}{2} \Delta^h u^n - [(u \cdot \nabla)u]^{n+\frac{1}{2}} - \frac{\nabla^h p^{n+\frac{1}{2},k}}{\rho} + \frac{f^{n+\frac{1}{2}}}{\rho}\]

Solve \(\Delta^h p^{n+\frac{1}{2},k+1} = \frac{\rho}{\Delta t} \nabla^h \cdot u^{*,k} + \Delta^h p^{n+\frac{1}{2},k}\)

End of for loop

\[p^{n+\frac{1}{2}} = p^{n+\frac{1}{2},k+1}\]

\[u^{n+1} = u^{*,k} + \frac{\Delta t}{\rho} (\nabla^h p^{n+\frac{1}{2},k} - \nabla^h p^{n+\frac{1}{2},k+1})\]

Algorithm 2 Explicit Immersed Boundary Solver Algorithm

\[(u^{n+1}, X^{n+1}) = \text{Explicit IB Timestep}(u^n, X^n, \Delta t)\]

\[F^n_k = \sum_i \frac{T_0}{\Delta s_{ik}} (||X^n_i - X^n_k|| - \ell_0) \frac{1}{\Delta s_k} \frac{X^n_i - X^n_k}{||X^n_i - X^n_k||}\]

\[f^n_{ij} = \sum_k F^n_k \delta_h(x_{ij} - X^n_k) \Delta s_k\]

Solve \((u^{n+1}, p^{n+\frac{1}{2}}) = \text{Navier-Stokes Solve} (u^n, f^n, f^n, \Delta t)\) using Algorithm 1

\[\frac{X^{n+1}_k - X^n_k}{\Delta t} = \sum_{ij} u^{n+\frac{1}{2}}_{ij} \delta_h(x_{ij} - X^n_k) \Delta x \Delta y\]
CHAPTER 3

STABILITY

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

3.1 Temporal Discretization

In order to simplify and make analysis easier, we first switch to a spatially continuous and temporally discrete system. We do this because the choice of temporal discretization appears to be much more important to stability than the spatial discretization—we will show stability for a wide class of spatial discretizations but only for two types of temporal discretizations. The spatially continuous case provides us an easy way to show the idea behind verifying stability without getting
bogged down with too many discretization details. We analyze unsteady Stokes flow because, as mentioned in Chapter 1, the advection terms of the Navier-Stokes equations are not the cause of instability in these methods. We will, however, include the advection terms in some computational tests in Section 3.3.

There are various temporal discretizations possible. Most IB and IIM implementations tend to be of the following form

\[
\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} + \nabla p^{n+\frac{1}{2}} = \frac{\mu}{2} \Delta(\mathbf{u}^{n+1} + \mathbf{u}^n) + \mathbf{f} \quad (3.1)
\]

\[
\nabla \cdot \mathbf{u}^{n+1} = 0 \quad (3.2)
\]

\[
\frac{\mathbf{X}^{n+1} - \mathbf{X}^n}{\Delta t} = \mathbf{U}, \quad (3.3)
\]

where different discretization possibilities for \( \mathbf{f} \) and \( \mathbf{U} \) will be discussed below. Let us define \( S_m \) and \( S_m^* \) through the formulas

\[
S_m(\mathbf{F}) = \int_{\Gamma} \mathbf{F}(s) \delta(\mathbf{x} - \mathbf{X}^m(s)) \, ds, \quad (3.4)
\]

and

\[
S_m^*(\mathbf{u}) = \int_{\Omega} \mathbf{u}(\mathbf{x}) \delta(\mathbf{x} - \mathbf{X}^m(s)) \, d\mathbf{x}. \quad (3.5)
\]

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

1) \( S_{n+1} \mathcal{A}_f \mathbf{X}^{n+1} \)

2) \( \frac{1}{2} S_n \mathcal{A}_f \mathbf{X}^n + \frac{1}{2} S_{n+1} \mathcal{A}_f \mathbf{X}^{n+1} \)

3) \( S_{n+\frac{1}{2}} \mathcal{A}_f \mathbf{X}^{n+\frac{1}{2}} \),

where \( \mathbf{X}^{n+\frac{1}{2}} \) is approximated by \( \frac{1}{2}(\mathbf{X}^n + \mathbf{X}^{n+1}) \), while common discretization choices for \( \mathbf{U} \) are
Some implicit methods also lag the spreading and interpolation operators in time, meaning that $S_n$ and $S_n^*$ are used to replace occurrences of $S$ and $S^*$ evaluated at other times in the above choices for $f$ and $U$.

Where the discretization choices for $f$ and $U$ can be found in the literature, many variations are used. Mayo and Peskin [23] use method 1A (i.e., take choice 1 for $f$ and A for $U$ from the lists above). Tu and Peskin [41] use a modified version of 1A with time-lagged spreading and interpolation operators (i.e., $S_n$ and $S_n^*$ instead of $S_{n+1}$ and $S_{n+1}^*$) with steady Stokes flow. Roma et al. [37] and Lee and Leveque [20] both use method 2B, whereas Griffith and Peskin [14] approximate 2B using an explicitly calculated $X^{n+1}$. Mori and Peskin [27] use method 3C with lagged spreading and interpolation operators, though with the addition of boundary mass; Peskin [32] and Lai and Peskin [18] approximate method 3C with an explicitly calculated $X^{n+1}$; and we experimented with method 3B. We prove below that both 1A and 3C are unconditionally stable and that the stability is not affected by the location at which $S$ and $S^*$ are evaluated, provided both are evaluated at the same location.

3.2 Unconditionally Stable Schemes

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

3.2.1 Crank-Nicolson for the Spatially Continuous Problem

We begin with method 3C for the spatially continuous problem, showing that the energy of the system
is nonincreasing. Here the inner products are on $L^2(\Omega)$ and $L^2(\Gamma)$, respectively. This energy represents the sum of the kinetic energy of the fluid and the potential energy in the elasticity of the boundary. Note that $A_f$ (the force generation operator on the immersed boundary) must be negative-definite for this definition of energy to make sense. We further require in the proof that $A_f$ be self-adjoint and linear (conditions which are satisfied by the common choice $A_f = \frac{\partial^2}{\partial s^2}$). We also assume that divergence-free velocity fields are orthogonal to gradient fields (which may put limitations on the boundary conditions but is satisfied, for example, by periodic boundary conditions). We also make use of the fact that $S$ and $S^*$ are adjoints, which can be seen from the following calculation with $\mathbf{F} \in L^2(\Gamma)$ and $\mathbf{w} \in L^2(\Omega)$

\[
(\mathcal{S}(\mathbf{F}(s,t)), \mathbf{w}(\mathbf{x}, t)) = \int_{\Omega} \mathcal{S}(\mathbf{F}(s,t))(\mathbf{x}, t) \cdot \mathbf{w}(\mathbf{x}, t) \, d\mathbf{x} \\
= \int_{\Omega} \int_{\Gamma} \mathbf{F}(s,t)\delta(\mathbf{x} - \mathbf{X}(s,t)) \, d\mathbf{s} \cdot \mathbf{w}(\mathbf{x}, t) \, d\mathbf{x} \\
= \int_{\Gamma} \mathbf{F}(s,t) \cdot \int_{\Omega} \mathbf{w}(\mathbf{x}, t)\delta(\mathbf{x} - \mathbf{X}(s,t)) \, d\mathbf{x} \, d\mathbf{s} \\
= (\mathbf{F}(s,t), \mathcal{S}^*(\mathbf{w}(\mathbf{x}, t)))_{\Gamma}.
\] (3.8)

Recall that the IB method using 3C is

\[
\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} + \nabla p^{n+\frac{1}{2}} = \frac{\mu}{2} \Delta (\mathbf{u}^{n+1} + \mathbf{u}^n) + \mathcal{S}_{n+\frac{1}{2}} A_f \left( \frac{1}{2}(\mathbf{X}^n + \mathbf{X}^{n+1}) \right) \quad (3.9)
\]

\[
\nabla \cdot \mathbf{u}^{n+1} = 0 \quad (3.10)
\]

\[
\frac{\mathbf{X}^{n+1} - \mathbf{X}^n}{\Delta t} = \mathcal{S}_{n+\frac{1}{2}}^* \left( \frac{1}{2}(\mathbf{u}^n + \mathbf{u}^{n+1}) \right). \quad (3.11)
\]

Multiplying through by $\Delta t$, taking the inner product of equation (3.9) with $\mathbf{u}^{n+1} + \mathbf{u}^n$ and the inner product of (3.11) with $-A_f(\mathbf{X}^{n+1} + \mathbf{X}^n)$ yields

\[
E[\mathbf{u}, \mathbf{X}] = \rho \langle \mathbf{u}, \mathbf{u} \rangle_{\Omega} + \langle -A_f \mathbf{X}, \mathbf{X} \rangle_{\Gamma} \\
= \rho \|\mathbf{u}\|_{\Omega}^2 + \|\mathbf{X}\|_{\Gamma}^2
\] (3.6)
\[
\rho \langle \mathbf{u}^{n+1} + \mathbf{u}^n, \mathbf{u}^{n+1} - \mathbf{u}^n \rangle = -\Delta t \left( \mathbf{u}^{n+1} + \mathbf{u}^n, \nabla p^{n+\frac{1}{2}} \right)_\Omega \\
+ \frac{\mu \Delta t}{2} \left( \mathbf{u}^{n+1} + \mathbf{u}^n, \Delta (\mathbf{u}^{n+1} + \mathbf{u}^n) \right)_\Omega \\
+ \frac{\Delta t}{2} \left( \mathbf{u}^{n+1} + \mathbf{u}^n, S_{n+\frac{1}{2}} A_f \mathbf{X}^n + \mathbf{X}^{n+1} \right)_\Omega \\
\langle -A_f(\mathbf{X}^{n+1} + \mathbf{X}^n), \mathbf{X}^{n+1} - \mathbf{X}^n \rangle = \frac{\Delta t}{2} \left( \langle -A_f(\mathbf{X}^{n+1} + \mathbf{X}^n), S^*_{n+\frac{1}{2}} (\mathbf{u}^{n+1} + \mathbf{u}^n) \rangle \right)_\Gamma.
\]

(3.12)

Adding these two equations gives us

\[
\rho \langle \mathbf{u}^{n+1} + \mathbf{u}^n, \mathbf{u}^{n+1} - \mathbf{u}^n \rangle + \langle -A_f(\mathbf{X}^{n+1} + \mathbf{X}^n), \mathbf{X}^{n+1} - \mathbf{X}^n \rangle \\
= -\Delta t \left( \mathbf{u}^{n+1} + \mathbf{u}^n, \nabla p^{n+\frac{1}{2}} \right)_\Omega \\
+ \frac{\mu \Delta t}{2} \left( \mathbf{u}^{n+1} + \mathbf{u}^n, \Delta (\mathbf{u}^{n+1} + \mathbf{u}^n) \right)_\Omega \\
+ \frac{\Delta t}{2} \left( \mathbf{u}^{n+1} + \mathbf{u}^n, S_{n+\frac{1}{2}} A_f \mathbf{X}^n + \mathbf{X}^{n+1} \right)_\Omega \\
+ \frac{\Delta t}{2} \left( \langle -A_f(\mathbf{X}^n + \mathbf{X}^{n+1}), S^*_{n+\frac{1}{2}} (\mathbf{u}^{n+1} + \mathbf{u}^n) \rangle \right)_\Gamma.
\]

(3.13)

The last two terms on the right-hand side cancel by the adjointness of \( S_{n+\frac{1}{2}} \) and \( S^*_{n+\frac{1}{2}} \), the first term on the right-hand side is zero because \( \mathbf{u}^{n+1} \) and \( \mathbf{u}^n \) are divergence free while \( \nabla p^{n+\frac{1}{2}} \) is a gradient field, and the left-hand side of the equation can be simplified using the linearity and self-adjointness of \( A_f \). This leaves us with

\[
\rho \langle \mathbf{u}^{n+1}, \mathbf{u}^{n+1} \rangle - \rho \langle \mathbf{u}^n, \mathbf{u}^n \rangle + \langle -A_f \mathbf{X}^{n+1}, \mathbf{X}^{n+1} \rangle - \langle -A_f \mathbf{X}^n, \mathbf{X}^n \rangle \\
= \frac{\mu \Delta t}{2} \left( \mathbf{u}^{n+1} + \mathbf{u}^n, \Delta (\mathbf{u}^{n+1} + \mathbf{u}^n) \right)_\Omega.
\]

(3.15)

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

\[
E[\mathbf{u}^{n+1}, \mathbf{X}^{n+1}] - E[\mathbf{u}^n, \mathbf{X}^n] \leq 0.
\]

(3.16)

In other words, the energy of the system is bounded for all time, and thus the system remains stable. The key to the proof is that the energy terms representing the work done by the fluid on the boundary and the work done by the boundary on the fluid (the terms involving \( S_{n+\frac{1}{2}} \) and \( S^*_{n+\frac{1}{2}} \) exactly cancel. This is a property
not shared by any of the other methods, though in Section 3.2.3 we show that the remaining terms for 1A result in a decrease of energy in addition to the energy loss from viscosity, and so scheme 1A is also stable.

It is useful to note that the proof is still valid even if $S_n$ and $S^*_n$ are used in place of $S_{n+\frac{1}{2}}$ and $S^*_{n+\frac{1}{2}}$; i.e., the system need not be fully implicit in order to achieve unconditional stability. This is contrary to what the community expected (as pointed out in Chapter 1), and it can be exploited to simplify the set of implicit equations that need to be solved, as we do in Section 3.3, without sacrificing stability.

### 3.2.2 Projection Methods and Discrete Delta Functions

We now show that the unconditional stability of method 3C extends to the usage of discrete delta functions and (exact) discrete projection methods. We show computationally in Section 3.3 that an approximate projection method appears to be sufficient for unconditional stability, but use an exact projection for purposes of the proof. Let $\nabla^h$, $\nabla^h_\cdot$, and $\Delta^h$ be discrete analogs of $\nabla$, $\nabla_\cdot$, and $\Delta$ satisfying $\Delta^h = \nabla^h \cdot \nabla^h$ (so that the projection is exact). Let $A_f$ be a discrete analog of $A_f$ maintaining linearity, negative-definiteness, and self-adjointness. Let $S_{n+\frac{1}{2}}$ and $S^*_{n+\frac{1}{2}}$ be discrete analogs of $S_{n+\frac{1}{2}}$ and $S^*_{n+\frac{1}{2}}$ preserving adjointness. We assume boundary conditions for which discretely divergence-free fields and discrete gradient fields are orthogonal and for which $\nabla^h$ and $\Delta^h$ commute. We employ a pressure-free projection method in the proof. Some other projection methods, such as pressure update projection methods, also work, but others might not; in particular, it appears that it was the projection employed by Mayo and Peskin [23] that caused the instability they saw, as we discuss in Section 3.2.3.

We note that many, if not most, existing IB implementations satisfy these conditions. For example, an implementation with periodic boundary conditions, employing a (exact) projection method, using the force generation operator $A_f = \gamma D_+D_-$, and using any of the standard discrete delta functions [32] will satisfy all of these conditions. All of these particular choices are quite common. Thus,
as we prove below, such implementations could be made unconditionally stable by switching to discretization method 3C.

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

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

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

$$\langle v, w \rangle_{\Omega_h} = \sum_{ij} v_{ij} \cdot w_{ij} \Delta x \Delta y \quad (3.18)$$

and

$$\langle Y, Z \rangle_{\Gamma_h} = \sum_k Y_k \cdot Z_k \Delta s. \quad (3.19)$$

The calculation to show this adjointness property for the standard tensor product discrete delta functions [32] is very similar to (3.8)

$$\langle S(F), w \rangle_{\Omega_h} = \sum_{ij} S(F)_{ij} \cdot w_{ij} \Delta x \Delta y$$

$$= \sum_{ij} \sum_k F_k \delta_h(x_{ij} - X_k) \Delta s \cdot w_{ij} \Delta x \Delta y$$

$$= \sum_k F_k \cdot \sum_{ij} w_{ij} \delta_h(x_{ij} - X_k) \Delta x \Delta y \, ds$$

$$= \langle F, S^*(w) \rangle_{\Gamma_h}. \quad (3.20)$$

With these definitions and assumptions, using the pressure update needed for second order accuracy [6], and utilizing method 3C, our discrete system becomes

$$\rho \frac{u^* - u^n}{\Delta t} = \frac{\mu}{2} \Delta^h (u^* + u^n) + \frac{1}{2} S_{n+\frac{1}{2}} A_f(X^n + X^{n+1}) \quad (3.21)$$

$$\Delta^h \phi = \frac{\rho}{\Delta t} \nabla^h \cdot u^* \quad (3.22)$$

$$u^{n+1} = u^* - \frac{\Delta t}{\rho} \nabla^h \phi \quad (3.23)$$

$$p^{n+\frac{1}{2}} = \phi - \frac{\nu}{2} \nabla^h \cdot u^* \quad (3.24)$$

$$\frac{X^{n+1} - X^n}{\Delta t} = \frac{1}{2} S_{n+\frac{1}{2}}^{*}(u^n + u^{n+1}). \quad (3.25)$$

Note that these equations do not specify the method for obtaining $X^{n+1}$, which is needed in the momentum solve (3.21) as the first step of the projection method.
Solving for it is nontrivial since both (3.21) and (3.25) depend on both $u^{n+1}$ and $X^{n+1}$. In order to avoid limiting the proof to a particular scheme, we simply assume that the value of $X^{n+1}$ that solves these equations is somehow obtained. Several examples for doing so, such as iterating (3.21)-(3.25) on approximations to $X^{n+1}$ (using fixed point, Newton, or Quasi-Newton methods), are explored in Chapter 5.

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

$$E[u, X] = \rho \langle u, u \rangle_{\Omega_h} + \langle -A_f X, X \rangle_{\Gamma_h}$$

is nonincreasing. Taking the inner product of (3.21) with $\Delta t (u^{n+1} + u^n)$, taking the inner product of (3.23) with $\rho(u^{n+1} + u^n)$ and taking the inner product of (3.25) with $\Delta t A_f (X^{n+1} + X^n)$ yields

$$\rho \langle u^{n+1} + u^n, u^* - u^n \rangle_{\Omega_h} = \frac{\mu \Delta t}{2} \langle u^{n+1} + u^n, \Delta^h (u^* + u^n) \rangle_{\Omega_h}$$

$$+ \frac{\Delta t}{2} \left\langle u^{n+1} + u^n, S_{n+\frac{1}{2}} A_f (X^n + X^{n+1}) \right\rangle_{\Omega_h}$$

$$\rho \langle u^{n+1} + u^n, u^{n+1} - u^n \rangle_{\Omega_h} = -\Delta t \left\langle u^{n+1} + u^n, \nabla^h \phi \right\rangle_{\Omega_h}$$

$$\left\langle -A_f (X^{n+1} + X^n), X^{n+1} - X^n \right\rangle_{\Gamma_h} = \frac{\Delta t}{2} \left\langle -A_f (X^{n+1} + X^n), S^*_{n+\frac{1}{2}} (u^n + u^{n+1}) \right\rangle_{\Gamma_h}.$$
\[
\rho \langle \mathbf{u}^{n+1}, \mathbf{u}^{n+1} \rangle_{\Omega_h} - \rho \langle \mathbf{u}^n, \mathbf{u}^n \rangle_{\Omega_h} + \langle -A_f \mathbf{X}^{n+1}, \mathbf{X}^{n+1} \rangle_{\Gamma_h} - \langle -A_f \mathbf{X}^n, \mathbf{X}^n \rangle_{\Gamma_h} = \Delta t \langle \mathbf{u}^{n+1} + \mathbf{u}^n, \nabla^h \phi \rangle_{\Omega_h} \\
+ \frac{\mu \Delta t}{2} \langle \mathbf{u}^{n+1} + \mathbf{u}^n, \Delta^h ((\mathbf{u}^{n+1} + \Delta t \nabla^h \phi) + \mathbf{u}^n) \rangle_{\Omega_h}.
\]

Making use of commutativity of \( \nabla^h \) and \( \Delta^h \) and orthogonality of discretely divergence-free vector fields and discrete gradient fields, we find

\[
\rho \langle \mathbf{u}^{n+1}, \mathbf{u}^{n+1} \rangle_{\Omega_h} - \rho \langle \mathbf{u}^n, \mathbf{u}^n \rangle_{\Omega_h} + \langle -A_f \mathbf{X}^{n+1}, \mathbf{X}^{n+1} \rangle_{\Gamma_h} - \langle -A_f \mathbf{X}^n, \mathbf{X}^n \rangle_{\Gamma_h} = \frac{\mu \Delta t}{2} \langle \mathbf{u}^{n+1} + \mathbf{u}^n, \Delta^h (\mathbf{u}^{n+1} + \mathbf{u}^n) \rangle_{\Omega_h}.
\]

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

\[
E[\mathbf{u}^{n+1}, \mathbf{X}^{n+1}] - E[\mathbf{u}^n, \mathbf{X}^n] \leq 0,
\]

showing that the energy of the system is nonincreasing and thus implying that the system is stable.

Just as with the spatially continuous case, nothing in this proof required the system to be fully implicit to achieve unconditional stability; the proof is still valid if \( S_n \) and \( S^*_n \) (or indeed spreading and interpolation operators evaluated at any location as long as they are adjoints) are used in place of \( S_{n+\frac{1}{2}} \) and \( S^*_{n+\frac{1}{2}} \). Again, this is a fact we demonstrate in our computational experiments in Section 3.3.

### 3.2.3 Unconditional Stability of Backward Euler

We now demonstrate that temporal discretization 1A is unconditionally stable, and in fact that it dissipates more energy from the system than one would find from the effect of viscosity alone. The method and assumptions are the same as in Section 3.2.2; the only difference is that more work is required to show that extra noncancelling terms are in fact negative.

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

\[
\begin{align*}
\rho \frac{u^n - u^*}{\Delta t} &= \mu \Delta h u^* + S_{n+1} A_f X^{n+1} \\
\frac{u^{n+1}}{\Delta t} &= u^* - \frac{\Delta t}{\rho} \nabla h \phi \tag{3.35} \\
\frac{X^{n+1} - X^n}{\Delta t} &= S_{n+1} u^{n+1}. \tag{3.36}
\end{align*}
\]

Taking the inner product of (3.34) with \(\Delta t (u^{n+1} + u^n)\), taking the inner product of (3.35) with \(\frac{\rho (u^{n+1} + u^n)}{\rho}\), and taking the inner product of (3.36) with \(-\Delta t A_f (X^{n+1} + X^n)\) yields

\[
\begin{align*}
\rho \left\langle u^{n+1} + u^n, u^* - u^n \right\rangle_{\Omega_h} &= \Delta t \left\langle u^{n+1} + u^n, \mu \Delta h u^* + S_{n+1} A_f X^{n+1} \right\rangle_{\Omega_h} \tag{3.37} \\
\rho \left\langle u^{n+1} + u^n, u^{n+1} - u^n \right\rangle_{\Omega_h} &= -\Delta t \left\langle u^{n+1} + u^n, \nabla h \phi \right\rangle_{\Omega_h} \\
\left\langle -A_f (X^{n+1} + X^n), X^{n+1} - X^n \right\rangle_{\Gamma_h} &= \Delta t \left\langle -A_f (X^{n+1} + X^n), S_{n+1} u^{n+1} \right\rangle_{\Gamma_h}. \tag{3.38}
\end{align*}
\]

We can simplify (3.39) by making use of the assumption that \(A_f\) is linear and self-adjoint and by using equation (3.36) to eliminate the appearance of \(X^n\) on the right-hand side. Making these simplifications and adding all three equations, we obtain

\[
\begin{align*}
\rho \left\langle u^{n+1}, u^{n+1} \right\rangle_{\Omega_h} - \rho \left\langle u^n, u^n \right\rangle_{\Omega_h} + \left\langle -A_f X^{n+1}, X^{n+1} \right\rangle_{\Gamma_h} - \left\langle -A_f X^n, X^n \right\rangle_{\Gamma_h} \tag{3.40} \\
= \Delta t \left\langle u^{n+1} + u^n, \mu \Delta h u^* + S_{n+1} A_f X^{n+1} - \nabla h \phi \right\rangle_{\Omega_h} \\
+ \Delta t \left\langle -2A_f X^{n+1} + A\Delta t S^n_{n+1} u^{n+1}, S_{n+1} u^{n+1} \right\rangle_{\Gamma_h}.
\end{align*}
\]

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

\[
\begin{align*}
E[u^{n+1}, X^{n+1}] - E[u^n, X^n] \tag{3.41} \\
= \Delta t \left\langle u^n - u^{n+1}, \mu \Delta h u^* + S_{n+1} A_f X^{n+1} - \nabla h \phi \right\rangle_{\Omega_h} \\
+ \Delta t \left\langle 2u^{n+1}, \mu \Delta h u^* + S_{n+1} A_f X^{n+1} - \nabla h \phi \right\rangle_{\Omega_h} \\
+ \Delta t \left\langle -2A_f X^{n+1} + A\Delta t S^n_{n+1} u^{n+1}, S_{n+1} u^{n+1} \right\rangle_{\Gamma_h}.
\end{align*}
\]
We can use equations (3.34) and (3.35) to replace the first occurrence of $\mu \Delta^h u^* + S_{n+1} A_f X^{n+1} - \nabla^h \phi$ with $\rho \frac{u^{n+1} - u^n}{\Delta t}$. Simultaneously using equation (3.35) to eliminate the second appearance of $u^*$ we obtain

$$E[u^{n+1}, X^{n+1}] - E[u^n, X^n] = \Delta t \left< u^n - u^{n+1}, \rho \frac{u^{n+1} - u^n}{\Delta t} \right>_{\Omega_h}$$

$$+ \Delta t \left< 2u^{n+1}, \mu \Delta^h u^{n+1} \right>_{\Omega_h}$$

$$+ \Delta t \left< 2u^{n+1}, \frac{\Delta t}{\rho} \Delta^h \nabla^h \phi - \nabla^h \phi \right>_{\Omega_h}$$

$$+ \Delta t \left< 2u^{n+1}, S_{n+1} A_f X^{n+1} \right>_{\Omega_h}$$

$$+ \Delta t \left< -2A_f X^{n+1}, S_{n+1}^* u^{n+1} \right>_{\Gamma_h}$$

$$- \Delta t^2 \left< -AS_{n+1}^* u^{n+1}, S_{n+1}^* u^{n+1} \right>_{\Gamma_h}.$$

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

$$E[u^{n+1}, X^{n+1}] - E[u^n, X^n] =$$

$$- \rho \left< u^{n+1} - u^n, u^{n+1} - u^n \right>_{\Omega_h}$$

$$+ 2 \mu \Delta t \left< u^{n+1}, \Delta^h u^{n+1} \right>_{\Omega_h}$$

$$- \Delta t^2 \left< -AS_{n+1}^* u^{n+1}, S_{n+1}^* u^{n+1} \right>_{\Gamma_h}.$$

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

We note that Mayo and Peskin [23] also used a backward Euler discretization of both the viscous and IB terms, but reported a lack of stability in their method that was also confirmed by Stockie and Wetton [38]. However, they did not solve the system of equations (3.34)-(3.36). The salient difference between method 1A and their method is that they moved the forcing terms from the momentum equation into the projection step to obtain a system of equations of the form

\[ \frac{u^* - u^n}{\Delta t} = \mu \Delta^h u^* \]  
\[ \Delta^h \phi = \frac{\rho}{\Delta t} \nabla^h \cdot u^* + \nabla^h \cdot (SA_f X^{n+1}) \]  
\[ u^{n+1} = u^* - \frac{\Delta t}{\rho} \nabla^h \phi + \frac{\Delta t}{\rho} (SA_f X^{n+1}) \]  
\[ \frac{X^{n+1} - X^n}{\Delta t} = S^* u^{n+1}. \]

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

### 3.3 Computational Results

In this section, we demonstrate the unconditional stability of our discretization computationally and indicate how stability is affected by some modifications not covered in the proofs of the preceding section. Since approximate projections are an increasingly commonly used method in the community that provides a velocity field that is not quite discretely divergence-free, it is the first modification that we test. We use an approximate projection that has an \( O(h^2) \) error in the divergence-free condition for the velocity. The other modification that we test is the addition of advection terms from the Navier-Stokes equations.
The test problem we use is one typically seen in the literature, in which the immersed boundary is a closed loop initially in the shape of an ellipse [20, 21, 23, 37, 38, 41]. We choose an ellipse initially aligned in the coordinate directions with horizontal semiaxis $a = 0.28125$ cm and vertical semiaxis $b = 0.75a$ cm. The fluid is initially at rest in a periodic domain, $\Omega$, with $\Omega = [0, 1] \times [0, 1]$. For this test problem, the boundary should perform damped oscillations around a circular equilibrium state with the same area as that of the original ellipse. The configuration of the boundary at different times can be seen in Figure 3.1.

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

![Figure 3.1](image-url)

**Figure 3.1.** Computed immersed boundary positions at successive times; the solid line is the initial shape, and the dashed-dotted and dashed lines show the configuration of the boundary later in the simulation.
\[
(\nabla^h \cdot \mathbf{u})_{ij} = \frac{u_{i+1,j} - u_{i-1,j}}{2h} + \frac{v_{i,j+1} - v_{i,j-1}}{2h}
\]
\[
(\nabla^h p)_{ij} = \left( \frac{p_{i+1,j} - p_{i-1,j}}{2h}, \frac{p_{i,j+1} - p_{i,j-1}}{2h} \right)
\]
\[
(\Delta^h_{\text{wide}} p)_{ij} = \frac{p_{i+2,j} - 2p_{i,j} + p_{i-2,j}}{4h^2} + \frac{p_{i,j+2} - 2p_{i,j} + p_{i,j-2}}{4h^2}
\]
\[
(\Delta^h_{\text{tight}} p)_{ij} = \frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{h^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{h^2}
\]

where $\Delta^h = \Delta^h_{\text{wide}}$ will be used for our Poisson solve in all tests other than those from Section 3.3.2, and $\Delta^h = \Delta^h_{\text{tight}}$ will be used for the viscous terms as well as the Poisson solve in Section 3.3.2. We define $A_f$ as $\gamma \frac{\partial^2}{\partial x^2}$ and choose $N_B$, the number of IB points, to approximately satisfy $N_B = \frac{2L_B}{h}$, where $L_B$ is the arclength of the immersed boundary. In all tests, except where otherwise noted, we use (spatially discrete) method 3C with lagged spreading and interpolation operators. We also use the common four point delta function

\[
\delta_h(x, y) = \delta_h(x) \delta_h(y)
\]
\[
\delta_h(x) = \begin{cases} 
\frac{1}{4h} (1 + \cos(\frac{x}{2h})) & |x| \leq 2h \\
0 & |x| \geq 2h.
\end{cases}
\]

The immersed boundary update equation,

\[
\frac{X^{n+1} - X^n}{\Delta t} = \frac{1}{2} S_n^*(u^n + u^{n+1}),
\]

can be re-written as

\[
g(X) = X - X^n - \frac{\Delta t}{2} S_n^*(u^n + u^{n+1}) = 0
\]

so that we can write the implicit system that we must solve as $g(X^{n+1}) = 0$ (note that $u^{n+1}$ depends on $X^{n+1}$, too). To solve this implicit system of equations, we use an approximate Newton solver (by employing finite difference approximations to obtain the Jacobian $g'$, which requires $O(N_B)$ fluid solves per implicit iteration to compute and is thus extremely slow). We discuss the use of the approximate Newton solver further in Section 3.4.
3.3.1 Computational Verification of the Method

3.3.1.1 Refinement Study

We begin with a simple convergence study to verify that the implicit discretization results in a consistent method. We expect only first order accuracy for two reasons: the IB method has been shown to exhibit first order accuracy on problems with a sharp interface, and our lagging of the spreading and interpolation operators makes the discretization formally first order. The IB method has three numerical parameters affecting the accuracy — the Eulerian mesh width, $h$; the average Lagrangian mesh width, $\Delta s$; and the timestep, $\Delta t$. The values of these numerical parameters that we use in our refinement study are given in Table 3.1. However, since we always choose $N_B$ so that $\Delta s$ is approximately $h/2$, we report the size of $N_B$ instead. For our convergence problem, we set $\gamma = 1$ (g·cm/$s^2$), $\rho = 1$ (g/cm$^2$), and $\mu = 0.01$ (g/cm·s).

The results of the refinement study are displayed in Table 3.2. Because an analytic solution is not available, we estimate the convergence rate by comparing the differences of the numerical solutions between successive grid levels. We use the $\ell^2$ norms induced by the inner products in equations (3.18) and (3.19).

3.3.1.2 Comparison to Explicit Method

For sufficiently small timesteps, the explicit and implicit methods produce nearly identical results, as expected. For larger timesteps, the energy measure defined by equation (3.26) gives us a useful way to compare the two methods. In Figure 3.2, we show the energy of the system as a function of time for four different simulations. All four use $\mu = 0.01$ and $\gamma = 1$, but differ on timestep size and whether the

<table>
<thead>
<tr>
<th>Level #</th>
<th>$h$</th>
<th>$N_B$</th>
<th>$\Delta t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/16</td>
<td>50</td>
<td>0.05</td>
</tr>
<tr>
<td>2</td>
<td>1/32</td>
<td>100</td>
<td>0.025</td>
</tr>
<tr>
<td>3</td>
<td>1/64</td>
<td>200</td>
<td>0.0125</td>
</tr>
<tr>
<td>4</td>
<td>1/128</td>
<td>400</td>
<td>0.000625</td>
</tr>
</tbody>
</table>

Table 3.1 Parameters used in numerical refinement study
Table 3.2 Results of a refinement study showing first order convergence of method 3C. The convergence rate is defined as $\text{sqrt}(E_1/E_3)$. $E_i(q)$ measures the error in variable $q$ at level $i$, defined by $\| q_i - \text{coarsen}(q_{i+1}) \|_2$. Here, $q_i$ is the value of variable $q$ at time $t = 0.2$ (seconds) computed with the numerical parameters for level $i$ from Table 3.1. The coarsen operator is simple averaging of nearest values for Eulerian quantities, and injection from identical gridpoints for Lagrangian quantities.

<table>
<thead>
<tr>
<th>$q$</th>
<th>$E_1(q)$</th>
<th>$E_2(q)$</th>
<th>$E_3(q)$</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u$</td>
<td>1.06e-03</td>
<td>4.17e-04</td>
<td>1.76e-04</td>
<td>2.4485</td>
</tr>
<tr>
<td>$X$</td>
<td>5.25e-04</td>
<td>2.96e-04</td>
<td>1.58e-04</td>
<td>1.8239</td>
</tr>
</tbody>
</table>

Figure 3.2. Energy of the system for four different simulations with $\mu = 0.01$. $o$: implicit method, $\Delta t = 6 \times 10^{-3}$; $\triangledown$: implicit method, $\Delta t = 6 \times 10^{-2}$; $x$: explicit method, $\Delta t = 6 \times 10^{-3}$; $\rightarrow$: explicit method, $\Delta t = 6 \times 10^{-2}$
explicit or implicit method is used. When $\Delta t = 6 \times 10^{-3}$, both the explicit and the implicit methods give nearly identical results until $t = 0.2$, at which point the explicit method becomes unstable. When $\Delta t = 6 \times 10^{-2}$, the implicit method gives results nearly identical to the implicit method with the smaller timestep, but the explicit method goes unstable immediately.

3.3.1.3 Parameter Testing on the Inviscid Problem

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

Note that inviscid simulations are a particularly good check for the method since, as can be seen from the energy proof of Section 3.2.2, the energy defined by equation (3.26) should remain constant. We ran all our simulations until time $t \geq 0.5$ s, and verified that for all 20 combinations of these parameters, the solution to the implicit system remained stable throughout the simulation and that the energy of the system was indeed constant. Figure 3.3 shows the energy of the system for the set of parameters $\gamma = 10^2, \Delta t = 10^{-2}$. Computing with large timesteps (e.g., $\Delta t = 10^{10}$) may not yield particularly accurate results (because errors of $O(\Delta t)$ obviously need not be small), but it does illustrate the stability of the method —
the boundary configuration was still elliptical at the end of the simulation, points
on the immersed boundary moved much less than the length of the computational
domain, and there was no change in energy during the simulation.

3.3.1.4 Comparison to Method 1A

The only reason for the stability proof of method 1A given in Section 3.2.3
was to assist the investigation of why the Mayo and Peskin scheme failed to be
unconditionally stable. However, implementing method 1A with lagged spreading
and interpolation operators requires only a minor change to the code and provides
an additional test of the analytical results. We present three simulations, all with
$\Delta t = 10^{-2}$ and $\gamma = 1$. The three simulations were method 3C with no viscosity,
method 1A with no viscosity, and method 3C with $\mu = 0.01$. The results are
plotted in Figure 3.4. Since method 1A is run with no viscosity, the solution to
the continuous equations will have constant energy, but we showed in Section 3.2.3
that method 1A will have energy dissipation other than from viscosity. From the

Figure 3.3. Energy of the system during the course of
an inviscid simulation with the implicit method, showing
perfect energy conservation.
Figure 3.4. Energy of the system with $\gamma = 1$. -: method 1A with no viscosity; --: method 3C with no viscosity, \cdot--\cdot: method 3C with $\mu = 0.01$.

In the figure, we see indeed that this is the case and that method 1A loses slightly more energy than method 3C loses with a viscosity of 0.01.

### 3.3.2 Unconditional Stability with an Approximate Projection

Calculations with a pressure-free approximate projection version of method 3C were run for the same values of $\Delta t$, $\gamma$ as in Section 3.3.1.3 and with $\mu = 0$ or 0.01. In all cases, the solution to the implicit system remained stable throughout the simulation, and the energy of the system, as defined by equation (3.26), did not increase. As with the exact-projection calculation, the boundary configuration was elliptical at the end of each simulation. The simulations with $\mu = 0$ are particularly interesting. For such simulations, the energy remains constant using an exact projection, so this allowed us to determine more clearly the effect of the approximate projection. In all cases, we found that the energy was nonincreasing, meaning that the approximate projection appears to have a neutral or stabilizing effect.
The approximate projection, \( \tilde{P} \), has the interesting property that when iterated it will converge to an exact projection, \( P \); i.e., \( \tilde{P}^m \to P \) as \( m \to \infty \) \cite{2}. This means that performing additional (approximate) projections per fluid solve must eventually reduce the additional energy dissipation due to using approximate projections. Figure 3.5 shows this effect. This simulation was run with \( \gamma = 1, \mu = 0, \Delta t = 10^{-2} \), and run until \( t = 0.1 \). Since approximate projections do not exactly enforce the incompressibility constraint, it is also interesting to note how approximate projections affect the volume conservation of the enclosed membrane. The IB method is well known to exhibit volume loss for closed pressurized membranes \cite{8, 21, 32, 35}, even when using an exact projection. Figure 3.5 demonstrates how approximate projections also have the effect of increasing the amount of volume loss. For comparison, an exact projection has a volume loss of 3.12% (almost exactly where the dashed line ends up in Figure 3.5) and an energy loss of \( 5.55 \times 10^{-14} \% \). (The energy loss is not exactly 0 with the exact projection because the implicit equations are solved only to a certain tolerance and because of the presence of round-off errors.)

### 3.3.3 Stability for the Full Navier-Stokes Equations

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

\[
H(x) = \begin{cases} 
1 & x > 0 \\
0 & \text{otherwise} 
\end{cases}
\]

\[
((u \cdot \nabla)u)_{ij}^n = \left[ \begin{array}{c}
H(u_{ij}^n) u_{ij}^n \frac{u_{i+1,j}^n - u_{ij}^n}{h} + H(-u_{ij}^n) u_{ij}^n \frac{u_{ij}^n - u_{ij-1}^n}{h} \\
H(v_{ij}^n) v_{ij}^n \frac{v_{i,j+1}^n - v_{ij}^n}{h} + H(-v_{ij}^n) v_{ij}^n \frac{v_{ij}^n - v_{ij-1}^n}{h} \\
H(u_{ij}^n) u_{ij}^n \frac{v_{i,j+1}^n - v_{ij}^n}{h} + H(-u_{ij}^n) u_{ij}^n \frac{v_{ij}^n - v_{ij-1}^n}{h} \\
H(v_{ij}^n) v_{ij}^n \frac{v_{i,j+1}^n - v_{ij}^n}{h} + H(-v_{ij}^n) v_{ij}^n \frac{v_{ij}^n - v_{ij-1}^n}{h}
\end{array} \right].
\]

We ran the simulation with \( \mu = 0 \) and at the CFL constraint \( \Delta t = \frac{h}{\|u\|_{\infty}} \) for each of \( \gamma = 1, 10^2, 10^5, 10^{10} \). Since the fluid is initially at rest, we (somewhat arbitrarily) set \( \Delta t \) for the first timestep to be about one-fifth the length of time needed for one oscillation. In each case, we ran until two full oscillations of the boundary had
Figure 3.5. Energy (---) and volume (--) loss by time $t = 0.1$ in an inviscid simulation as a function of the number of approximate projections performed. Computations performed with $\gamma = 1$ and $\Delta t = 10^{-2}$.

occurred and verified that the energy of the system was decreasing in all cases. Figure 3.6 plots the energy as a function of time for the simulation where $\gamma = 10^{10}$.

3.3.4 Volume Loss and Stability

Even though the velocity field on the Eulerian grid will be discretely divergence-free when an exact projection is used, this does not guarantee that the interpolated velocity field (in which the immersed boundary moves) is continuously divergence-free. Our implicit method does not fix this problem, revealing an interesting and unexpected view of the relationship of the incompressibility constraint and stability. When running our implicit method with a timestep sufficiently small that the explicit scheme is stable, we observed that the explicit method and our implicit method exhibit nearly identical volume loss. Since the total volume loss by the end of the simulation is an increasing function of the size of the timestep, and because the implicit method is generally used with a larger timestep than the explicit method, use of the implicit method will generally result in a greater
volume loss. Stockie and Wetton [38] observed similar volume loss for the Mayo and Peskin scheme and concluded that it was this accumulation of error in the incompressibility condition that caused the stability limitation on the timestep for that scheme. While a reasonable hypothesis, our implicit scheme shows similarly increasing volume loss despite being unconditionally stable. In fact, partly based on correlation of energy and volume loss observed in Section 3.3.2, we conjecture the opposite — that the loss of volume in the IB method actually aids stability. A loss of volume results in a smaller elastic force due to the shortened distance between points, making the resulting grid forces smaller and thus making the system less likely to over correct for any previous errors.

3.4 Discussion

In [32], Peskin provides an overview of the current state of the IB method. In his outline of active research directions and outstanding problems, the first issue addressed is the “severe restriction on the time-step duration” to which existing
implementations are subject. Pointers to research that has been ongoing in this area for over a decade are provided, and Peskin states that “it would be a huge improvement if this [restriction] could be overcome.” The analysis of this chapter demonstrates that unconditionally stable IB methods are possible and identifies specific features of the method required to achieve unconditional stability. This chapter therefore solves the stability problem posed by Peskin — with some caveats. For example, our results do not apply to all extensions of the IB method, and other outstanding problems of the IB method still exist. The results of this chapter raise several interesting questions as well. Can such a method be made efficient? Can it be made more accurate? Can the volume loss be mitigated or removed? Does the stability carry over to common extensions and modifications of the standard IB method? Is there anything that can be done to improve existing explicit codes without switching to an implicit solver? We discuss these questions in this section with potential solutions suggested by results from this chapter.

3.4.1 Efficiency

Perhaps the most prominent question is whether the method can be made efficient. Indeed, this question seems to be outstanding in the community for the various implicit methods that have been proposed and used, as evidenced by the fact that Stockie and Wetton [38] is the only paper of which we are aware that has concretely compared computational costs of explicit and implicit methods. In some sense, we have returned to where Tu and Peskin [41] left off. There are similarities between Tu and Peskin’s paper and our analysis: they presented a method that appeared to be unconditionally stable; they solved their implicit system of equations using Newton’s method, which was therefore far less efficient than an explicit method; and they then presented a challenge to come up with a more efficient version. There are some important differences: our method applies to more than just steady Stokes flow; we have proven that some methods actually are unconditionally stable; we have discovered why the paper written to meet the challenge from Tu and Peskin (namely Mayo and Peskin [23]) failed to remain
stable; and we have demonstrated that aspects of the problem that the community suspected to be causing the instability of previous implicit methods (namely, volume loss and lagging of spreading and interpolation operators) were not the actual source of instability.

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

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

3.4.2 Accuracy

By lagging the spreading and interpolation operators, we have a method that is only first order accurate in time. Since the IB method is only first order accurate for sharp interface problems and it is to such problems that the method is typically applied, we did not address this issue in this chapter. However, the IB method is beginning to be applied to problems with a thick interface [14, 27] where the IB method can achieve higher order accuracy. Also, the IIM can be used in problems with a sharp interface and get second order accuracy.

One possibility for obtaining a second order accurate scheme comes from realizing that our proof for stability did not rely at all on where $S$ and $S^*$ were evaluated — as long as they were evaluated at the same location. This means that we can employ a two-step method, where $X^{n+\frac{1}{2}}$ is computed in the first step to at least first order accuracy, and then the value of $X^{n+\frac{1}{2}}$ is used as the location of the spreading
and interpolation operators in a second step to solve for $X^{n+1}$ and $u^{n+1}$. See Lai and Peskin [18] for an example of a very similar explicit two-step approach used to obtain formal second order accuracy.

### 3.4.3 Volume Conservation

Another important issue in some applications concerns the volume loss that occurs with the IB method. As discussed in Section 3.3.4, the implicit method will generally exhibit greater volume loss than the explicit method due to the use of a larger timestep. One solution is to use the IIM [21, 22] or a hybrid IB/II method [20], though the stability results of this chapter might not hold with the different spatial discretizations used for the “spreading” and interpolation operators in such methods. Another solution would be to use the modified stencils of Peskin and Printz [35], which satisfy the necessary conditions in the proofs in this chapter. Peskin and Printz reported substantially improved volume conservation, but at the cost of wider stencils and more computationally expensive interpolation and spreading. We explore a new possibility in Chapter 4.

### 3.4.4 Extensions

There are many extensions to the IB method as well as variations on how the fluid equations are solved. One extension that has been mentioned many times already is the IIM and hybrid IB/II methods. These methods modify the finite difference stencils of the fluid solver near the immersed boundary instead of utilizing discrete delta functions to spread the force from the Lagrangian to the Eulerian grid. They also typically use a different interpolation scheme, such as bilinear interpolation. For these methods, the necessary adjoint property of the “spreading” and interpolation operators does not hold. Whether the stability results of this chapter can be extended for such schemes or whether those schemes can be modified to be made unconditionally stable is unknown. Similarly, it is unknown how the stability is affected by more complicated discretizations such as the double projection method proposed by Almgren et al. [2], an L0-stable
discretization of the viscous terms from Twizell et al. [42], or some methods of incorporating boundary mass in the IB method [17, 44].

3.4.5 Explicit Method Stability

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

3.5 Conclusions

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

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

Another unexpected result was that the error in the incompressibility constraint exhibited by existing implementations apparently does not adversely affect the stability of the computations. We proved that for implementations employing an exact projection and satisfying the necessary conditions from Section 3.2.2, the method is unconditionally stable, despite the fact that such methods suffer volume change due to the interpolated velocity field on the Lagrangian grid not being divergence-free. We further demonstrated computationally that a common method (namely, approximate projections) that results in the divergence-free constraint being satisfied only approximately on both the Eulerian and Lagrangian grids still appears to be unconditionally stable.

Finally, we demonstrated computationally that the advection terms from the Navier-Stokes equations present no additional difficulty beyond the stability (CFL) constraint of advection alone.
CHAPTER 4

VOLUME CONSERVATION

The IB method is well known to exhibit volume loss for closed pressurized membranes, even when using an exact projection. Multiple methods have been introduced to fix this problem [8, 21, 35], but we present a new method that requires only a small modification to existing codes and that substantially improves volume conservation with only a marginal computational cost. This new method works by correcting the interpolated velocity field to satisfy a discrete divergence-free constraint defined on the Lagrangian interface.

4.1 Continuous vs. Discretely Divergence-free Vector Fields

In the continuous set of equations, the continuity equation we want to satisfy is

\[ \nabla \cdot \mathbf{u} = 0, \quad (4.1) \]

i.e., the velocity field must be divergence free. However, exact projection methods solve a different constraint, namely

\[ \nabla_h \cdot \mathbf{u} = 0. \quad (4.2) \]

A continuous velocity field satisfying equation (4.1) will not in general satisfy (4.2) and vice-versa. Also, the choice of \( \nabla_h \) is not unique, and thus a vector field that is discretely divergence free under one discrete divergence operator will not in general be discretely divergence free according to other discrete divergence operators.

This point is important because IB and IIM methods make use of two computational grids. When an exact projection is employed in these methods, the computed velocity field on the Eulerian grid is discretely divergence free, but this velocity
field is then interpolated to the Lagrangian grid with interpolation operators that ignore the divergence free constraint. In general, this results in a velocity field on the Lagrangian grid that is $O(h^2)$ away from being divergence free (assuming a second order interpolation scheme is used). This can result in the volume of an enclosed pressurized object shrinking.

Peskin and Printz [35] addressed this problem by modifying the stencil of their divergence operator relative to their chosen interpolation operator, such that a velocity field that is discretely divergence free under their modified divergence operator would remain discretely divergence free on the Lagrangian grid when interpolated with their specific interpolation operator.

The IIM method [21] modifies the finite difference stencils of the fluid solver near the immersed boundary instead of utilizing discrete delta functions to spread the force from the Lagrangian to the Eulerian grid. The modified stencils result in a more accurate computed Eulerian velocity and pressure by discretizing the discontinuities in these variables near the immersed boundary. These modified finite difference stencils can be viewed as using modified divergence, gradient, and laplacian operators (as discretizing the discontinuities results in so-called “correction terms” for the finite difference stencils). It turns out that this method also has much better volume conservation as well.

The Blob projection method [8] is based on finding an analytic expression that represents the projection of a regularized form of the forces along the boundary onto the space of divergence-free vector fields without first transferring the forces from the Lagrangian to the Eulerian grid. This projected force field is used in combination with a projected momentum equation (projecting each term of the equation onto the space of divergence-free vector fields). This method achieves higher order accuracy of IB position and grid velocity, as well as providing much better volume conservation.

All these previous attempts at fixing volume conservation focused on modifying the stencils of the discrete differential operators on the Eulerian grid in a way that resulted in the interpolated velocity field being nearly incompressible. None
of these methods tried to fix the problem by altering the interpolation operator to maintain the incompressibility constraint. While the IIM does have a modified interpolation operator, the modifications were made to accurately account for jump discontinuities in the velocity rather than to account for the incompressibility constraint. Also, the hybrid IB/II method uses simple bilinear interpolation and achieves roughly the same volume conservation improvements that the full IIM obtains. The idea of our method is much simpler: modify the interpolation operator to maintain the incompressibility constraint when transferring the velocity between grids. We make no modifications to the discrete differential operators used on the Eulerian grid.

4.2 Discrete Incompressibility Constraint on a Surface

In order to enforce the incompressibility constraint on the Lagrangian grid, we need some discrete equation that makes sense on that grid. Returning to the continuous equation, we have

\[ 0 = \nabla \cdot \mathbf{u}, \]  

which we can weaken to integral form:

\[ 0 = \int_{\Omega} \nabla \cdot \mathbf{u} \, dV. \]  

(4.4)

However, what we really care about is the volume inside any enclosed membrane; as long as the integral of the divergence within that membrane is 0 we will still have volume conservation. Thus, defining \( \Omega^- \) as the region interior to the membrane, we are concerned only that

\[ 0 = \int_{\Omega^-} \nabla \cdot \mathbf{u} \, dV. \]  

(4.5)

Applying the divergence theorem we obtain

\[ 0 = \int_{\partial \Omega^-} \mathbf{u} \cdot \mathbf{n} \, dS, \]  

(4.6)
where \( \mathbf{n} \) is the outward unit normal to the boundary, \( \partial \Omega^- \). This last equation is defined solely on the Lagrangian interface, so it is this equation that we discretize to obtain our constraint

\[
\sum_i \mathbf{u}_i \cdot \mathbf{n}_i \Delta S_i = 0,
\]

where \( \Delta S_i \) is some discretization of arclength in the current configuration. (It is worth clarifying that \( \Delta S_i \neq \Delta s_i \) since the latter is used in other chapters to refer to arclength in some reference configuration.) Note that it is not important whether the \( \mathbf{n}_i \)'s are outward or inward normals, as long as they are consistent.

To use this constraint, we need the normal vector at each IB point, i.e., at the vertices of the polygonal discretization of the immersed boundary. To obtain the normal vector, we first define the (unnormalized) tangent vector at the \( i \)th IB point as

\[
\mathbf{\tau}_i = (X_{i+1} - X_{i-1}, Y_{i+1} - Y_{i-1}).
\]

This is just the secant line connecting IB points \( i-1 \) and \( i+1 \). Given this definition, we can define \( \Delta S_i \) by

\[
\Delta S_i = \frac{1}{2} \| \mathbf{\tau}_i \|.
\]

The normal vectors to \( \mathbf{\tau}_i \) (of the same magnitude) are just

\[
\mathbf{\eta}_i = \pm (Y_{i+1} - Y_{i-1}, X_{i-1} - X_{i+1}).
\]

Now, we can finally define \( \mathbf{n}_i \):

\[
\mathbf{n}_i = \frac{\mathbf{\eta}_i}{\| \mathbf{\eta}_i \|} = \frac{\mathbf{\eta}_i}{\| \mathbf{\tau}_i \|} = \frac{\mathbf{\eta}_i}{2 \Delta S_i}.
\]

Enforcing (4.7) on the discrete velocity \( \mathbf{U} \) that lives on the Lagrangian grid is rather simple. First, for each closed IB object, we find the mean of \( \mathbf{U}_i \cdot \mathbf{n}_i \) on that object:

\[
M = \frac{1}{\sum_i \Delta S_i} \sum_i \mathbf{U}_i \cdot \mathbf{n}_i \Delta S_i,
\]
and then make the replacement

$$U_i := U_i - M n_i.$$  \hfill (4.13)

We can view (4.13) as a modification of the interpolation operator using global information from the entire IB object under consideration. This modification will enforce the condition

$$\sum_i U_i \cdot n_i \Delta S_i = 0.$$  \hfill (4.14)

We expect the modified interpolation operator, defined by applying (4.13) as a postprocessing step after applying the IB interpolation operator, to remain second order. The reasoning is as follows: $U = S^* u$ is a second order interpolation of the Eulerian velocity field $u$ and $u$ satisfies the continuous divergence-free condition to $O(h^2)$, so it should be the case that $M$ defined in equation (4.12) is $O(h^2)$ and thus that the correction (4.13) does not lower the order of accuracy of the modified interpolation operator.

While the modified interpolation operator will enforce the Lagrangian incompressibility condition (4.14), in practice the object is still observed to shrink, though at a dramatically reduced rate. Perfect volume conservation of the non-discretized object would require satisfying (4.6) rather than (4.14), but our immersed boundary is discretized and described by a collection of points so measurement of continuous volume does not make sense. We need some discrete measurement of volume.

### 4.3 Polygonal Area

While our velocity correction should improve volume conservation, we would like to have some way of measuring the improvement. Our immersed boundary is represented as a collection of points, so we need to define some way to fit a surface to those points and then measure the enclosed volume. The most natural choice is to simply use the “springs” (straight lines) connecting adjacent IB points and then
compute the area of the resulting polygon. The formula for this area (assuming no self-intersections and counter-clockwise ordering [36]) is simply

\[
A_{\text{polygon}} = \frac{1}{2} \sum X_i (Y_{i+1} - Y_{i-1}) = \frac{1}{4} \sum X_i (Y_{i+1} - Y_{i-1}) + \frac{1}{4} \sum Y_i (X_{i-1} - X_{i+1}).
\]

We look at the general update formulas

\[
X_i^{n+1} = X_i^n + \Delta t U_i, \tag{4.17}
\]
\[
Y_i^{n+1} = Y_i^n + \Delta t V_i, \tag{4.18}
\]

and do not assume that \(U_i\) and \(V_i\) have been "corrected" yet, i.e., we do not assume they satisfy any condition like (4.14). Now, if we substitute these into the formula for the area at time level \(n+1\) we obtain

\[
A^{n+1} = \frac{1}{4} \sum X_i^{n+1} (Y_{i+1}^{n+1} - Y_{i-1}^{n+1}) + \frac{1}{4} \sum Y_i^{n+1} (X_{i-1}^{n+1} - X_{i+1}^{n+1}) \tag{4.19}
\]

\[
= \frac{1}{4} \sum [X_i^n + \Delta t U_i] [Y_{i+1}^n - Y_{i-1}^n + \Delta t V_{i+1} - \Delta t V_{i-1}]
\]
\[
+ \frac{1}{4} \sum [Y_i^n + \Delta t V_i] [X_{i+1}^n - X_{i-1}^n + \Delta t U_{i-1} - \Delta t U_{i+1}]. \tag{4.20}
\]

Distributing and collecting similar terms we find that

\[
A^{n+1} = \frac{1}{4} \sum X_i^n (Y_{i+1}^n - Y_{i-1}^n) + \frac{1}{4} \sum Y_i^n (X_{i-1}^n - X_{i+1}^n)
\]
\[
+ \frac{\Delta t}{4} \sum U_i (Y_{i+1}^n - Y_{i-1}^n) + \frac{\Delta t}{4} \sum V_i (X_{i-1}^n - X_{i+1}^n)
\]
\[
+ \frac{\Delta t^2}{4} \sum X_i^n (V_{i+1} - V_{i-1}) + \frac{\Delta t^2}{4} \sum Y_i^n (U_{i-1} - U_{i+1})
\]
\[
+ \frac{\Delta t^2}{4} \sum U_i (V_{i+1} - V_{i-1}) + \frac{\Delta t^2}{4} \sum V_i (U_{i-1} - U_{i+1}). \tag{4.21}
\]

The first two terms on the right-hand side are just the polygonal area at timestep \(n\). The two lines after that can be combined by rearranging the summations. This gives us

\[
A^{n+1} = A^n
\]
\[
+ \frac{\Delta t}{2} \sum U_i (Y_{i+1}^n - Y_{i-1}^n) + \frac{\Delta t}{2} \sum V_i (X_{i-1}^n - X_{i+1}^n) \tag{4.22}
\]
\[
+ \frac{\Delta t^2}{4} \sum U_i (V_{i+1} - V_{i-1}) + \frac{\Delta t^2}{4} \sum V_i (U_{i-1} - U_{i+1}).
\]

If we want polygonal area of the IB to be conserved, we find that this standard update gives us an error that is \(O(\Delta t)\) in each timestep. Because the error is
roughly linear in $\Delta t$, we expect that the volume loss by a given time will likely be nearly independent of the timestep used in the simulation. This means that the volume loss will not converge to 0 as $\Delta t \to 0$ (though it will as $h \to 0$). We have observed this in practice.

The first of the “error terms” in (4.22) (the term after $A^n$) is of the form $\sum U_i \cdot \eta_i$, an expression for the average flux of volume across the boundary. It is not quite so clear what the other term represents, though it can be related to the acceleration (through differentiation of (4.24) from the next section); specifically, it is of the form $-\frac{1}{\Delta t} \sum U'_i \cdot \eta_i$.

In the next two sections, we determine the effect of the volume/velocity correction (4.13) and derive a correction that will ensure exact preservation of polygonal area.

### 4.4 Effect of Velocity Correction

We want to determine how (4.14) affects the error terms in (4.22) for the standard IB point update. Noting that equation (4.14) can be rewritten as

$$0 = \sum_i U_i \cdot n_i \Delta S_i$$

$$= \sum_i U_i \cdot \frac{\eta_i}{\|\eta_i\|} \Delta S_i$$

$$= \sum_i \frac{1}{2} U_i \cdot \eta_i$$

$$= \sum_i \frac{1}{2} U_i(Y_{i+1}^n - Y_{i-1}^n) + \frac{1}{2} V_i(X_{i-1}^n - X_{i+1}^n),$$

which implies that

$$0 = \sum_i U_i(Y_{i+1}^n - Y_{i-1}^n) + V_i(X_{i-1}^n - X_{i+1}^n),$$

we see that (4.22) simplifies to

$$A^{n+1} = A^n + \frac{\Delta t^2}{4} \sum U_i(V_{i+1} - V_{i-1}) + \frac{\Delta t^2}{4} \sum V_i(U_{i-1} - U_{i+1}).$$

This means that enforcing the Lagrangian incompressibility condition (4.14) will cause our error in preserving polygonal volume to be much smaller than with the
standard IB update. Further, since the only error term in conserving polygonal volume is $O(\Delta t^2)$, the percentage volume loss of the immersed boundary should converge to 0 as $\Delta t \to 0$, independent of $h$.

4.5 Exact Conservation of Polygonal Area

Given that we know the error terms in conserving the polygonal volume, it seems we could derive a method that eliminates those errors entirely. Also, since the remaining error term after our previous velocity correction is higher order than the one that it eliminated, we believe that a modification that perfectly conserves polygonal volume should not reduce the overall second order accuracy of the interpolation operator.

In the previous section, we assumed the definition of $n_i$ and $S_i$ defined in Section 4.2. This simplified the computations for clarity, but we would like to allow the use of other definitions of $n_i = (n_{i,1}, n_{i,2})$ and $S_i$ in our derivation here. Our method is to write the volume correction we want to perform as

$$U_i := U_i - Mn_{i,1} \quad (4.26)$$
$$V_i := V_i - Mn_{i,2} \quad (4.27)$$

and leave $M$ to be determined later. Plugging this corrected value of $U$ into (4.22) we get

$$A^{n+1} = A^n + \frac{\Delta t}{2} \sum (U_i - Mn_{i,1})(Y^n_{i+1} - Y^n_{i-1})$$
$$+ \frac{\Delta t}{2} \sum (V_i - Mn_{i,2})(X^n_{i+1} - X^n_{i+1})$$
$$+ \frac{\Delta t^2}{4} \sum (U_i - Mn_{i,1})(V_{i+1} - V_{i-1} - (Mn_{i+1,2} - Mn_{i-1,2}))$$
$$+ \frac{\Delta t^2}{4} \sum (V_i - Mn_{i,2})(U_{i+1} - U_{i+1} - (Mn_{i+1,1} - Mn_{i+1,1})).$$

If we pick $M$ to enforce $A^{n+1} = A^n$, then we get a simple quadratic equation for $M$. The reason for there being two possible values for $M$ is that we can preserve the volume by moving the boundary far enough to entirely invert it. We really do not want to do that, so we pick the value of $M$ that is smallest in magnitude.
4.6 Unequally Spaced Surface Points

Equations (4.8) and (4.10) for the tangent and normal vectors were derived assuming that the points on the Lagrangian grid were evenly spaced. While this should be true in a rough sense, it could be off by 30% or more in common IB simulations. When points are not equally spaced, the definition of normals in equation (4.10) has lower accuracy. We can improve this by first defining

\[ \mathbf{\tau}_{i+} = (X_{i+1} - X_i, Y_{i+1} - Y_i) \]  
\[ \mathbf{\tau}_{i-} = (X_i - X_{i-1}, Y_i - Y_{i-1}), \]

which are the secant lines connecting IB points \( i \) and \( i+1 \), and connecting IB points \( i-1 \) and \( i \), respectively. We define

\[ \Delta S_{i+} = \| \mathbf{\tau}_{i+} \| \]  
\[ \Delta S_{i-} = \| \mathbf{\tau}_{i-} \| \]  
\[ \Delta S_i = \frac{1}{2}(\Delta S_{i+} + \Delta S_{i-}). \]

We then normalize the secant lines

\[ \mathbf{t}_{i+} = \frac{\mathbf{\tau}_{i+}}{\Delta S_{i+}} \]  
\[ \mathbf{t}_{i-} = \frac{\mathbf{\tau}_{i-}}{\Delta S_{i-}} \]

and define a weighted “tangent” at IB point \( i \) as

\[ \mathbf{\tau}_i = \frac{\Delta S_{i-}\mathbf{t}_{i+} + \Delta S_{i+}\mathbf{t}_{i-}}{\Delta S_{i-} + \Delta S_{i+}}, \]

with a corresponding normal at IB point \( i \) as

\[ \mathbf{n}_i = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \mathbf{\tau}_i = \begin{pmatrix} \tau_{i,2} \\ -\tau_{i,1} \end{pmatrix}. \]

Now we can define \( \mathbf{n}_i \) as

\[ \mathbf{n}_i = \frac{\mathbf{\eta}_i}{\| \mathbf{\eta}_i \|} \]

which we can use in place of (4.11) in equations (4.12) and (4.13).
4.7 Computational Results

In this section, we demonstrate the much improved volume conservation obtained with the velocity correction postprocessing step suggested in this chapter. The test problem we use is the same one used in Section 3.3, other than the velocity correction. We use the definition of normals given in Section 4.6 and the correction defined in equation (4.13). We get nearly identical results by using either the value of $M$ defined in equation (4.12) or by using the value of $M$ that enforces $A_{n+1} = A^n$ in equation (4.28), so we present the results from only the latter.

4.7.1 Refinement Study

We begin with a simple convergence study to verify that the volume correction does not adversely affect the convergence of the method. This is a repeat of the refinement study test done in Section 3.3.1 with the exception that the volume correction is included in the computations.

The results of the refinement study are displayed in Table 4.1. Because an analytic solution is not available, we estimate the convergence rate by comparing the differences of the numerical solutions between successive grid levels. We use the $l^2$ norms induced by the inner products in equations (3.18) and (3.19). See Section 3.3.1 for more details of the parameters in the system, and note how close these results are to the ones in that section.

**Table 4.1** Results of a refinement study showing first order convergence of method 3C. The convergence rate is defined as $\sqrt{E_1/E_3}$. $E_i(q)$ measures the error in variable $q$ at level $i$, defined by $\|q_i - \text{coarsen}(q_{i+1})\|_2$. Here, $q_i$ is the value of variable $q$ at time $t = 0.2$ (seconds) computed with the numerical parameters for level $i$ from Table 3.1. The coarsen operator is simple averaging of nearest values for Eulerian quantities, and injection from identical gridpoints for Lagrangian quantities.

<table>
<thead>
<tr>
<th>$q$</th>
<th>$E_1(q)$</th>
<th>$E_2(q)$</th>
<th>$E_3(q)$</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u$</td>
<td>1.06e-03</td>
<td>4.17e-04</td>
<td>1.76e-04</td>
<td>2.4486</td>
</tr>
<tr>
<td>$X$</td>
<td>5.26e-04</td>
<td>2.96e-04</td>
<td>1.58e-04</td>
<td>1.8241</td>
</tr>
</tbody>
</table>
4.7.2 Volume Conservation Comparison

Figure 4.1 shows the volume as a function of time. This graph compares the discretization given in Chapter 3 and the same discretization modified to conserve polygonal volume as outlined in Section 4.5, showing perfect volume conservation for the modified version. We have run the simulation about 6 times longer than we did in Chapter 3 in order to demonstrate more clearly the long-term volume loss problems presented by the standard IB method.

4.7.3 Effect on Energy Conservation

Unfortunately, these volume correction methods have a drawback as well. By modifying the velocity field in which the IB moves, our energy conservation results from Chapter 3 no longer hold. In fact, the system can go unstable; whether it does depends on the particular choices of $\nu$, $\gamma$, $\Delta t$, and $h$. Figure 4.2 shows the energy as a function of time, in an implicit simulation which has been run much longer in order to show the long-term effects on energy. From Figure 4.2, it can be seen that the system with the volume correction is going unstable. This simulation

![Figure 4.1. Comparison of volume of the system with and without the volume correction](image)
Figure 4.2. Comparison of energy of the system with and without the volume correction

represents 500 evenly spaced timesteps, so this instability is not as severe as found in the standard (explicit) IB method (which rapidly goes unstable within just a few timesteps).

We have tried various methods for fixing the instability introduced by this volume correction, which are covered in Appendix B. Unfortunately, we have not had much success. However, it should be noted that many simulations may not be affected by this instability. Early results led us initially to believe that the method would always remain stable, and for some applications (in cell dynamics) we had no problems with these corrections. It was not until we tried to more thoroughly test parameter space that we noticed instabilities. Also worth noting is that Figure 4.2 was run with a very coarse value of $h$, 1/32. Cutting that value of $h$ in half to 1/64 resulted in the stable system shown in Figure 4.3.

4.8 Conclusions

We have presented a method for preserving the volume of closed objects in the IB method that only requires minor modification of existing codes and that has a
negligible computational cost. The method can achieve perfect conservation of the polygonal area connecting the IB points, though it can destabilize the numerical scheme depending on the values of $\Delta t$, $\nu$, $\gamma$, and $h$. 

**Figure 4.3.** Comparison of energy of the system with and without the volume correction, using a slightly smaller value of $h$
CHAPTER 5

EFFICIENCY — SOLVER POSSIBILITIES

In this chapter, we highlight several possible methods for solving the implicit system of equations derived in Chapter 3. We only briefly outline methods in this chapter, leaving more detailed exploration of a subset of the possibilities to Chapters 6 and 7.

The implicit system derived in Chapter 3 that we want to solve (implicit discretization 3C), leaving the time at which we are evaluating the spreading and interpolation operators unspecified, is

\[
\begin{align*}
\frac{\rho u^{n+1} - u^n}{\Delta t} &= -\nabla^h p^{n+\frac{1}{2}} - \rho [(u \cdot \nabla)u]^{n+\frac{1}{2}} \\
&\quad + \frac{\mu}{2} \Delta^h (u^{n+1} + u^n) + \frac{1}{2} S_A f (X^n + X^{n+1}) \\
\nabla^h \cdot u^{n+1} &= 0 \\
\frac{X^{n+1} - X^n}{\Delta t} &= \frac{1}{2} S^* (u^n + u^{n+1}).
\end{align*}
\] (5.1)

(5.2)

(5.3)

There are three variables to determine: \(u^{n+1}, \nabla^h p^{n+\frac{1}{2}}, \) and \(X^{n+1}\). We write this in the form of one large system; after multiplying (5.1) and (5.3) through by \(\Delta t\) we have

\[
\begin{pmatrix}
\rho I - \frac{\mu \Delta t}{2} \Delta^h & \Delta t \nabla^h - \frac{\Delta t}{2} S_A f \\
\nabla^h & 0 & 0 \\
-\frac{\Delta t}{2} S^* & 0 & I
\end{pmatrix}
\begin{pmatrix}
\frac{u^{n+1}}{p^{n+\frac{1}{2}}} \\
X^{n+1}
\end{pmatrix}
= \begin{pmatrix}
\frac{u^n}{p^{n-1/2}} \\
X^n
\end{pmatrix}.
\] (5.4)

In equation (5.4), we have assumed that \(S, A_f, \) and \(S^*\) are linear. This is true if we lag the time evaluation of the spreading and interpolation operators as in Chapter 3.
and have an appropriate constitutive law for the elastic forces, such as a linearly elastic material with zero resting length.

5.1 Direct Application of Linear Solvers

Perhaps the most obvious way to solve equation (5.4) is to apply linear system solvers directly to it. However, this system poses a number of difficulties for doing so. The system is singular (the gradient operator has a nontrivial nullspace). Direct matrix factorization methods such as LU decomposition would result in a huge amount of fill-in in an otherwise very sparse system, and methods like GMRES may fail to converge without a good preconditioner (and it is not clear what would be a good preconditioner for this system). Such methods do not look promising to us because of these potential drawbacks, and we are not aware of any work to explore these options.

5.2 Newton-like Methods

One of the features of the IB method that makes it appealing is the ability to use standard Navier-Stokes solver codes without modification. There were two keys to the IB method and its implementations that allowed this: one, the use of a singular force to represent the immersed objects in order to allow a single set of fluid dynamics equations to hold throughout the domain with no internal boundary conditions; and two, the fact that the IB terms have traditionally been discretized explicitly in order to decouple their solution from that of the velocity and pressure. Implicit discretizations do not decouple the IB position from the velocity and pressure, making the use of unmodified standard Navier-Stokes solver codes less straightforward.

Before [28], it was commonly believed that fully implicit discretizations were necessary to obtain an unconditionally stable IB method. Because of this, many implementations were based on fully implicit discretizations. Fully implicit discretizations require the use of spreading and interpolation operators that are not lagged in time, and this makes the system of equations nonlinear in $X$. Thus, for such systems a nonlinear solver is needed.
Leveque and Li [21] introduced a Newton-like method that handles both situations mentioned above, namely dealing with the nonlinearity in $X$ and allowing Navier-Stokes codes to be used unmodified as a subsolver. Their method was based on rearranging equation (5.3) to obtain

$$g(X^{n+1}) = X^{n+1} - X^n - \frac{\Delta t}{2} S_s(u^n + u^{n+1}(X^{n+1})) = 0,$$

(5.5)

where the notation $u^{n+1}(X^{n+1})$ is used to make it clear that each evaluation of $g$ involves solving the Navier-Stokes equations. This equation can then be solved with Newton’s method or a Quasi-Newton method. It is worth noting that Tu and Peskin proposed a similar method [41], though it was restricted to the case of steady Stokes flow and written in a way that was tightly coupled to a specialized Stokes flow solver.

### 5.2.1 Newton’s Method

Solving equation (5.5) via Newton’s method requires an initial guess for the new IB point positions, such as $X^{n+1,0} = X^n$, and then iterating over the steps

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

In these steps, $\mathbf{e}_j$ is the vector with $i$th component equal to $\delta_{ij}^{\text{Kroenecker}}$, and $h$ is some small parameter (e.g., $\sqrt{\epsilon_{\text{machine}}}$). Unfortunately, the notation used in this context has been slightly ambiguous and not fully consistent. $X$ is often thought of as an $N_B$-dimensional array of vectors (where $N_B$ is the number of Lagrangian gridpoints) each of length 2 (in 2D), but in this case we are thinking of it as an array of scalars of length $2N_B$. $s$ and $\mathbf{e}_j$ are also arrays of scalars of length $2N_B$, while $J$ is a matrix of size $2N_B \times 2N_B$. This means that the subscripts $j$ and $i$ each range from 1 to $2N_B$, not from 1 to 2 or from 1 to $N_B$.

Newton’s method has been used in both [28] and [41], and was reported to be computationally inefficient. The reason for this is that computing the Jacobian at
each iteration involves $4N_B$ fluid solves, where $N_B$ is the number of points in the immersed boundary. By using one-sided differences to compute the Jacobian, this cost could be cut in half (only requiring $2N_B + 1$ fluid solves), but it will still be orders of magnitude too slow.

5.2.2 Quasi-Newton Methods

Solving equation (5.5) via a Quasi-Newton method is very similar to using Newton’s method. It requires having an initial guess for the new IB point positions, such as $X^{n+1,0} = X^n$, as well as an initial guess for the Jacobian, such as $J^{n+1,0} = J^n$ or $J^{n+1,0} = I$. The reason that $J^{n+1,0} = I$ may be a good choice is that

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

so $J \to I$ as $\Delta t \to 0$. Once the initial guesses are selected, Quasi-Newton methods then proceed by iterating over the steps

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

There are many different ways to update the Jacobian in the final step of each iteration, corresponding to different Quasi-Newton methods. The only Quasi-Newton method that has appeared in the IB literature is BFGS. Defining $H = (J^{n+1,m})^{-1}$, the BFGS update [9] is

$$(J^{n+1,m+1})^{-1} = H + \frac{(s - Hy)s^T + s(s - Hy)^T}{s^T y} - \frac{(s - Hy)^T y s s^T}{(s^T y)^2}.$$ (5.7)

This method is cheap in comparison to the Newton method as it involves only one fluid solve per iteration (note that the computation of $g(X^{n+1,m+1})$ in step three of each iteration can be reused in steps 1 and 3 of the following iteration). In fact, the most expensive part of this iteration may be the dense $2N_B \times 2N_B$
matrix multiply in step 1 (BFGS stores approximations to $J^{-1}$ rather than $J$). The computational cost of such a matrix multiplication grows quadratically with the number of IB points, $N_B$; further, $J$ becomes increasingly ill-conditioned both as $N_B$ grows and as the IB points get closer together [21]. However, for some problems $N_B$ will be small enough relative to the number of Eulerian gridpoints, $N$, that this method may be reasonable.

This solution method has only been used in IIM implementations, possibly due to the fact that the representations of the boundary commonly used with the IIM allows fewer control points, thus allowing $N_B$ to be significantly smaller. This method was first used by Leveque and Li [21], and has been adopted by others using the IIM; see also [19, 20].

The use of BFGS is an interesting choice since it is more commonly used for minimization problems rather than nonlinear solves. In particular, it is designed with the assumption that the Jacobian is symmetric and positive definite, and only produces approximations to the Jacobian with those attributes [9]. While the Jacobian is clearly symmetric and positive definite in the limit $\Delta t \to 0$ (since $J \to I$), it is not at all clear that the Jacobian is symmetric and positive definite for finite $\Delta t$. However, none of the IIM papers using this method even mention this issue, let alone argues whether these assumptions are valid or why we can expect the method to work.

In private correspondence with Zhilin Li, he conjectured that the Jacobian would satisfy those assumptions “near” the solution. However, in our testing with a hybrid IB/II method (similar to Lee and Leveque’s work in [20], except that we used temporal discretization 3B instead of 2B) with a value of $\Delta t$ near the stability limit of the explicit method, we found that the skew-symmetric part of the Jacobian was as large in norm as the symmetric part of the Jacobian, and that the Jacobian was in fact indefinite with at least one eigenvalue having very large negative real part. This held true whether our iterate, $X^{n+1,m}$, was near the solution of the implicit equations or not. While it is known that the approximate Jacobian does not need to converge to the true Jacobian in order for Quasi-Newton methods to converge to
the solution, we are worried about the robustness of using such methods. In fact, we found in our testing that the Quasi-Newton method fails to converge when $\Delta t$ is taken to be unrealistically large, such as in our parameter stress-test problems of Chapter 3, and that the Newton method sometimes also breaks down under those cases without extra work (such as implementation of line searches). We did not fully investigate the threshold at which the Quasi-Newton methods failed to converge other than “it seemed to fail a lot sooner than the Newton method.”

5.3 Schur Complement Systems

Recall that the singular system of equations that we want to solve, equations (5.1)-(5.3), is

$$\rho \frac{u^{n+1} - u^n}{\Delta t} = -\nabla^h p^{n+\frac{1}{2}} - \rho [(u \cdot \nabla)u]^{n+\frac{1}{2}} + \mu \Delta^h (u^{n+1} + u^n) + \frac{1}{2} Sa_f (X^n + X^{n+1})$$

$$\nabla^h \cdot u^{n+1} = 0$$

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

Applying a Schur Complement approach to eliminate one or more of these variables is not so much a method of solving this system as it is a method for modifying it in an attempt to make it easier to solve. We derive multiple such modified systems in this section.

We begin by eliminating $p^{n+\frac{1}{2}}$ from equations (5.8)-(5.10). We do this by applying $P = (I - \nabla^h (\nabla^h \cdot \nabla^h)^{-1} \nabla^h \cdot )$, the projection operator that projects orthogonally in $L^2(\Omega^h)$ onto the space of (discretely) divergence-free vector fields, to each term of equation (5.8). This eliminates $p^{n+\frac{1}{2}}$ as well as the need for the separate incompressibility constraint equation and yields

$$\rho \frac{u^{n+1} - u^n}{\Delta t} = -\rho P [(u \cdot \nabla)u]^{n+\frac{1}{2}} + \frac{\mu}{2} P \Delta^h (u^{n+1} + u^n)$$

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

$$\nabla^h \cdot u^{n+1} = 0$$

$$\frac{X^{n+1} - X^n}{\Delta t} = \frac{1}{2} S^* (u^n + u^{n+1}).$$
Others who have used this particular Schur complement idea [23, 27, 38] have applied the projection before discretizing spatially. When the equations are still spatially continuous (or if we assume periodic boundary conditions), the operators $\mathbb{P}$ and $\Delta$ in equation (5.11) commute. Further, since $\mathbf{u}^{n+1}$ and $\mathbf{u}^n$ are already divergence free, we can use $\mathbf{u}^{n+1} = \mathbb{P}\mathbf{u}^n$ and $\mathbf{u}^n = \mathbb{P}\mathbf{u}^n$ in equation (5.11) to eliminate the occurrence of $\mathbb{P}$ in the viscous term. Doing that and solving for $\mathbf{u}^{n+1}$ in (5.11) and for $\mathbf{X}^{n+1}$ in (5.12), we obtain

$$\mathbf{u}^{n+1} = M \left( I + \frac{\nu \Delta t}{2} \Delta h \right) \mathbf{u}^n - \Delta t M \mathbb{P}[\mathbf{u} \cdot \nabla \mathbf{u}]^{n+\frac{1}{2}}$$

(5.13)

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

(5.14)

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

From here we can eliminate either $\mathbf{u}^{n+1}$ or $\mathbf{X}^{n+1}$. Plugging the value of $\mathbf{u}^{n+1}$ into equation (5.14) and collecting $\mathbf{X}^{n+1}$ terms to the left-hand side gives

$$\left( I - \frac{\Delta t^2}{4 \rho} \mathbb{S}^* M \mathbb{P} S A_f \right) \mathbf{X}^{n+1} = \left( I + \frac{\Delta t^2}{4 \rho} \mathbb{S}^* M \mathbb{P} S A_f \right) \mathbf{X}^n$$

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

(5.15)

Note that everything on the right-hand side of (5.15) can be calculated explicitly. Similarly, we could plug the value of $\mathbf{X}^{n+1}$ from (5.14) into equation (5.13) and collect $\mathbf{u}^{n+1}$ terms to the left-hand side, to obtain

$$\left( I - \frac{\Delta t^2}{4 \rho} M \mathbb{P} S A_f S^* \right) \mathbf{u}^{n+1} = \left( M \left( I + \frac{\nu \Delta t}{2} \Delta h \right) + \frac{\Delta t^2}{4 \rho} M \mathbb{P} S A_f S^* \right) \mathbf{u}^n$$

$$- \Delta t M \mathbb{P}[\mathbf{u} \cdot \nabla \mathbf{u}]^{n+\frac{1}{2}} + \frac{\Delta t}{\rho} M \mathbb{P} S A_f \mathbf{X}^n,$$

(5.16)

where again everything on the right-hand side of (5.16) can be calculated explicitly.

Equations (5.15) and (5.16) give us two possible avenues of attack, which we will discuss later. We can come up with others by beginning again with equations (5.8)-
If we solve equation (5.10) for $X^{n+1}$, substitute it into equation (5.8) to eliminate $X^{n+1}$, and then collect occurrences of $u^{n+1}$, we obtain

\[
\left( I - \frac{\nu \Delta t}{2} \Delta h - \frac{\Delta t^2}{4\rho} SA_f S^* \right) u^{n+1} = \left( I + \frac{\nu \Delta t}{2} \Delta h + \frac{\Delta t^2}{4\rho} SA_f S^* \right) u^n - \frac{\Delta t}{\rho} \nabla h p^{n+\frac{1}{2}} - \Delta t [(u \cdot \nabla) u]^{n+\frac{1}{2}} + \frac{\Delta t}{\rho} SA_f X^n \quad (5.17)
\]

\[
\nabla h \cdot u^{n+1} = 0.
\quad (5.18)
\]

Note that multiplying both sides of (5.17) by $MP$ will yield equation (5.16) (modulo one instance of commutativity of $P$ and $\Delta h$).

Equation (5.17) has some interesting properties. Defining $A_1 = \left( I - \frac{\nu \Delta t}{2} \Delta h \right)$, $A_2 = -\frac{\Delta t^2}{4\rho} SA_f S^*$, and $A = A_1 + A_2$, equation (5.17) simplifies to

\[
Au^{n+1} = (2I - A) u^n - \frac{\Delta t}{\rho} \nabla h p^{n+\frac{1}{2}} - \Delta t [(u \cdot \nabla) u]^{n+\frac{1}{2}} + \frac{\Delta t}{\rho} SA_f X^n.
\quad (5.19)
\]

If we assume $A_f$ is self-adjoint and negative semidefinite (as done in Chapter 3), then the adjointness of $S$ and $S^*$ imply that $A_2$ is symmetric and positive semidefinite. Combined with the fact that $\Delta h$ is symmetric and negative semidefinite, we see that $A$ is symmetric and positive definite, with all of its eigenvalues greater than or equal to 1. Additionally, $\lim_{\Delta t \to 0} A = I$.

One approach to solving equations (5.17) and (5.18) is to apply a Schur complement one more time to remove either $p^{n+\frac{1}{2}}$ or $u^{n+1}$. By applying $\frac{\Delta t}{\rho} \nabla h \cdot A^{-1}$ to every term of (5.19) and making use of (5.18), we can obtain an equation for $p^{n+\frac{1}{2}}$

\[
\nabla h \cdot A^{-1} \nabla h p^{n+\frac{1}{2}} = \nabla h \cdot A^{-1} \left[ \frac{\rho}{\Delta t} (2I - A) u^n - \rho [(u \cdot \nabla) u]^{n+\frac{1}{2}} + SA_f X^n \right].
\quad (5.20)
\]

Similarly, we could also apply $P$ to every term of (5.19), substitute $u^{n+1} = P u^{n+1}$ (which is valid due to equation (5.18)) and obtain

\[
PAP u^{n+1} = P \left[ (2I - A) u^n - \Delta t [(u \cdot \nabla) u]^{n+\frac{1}{2}} + \frac{\Delta t}{\rho} SA_f X^n \right].
\quad (5.21)
\]

Since $P$ is self-adjoint, $PAP$ is symmetric positive semidefinite (positive definite on the subspace of divergence-free vector fields). Although equation (5.21) is basically
equivalent to (5.16), \( \left( I - \frac{\Delta t^2}{4\rho} M\mathbb{P}S_A f^* \right) \) is not symmetric or positive definite. Thus, equation (5.21) may be the preferred form for applying linear solvers.

Equations (5.15), (5.16), (5.19)-(5.18), (5.20), and (5.21) each provide possibilities on which to base solvers. Each will be discussed in subsequent sections.

5.3.1 Fixed Point Methods

Equation (5.15) is of the form

\[
\left( I - \frac{\Delta t^2}{4\rho} S^* M\mathbb{P}S_A f \right) X^{n+1} = Z^n, \tag{5.22}
\]

where \( Z^n \) represents known quantities. We can rewrite this as a fixed point problem

\[
X^{n+1} = \frac{\Delta t^2}{4\rho} S^* M\mathbb{P}S_A f X^{n+1} + Z^n. \tag{5.23}
\]

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

\[
X^{n+1,m+1} = \frac{\Delta t^2}{4\rho} S^* M\mathbb{P}S_A f X^{n+1,m} + Z^n, \tag{5.24}
\]

or equivalently,

\[
X^{n+1,m+1} - X^{n+1,m} = Z^n - \left( I - \frac{\Delta t^2}{4\rho} S^* M\mathbb{P}S_A f \right) X^{n+1,m}. \tag{5.25}
\]

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

\[
(I - \lambda A_f)(X^{n+1,m+1} - X^{n+1,m}) = Z^n - \left( I - \frac{\Delta t^2}{4\rho} S^* M\mathbb{P}S_A f \right) X^{n+1,m}, \tag{5.26}
\]

which solves the same fixed point equation but may converge faster. Here \( \lambda \) is some simple approximation to \( \frac{\Delta t^2}{4\rho} S^* M\mathbb{P}S \).

A method of this form was first introduced by Mayo and Peskin [23]. They use a different temporal (as well as spatial) discretization than we have employed, but the form remains the same. In their work, they make \( \lambda \) a diagonal matrix:

\[
\lambda = \frac{\Delta t^2}{4\rho} S^* S 1, \tag{5.27}
\]

where \( 1(s) \) is a function that takes the value 1 for every \( s \), that is, \( 1(s) \) is 1 at each IB point. (Technically, they lump \( \Delta t^2/4\rho \) in with \( A_f \) rather than \( \lambda \) but the
product is equivalent either way). They report no computational timings other than “[it] often converged slowly.” They also explored Aitken extrapolation to try to accelerate convergence of the fixed point iterates, and later tried a Krylov solver directly on (5.22). Unfortunately, these other methods were not explored in detail and the comparisons they provide of these methods were only cursory, but they did state that the Krylov solver method was the fastest when it converged. As such, we focus our effort on Krylov methods for solving (5.22) instead.

This fixed point method of Mayo and Peskin was also used by Stockie and Wetton [38] in their analysis of explicit and implicit IB solvers. It also motivated a very similar fixed point method used by Roma et al.[37].

5.3.2 Projection Methods

Equations (5.19) and (5.18),

\[
A u^{n+1} = (2I - A) u^n - \frac{\Delta t}{\rho} \nabla^h p^{n+\frac{1}{2}} - \Delta t [(u \cdot \nabla)u]^{n+\frac{1}{2}} + \frac{\Delta t}{\rho} S A_f X^n \tag{5.28}
\]

\[
\nabla^h \cdot u^{n+1} = 0 \tag{5.29}
\]

where \(A = A_1 + A_2 = (I - \frac{\mu \Delta t}{2} \Delta^h) + \left(-\frac{\Delta t^2}{4\rho} S A_f S^* \right)\), look similar to the discretized Navier-Stokes equations. Since projection methods are a common method used to solve the incompressible Navier-Stokes equations, they may be useful here as well. For example, one could imagine a method where an approximation to \(p^{n+\frac{1}{2}}\) is used to solve for a nondivergence-free \(u^*\) approximation to the velocity field \(u^{n+1}\), which is then projected onto the space of divergence-free vector fields to obtain \(u^{n+1}\). To our knowledge, no one has tried such methods. We explore such possibilities in Chapter 6.

5.3.3 Krylov Solvers

The double Schur complement equations, (5.15), (5.16), (5.20), and (5.21) are all in the standard form \(A x = b\), to which we can apply Krylov subspace methods. Once any of those four equations is solved, the solution can be used to solve for the other variables, typically also with a Krylov subspace solver. For example, if
(5.20) is solved for $\nabla^h p^{n+\frac{1}{2}}$, then the result can be plugged into (5.17) to solve for $u^{n+1}$, which can in turn be plugged into (5.3) to solve for $X^{n+1}$.

The double Schur complement $X$ equation, equation (5.15), is

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

(5.30)

This was solved by Mayo and Peskin in [23] with a conjugate gradient squared solver, using a preconditioner of $I - \lambda A_f$. As with their fixed point scheme (see Section 5.3.1), $\lambda$ was defined as

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

(5.31)

Unfortunately, only cursory results were given. Mori and Peskin [27] also used the double Schur complement $X$ equation as the basis for a solver, using GMRES as their Krylov method. They also mention trying Biconjugate Gradient Stabilized, Conjugate Gradient Squared, and TFQMR iterations, but found GMRES to be the most efficient. They note that a preconditioner is essential for efficiency, and present one similar to the Mayo and Peskin preconditioner but that also accounts for the fact that they have thick immersed boundaries and additional boundary mass.

The nonsymmetric double Schur complement $u$ equation, equation (5.16), is

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

(5.32)

This equation was also derived in Mori and Peskin [27]. They could not find a simple way to form an efficient preconditioner for this equation, and so focused on the double Schur complement $X$ equation.

The double Schur complement $p$ equation, equation (5.20), is

$$
\nabla^h \cdot A^{-1} \nabla^h p^{n+\frac{1}{2}} = \nabla^h \cdot A^{-1} \left[ \frac{\rho}{\Delta t} (2I - A) u^n - \rho \left[ (u \cdot \nabla) u \right]^{n+\frac{1}{2}} + S A_f X^n \right].
$$

(5.33)
We are not aware of anyone having used this equation. It has some interesting properties, however. Since $A \to I$ as $\Delta t \to 0$, this looks like a generalized Poisson equation. Because $A$ is symmetric and positive definite, $A^{-1}$ is as well. Because $\nabla^h$ and $\nabla^h \cdot$ are adjoints (at least for uniform Cartesian grids and standard spatial discretizations), $\nabla^h \cdot A^{-1} \nabla^h$ is symmetric and positive definite (excluding the subspace of constants). This looks like an equation to which conjugate gradient could be applied, though it would require an inner solver for every $A^{-1}$ evaluation, which may make it expensive.

The symmetric double Schur complement $u$ equation, equation (5.21), is

$$
\mathbb{P} A \mathbb{P} u^{n+1} = \mathbb{P} \left[ (2I - A) u^n - \Delta t \left[ (u \cdot \nabla) u \right]^{n+\frac{1}{2}} + \frac{\Delta t}{\rho} S A_f X^n \right].
$$

(5.34)

This equation has also not appeared in the literature, but it too has some interesting properties. Because $\mathbb{P}$ is self-adjoint, $\mathbb{P} A \mathbb{P}$ is symmetric and positive definite on the subspace of divergence-free vector fields. Thus, conjugate gradient could be applied to this equation as well. Further, $\mathbb{P} A \mathbb{P}$ need not be applied for every matrix vector product in the conjugate gradient calculations; applying $\mathbb{P} A$ will be sufficient. The reason for this is that divergence-free vector fields are closed under addition and scalar multiplication, and the fact that the CG algorithm only performs additions, subtractions, multiplication by scalars, and applications of $\mathbb{P} A$ on the vectors it is given.
CHAPTER 6

EFFICIENCY — PROJECTION METHODS

In Section 5.3.2, an unexplored possibility for solving the implicit system of equations derived in Chapter 4 was presented: using projection methods to solve

\[ Au^{n+1} + \frac{\Delta t}{\rho} \nabla^h p^{n+\frac{1}{2}} = (2I - A) u^n - \Delta t[(u \cdot \nabla)u]^{n+\frac{1}{2}} + \frac{\Delta t}{\rho} SA_f X^n \]  
\[ \nabla^h \cdot u^{n+1} = 0, \]  

(6.1)

(6.2)

where \( A = I - \frac{\nu}{2} \Delta^h - \frac{\Delta^2}{4\rho} SA_f S^* \). This system of equations was derived by applying a Schur complement to eliminate \( X^{n+1} \). In this chapter we explore this approach.

We begin with an overview of the theory behind projection methods for solving equations of the form

\[ \rho u_t + \rho(u \cdot \nabla) u = -\nabla p + \mu \Delta u + f(u) \]  
\[ \nabla \cdot u = 0, \]  

(6.3)

(6.4)

which can represent both the incompressible Navier-Stokes equations and the IB equations. For the standard Navier-Stokes case, \( f \) is an explicit function of \( x \) and \( t \), while for the IB case, \( f \) is also a function of \( u \). We then cover important properties of the IB equations that affect the feasibility of projection methods, derive the splitting error of a projection method, introduce some new variants to improve the method, perform tests that demonstrate that these methods are not competitive, and include alternative methods suggested by our analysis.

We also include some further generalizations from the literature, listing their potential problems and advantages so that others who want to investigate projection method modifications further need not duplicate our efforts.
6.1 Basic Theory

The basic idea behind projection methods is to rewrite (6.3) in the form of a Hodge Decomposition

\[ \mathbf{w} + \nabla p = -\rho (\mathbf{u} \cdot \nabla) \mathbf{u} + \mu \Delta \mathbf{u} + \mathbf{f} \]  \hspace{1cm} (6.5)

where

\[ \mathbf{w} = \rho \frac{\partial}{\partial t} \mathbf{u}. \]  \hspace{1cm} (6.6)

The reason for writing (6.3) as a Hodge Decomposition is that it suggests a possible numerical method for solving the equations. Following the constructive proof of the Hodge Decomposition, one first takes the divergence of equation (6.5) to get a Poisson equation for the pressure

\[ \nabla \cdot \nabla p = \nabla \cdot (-\rho (\mathbf{u} \cdot \nabla) \mathbf{u} + \mu \Delta \mathbf{u} + \mathbf{f}), \]  \hspace{1cm} (6.7)

after which \( \mathbf{w} \) can be obtained via

\[ \mathbf{w} = -\rho (\mathbf{u} \cdot \nabla) \mathbf{u} + \mu \Delta \mathbf{u} + \mathbf{f} - \nabla p. \]  \hspace{1cm} (6.8)

There are two interesting things about using this decomposition as the basis of a numerical method: with an implicit discretization the vector field being decomposed (the right-hand side of (6.5)) has unknowns in it, and this decomposition enforces \( \nabla \cdot \mathbf{w} = 0 \) rather than \( \nabla \cdot \mathbf{u} = 0 \). However, \( \nabla \cdot \) and \( \rho \frac{\partial}{\partial t} \) commute (since \( \rho \) is assumed to be constant) so \( \nabla \cdot \mathbf{w} = 0 \) implies \( \nabla \cdot \mathbf{u} = 0 \) provided \( \nabla \cdot \mathbf{u} = 0 \) initially.

Having unknowns in the vector field being decomposed would make it difficult to perform the decomposition via equations (6.7) and (6.8); it would mean solving a Poisson equation where the right-hand side is unknown. The simplest way to work around this is to use an explicit time discretization, in order to ensure the vector field being decomposed consists entirely of known values from the previous time-level. Using the Hodge Decomposition as the basis of a numerical scheme with an implicit or mixed explicit/implicit discretization requires a slightly different strategy in order to work around this problem.
One way of handling implicit or partially implicit time discretizations is simply to rewrite the decomposition so that only known values are being decomposed. For example, typical Navier-Stokes and IB solvers handle the viscous terms implicitly and everything else explicitly. Thus, we could write the decomposition as

$$\left( \rho \frac{\partial}{\partial t} - \mu \Delta \right) \mathbf{u} + \nabla p = -\rho (\mathbf{u} \cdot \nabla) \mathbf{u} + \mathbf{f}. \quad (6.9)$$

More general temporal discretization choices could be handled by writing decompositions of the form

$$\mathcal{A} \mathbf{u} + \nabla p = \mathbf{r}. \quad (6.10)$$

However, in contrast to the case when $\mathcal{A} = \rho \frac{\partial}{\partial t}$, more general choices of $\mathcal{A}$ may not commute with $\nabla \cdot$. There are also special cases where $\mathcal{A}$ may commute with $\nabla \cdot$ as a continuous operator, but where the discretized operator does not commute with $\nabla^h \cdot$. One example is the choice of $\mathcal{A}$ in (6.9), where the lack of commutativity of the corresponding discrete operators occurs only at the boundary of the domain. This lack of commutativity of $\nabla \cdot$ and $\mathcal{A}$ (or their corresponding discretizations) means that the numerical method for solving (6.10) becomes a bit more complicated. Many different methods exist based on differences in how this lack of commutativity is handled.

### 6.1.1 Subtleties in Meaning of Commutativity

We have discussed $\mathcal{A}$ commuting with $\nabla \cdot$, but strictly speaking this does not make sense. $\mathcal{A}$ maps vector fields to vector fields, but $\nabla \cdot$ operates on vector fields to produce scalar fields. Having these operators “commute” can make sense only for very specialized forms of $\mathcal{A}$ where it can be thought of as a scalar operator that acts componentwise, i.e., where we can abuse notation to write $\mathcal{A} \mathbf{u} = (\mathcal{A}u_1, \mathcal{A}u_2)$. A good example of such an operator is $\Delta$; in fact, the most common way of writing the Navier-Stokes equations fosters this confusion because $\Delta$ is defined for both scalar and vector fields so that we can write $\Delta \mathbf{u} = (\Delta u_1, \Delta u_2)$.

An example of $\mathcal{A}$ that does “commute” with $\nabla \cdot$ (and hence also $\nabla$) is $\mathcal{A} = \rho \frac{\partial}{\partial t} - \mu \Delta$ (except near boundaries). Of course, not all $\mathcal{A}$ that can be thought of
as both a scalar and a vector operator will commute with \( \nabla \cdot \); the choice of \( \mathcal{A} \) arising from implicit IB discretizations assuming a linearly elastic material with zero resting length is one such example.

### 6.1.2 Common Assumptions

There are some common assumptions shared by different projection methods that are worth noting. Because the Navier-Stokes equations are typically solved with an implicit treatment of the viscous terms and an explicit treatment of all other terms, many projection methods are designed assuming \( \mathcal{A} = \rho \frac{\partial}{\partial t} - \mu \Delta \). Because of the prevalence of this particular case, it is worth looking at a common discretization of equation (6.10) for this case:

\[
(\frac{\rho}{\Delta t} I - \frac{\mu}{2} \Delta^h) u^{n+1} + \nabla^h p^{n+\frac{1}{2}} = \left( \frac{\rho}{\Delta t} I + \frac{\mu}{2} \Delta^h \right) u^n + \mathbf{r}^n. \tag{6.11}
\]

Note here that part of the discrete form of \( \mathcal{A} \) has moved to the right-hand side since it involves the known quantity \( u^n \). The remaining part, \( \frac{\rho}{\Delta t} I - \frac{\mu}{2} \Delta^h \), which we call \( A \), has a number of useful properties: it is a small perturbation of a scaled identity matrix, it is symmetric and positive definite, and it has eigenvalues greater than or equal to \( \rho/\Delta t \). Further, assuming a Cartesian grid and standard second order spatial discretizations, \( \Delta^h \) “commutes” with \( \nabla^h \), in the interior of the domain. Since the identity matrix also commutes with everything, ignoring lack of commutativity of \( A \) and \( \nabla^h \) will introduce only a small error at the boundary of the domain. Projection methods often rely heavily on these properties (though the practitioners may not be aware that they are doing so, let alone how they are doing so).

### 6.1.3 Velocity-correction Projection Methods

If we were simply to ignore the lack of commutativity of \( \mathcal{A} \) and \( \nabla \cdot \), we could approximately solve equation (6.10) by first performing the Hodge Decomposition

\[
\mathbf{w} + \nabla p = \mathbf{r}, \tag{6.12}
\]

and then solving

\[
\mathcal{A} \mathbf{u} = \mathbf{w}. \tag{6.13}
\]
This method is known as a (nonincremental) velocity-correction projection method. It was introduced by Orszag et al. [29] and Karniadakis et al. [16]. In general, this method results in a solution satisfying $\nabla \cdot \mathbf{A} \mathbf{u} = 0$ and not satisfying the actual constraint $\nabla \cdot \mathbf{u} = 0$. However, if $\nabla \cdot$ and $\mathbf{A}$ “almost” commute, as is the case under the common assumptions in Section 6.1.2, the errors in both the solution and the incompressibility constraint are small.

This method is often improved by making an approximation to the unknown viscous terms (we are assuming the special case $\mathbf{A} = \rho \frac{\partial}{\partial t} - \mu \Delta$ outlined in Section 6.1.2), and then subtracting this approximation to the viscous terms from both sides. For $\mathbf{A} = \rho \frac{\partial}{\partial t} - \mu \Delta$, this yields the system

$$\left( \rho \frac{\partial}{\partial t} - \mu \Delta \right) \mathbf{u} + \mu \nabla \mathbf{u} + \nabla p = -\rho (\mathbf{u} \cdot \nabla) \mathbf{u} + \mathbf{f} + \mu \nabla \mathbf{u}, \quad (6.14)$$

where $\mu \nabla \mathbf{u}$ represents an approximation to the unknown viscous terms. This system is solved in two steps, first by performing the Hodge Decomposition

$$\mathbf{w} + \nabla p = -\rho (\mathbf{u} \cdot \nabla) \mathbf{u} + \mathbf{f} + \mu \nabla \mathbf{u}, \quad (6.15)$$

and then by solving

$$\left( \rho \frac{\partial}{\partial t} - \mu \Delta \right) \mathbf{u} = \mathbf{w} - \mu \nabla \mathbf{u}. \quad (6.16)$$

A good approximation to the viscous terms makes (6.16) of the form $\rho \frac{\partial}{\partial t} \mathbf{u} \approx \mathbf{w}$. Since $\nabla \cdot \mathbf{w} = 0$ and $\rho \frac{\partial}{\partial t}$ commutes with $\nabla \cdot$, it turns out that this modification results in a velocity field that is closer to being divergence-free than one would obtain with the nonincremental velocity-correction scheme. This modification results in a method known as the standard incremental velocity-correction scheme.

The standard incremental velocity-correction scheme could also be made into an iterative method, with the resulting velocity used to obtain a better approximation to the viscous terms in the next iteration. However, it turns out that with the assumptions in Section 6.1.2 (that $\nabla \cdot$ and $\mathbf{A}$ almost commute), a single step of such an iterative scheme is typically sufficient. A further improvement to the standard incremental velocity-correction scheme can be obtained as well; see [15] for details.
6.1.4 Pressure-correction Projection Methods

The velocity-correction projection methods are based on doing a Hodge Decomposition of (6.10) first, then inverting $A$. Pressure-correction projection methods can be thought of as methods that reverse this order. In the first step, the pressure and incompressibility constraint are ignored in solving

$$A w = r. \quad (6.17)$$

This is followed by projecting $w$ onto the space of divergence-free vector fields with a Hodge Decomposition,

$$u + \nabla q = w. \quad (6.18)$$

However, the Hodge Decomposition is usually written in the form

$$\rho u_t + \nabla p = \rho w_t, \quad (6.19)$$

which provides a divergence-free $u$ since $\nabla \cdot$ and $\rho \frac{\partial}{\partial t}$ commute. This alternate form and choice of pressure is based on the common assumption $A = \rho \frac{\partial}{\partial t} - \mu \Delta$ from Section 6.1.2. Comparing the equation we want to solve,

$$\rho u_t + \nabla p = -\rho(u \cdot \nabla)u + f + \mu \Delta u, \quad (6.20)$$

with a simple rewrite of (6.17),

$$\rho w_t = -\rho(u \cdot \nabla)u + f + \mu \Delta w, \quad (6.21)$$

Equation (6.19) follows if the difference between the $\mu \Delta u$ and $\mu \Delta w$ terms is ignored, yielding the standard pressure-correction method. This difference introduces an error which results in the momentum equation not being satisfied exactly. However, if $A = \rho \frac{\partial}{\partial t} - \mu \Delta$, the error in the computed velocity and in the momentum equation may be small, as will be discussed in more detail in Section 6.3.

There are some interesting contrasts between the velocity-correction and pressure-correction projection methods. The velocity-correction projection method obtains a solution that satisfies the momentum equation but not the incompressibility
constraint, whereas the pressure-correction projection method obtains a solution that satisfies the incompressibility constraint but not the momentum equation. When $A$ and $\nabla \cdot$ commute, the velocity-correction method obtains the exact solution of (6.10), but the pressure-correction method does not. The reason for this is that in the pressure-correction method, the correct pressure satisfies

$$A u + \nabla p = Aw,$$  

(subject to the constraint $\nabla \cdot u = 0$) rather than equation (6.19). (This is just (6.10) combined with (6.17).) Multiplying through by $A^{-1}$ we see that

$$u + A^{-1} \nabla p = w.$$  

(6.23)

Now, if $A$ commutes with $\nabla \cdot$ then it will also commute with $\nabla$ (since $\nabla$ and $\nabla \cdot$ are negative adjoints), which implies that

$$u + \nabla A^{-1} p = w.$$  

(6.24)

We can contrast the values of $p$ obtained by (6.24) and (6.19) by comparing them to $q$ from (6.18). Equation (6.19) gives $p = \rho \frac{\partial q}{\partial t}$, whereas the correct pressure given by (6.24) is $p = Aq$. Therefore, when $\nabla \cdot$ and $A$ commute, the pressure-correction method yields the wrong pressure even though the velocity field is correct.

When $\nabla \cdot$ and $A$ commute, a more accurate pressure in the sense of solving (6.10) exactly can be obtained by setting $p = Aq$. This more accurate pressure step was pointed out by Brown et al. in [6] and is known as the rotational (nonincremental) pressure-correction projection method. It is worth noting that when $\nabla \cdot$ and $A$ do not commute, $A^{-1} \nabla p$ from (6.23) may not be a gradient field, degrading the accuracy of the pressure-correction method.

Just as with the velocity-correction projection method, there is an incremental form of the pressure-correction projection method. We can get a more accurate solution by approximating the pressure gradient and subtracting it from both sides of the momentum equation,

$$A u + \nabla p - \hat{\nabla} p = r - \hat{\nabla} p.$$  

(6.25)
This is followed by solving

$$\mathcal{A} \mathbf{w} = \mathbf{r} - \nabla \hat{p}, \quad (6.26)$$

after which the (standard) incremental pressure-correction method performs the Hodge Decomposition

$$\rho \mathbf{u}_t + \nabla \hat{q} = \rho \mathbf{w}_t, \quad (6.27)$$

and sets $\nabla p = \nabla \hat{p} + \nabla \hat{q}$. Again, note how (6.27) ignores all parts of $\mathcal{A}$ other than $\rho \frac{\partial}{\partial t}$, just like (6.19).

The rotational and incremental methods can be combined by replacing (6.27) with

$$\mathcal{A} \mathbf{u} + \nabla \hat{q} = \mathcal{A} \mathbf{w}, \quad (6.28)$$

which, if that $\mathcal{A}$ and $\nabla$ commute, becomes

$$\mathbf{u} + \nabla q = \mathbf{w}, \quad (6.29)$$

where $q = \mathcal{A}^{-1} \hat{q}$. After performing the Hodge Decomposition (6.29), the rotational incremental pressure-correction projection method involves setting $\nabla p = \nabla \hat{p} + \nabla \mathcal{A} q$. Both the standard and the rotational forms of the incremental projection method can be embedded into an iterative method, using the resulting pressure from one step to obtain a better approximation $\nabla \hat{p}$ for the following iteration. As with the incremental velocity-correction projection method, if the assumptions in Section 6.1.2 are valid (namely that the discrete form of $\mathcal{A}$ is a “small” perturbation of a scaled identity operator and $\mathcal{A}$ “almost” commutes with $\nabla \cdot$), then a single step of such an iterative method is typically sufficient.

### 6.1.5 Alternative Decomposition

When $\mathcal{A}$ and $\nabla \cdot$ (or $\nabla$) do not even approximately “commute” then both the velocity-correction and pressure-correction projection methods can yield large splitting errors. Stepping back a bit, the reason for the large splitting errors is
that these methods rely on commutativity in order to be able to use the Hodge Decomposition as the basis of a numerical scheme. The Hodge Decomposition is useful for decomposing vectors \( \mathbf{\hat{r}} \) into the form

\[ \mathbf{u} + \nabla p = \mathbf{\hat{r}}, \tag{6.30} \]

with \( \nabla \cdot \mathbf{u} = 0 \). More concretely, the Hodge Decomposition states that any vector field can be decomposed uniquely and orthogonally into a divergence-free vector field and a gradient field, i.e.,

\[ \forall \mathbf{\hat{r}} \exists! \mathbf{w}, \nabla \psi \]

with \( \nabla \cdot \mathbf{w} = 0 \) and \( \langle \mathbf{w}, \nabla \psi \rangle = 0 \)

such that \( \mathbf{\hat{r}} = \mathbf{w} + \nabla \psi \).

We are glossing over boundary conditions here; the Hodge Decomposition can be stated for various boundary conditions, including periodic. This decomposition is a beautiful result, especially since it has such a simple constructive proof. It is common to define \( P \) to be the orthogonal (in \( L^2 \)) projection onto the space of divergence-free fields from this Hodge Decomposition, i.e., \( \mathbf{w} = P \mathbf{\hat{r}} \). As per the constructive proof of the Hodge Decomposition (outlined in Section 2.3.2), \( P \) is given by \( I - \nabla(\nabla \cdot)^{-1} \nabla \cdot \). This decomposition and its proof can also easily be generalized to other orthogonal subspaces.

Rather than decomposing vectors into the form (6.30), it would be much more useful to decompose them into the form

\[ \mathcal{A}\mathbf{u} + \nabla p = \mathbf{\hat{r}}, \tag{6.31} \]

with \( \nabla \cdot \mathbf{u} = 0 \). Such a decomposition would eliminate the commutativity problems appearing in the velocity-correction and pressure-correction schemes. It turns out that a more general decomposition can be defined; this decomposition states that for any positive definite operator \( \mathcal{C} \) that

\[ \forall \mathbf{\hat{r}} \exists! \mathbf{w}, \nabla \psi \]

with \( \nabla \cdot \mathbf{Cw} = 0 \) and \( \langle \mathbf{w}, \nabla \psi \rangle_{\mathcal{C}} = \langle \mathbf{Cw}, \nabla \psi \rangle = 0 \)

such that \( \mathbf{\hat{r}} = \mathbf{w} + \nabla \psi \).
The proof that this new decomposition exists is basically identical to the one for the Hodge Decomposition (which is outlined in 2.3.2). As with the Hodge Decomposition, we define $P_C$ to be the projection onto the space orthogonal to gradients in this new decomposition, i.e., $w = P_C \tilde{r}$. Thus $P_T = P$ is the Hodge Decomposition projection operator. Similar to $P$, $P_C$ is given by $I - \nabla(\nabla \cdot C \nabla)^{-1} \nabla \cdot C$.

This new decomposition with the choice $C = A^{-1}$ solves equation (6.31) exactly, but the generalized Poisson problem involved in performing the decomposition,

$$\nabla \cdot C \nabla \psi = \nabla \cdot \tilde{r}, \tag{6.32}$$

may be too computationally expensive to solve with this choice of $C$. Noting that the pressure-correction and velocity-correction methods are based on the choice $C = I$, which is a rather crude approximation to $A^{-1}$, it may be possible to use a variant of the pressure-correction and velocity-correction methods based on the new decomposition with a choice of $C$ corresponding to some approximation of $A^{-1}$.

As pointed out in the previous sections, the incremental velocity-correction and pressure-correction methods can be iterated. Both methods are based on making an approximation to a subset of the terms (namely, either the pressure gradient or viscous terms), performing a couple of steps (inverting $\mathcal{A}$ and doing a Hodge Decomposition) in which commutation errors are ignored, and then using what is hopefully a better approximation to the solution to make a better approximation to either the viscous terms or the pressure gradient in order to perform the next iteration. When $\nabla \cdot$ and $\mathcal{A}$ commute or “almost” commute, then only a single iteration may be necessary in order to obtain a sufficiently accurate approximation to the solution of (6.10). However, in the IB case the commutation error may be large. This means that a single iteration will not be sufficient, and also that these iterative methods may converge slowly or even fail to converge entirely. Choosing $C$ to be some approximation to $\mathcal{A}^{-1}$ better than $I$ will reduce these commutation errors, and may result in an efficient method if the choice of $C$ is such that (6.32) is not much more expensive than a standard Poisson solve. This will be discussed in more detail in Sections 6.4 and 6.5.
6.2 Important Properties of IB Equations

The IB equations, specifically in the Schur complement form (6.1)-(6.2), have a number of properties that strongly influence the effectiveness of various solution methods. This section explores those properties.

As pointed out in [22], the velocity and pressure which solve the IB equations are not \( C^\infty \); they are piecewise-\( C^\infty \). While the velocity is continuous, its derivatives are not, and the pressure is discontinuous. While these discontinuities are “smeared-out” when solving via the IB method with the regularized delta functions, extrapolations of these variables still suffer from low order accuracy near the interface (zeroth order in the case of the pressure). This is important because projection methods (the incremental forms, anyway) are based on making an approximation to a subset of the terms in the momentum equation (such as the viscous term or pressure gradient) using an extrapolation of previous values of the velocity or the pressure. The accuracy and efficiency of the method relies on (among other factors) getting an accurate approximation to these terms, but the lack of smoothness in the velocity and pressure will cause the accuracy of these approximations to suffer, especially in the case of the pressure.

As already noted in Section 5.3, \( A = A_1 + A_2 = (I - \frac{\nu \Delta t}{2} \Delta^k) + \left( -\frac{\Delta t^2}{4\rho} SA_f S^* \right) \) is symmetric positive definite, with all its eigenvalues greater than or equal to 1, and \( \lim_{\Delta t \to 0} A = I \). If we were discretizing the incompressible Navier-Stokes equations instead of the IB equations, then \( A_1 \) would be used in the place of \( A \) in equation (6.1). \( A_2 \) represents the elastic force from the immersed boundary and scales linearly with the elastic tension parameter, \( \gamma \). Typically, \( A_2 \) is the dominant term because \( \gamma \) is large. In both the Navier-Stokes and IB cases, we can write \( A = I + \Delta t A_t \) since both \( A_1 - I \) and \( A_2 \) have a multiplicative factor of \( \Delta t \). Further, \( A_t \) is symmetric and positive semidefinite in both cases. However, we cannot assume that \( A_t \) is \( O(1) \) in the IB case. In fact, we often cannot even assume that \( \Delta t A_t \) is \( O(1) \) since the presence of the large elastic tension may require a restrictively small a timestep for this to be true.
Just as with $A$, we will occasionally abuse the notation for $A$ and use it both as an operator on vector grid quantities and as an operator on scalar grid quantities (as discussed in Section 6.1.1). Thus, in 2D problems, we could equivalently write either $Au$ or $(Au_1, Au_2)$ and both expressions would make sense and mean the same thing, despite the obvious difference in size between $u$ and its individual components. Such abuse of $A$ only makes sense when $A$ is block diagonal, with each block being equal. This occurs, for example, when zero resting length springs are used in the constitutive model (because of the special form of $A_f$ that occurs with cancellation of the $\| \frac{\partial X}{\partial s} \|$ terms), but may not be valid for other cases. When $A$ does not make sense as a scalar operator, the choice of projection methods may be limited, as will be discussed later.

Some properties of $A$ are shown graphically in Figures 6.1-6.3. All were computed with $h = 1/32$, $\gamma = 10^2$, $\Delta t = 5 \times 10^{-2}$, $\nu = 0.01$, and 200 IB points. The matrix structure of $A$ can be seen in Figure 6.1, which also shows the block-diagonal componentwise behavior of $A$. In Figure 6.2, we give a measure of the diagonal dominance of $A$. For each row, we computed the diagonal of $A$ divided by the sum of the absolute values of the off-diagonal entries of that row. Figure 6.2 results from sorting these values. In Figure 6.3, we plot the eigenvalues of $A$ sorted by magnitude. Note that the long gradual initial increase corresponds to the eigenvalues of $A_1$ (since $A_2$ has a large null space), and the sudden increase corresponds predominantly to nonzero eigenvalues of $A_2$.

### 6.3 Splitting Error

In this section we derive and analyze the splitting error involved in using the fractional step approach of the pressure-correction projection methods. We use $G$ to denote the matrix operator for $\nabla^h$ and $G_T$ to denote the matrix operator for $-\nabla^h$, and we assume periodic boundary conditions for simplicity so that there are no additional contributions to the equations from boundary conditions. This allows us to write (6.1)-(6.2) in the algebraic form
Figure 6.1. Sparsity pattern of $A$; immersed boundary in initial elliptical configuration used in tests from Chapters 3 and 4.
Figure 6.2. Measure of diagonal dominance of the rows of A. Each point on the y-axis measures an entry on the diagonal of A divided by the sum of the absolute values of the off-diagonal entries of that row. Thus a value greater than 1 indicates diagonal dominance for that row.

Figure 6.3. Sorted eigenvalues of A. Note that the spike at the right corresponds to the effect of $A_2$. 
\[
\begin{pmatrix}
A & \frac{\Delta t}{p} G \\
G^T & 0
\end{pmatrix}
\begin{pmatrix}
u \\
p
\end{pmatrix} =
\begin{pmatrix}
r \\
-z
\end{pmatrix}.
\] (6.33)

We refer to the first row of (6.33) as the momentum equation, and the second row as the constraints equation. Note here that \( z = 0 \) for discretizations of the standard Navier-Stokes or IB equations.

Our results generalize to cases where \( G \) and \( G^T \) are something other than gradient and divergence operators. For example, in the work of Taira and Colonius\[39\], there are two constraints on the motion of the fluid—the standard incompressibility condition on the fluid plus prescribed motion of a structure embedded within the fluid. In Taira and Colonius, \( p \) represents more variables than the pressure; just as pressure behaves as a Lagrange multiplier to enforce the incompressibility constraint, they also use the force on parts of the immersed boundary as a Lagrange multiplier to enforce the prescribed motion of those parts of the immersed boundary. Thus, \( p \) may represent more than one variable that acts as a Lagrange multiplier, and \( G \) and \( G^T \) may represent various operators on those Lagrange multipliers. Such cases add to the violation of “commutativity” of \( A \) and \( G^T \) and cause “commutativity” not to make sense. Because we concentrate on the case that \( p \) is pressure and that \( G \) and \( G^T \) are gradient and divergence operators respectively, we often refer to \( p \) as pressure, to \( G \) and \( G^T \) as gradient and divergence operators, or to the constraints equation as the continuity equation or incompressibility constraint. However, our analysis does generalize and has some interesting ramifications that we will occasionally comment on. Extra details about more generalized \( G, G^T, \) and \( p \) can be found in Appendix A.2.

We concentrate on pressure-correction projection methods because we are aware of far more modifications and generalizations of these methods in the literature than the velocity-correction methods, though we will comment more on the latter in Section 6.7. The pressure-correction projection methods for solving (6.33) are of the form
Equation (6.34) represents the momentum solve ignoring the constraints, while (6.35)-(6.36) represent the projection step to enforce the constraints and (6.37) is the pressure update. Equations (6.35) and (6.36) are referred to as the Poisson solve and velocity correction steps, respectively. For the nonincremental pressure-correction methods, \( \hat{p} = 0 \), whereas for the incremental versions \( \hat{p} \) is some approximation to the pressure (such as \( p^{n-1/2} \)). In the nonrotational pressure-correction methods \( Y = I \), whereas \( Y = A \) for the rotational versions.

We seek to derive an expression for the splitting error of this fractional step scheme. We begin by writing (6.34) and (6.35) in system form

\[
\begin{pmatrix}
A & 0 \\
-G^T & \frac{\Delta t}{\rho} G^T G
\end{pmatrix}
\begin{pmatrix}
u^* \\
\phi
\end{pmatrix} = \begin{pmatrix}
r \\
z
\end{pmatrix} + \begin{pmatrix}
-\frac{\Delta t}{\rho} G \hat{p} \\
0
\end{pmatrix},
\]

and do likewise for (6.36) and (6.37) to obtain

\[
\begin{pmatrix}
I & -\frac{\Delta t}{\rho} G \\
0 & Y
\end{pmatrix}
\begin{pmatrix}
u^* \\
\phi
\end{pmatrix} = \begin{pmatrix}
u \\
p
\end{pmatrix} + \begin{pmatrix}0 \\
-\hat{p}
\end{pmatrix}.
\]

Solving (6.39) for (\( \nu^*, \phi \)) yields

\[
\begin{pmatrix}
u^* \\
\phi
\end{pmatrix} = \begin{pmatrix}
I & \frac{\Delta t}{\rho} G Y^{-1} \\
0 & Y^{-1}
\end{pmatrix} \left[ \begin{pmatrix}
u \\
p
\end{pmatrix} + \begin{pmatrix}0 \\
-\hat{p}
\end{pmatrix} \right],
\]

which we can plug into equation (6.38) to obtain

\[
\begin{pmatrix}
A & 0 \\
-G^T & \frac{\Delta t}{\rho} G^T G
\end{pmatrix}
\begin{pmatrix}
I & \frac{\Delta t}{\rho} G Y^{-1} \\
0 & Y^{-1}
\end{pmatrix}
\left[ \begin{pmatrix}
u \\
p
\end{pmatrix} + \begin{pmatrix}0 \\
-\hat{p}
\end{pmatrix} \right]
= \begin{pmatrix}
r \\
z
\end{pmatrix} + \begin{pmatrix}
-\frac{\Delta t}{\rho} G \hat{p} \\
0
\end{pmatrix}.
\]
Multiplying matrices and bringing known quantities to the right-hand side we obtain
\[
\begin{pmatrix}
A & \frac{\Delta t}{\rho} AGY^{-1} \\
-G^T & 0
\end{pmatrix}
\begin{pmatrix}
u \\
p
\end{pmatrix}
= \begin{pmatrix}
r \\
z
\end{pmatrix} + \begin{pmatrix}
\frac{-\Delta t}{\rho} G\hat{p} \\
0
\end{pmatrix} - \begin{pmatrix}
A & \frac{\Delta t}{\rho} AGY^{-1} \\
-G^T & 0
\end{pmatrix} \begin{pmatrix}
0 \\
-\hat{p}
\end{pmatrix},
\]
which simplifies to
\[
\begin{pmatrix}
A & \frac{\Delta t}{\rho} AGY^{-1} \\
-G^T & 0
\end{pmatrix}
\begin{pmatrix}
u \\
p
\end{pmatrix}
= \begin{pmatrix}
r \\
z
\end{pmatrix} + \frac{\Delta t}{\rho} \begin{pmatrix}
-G + AGY^{-1} \\
0
\end{pmatrix} \hat{p}.
\]
Comparing (6.43) to (6.33), we see that we correctly solved the constraints equation, but the momentum equation has been modified with the following splitting error
\[
\frac{\Delta t}{\rho} G\hat{p} - \frac{\Delta t}{\rho} AGY^{-1} p - \frac{\Delta t}{\rho} (G - AGY^{-1})\hat{p}
= \frac{\Delta t}{\rho} (G - AGY^{-1}) (p - \hat{p}).
\]
If we can assume that \(G - AGY^{-1}\) and \(p - \hat{p}\) are \(O(1)\), then this splitting error is \(O(\Delta t)\). It is reasonable to assume that \(p\) is \(O(1)\) and that we use an approximation \(\hat{p}\) to \(p\) no worse than 0. Also, given that \(\|A^{-1}\| \leq \|I\| = 1\) in both the IB and Navier-Stokes cases (since \(A\) is symmetric with eigenvalues greater than or equal to 1), we have that \(\|Y^{-1}\| \leq 1\) for these methods when \(Y = I\) or \(Y = A\). Because of these reasons, the splitting error (6.44) should be \(O(\Delta t)\) whenever \(\|A\|\) is \(O(1)\). We can show that the splitting error is higher order than \(O(\Delta t)\) under further assumptions, discussed below. Note, however, that while \(\|A\|\) is \(O(1)\) in the Navier-Stokes case, this is not true in the IB case.

If \(\hat{p}\) is an \(O(\Delta t)\) approximation to \(p\), i.e., if \(p = \hat{p} + \Delta t p_b\), then the splitting error is
\[
\frac{\Delta t^2}{\rho} (G - AGY^{-1}) p_b,
\]
which means that the incremental forms of the pressure-correction method can be more accurate than the nonincremental forms (which use \(\hat{p} = 0\)). However, the
incremental forms require a good approximation to $p$ in order for this to be valid. While that is easily obtained for the Navier-Stokes case (by using e.g., $\hat{p} = p^{n-1/2}$), the discontinuities in the pressure in the IB case make this difficult to achieve.

In the nonrotational forms of the pressure-correction projections, where $Y = I$, the splitting error simplifies further to

$$\frac{\Delta t}{\rho} (I - A) G (p - \hat{p}).$$

(6.46)

For the Navier-Stokes case, $A$ is a small perturbation of the identity and is of the form $I - \kappa \Delta t L$. This makes the splitting error $-\frac{\kappa \Delta t^2}{\rho} L G (p - \hat{p})$, which is diffusive and tends to aid stability [30]. Further, since $L$ and $G$ “almost” commute, this error is predominantly a gradient and the error in the solution is mostly contained in the pressure.

In the IB case, since $A_2$ (the dominant part of $A$) scales with $\gamma$ and is often quite large, not only is $A$ not a small perturbation of the identity, it is not even $O(1)$. Thus, the splitting error is quite large. Further, since $I - A$ does not even approximately commute with $G$ in the IB case, this error will be reflected in both the velocity and the pressure.

In the rotational forms of the pressure-correction, i.e., where $Y = A$, the splitting error is

$$\frac{\Delta t}{\rho} (G - AGA^{-1}) (p - \hat{p}).$$

(6.47)

When $A$ and $G$ commute, which happens for the Navier-Stokes equations with periodic boundary conditions, the splitting error is 0. If $A$ and $G$ nearly commute, which occurs in the Navier-Stokes case, then $G - AGA^{-1} \approx 0$ and thus (6.47) is nearly 0. Even in the IB case, where $A$ and $G$ do not commute, this splitting error looks interesting. We expect the dominant part of the splitting error, (6.44), to be $\frac{\Delta t}{\rho} AGY^{-1} (p - \hat{p})$ in the IB case. So we are interested in $\|AGY^{-1}\|$. Thus, $\|AGA^{-1}\| \leq \|AG\| \|A^{-1}\| \leq \|AG\|$, from which we suspect that (6.47) will generally be smaller than (6.46).

For some problems, the choice of $Y = A$ is inapplicable; it simply may not make sense. $Y$ and $A$ are both square matrices, but they operate on different
variables; $Y$ operates on $p$ while $A$ operates on $u$. So it is only under special circumstances that the abuse of notation $Y = A$ makes sense. In particular, $A$ needs to be block diagonal and to perform the same operation on each component of the grid-vector to which it is being applied (as discussed in Section 6.2), and $p$ needs to represent a single grid-scalar (i.e., representing just pressure and not a combination of Lagrange multipliers as in the work of Taira and Colonius). If these conditions are not satisfied, not only do $G$ and $A$ not "commute," the rotational forms of the pressure-correction methods do not make sense and cannot be applied.

From this splitting error analysis, we see that there are four things that make these projection methods work well for solving the incompressible Navier-Stokes equations:

- $A$ is relatively small ($O(1)$) in norm
- a good initial guess to the pressure is readily available
- $A$ is a small ($O(\Delta t)$) perturbation of the identity
- $A$ "almost" commutes with $G$

All four break down when considering the IB equations.

### 6.4 Improvements: The New Update Operator

Noting the improvement with the choice $Y = A$, particularly when $A$ and $G$ nearly commute, we tried selecting $Y = A_1$ since $A_1$ is the portion of $A$ that nearly commutes with $G$. This did not provide a significant improvement (see Appendix A.4 for details), but it led us to investigate other possibilities for the choice of $Y$.

Looking at (6.43),

$$
\begin{pmatrix}
  A & \frac{\Delta t}{\rho} AGY^{-1}
  \\
  -G^T & 0
\end{pmatrix}
\begin{pmatrix}
u
  \\
  p
\end{pmatrix}
= 
\begin{pmatrix}
r
  \\
z
\end{pmatrix}
+ \frac{\Delta t}{\rho}
\begin{pmatrix}
-G + AGY^{-1}
  \\
0
\end{pmatrix}
\hat{p}.
$$

there is no splitting error if $AGY^{-1} = G$. If $A$ and $G$ "commute," then this can be achieved by employing the rotational pressure-correction choice $Y = A$. When $G$
and $A$ do not “commute” we would like to know how to select $Y$ to make $AGY^{-1}$ as close to $G$ as possible. Towards that end we write

\begin{align}
AGY^{-1} &\approx G \
AG &\approx GY,
\end{align}

which we would like to enforce exactly. However, due to the difference in sizes of $Y$ and the lack of invertibility of $G$, this equation is only solvable in special cases. So we instead seek an approximate solution by multiplying both sides by $G^TC$ (where $C$ is any positive definite operator, discussed more below) to obtain

\begin{align}
G^TCAG &= G^TCGY \
(G^TC)^{-1}G^TCAG &= Y.
\end{align}

We define $N$ to be this particular choice for $Y$, though we generalize it in Appendix A.1.5. (Note that $G^TCG$ technically is not invertible, much like $G^TG$. However, $(G^TG)^{-1}$ is often used in standard projection methods because it is understood that only the subspace orthogonal to the nullspace (namely, constants, at least with periodic boundary conditions) is being operated on. We do the same with $G^TCG$.) This operator has many interesting properties, which we explore in this section.

### 6.4.1 Splitting Error

The splitting error of (6.43) is limited to the momentum equation. When $Y = N$, the splitting error in the momentum equation is given by

\begin{align}
\frac{\Delta t}{\rho}Gp - \frac{\Delta t}{\rho}AGN^{-1}p - \frac{\Delta t}{\rho}G\hat{p} + \frac{\Delta t}{\rho}AGN^{-1}\hat{p} \\
= \frac{\Delta t}{\rho}(G - AGN^{-1})(p - \hat{p}).
\end{align}

This splitting error is very similar to the splitting error for the “standard” pressure-correction methods (see Section A.1.1), $\frac{\Delta t}{\rho}(G - AGY^{-1})(p - \hat{p})$. We will demonstrate computationally that the splitting error with $Y = N$ is much smaller than for the standard methods ($Y = I$ or $Y = A$) in Section 6.6.
6.4.2 Relationship to the New Decomposition

The splitting error, $\frac{\Delta t}{\rho} (G - AGN^{-1})(p - \hat{p})$ has a very special form, as can be seen in the following derivation:

$$
G - AGN^{-1} = GNN^{-1} - AGN^{-1}
= G(G^TCG)^{-1}G^T CAGN^{-1} - AGN^{-1}
= (G(G^TCG)^{-1}G^T C - I)AGN^{-1}
= -PCAGN^{-1},
$$

(6.53)

where $PC$ is a discrete form of $\mathbb{P}_C$ from the new decomposition in Section 6.1.5.

6.4.3 Generalizes Pressure Update Operator

$N$ is a generalization of the update operator for the rotational scheme in cases when $A$ and $G$ do not commute. In fact, not only does it give a smaller splitting error than the choice $Y = A$ from the rotational scheme when $A$ and $G$ do not commute, it provides a choice for $Y$ that makes sense when $Y = A$ does not (e.g., when $A$ is strictly a vector operator or $G$, $G^T$, and $p$ are more generalized choices such as those from Appendix A.2).

6.4.4 Consistency of the Pressure

From the special form of the splitting error in the momentum equation derived in Section 6.4.2,

$$
-\frac{\Delta t}{\rho} PCAGN^{-1}(p - \hat{p}),
$$

we see that $u$ and $p$ satisfy

$$
Au + \frac{\Delta t}{\rho} Gp + \frac{\Delta t}{\rho} PCAGN^{-1}(p - \hat{p}) = r
$$

(6.55)

instead of

$$
Au + \frac{\Delta t}{\rho} Gp = r.
$$

(6.56)

It is interesting to note that if we apply $G^TC$ to either equation (6.55) or (6.56) then we get

$$
G^TCAu + \frac{\Delta t}{\rho} G^T CGp = G^TCr.
$$

(6.57)
Since $G^T u = 0$ (i.e., $u$ satisfies the constraints of the problem), $C = A^{-1}$ provides the exact solution for the pressure. Thus, if $C$ is a good approximation to $A^{-1}$, then $p$ will likely be a very good approximation to the real pressure. Note that $p$ satisfies (6.57) only because of the special form of the splitting error that exists due to the use of the new pressure update operator.

### 6.4.5 Postprocessing to Reduce the Splitting Error

As shown in Section 6.4.2, the splitting error $e$ is of the special form

$$e = -\frac{\Delta t}{\rho} P_C AGN^{-1}(p - \hat{p}).$$

Because of this special form of $e$, $G^T Ce = 0$, i.e., $Ce$ is a vector field that satisfies all the constraints of the problem. We can take advantage of this and, as a postprocessing step, subtract $Ce$ from the velocity field to obtain a smaller splitting error. (Note that, to machine precision, because the splitting error is limited to the momentum equation, the splitting error will be the negative of the residual to the momentum equation if subsolvers of the projection method are held to a sufficiently tight tolerance.) In general this would change the splitting error to

$$-\frac{\Delta t}{\rho} (I - AC)P_C AGN^{-1}(p - \hat{p}).$$

This can be seen by starting with equation (6.55),

$$Au + \frac{\Delta t}{\rho} Gp + \frac{\Delta t}{\rho} P_C AGN^{-1}(p - \hat{p}) = r$$

and substituting $u_{\text{postprocessed}} = u + \frac{\Delta t}{\rho} CP_C AGN^{-1}(p - \hat{p})$ for $u$ to obtain

$$A(u + \frac{\Delta t}{\rho} CP_C AGN^{-1}(p - \hat{p})) + \frac{\Delta t}{\rho} Gp + \frac{\Delta t}{\rho} P_C AGN^{-1}(p - \hat{p})$$

$$= r + \frac{\Delta t}{\rho} ACP_C AGN^{-1}(p - \hat{p})$$

$$Au_{\text{postprocessed}} + \frac{\Delta t}{\rho} Gp + \frac{\Delta t}{\rho} (I - AC)P_C AGN^{-1}(p - \hat{p}) = r.$$
One special case is worth pointing out for (6.59). If $A_t$ is small (where $A = I + \Delta t A_t$), as is the case in the Navier-Stokes equations, then letting $C = I$ the splitting error would be

$$\frac{\Delta t^2}{\rho} A_t P_t A G N^{-1} (p - \hat{p}),$$

revealing that this splitting error is smaller in magnitude than one might assume from (6.59).

### 6.4.6 Summary

Using the new pressure update operator, $Y = N$, provides a method with a lower splitting error, especially when $A$ and $G$ do not nearly “commute.” This pressure update also allows a velocity postprocessing step to be employed to potentially reduce the error further. (There are a few additional properties in a slightly generalized version in Section A.1.5.) The complete projection method then becomes

$$Au^* = r - \Delta t G \hat{p}$$

(6.63)

$$\frac{\Delta t}{\rho} G^T G \phi_1 = G^T u^* + z$$

(6.64)

$$\hat{u} = u^* - \frac{\Delta t}{\rho} G \phi_1$$

(6.65)

$$G^T C G \phi_2 = G^T C A G \phi_1$$

(6.66)

$$p = \hat{p} + \phi_2$$

(6.67)

$$u = \hat{u} + C (r - \Delta t \frac{1}{\rho} G p - A \hat{u}),$$

(6.68)

where (6.66) and (6.67) combined are a long form of $p = \hat{p} + N \phi_1$. This method, while an improvement over former methods, unfortunately still provides too large of an error for the IB method when $C = I$.

We believe that it would work if a suitable choice of $C$ could be found that both approximates $A^{-1}$ well and does not make (6.66) computationally much more expensive than (6.64). We have not yet been able to find any such $C$; $I$ and $A_1^{-1}$ are the best approximations to $A^{-1}$ we have been able to find.
6.5 Connection to Fixed Point Schemes

Our analysis, backed up by computational tests in Section 6.6, show that the methods presented so far will not work for the IB equations; the splitting error is simply too large. However, in Section 6.1.4 we mentioned that incremental pressure-correction projection methods could be turned into iterative schemes. Using such iterative schemes may provide a way to reduce the splitting error to an acceptable level. Our generalized improved scheme could also be combined with such an approach. Further, it turns out that there is an alternative way to generate an iterative scheme with our generalized method. In this section we derive both iterative schemes, demonstrate that both are essentially fixed point methods, and use the fixed point equations to derive alternative schemes.

6.5.1 Iterative Schemes

Equations (6.63)-(6.68) specify our generalized and improved system, using the new update operator. Steps (6.63)-(6.67), ignoring (6.65), can be seen as a method for taking an approximation \( \hat{p} \) to \( p \) and using it to obtain a better approximation. These steps could be repeated in order to obtain

\[
Au^{*,k} = r - \Delta tG\hat{p}^{(k)} \quad (6.69)
\]

\[
\Delta tG^T\hat{p}^{(k)} = G^T\hat{u}^{*,k} + z \quad (6.70)
\]

\[
\hat{p}^{(k+1)} = \hat{p}^{(k)} + Y\phi^{(k)} \quad (6.71)
\]

where we have replaced \( N \) with \( Y \) in order to make it more general. This gives us an iterative scheme for the pressure.

There is an alternative iterative method that we can obtain, based on using approximations to \( u \) in an attempt to obtain successively better approximations. Equations (6.66)-(6.67) give the same pressure as would using \( \hat{u} \) in the pressure consistency equation (6.57),

\[
G^TCAu + \frac{\Delta t}{\rho}G^TGp = G^TCr \quad (6.72)
\]

to solve for a new pressure \( p \). From this point of view, equations (6.63)-(6.65) merely serve to provide a constraints-satisfying vector field, \( \hat{u} \), which we can then
use in (6.72). However, we already had a constraints-satisfying vector field, namely $u^n$, and thus do not really need to perform steps (6.63)-(6.65). We can skip straight to the pressure solve (6.72) for $p$ using $u^n$ instead of $\hat{u}$, then do the velocity postprocessing step (6.68) to obtain a better approximation to the velocity. Repeating these two steps, we obtain the iterative method

$$\Delta t G^T C G p^{n+\frac{1}{2},k+1} = G^T C (r - A u^{n+1,k})$$  

(6.73)

$$u^{n+1,k+1} = u^{n+1,k} + C (r - A u^{n+1,k} - \Delta t G p^{n+\frac{1}{2},k+1}).$$  

(6.74)

Applying $G^T$ to (6.74) and using (6.73) shows that $G^T u^{n+1,k+1} = G^T u^{n+1,k}$. So, if $u^{n+1,k}$ satisfies the constraints equation then $u^{n+1,k+1}$ will as well.

### 6.5.2 Fixed Point on $p$

By solving (6.69) for $u^{*,k}$, plugging it into (6.70), solving for $\phi^{(k)}$, and plugging the result into (6.71), we come up with a recurrence relation for the $\hat{p}^{(k)}$

$$\hat{p}^{(k+1)} = (I - Y (G^T G)^{-1} G^T A^{-1} G) \hat{p}^{(k)} + Y (G^T G)^{-1} (G^T A^{-1} r + z).$$  

(6.75)

This is a fixed point equation for $\hat{p}^{(k)}$. Thus, typical pressure-correction projection methods can be interpreted as a single step of a fixed point method.

Given that pressure-correction projection methods are essentially fixed point iterations, and ones which often only use a single iteration, it is somewhat remarkable that these methods have seen such wide use. The four reasons that combine to make a single iteration provide an accurate enough solution in the case of the incompressible Navier-Stokes equations were detailed in Section 6.3: $A$ is relatively small in norm, a good initial guess to the pressure is readily available, $A$ is a small perturbation of the identity, and $A$ “almost” commutes with $G$. All four break down when considering the IB equations. We could potentially make up for $A$ being large by using approximations of $A^{-1}$ for $C$ (appearing in $Y$ when $Y = N$), but the pressure problem and lack of commutativity cannot easily be corrected.
This \( p^{(k)} \) fixed point method will converge iff
\[
\rho((I - Y(G^T G)^{-1}G^T A^{-1}G) P_{\perp \text{Null}(G)}) < 1, \tag{6.76}
\]
where \( \rho(\cdot) \) represents the spectral radius of a given matrix and \( P_{\perp \text{Null}(G)} \) represents the orthogonal projection onto the space orthogonal to the nullspace of \( G \). (The reason for the multiplication by \( P_{\perp \text{Null}(G)} \) on the right is that the iteration matrix is applied only to the subspace orthogonal to the nullspace of \( G \); the spectral radius would be (at least) 1 if this nullspace were not excluded.) For the typical choice of \( Y = A \), this fixed point scheme will actually diverge when \( A \) is not a small perturbation of the identity, as shown in numerical tests in Section 6.6. It always appears to converge when \( Y = I \), but extraordinarily slowly if \( A \) is large. Seeing that this method is problematic when \( A \) is large, we seek a better method than fixed point iteration for solving for the pressure. The solution to the fixed point problem satisfies
\[
p^{n+\frac{1}{2}} = (I - Y(G^T G)^{-1}G^T A^{-1}G) p^{n+\frac{1}{2}}
+ Y(G^T G)^{-1}(G^T A^{-1}r + z), \tag{6.77}
\]
or equivalently
\[
Y(G^T G)^{-1}G^T A^{-1}G p^{n+\frac{1}{2}} = Y(G^T G)^{-1}(G^T A^{-1}r + z), \tag{6.78}
\]
or
\[
G^T A^{-1}G p^{n+\frac{1}{2}} = (G^T A^{-1}r + z). \tag{6.79}
\]
This is the Uzawa or double Schur complement \( p \) method, which is discussed further in Section A.1.2.

### 6.5.3 Fixed Point on \( u \)

By solving (6.73) for \( p^{n+\frac{1}{2},k+1} \), plugging it into (6.74), and making use of the definition of \( P_C \), we come up with a recurrence relation for the \( u^{n+1,k} \)
\[
u^{n+1,k+1} = u^{n+1,k} + C(r - Au^{n+1,k} - \Delta t G p^{n+\frac{1}{2},k+1}) \tag{6.80}
= u^{n+1,k} + C P_C (r - Au^{n+1,k}) \tag{6.81}
= (I - C P_C A) u^{n+1,k} + C P_C r. \tag{6.82}
\]
Note that when $C = A^{-1}$ this method converges in one step, assuming the initial guess for the velocity is divergence-free.

From equation (6.82) it is clear that this is simply a fixed point method for the velocity field. This method will converge iff

$$\rho((I - CP_C A)P) < 1. \quad (6.83)$$

(The reason for the multiplication by $P$ on the right is that the iteration matrix is applied only to divergence-free vector fields; thus we are not interested in eigenvalues corresponding to the nondivergence-free subspace.) If $C = I$ and $A$ is large, then not only will (6.83) be violated but the fixed point iteration will diverge, as shown numerically in Section 6.6. So, we need to ensure that $C$ is a sufficiently close approximation to $A^{-1}$ if we want to use this fixed point method.

As with the pressure-correction projection methods, we seek a method other than fixed point iteration for solving these equations, and note that the fixed point solution satisfies

$$u^{n+1} = (I - CP_C A)u^{n+1} + CP_C r, \quad (6.84)$$

or

$$CP_C A u^{n+1} = CP_C r, \quad (6.85)$$

which is a double Schur complement $u$ method that differs from equations (5.16) and (5.21) from Section 5.3.

Noting that this is also a nonsymmetric double Schur complement method, we can look for alternatives to (6.85) to try to find a symmetric positive definite system to which we can apply Conjugate Gradient. If we set $w = C^{-1}u$, then (since $G^T u = 0$ implies $P_C w = P_C C^{-1} u = C^{-1} u = w$) we can write (6.85) as

$$CP_C ACP_C w = CP_C r \quad (6.86)$$
where \( u \) can be recovered in the postprocessing step \( u = Cw \). Assuming \( C = C^T \) and \( A \) is symmetric and positive-definite, \( CP_CACP_C \) is symmetric and positive definite (on the subspace orthogonal to gradients) because

\[
(CP_C)^T = (C(I - G(G^T CG)^{-1}G^T C))^T \tag{6.87}
\]
\[
= (I^T - C^T G(G^T CG)^{-1}G^T C)^T \tag{6.88}
\]
\[
= (I - CG(G^T CG)^{-1}G^T C) \tag{6.89}
\]
\[
= C - CG(G^T CG)^{-1}G^T C \tag{6.90}
\]
\[
= C(I - G(G^T CG)^{-1}G^T C) \tag{6.91}
\]
\[
= CP_C. \tag{6.92}
\]

In the special case \( C = I \), (6.86) reduces to

\[
PAPw = Pr, \tag{6.93}
\]

which is the symmetric double Schur complement \( u \) method from Section 5.3 (since \( Pu = u = IP_I w = Pw \)).

### 6.6 Computational Tests

In this section, we present plots of spectral radii and norms of matrices from our analysis to demonstrate the size of splitting errors and convergence of iterative methods as a function of \( \gamma \) (the elastic tension). In all computations in this section we use \( \Delta t = 5 \times 10^{-2} \), \( h = 2\Delta s = 1/32 \), and \( \nu = 0.01 \), and place the immersed boundary in the initial elliptical configuration used in tests from Chapters 3 and 4.

Figure 6.4 compares the matrix 2-norms of the splitting errors of the nonrotational, rotational, and new projection methods. Recall that the splitting error for these methods, from (6.44), is \( \frac{\Delta t}{\rho}(G - AGY^{-1})(p - \hat{p}) \). In Figure 6.4, we plot the norms of \( G - AGY^{-1} \) and see that the new projection method significantly reduces the splitting error, though the splitting error remains large.

Figure 6.5 plots the spectral radii of the pressure iteration matrices from Section 6.5.2, \( (I - Y(G^T G)^{-1}G^T A^{-1} G)P_{\perp \text{constants}} \), for various \( \gamma \) and \( Y \). Interestingly, despite the fact that the splitting error for a single iteration is smaller with \( Y = A \)
or $Y = N$, the only pressure iteration matrix that results in a convergent iterative scheme is the one that uses $Y = I$. However, even that method converges very slowly as the spectral radius becomes arbitrarily close to 1 as $\gamma \to \infty$. Further, as can be seen in Figure 6.6, the 2-norm of the iteration matrix when $Y = I$ can be well above 1 (up to about 1.64 for the range of $\gamma$ used), so the method would actually appear to diverge in practice. The spectral radii and norms of the iteration matrix for the case $Y = I$ are also shown in Table 6.1 to illustrate how close to 1 they are.

Figure 6.7 plots the spectral radii of the velocity iteration matrix from Section 6.5.3, $(I - CP_C A) P$, for a range of values of $\gamma$. Interestingly, since the spectral radius is always larger than 1, this method does not converge even when $\gamma$ is small.

These computations reinforce the fact that these projection methods simply are not practical as a computational method when $\gamma \neq 0$ and $C = I$. We have tried several other choices for $C$: $A_1^{-1}$, $(I + D)^{-1}$ (where $D$ is $\text{diag}(SS^* \mathbf{1})$, an approximation to the diagonal of $A_2$), $\text{diag}(A_1 + D)^{-1}$, $(A_1 + D)^{-1}$, and $\text{diag}(A)^{-1}$.
Figure 6.5. Comparison of spectral radii of iteration matrices from equation (6.76) for various $Y$. These matrices are $(I - Y(G^T G)^{-1} G^T A^{-1} G) P\_\text{constants}$.

Figure 6.6. Comparison of norms for same operators as in Figure 6.5.
Table 6.1 Spectral radii and norms of the pressure iteration matrix $(I - (G^T G)^{-1} G^T A^{-1} G) P_{\perp \text{constants}}$.

<table>
<thead>
<tr>
<th>$\gamma$</th>
<th>Spectral radius</th>
<th>2-Norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.670862</td>
<td>0.670862</td>
</tr>
<tr>
<td>10$^{-1}$</td>
<td>0.677375</td>
<td>0.719706</td>
</tr>
<tr>
<td>10$^{0}$</td>
<td>0.879128</td>
<td>1.05431</td>
</tr>
<tr>
<td>10$^{1}$</td>
<td>0.960989</td>
<td>1.18176</td>
</tr>
<tr>
<td>10$^{2}$</td>
<td>0.986093</td>
<td>1.52109</td>
</tr>
<tr>
<td>10$^{3}$</td>
<td>0.997185</td>
<td>1.59039</td>
</tr>
<tr>
<td>10$^{4}$</td>
<td>0.999602</td>
<td>1.62293</td>
</tr>
<tr>
<td>10$^{5}$</td>
<td>0.999952</td>
<td>1.64296</td>
</tr>
</tbody>
</table>

Figure 6.7. Comparison of spectral radii of iteration matrices from equation (6.83), $(I - C P C A) P$, for $C = I$ and a range of $\gamma$. 
With the exceptions of $C = A^{-1}$ (which is extremely computationally costly) and $C = A_1^{-1}$ (which is computationally costly but provided slightly better results than $C = I$ in terms of norms, spectral radii, and splitting errors), all of the choices we have tried so far have been worse than $C = I$.

6.7 Velocity-correction Projection Methods

In this section we briefly argue that velocity-correction projection methods will face the same difficulties for the IB equations that make the pressure-correction projection methods not work well. We do so by looking at the splitting error and the resulting iteration matrix of the equivalent fixed point method. In the velocity-correction projection methods, the first step to solving

\[ Au + \Delta t Gp = r \]  

\[ G^T u = 0, \]  

is to do a Hodge Decomposition to split $r$ into divergence-free $Au$ and gradient $\Delta t Gp$ parts. This is done by solving

\[ \Delta t G^T Gp = G^T r, \]  

which implies that

\[ p = \frac{1}{\Delta t}(G^T G)^{-1}G^T r. \]  

The second step is solving for the velocity, given that we have an expression for $Au$:

\[ u = A^{-1}(r - \Delta t Gp). \]  

Because of the solution method, there is no error in the momentum equation, but the incompressibility constraint is not satisfied. Using our expression for $u$, we can determine an expression for how much the incompressibility constraint is violated:

\[ G^T u = G^T A^{-1} r - \Delta t G^T A^{-1} Gp \]

\[ = G^T A^{-1} r - G^T A^{-1} G(G^T G)^{-1} G^T r \]

\[ = G^T A^{-1} P r. \]
As an aside, a more direct route to this splitting error was by noting that the velocity-correction method enforces

\[ Au = Pr. \] (6.102)

We can also look at an incremental form of the velocity-correction projection method. For the Navier-Stokes equations, the incremental velocity-correction projection method involved approximating the viscous terms using a lagged velocity (see (6.14)). There is a question of how to extend this to the IB equations. The simplest possibility would be approximating just the viscous terms. A more consistent choice is to approximate all terms relating to portions of \( A \) that do not commute with \( G \); in both the Navier-Stokes and IB cases this means \( I - A \). Using this strategy, the incremental velocity-correction projection method means that we approximately solve

\[
Au^{n+1,k+1} + (I - A)u^{n+1,k} + \Delta tGp^{k+1} = r + (I - A)u^{n+1,k} \\
G^T u = 0
\] (6.103, 6.104)

by splitting with a Hodge Decomposition so that

\[
Au^{n+1,k+1} + (I - A)u^{n+1,k} = P(r + (I - A)u^{n+1,k}).
\] (6.105)

This decomposition implies

\[
u^{n+1,k+1} = A^{-1}Pr + A^{-1}(I - P)(A - I)u^{n+1,k},
\] (6.106)

which will converge iff

\[
\rho(A^{-1}(I - P)(A - I)) < 1.
\] (6.107)

Note that when \( A \) commutes with \( P \) (which occurs when \( A \) commutes with \( G \)), \( A^{-1}(I - P)(A - I)P = 0 \). This means that under this special condition the method will converge in one step if given a divergence-free initial guess. Thus we have the same problem that we repeatedly ran into with the pressure-correction projection methods: We have a matrix \( A \) which is very large and does not commute with
This means that not only will the projection method be slow, it can actually diverge as an iterative method.

Figure 6.8 shows the spectral radii and norms of the iteration matrix $A^{-1}(I - P)(A - I)$ for various values of $\gamma$. It is interesting to note that the spectral radii are nearly identical to those found in Table 6.1 for the pressure iteration method with $Y = I$, so this method is actually convergent. However, given the large norms the convergence would be extraordinarily slow; in practice it would just appear to be rapidly diverging for any reasonable number of iterations.

### 6.8 Conclusions

In this chapter, we provided an overview of the theory behind projection methods, outlined properties of the IB equations that differ from the incompressible Navier-Stokes equations, and analyzed the splitting error of projection methods. We showed that there were four reasons that combined to make projection methods
effective (result in only a small splitting error) in the case of the incompressible Navier-Stokes equations: $A$ is relatively small in norm, a good initial guess to the pressure is readily available, $A$ is a small perturbation of the identity, and $A$ “almost” commutes with $G$. All four of these reasons break down when considering the IB equations, in particular due to the nature of the large singular force reflected in $A$.

We introduced a new projection method variant, using the new pressure update operator, which significantly reduced the splitting errors and had many other beneficial properties. This method may have useful applications for other problems, but unfortunately we found the splitting error to be too large to be of use for solving the IB equations. This could potentially be remedied if an appropriate coarse approximation to $A^{-1}$ could be found (for $C'$), but our attempts so far have not uncovered any.

We also explored the connection between projection methods and fixed point iterations and re-derived some of the double Schur complement equations from Chapter 5, suggesting that these methods will perhaps be better suited to solving the IB equations. Exploring these options is the topic of Chapter 7.

We have also explored other projection method variations, including various generalizations from the literature. Our new update operator provides some interesting connections between some of these generalizations, but these generalizations have the same basic problems for the IB equations as the projection methods analyzed in this chapter. We leave the details of these generalizations in Appendix A for those wanting to investigate these options further and avoid duplicating our efforts.
CHAPTER 7

EFFICIENCY — COMPARISON OF IMPLICIT METHODS

Having ruled out a projection approach to solving the IB equations in Chapter 6, in this chapter we numerically compare a variety of implicit methods with the explicit method. The implicit methods we compare are the double Schur complement methods from Section 5.3, which are based on the double Schur complement $X$ equation,

$$
(I - \frac{\Delta t^2}{4\rho} S^* M \mathbb{P} S_{\rho} f) X^{n+1} = 
(I + \frac{\Delta t^2}{4\rho} S^* M \mathbb{P} S_{\rho} f) X^n
+ \frac{\Delta t}{2} S^* \left( u^n + M \left( I + \frac{\rho \Delta t}{2} \Delta h \right) u^n - \Delta t M \mathbb{P} \left( (u \cdot \nabla) u \right)^{n+\frac{1}{2}} \right),
$$

\hspace{20pt}\text{(7.1)}$$

the double Schur complement $p$ equation,

$$
\nabla^h \cdot A^{-1} \nabla^h p^{n+\frac{1}{2}} = \nabla^h \cdot A^{-1} \left( \frac{\rho}{\Delta t} \left( 2I - A \right) u^n - \rho \left( (u \cdot \nabla) u \right)^{n+\frac{1}{2}} + S_{\rho} f X^n \right),
$$

\hspace{20pt}\text{(7.2)}$$

and the symmetric double Schur complement $u$ equation,

$$
\mathbb{P} A \mathbb{P} u^{n+1} = \mathbb{P} \left( \left( 2I - A \right) u^n - \Delta t \left( (u \cdot \nabla) u \right)^{n+\frac{1}{2}} + \frac{\Delta t}{\rho} S_{\rho} f X^n \right).
$$

\hspace{20pt}\text{(7.3)}$$

7.1 Additional Test Problem

In addition to the ellipse problem considered in previous chapters, we consider a model of a moving wing problem adopted from Miller and Peskin [24]. The target equations of motion of the wing are as follows:

$$
x(t) = \frac{A_0}{2} \cos(2\pi ft) \hspace{20pt}\text{(7.4)}$$

and

$$
\alpha(t) = \alpha_0 + \beta \sin(2\pi ft + \phi) \hspace{20pt}\text{(7.5)}$$
where $A_0$ is stroke amplitude, $x(t)$ describes the horizontal position of the center of the wing as a function of time, $\alpha(t)$ describes the angle of attack relative to the $x$-axis as a function of time, $\phi$ sets the timing of rotation, and $\beta$ sets the change in angle of attack during stroke reversal. Basically, the wing flaps back and forth along a horizontal plane with a frequency of $f$. In this case, $A_0/c$ was set to 2.8 (where $c$ is the chord length of the wing), $\phi$ was set to 0, $\alpha_0$ was set to $\pi/2$, and $\beta$ was set to $\pi/4$. This provides a symmetric stroke with a minimum angle of attack of $45^\circ$. The value $7.5\nu/\pi c A_0$ is used for $f$.

The wing does not track exactly the motion specified by (7.4)-(7.5). Instead, a target IB tracks that motion while a second IB representing the real wing is connected to the target boundary by spring-like forces. Letting $Y(s,t)$ represent the target IB and $X(s,t)$ represent the real IB, the forces generated at $X(s,t)$ by the connection between boundaries is given by

$$F_{\text{targ}} = k_{\text{targ}}(Y(s,t) - X(s,t)).$$

(7.6)

$Y(s,t)$ moves according to equations (7.4) and (7.5) with no force generated at those points. There are additional forces generated at $X(s,t)$ representing damping, tension/compression, and bending forces. We consider only the last of these, given by

$$F_{\text{beam}} = k_{\text{beam}} \frac{\partial^4 X(s,t)}{\partial s^4}.$$  

(7.7)

so that our total force at the immersed boundary is

$$F_{\text{total}} = F_{\text{targ}} + F_{\text{beam}}.$$  

(7.8)

### 7.2 Results

In this section, we show computational timings of both the ellipse and wing problems to compare the efficiency of the explicit, double Schur complement $p$ (DSP), symmetric double Schur complement $u$ (DSU), and double Schur complement $X$ methods.
7.2.1 Ellipse Problem

Figure 7.1 shows the CPU time required by the explicit method for the ellipse problem as a function of the spatial stepsize, $h$, for a timestep $\Delta t$ that is near the stability limit. The plot for $\gamma = 100$ is nearly identical, other than all times being increased by a factor of about 10.

The “near stability limit” timestep for the explicit method is used when comparing against the implicit methods below as well. This timestep is found by an (expensive) search that involves repeating the simulation with several different values of $\Delta t$ until a timestep is found that both results in a stable simulation and is within 1% of the magnitude of a timestep that results in an unstable simulation. This search needs to be repeated whenever any parameters in the system (e.g., $h$, $\gamma$, initial IB configuration) change. The time required to find such a timestep is not reported; instead, only the CPU time the simulation took with the resulting “optimal” timestep is shown. This gives a somewhat pessimistic view of the implicit methods, since practical computations will often only guess at the stability limit.

![Figure 7.1. CPU time for the explicit method on the ellipse problem when $\gamma = 1$, as a function of spatial stepsize](image-url)
for explicit simulations and then divide by some safety factor, such as 5 or 10, in order to try to ensure stability. In fact, it is not too uncommon for practitioners to simply manually restart simulations with a smaller timestep when “safe” guesses turn out to be wrong, which can significantly increase the cost of real world use of the explicit method, though this cost is not measured or reported in practice.

Figures 7.2 and 7.3 show the percentage of time that the explicit, DSU, and DSX methods took relative to the explicit method (thus the explicit method always has a value of 1). DSP took around 15-25 times as long as the explicit method in all cases, swamping the time used by any other method. Plotting the timings of the DSP method in the same figure made it difficult to compare the other methods, and thus we do not show them here.

From Figures 7.2 and 7.3 we see that the DSU and DSX methods with the timestep computed from the CFL condition (at CFL 1) are a bit slower than the explicit method with an optimal timestep when $h$ is coarse. However, both rapidly improve relative to the explicit method as $h$ becomes finer, so that both

![Figure 7.2](https://example.com/figure7.2.png)

**Figure 7.2.** Comparison of explicit (with “optimal” timestep), DSU, and DSX (both with CFL 1 timestep) methods for the ellipse problem, using $\gamma = 1$. 
Figure 7.3. Comparison of explicit (with “optimal” timestep), DSU, and DSX (both with CFL 1 timestep) methods for the ellipse problem, using $\gamma = 100$.

are competitive with the optimal-timestep explicit method. Comparing between the figures, we see that increasing the stiffness of the boundary ($\gamma$) decreases the relative effectiveness of the implicit methods.

This appears to be a good improvement over the explicit method, but probably less than one would suspect. There are two reasons for this: the CFL constraint is not significantly larger than the explicit timestep limit (larger $\gamma$ results in faster fluid velocities in this particular problem, thus keeping the two timesteps within an order of magnitude), and larger timesteps result in more poorly conditioned systems, which reduce some of the savings of using a larger timestep.

### 7.2.2 Wing Problem

Figures 7.4 and 7.5 show the percentage of time that the explicit, DSU, and DSX methods took relative to the explicit method for the wing problem described in Section 7.1. The DSP method was again extremely slow in comparison to the other methods and was removed from the plots. The effect with decreasing $h$ is
unfortunate; the implicit methods become gradually slower relative to the explicit method. We do not know the precise reason for this, but suspect that the beam forces may cause the condition number of the system to increase more rapidly than the spring forces. However, on the positive side, we find that increasing the stiffness improves the advantage of the implicit methods. While this is the opposite of the effect in the ellipse problem, this makes sense since an increase of stiffness will not result in an increased velocity field in the wing problem, meaning that the CFL constraint in the wing problem does not decrease as the stiffness increases.

7.3 Conclusions

The DSP method appears to be much slower than all the other methods, and is thus not recommended. The DSU and DSX methods are not as fast as one would like due to the tradeoff of ill-conditioning for a larger timestep. However, both are competitive with the explicit method, even when the explicit method is run with an optimal timestep. Since an optimal timestep is not always available (and in fact

![Figure 7.4. Comparison of explicit, DSU, and DSX methods for the wing problem, with $\gamma = 10^4$ and $k_{beam} = 10^{-3}$ for various spatial stepsizes.](image-url)
Figure 7.5. Comparison of explicit, DSU, and DSX methods for the wing problem, with $\gamma = 10^6$ and $k_{\text{beam}} = 10^{-1}$ for various spatial stepsizes.

A timestep known to be stable may not be available either, these methods will likely be an order of magnitude or more faster than the explicit method in practice.

Both the DSU and DSX methods were run without any preconditioning. It may be possible to accelerate these methods with the use of an appropriate preconditioner. Mori and Peskin have claimed some success with preconditioning for an IB problem featuring thick boundaries and additional boundary mass using a method similar to DSX (Y. Mori, personal communication, July 18, 2006).
APPENDIX A

GENERALIZED PROJECTION METHOD
DETAILS

In this appendix, we add some details about additional generalizations of projection methods that we have considered, that were omitted from Chapter 6.

A.1 Further Generalizations

In Chapter 6, we discussed basic variants of pressure-correction projection methods and added a new update operator. There are many generalizations of pressure-correction projection methods in the literature. In an attempt to find an accurate and efficient method for the IB problem, we looked at many of these, including hybrids. While we did not find anything that looks promising for the IB equations, we document our work so that others who want to investigate these methods further need not repeat our efforts and can be aware of potential problems.

We begin by modifying (6.33) slightly to allow for boundary conditions other than periodic,

\[
\begin{pmatrix}
A & \frac{\Delta t}{\rho} G \\
G^T & 0
\end{pmatrix}
\begin{pmatrix}
u \\
p
\end{pmatrix}
= \begin{pmatrix}
r \\
-z
\end{pmatrix}
+ \begin{pmatrix}
bc_1 \\
-bc_2
\end{pmatrix}.
\]  

(A.1)

In [30], Perot analyzed and generalized the fractional step method by introducing a generalized Poisson solve and velocity correction. The Poisson solve involved inverting \(G^T B_1 G\) instead of \(G^T G\), whereas the velocity correction in his method involved subtracting some factor times \(B_2 G \phi\) rather than just \(G \phi\). Both \(B_1\) and \(B_2\) are thought of as approximations to \(A^{-1}\), though \(I\) is a common choice for both. Almgren et al.[2] investigated different choices of what to project when using approximate projections. Two of their choices correspond to projecting \(u^* - u^n\) instead of \(u^*\), but both of these were rejected in their analysis and we do not
investigate them further. Two of their other choices correspond to whether to include the approximation to the pressure gradient in the Poisson solve (which also affects the velocity correction) or in the pressure update step. They also introduce an additional hybrid projection method that can be written as a more complicated pressure update scheme. Combining the generalizations of both Perot and Almgren et al. with the incremental, rotational, and new update variations we obtain the generalized projection method

\[ Au^* = r + bc_1 - \frac{\Delta t}{\rho} G\hat{p} \]  
(A.2)

\[ \frac{\Delta t}{\rho} G^T B_1 G\phi = G^T u^* + z + bc_2 + \frac{\Delta t}{\rho} G^T B_3 G\hat{p} \]  
(A.3)

\[ u = u^* - \frac{\Delta t}{\rho} B_2 G\phi + \frac{\Delta t}{\rho} B_3 G\hat{p} \]  
(A.4)

\[ p = \delta_1 \hat{p} + Y \phi + \delta_2 (G^T G)^{-1} G^T \left( -\frac{\rho u^*}{\Delta t} + G\hat{p} \right) , \]  
(A.5)

which will be compared with (6.34)-(6.37).

Equations (A.2)-(A.5) contain several unspecified parameters and operators \((B_1, B_2, B_3, Y, \delta_1, \delta_2, \hat{p})\), that can be modified to get different projection or fractional step methods. The method may appear a bit overwhelming at first, but below we outline several special cases in order to highlight the purpose of the various terms. First, however, note that several terms have been underlined. All of the underlined terms potentially have a \(G^T G\) appearing in them (in equation (A.3) if \(B_1 = I\) or \(B_3 = I\), in equation (A.5) if \(\delta_2 \neq 0\), or if \(G^T G\) appears in \(Y\)). The reason for the underlining is to account for approximate projections, which use a discretization approximating \(G^T G\) rather than using \(G^T G\) exactly. We also note that using the same steps as in Section 6.3, equations (A.2)-(A.5) are equivalent to the following system for \(u\) and \(p\)
\[
\begin{pmatrix}
A \\
-G^T \\
\Delta t \rho G^T B_1 G Y^{-1}
\end{pmatrix}
\begin{pmatrix}
\tilde{p}
\end{pmatrix}
= \\
\begin{pmatrix}
\Delta t \rho G^T B_2 G Y^{-1} - \Delta t \rho G^T B_2 G Y^{-1}
\end{pmatrix}
\begin{pmatrix}
\Delta t \rho G^T B_3 G - G^T B_3 G
\end{pmatrix}
\begin{pmatrix}
\hat{p}
\end{pmatrix}
\]

\begin{eqnarray}
\frac{\Delta t}{\rho} \begin{pmatrix}
AB_2 G Y^{-1}
\end{pmatrix}
\begin{pmatrix}
G^T B_1 G Y^{-1} - G^T B_2 G Y^{-1}
\end{pmatrix}
\delta_1 \hat{p} + \\
\frac{\Delta t}{\rho} \begin{pmatrix}
AB_2 G Y^{-1}
\end{pmatrix}
\begin{pmatrix}
G^T B_1 G Y^{-1} - G^T B_2 G Y^{-1}
\end{pmatrix}
\delta_2 (G^T G)^{-1} G^T \left( \frac{-\rho u^n}{\Delta t} + G \hat{p} \right),
\end{eqnarray}

(A.6)

and that the splitting error can be found by comparing (A.6) to (A.1). See Section A.3 for the complete derivation of this equation.

The remainder of this section overviews special cases (including connections to previous cases already covered), pointing out results from the literature on these methods and potential problems in applying such methods to the IB case.

### A.1.1 Standard Pressure-correction Methods

Section 6.3 analyzed the standard pressure-correction projection methods, including incremental/nonincremental and rotational/nonrotational possibilities. These methods correspond to the choices \( B_1 = B_2 = C = I \) (no specialized Poisson solves or gradients), \( B_3 = 0 \) (no inclusion of the approximated pressure gradient in the Poisson solve step), \( \delta_1 = 1 \) (Poisson solve variable needs to be added to previous pressure approximation), \( \delta_2 = 0 \) (no need to use alternative pressure update since \( \delta_1 = 1 \)), and \( G^T G = G^T G \) (not using an approximate projection).

As noted in Section 6.3, \( \hat{p} = 0 \) for the nonincremental pressure-correction methods, \( \hat{p} \) is some approximation to \( p^{n+1/2} \) (such as \( p^{n-1/2} \)) for the incremental versions, \( Y = I \) in the nonrotational versions, and \( Y = A \) for the rotational versions.

As derived in Section 6.3, the splitting error for these methods is

\[
\frac{\Delta t}{\rho} (G - AG Y^{-1})(p - \hat{p}).
\]

(A.7)

It is worth noting that for the nonrotational versions, a scheme yielding the same \( u \) and \( p \) and splitting error can be obtained by changing \( B_3 \) to \( I \) (corresponding to including the approximate pressure gradient in the Poisson solve) and \( \delta_1 \) to 0 (corresponding to the fact that the Poisson solve variable already includes the...
approximate pressure). In fact, this was the method used in Section 2.3.2 (see (2.47)). However, in combination with the rotational versions, this change modifies the splitting error to

$$\frac{\Delta t}{\rho} (G - AGA^{-1})p - \frac{\Delta t}{\rho} (G - AG)\hat{p}. \quad (A.8)$$

In this splitting error, the higher order accuracy from a good guess for $\hat{p}$ is lost since the factors multiplying $p$ and $\hat{p}$ are not the same (which I refer to as an unbalanced splitting error). Also, note that the error will not be 0 when $A$ and $G$ commute, as is normally expected. Thus, what appears to be an innocuous change can have bad side effects when other generalizations are considered.

### A.1.2 Uzawa Method

If $B_1 = B_2 = A^{-1}$ (resulting in modified Poisson equation and gradient in the velocity correction), $Y = I$ (simple pressure update), $\hat{p} = 0$ (do not include an approximation to the pressure), $B_3 = \delta_1 = \delta_2 = 0$ (no need to handle an approximation to the pressure since we are not including one), and $G^T G = G^T G$ (and exact projection), then the set of equations that the generalized fractional step method (A.2)-(A.5) solves is

$$\begin{pmatrix} A & \frac{\Delta t}{\rho} G \\ -G^T & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} r \\ z \end{pmatrix} + \begin{pmatrix} bc_1 \\ bc_2 \end{pmatrix}, \quad (A.9)$$

which has no splitting error. Unfortunately, this method requires a nested inner/outer solve to invert $G^T A^{-1} G$, and has traditionally been avoided for that reason [30]. Note that this method is precisely the double Schur complement $p$ method from Section 5.3 (which consists of solving (5.20) followed by plugging the resulting pressure into (5.17) to get the new velocity field).

### A.1.3 Perot’s Generalization

In Perot’s analysis and generalization of the nonincremental nonrotational pressure-correction projection methods [30], he uses $B_3 = 0, \delta_1 = \delta_2 = 0, Y = I, \hat{p} = 0, G^T G = G^T G$, and allows $B_1$ and $B_2$ to vary. These methods simplify to

$$\begin{pmatrix} A & \frac{\Delta t}{\rho} AB_2 G \\ -G^T & \frac{\Delta t}{\rho} G^T (B_1 - B_2) G \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} r \\ z \end{pmatrix} + \begin{pmatrix} bc_1 \\ bc_2 \end{pmatrix}. \quad (A.10)$$
Perot notes several interesting things about this generalization. This system is an exact solve if $B_1 = B_2 = A^{-1}$ and, in fact, is the Uzawa method as shown in Section A.1.2. He also notes that the constraints equation is satisfied whenever $B_1 = B_2$. All of the error can be moved from the momentum equation to the constraints equation if $B_2 = A^{-1}$ and $B_1 \neq A^{-1}$. If $B_1$ is an explicitly known matrix (unlike $A_1$), then this system is much easier to solve than the Uzawa method (no inner iteration necessary). However, it does not appear to be popular, and Chang et al.\cite{7} argue that in most cases it is more important to satisfy incompressibility than to reduce the error in the momentum equation.

Writing $A = I + \Delta t A_t$, Perot suggested using $B_1 = B_2 = B_N \equiv I - \Delta t A_t + \Delta t^2 A_t^2 - \Delta t^3 A_t^3 + \ldots$, i.e., an $N$th order Taylor series expansion of $A^{-1}$. For this method, a restriction on the timestep might need to be imposed in order to maintain positive-definiteness of $B_N$. This is typically not a problem when solving the Navier-Stokes equations, but such a method will likely be ineffective for the IB method. This is because $A_t$ will approximately scale with $\gamma$ and impose too large a restriction on the timestep to maintain positive-definiteness. In fact, without a severe timestep restriction, this will be a divergent series rather than a good approximation to $A^{-1}$ in the IB case. (It is perhaps also worth noting that a second order expansion has the nice property of ensuring positive semidefiniteness of $B_N$. However, it is unlikely to be a good approximation to $A^{-1}$. Further, in order to ensure that the system is not singular or ill-conditioned, $\Delta t$ would have to be small enough to bound the eigenvalues of $B_N$ away from 0, and thus this expansion still imposes too stringent a timestep restriction in the IB case.)

**A.1.4 Almgren et al. Variants**

Almgren et al.\cite{2} investigate different choices of what to project when using approximate pressure-correction projection methods. In all their methods they use $B_1 = B_2 = Y = I$ (no generalized Poisson solve, gradient, or pressure update), $\hat{p} = p^{n-1/2}$, $G^T B_3 G = G^T G$ when $B_3 = I$, and for all other occurrences of $G^T G$ they use $L \neq G^T G$ (i.e., an approximate projection). They present five methods,
two of which they categorically reject as bad (the “velocity-update” projection methods), which we do not cover. The other three approximate methods are listed below with their splitting error.

Approximate PM1 corresponds to the choices $B_3 = 0$, $\delta_1 = 1$, and $\delta_2 = 0$, and solves the set of equations,

$$
\begin{pmatrix}
A & -G^T \\
-G^T & \frac{\Delta t}{\rho} L - \frac{\Delta t}{\rho} G^T G
\end{pmatrix}
\begin{pmatrix}
u \\
p
\end{pmatrix}
= \begin{pmatrix}
r \\
0
\end{pmatrix}
+ \begin{pmatrix}
bc_1 \\
bc_2
\end{pmatrix}
+ \frac{\Delta t}{\rho} \begin{pmatrix} -G + AG \\ L - G^T G \end{pmatrix} p^{n-1/2}.
$$
(A.11)

Approximate PM2 corresponds to the choices $B_3 = I$, $\delta_1 = 0$, and $\delta_2 = 0$, and solves the set of equations

$$
\begin{pmatrix}
A & -G^T \\
-G^T & \frac{\Delta t}{\rho} L - \frac{\Delta t}{\rho} G^T G
\end{pmatrix}
\begin{pmatrix}
u \\
p
\end{pmatrix}
= \begin{pmatrix}
r \\
0
\end{pmatrix}
+ \begin{pmatrix}
bc_1 \\
bc_2
\end{pmatrix}
+ \frac{\Delta t}{\rho} \begin{pmatrix} -G + AG \\ 0 \end{pmatrix} p^{n-1/2}.
$$
(A.12)

Approximate PM5 corresponds to the choices $B_3 = 0$, $\delta_1 = 0$, and $\delta_2 = 1$, and solves the set of equations

$$
\begin{pmatrix}
A & -G^T \\
-G^T & \frac{\Delta t}{\rho} L - \frac{\Delta t}{\rho} G^T G
\end{pmatrix}
\begin{pmatrix}
u \\
p
\end{pmatrix}
= \begin{pmatrix}
r \\
0
\end{pmatrix}
+ \begin{pmatrix}
bc_1 \\
bc_2
\end{pmatrix}
+ \frac{\Delta t}{\rho} \begin{pmatrix} -G + AG \\ 0 \end{pmatrix} p^{n-1/2}
\frac{\Delta t}{\rho} \begin{pmatrix} -G + AG \\ L - G^T G \end{pmatrix} L^{-1}G^T \left( -\frac{\rho}{\Delta t} n_1 + G p^{n-1/2} \right).
$$
(A.13)

Note that if $G^T B_3 G = L$ were chosen instead when $B_3 = I$, then approximate PM2 would become the same as approximate PM1.

While approximate projections are not the focus of our study, and all three of these methods are identical when replacing $L$ with $G^T G$, there are some interesting results from the work of Almgren et al. They noted that approximate PM2 always gave good results and remained stable. However, they noted that approximate PM1, while it occasionally went unstable, gave more accurate results (meaning smaller error in the incompressibility constraint) than approximate PM2. (This better accuracy could probably be guessed by comparing the splitting error of the two methods and noticing the balancing of the $p$ and $\dot{p}$ terms in the splitting error of the constraints equation for approximate PM1). They introduced approximate PM5 as a hybrid between the two that would yield a velocity with a divergence.
approximately as small as the one from approximate PM1 while remaining stable, like approximate PM2. The cost of the hybrid method is an extra Poisson solve, though one for which they had a very good initial guess.

The interesting thing to note with their method is that this is another case where the pressure increment choice matters. In fact, in this case it was PM2 (or PM5) that always gives acceptable results. In the rotational pressure-correction method or with the new update operator, the results were somewhat the opposite; as pointed out in Section A.1.1, PM2 provides an undesirable error (see (A.8)).

A.1.5 Generalizations of the New Update Operator

The methods with the new update operator in Section 6.4 correspond to the choices $B_3 = 0$ (no inclusion of the approximated pressure gradient in the Poisson solve step), $\delta_1 = 1$ (Poisson solve variable needs to be added to previous pressure approximation), $\delta_2 = 0$ (no need to use alternative pressure update since $\delta_1 = 1$), $G^G = G^T G$ (not using an approximate projection), and $B_1 = B_2 = I$ (no specialized Poisson solve or gradients). If we relax the $B_1 = B_2 = I$ constraint but still enforce $B_1 = B_2$, then we get a slightly more generalized method. This slightly more general method solves the set of equations

$$
\left( \begin{array}{ccc} 
A & \frac{\Delta t}{\rho} & AB_2 G Y^{-1} \\
-G^T & 0 & \\
0 & 0 & 
\end{array} \right) \begin{pmatrix} u \\ r \\ z 
\end{pmatrix} = \begin{pmatrix} 
\dot{u} \\
r \\
z 
\end{pmatrix} + \left( \begin{array}{c} bc_1 \\
bc_2 \end{array} \right) + \frac{\Delta t}{\rho} \begin{pmatrix} 
-G + AB_2 G Y^{-1} \\
0 
\end{pmatrix} \hat{p}.
$$

(A.14)

Following the same steps as in Section 6.4, the new update operator $N$ becomes

$$
N = (G^T C G)^{-1} G^T C A B_2 G.
$$

(A.15)

This causes a number of other changes, but also has some new interesting properties.

Using the generalized choice of $N$ in (A.15), the splitting error in the momentum equation given by (6.52) becomes

$$
\frac{\Delta t}{\rho} (G - AB_2 G N^{-1})(p - \hat{p}),
$$

(A.16)

which has the special form

$$
-\frac{\Delta t}{\rho} P C A B_2 G N^{-1}(p - \hat{p})
$$

(A.17)
in this case, due to the fact that \(\text{compare to (6.53)}\)

\[
G - AB_2GN^{-1} = -P_CAB_2GN^{-1}. \tag{A.18}
\]

Because of our inclusion of boundary conditions, (6.55)-(6.57) need to be modified to replace \(r\) with \((r + bc_1)\). Similarly, equations (6.59)-(6.61) need to be modified to replace \(r\) with \((r + bc_1)\) and to replace \(A\) with \(AB_2\). There is also an additional special case that is interesting for (6.59) (with \(A\) replaced by \(AB_2\), that is). If \(B_2 \approx A^{-1}\), then that splitting error approximately becomes

\[
\frac{\Delta t}{\rho} (AC - I) P_C I GN^{-1} (p - \dot{p}) = 0, \tag{A.19}
\]

since \(P_C G = 0\).

**A.1.5.1 Connects Uzawa and Rotational Schemes**

In Section 6.4.3, we mentioned some ways in which \(N\) generalized the optimal pressure update operator \(Y = A\) from the rotational method. With the modified choice of \(N\) from (A.15), there is an additional generalization provided. There are two special cases in which we can solve (A.1) with no splitting error. The first case is using the rotational pressure-correction method (i.e., \(B_2 = I, Y = A\)), under the assumption that \(A\) and \(G\) “commute.” The second case is using the Uzawa method (i.e., \(B_2 = A^{-1}, Y = I\)). In either case, choosing \(Y = N\) simplifies to the correct choice. So the new update operator gives us a way to generalize and connect the “optimal” Uzawa and rotational schemes and provides an update choice for the pressure for intermediate schemes such as those proposed by Perot.

**A.1.5.2 Alternative Incremental Pressure-correction Methods**

Just as with the rotational projection, the different forms of projections considered by Almgren et al. are not equivalent if we use the pressure update operator \(Y = N\), even when we use exact projections (i.e., when we use \(G^TG = G^TG\)). It turns out that keeping \(B_3 = 0\) and changing \(\delta_1 = 0, \delta_2 = 1\) (corresponding to PM5) will yield identical results to our current case \((B_3 = 0, \delta_1 = 1, \delta_2 = 0)\) when using an exact projection. However, \(B_3 = I, \delta_1 = 0, \delta_2 = 0\) (corresponding to PM2)
degrades the splitting error in the same way it does for the rotational scheme (see Section A.1.1).

### A.1.5.3 Full Method, Related Notes

For convenience, the full pressure-correction method using the generalized new pressure update operator and velocity postprocessing is

\begin{align}
Au^* &= r + bc_1 - \Delta tG\hat{p} \quad (A.20) \\
\Delta tG^T B_1 G\phi_1 &= G^T u^* + z + bc_2 \quad (A.21) \\
\hat{u} &= u^* - \Delta tB_2 G\phi_1 \quad (A.22) \\
G^T C G\phi_2 &= G^T CAB_2 G\phi_1 + bc_3 \quad (A.23) \\
p &= \hat{p} + \phi_2 \quad (A.24) \\
\hat{u} &= \hat{u} + C(r + bc_1 - \Delta tGp - A\hat{u}) \quad (A.25)
\end{align}

Note that (A.23) and (A.24) combined are a long form of \( p = \hat{p} + N\phi_1 \).

An alternative derivation of the new pressure update operator is provided in Section A.5, details of how to handle boundary conditions in the nonperiodic case are covered in Section A.6, and discussion of combining the new update operator with approximate projections is discussed in more detail in Section A.7.

### A.2 More Generalized Constraint Problems

In Section 6.3 we briefly mentioned more generalized fluid constraint problems; in this section we provide one example of such a system.

For most problems, \( G \) will be a discrete gradient operator and \(-G^T\) a discrete divergence operator. For such problems, the constraint on the motion of the fluid is the incompressibility condition \( G^T u = -z - bc_2 = 0 - bc_2 \). However, other choices of \( G \) are possible. In the work of Taira and Colonius[39], there are two constraints on the motion of the fluid — the standard incompressibility condition on the fluid plus prescribed motion of a structure embedded within the fluid. They employ two computational grids, an Eulerian one for the fluid equations, and a Lagrangian one for the embedded structure. Their system of equations is of the form
In the formulation of Taira and Colonius, $E$ is an interpolation operator and the third set of equations states that the velocity field interpolated from the Eulerian computational grid ($Eu$) must be equal to the prescribed velocity field ($u_p$) defined on the Lagrangian grid. Similar to the role the pressure plays as a Lagrange multiplier in the Navier-Stokes equation to enforce the incompressibility condition (the pressure is whatever it needs to be to ensure that the fluid is incompressible), in their method the force that the embedded interface exerts on the fluid ($\hat{f}$, which is defined on the Lagrangian grid) is also a Lagrange multiplier (the force is whatever is needed to achieve the prescribed motion). Given this algebraic formulation, they simply define $Q$ to be $(G, ET)$, $r_2$ to be $(0, u_p)^T$, $bc_{23}$ to be $(bc_2, bc_3)$, and $\lambda$ to be $(p, \hat{f})^T$ to put their system of equations in the form

$$\begin{bmatrix} A & G & ET \\ GT & 0 & 0 \\ E & 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ p \\ \hat{f} \end{bmatrix} = \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} bc_1 \\ bc_2 \\ bc_3 \end{bmatrix}. \quad (A.26)$$

If we relabel $G$ to be $Q$, $p$ to be $\lambda$, $(r, z) = (r, r_2)$, and $bc_2 = bc_{23}$ then this is equation (A.1).

### A.3 Derivation of Equivalent System

In this section we derive the splitting error of the generalized projection method presented in Section A.1.

Equations (A.2) and (A.3) in system form are

$$\begin{bmatrix} A & 0 \\ Q^T & 0 \end{bmatrix} \begin{bmatrix} u^r \\ \phi \end{bmatrix} = \begin{bmatrix} r \\ z \end{bmatrix} + \begin{bmatrix} bc_1 \\ bc_2 \end{bmatrix} + \begin{bmatrix} -\Delta t \hat{p} \\ \Delta t G T B_3 G \hat{p} \end{bmatrix}, \quad (A.28)$$

whereas equations (A.4) and (A.5) in system form are

$$\begin{bmatrix} I & -\Delta t B_2 G \\ 0 & \frac{Y}{G} \end{bmatrix} \begin{bmatrix} u^r \\ \phi \end{bmatrix} = \begin{bmatrix} \hat{u} \\ \hat{p} \end{bmatrix} + \begin{bmatrix} -\Delta t B_3 G \hat{p} \\ -\delta_1 \hat{p} - \delta_2 (G T G)^{-1} G T \left( \frac{-\hat{u}}{\Delta t} + G \hat{p} \right) \end{bmatrix}. \quad (A.29)$$
Solving equation (A.29) for \((u^*, \phi)\) gives

\[
\begin{pmatrix}
  u^* \\ \\
  \phi \\
\end{pmatrix} =
\begin{pmatrix}
  I & \Delta t B_2 G Y^{-1} \\
  0 & Y^{-1} \\
\end{pmatrix}
\begin{pmatrix}
  u \\ \\
  p \\
\end{pmatrix} +
\begin{pmatrix}
  -\Delta t B_3 G \hat{\phi} \\
  -\delta_1 \hat{\phi} - \delta_2 (G^T G)^{-1} G^T \left( \frac{-u}{\Delta t} + G \hat{\phi} \right) \\
\end{pmatrix}.
\]  

(A.30)

Plugging equation (A.30) into equation (A.28), we obtain

\[
\begin{pmatrix}
  A \\
  -G^T \\
\end{pmatrix}
\begin{pmatrix}
  0 \\
  \Delta t G^T B_1 G \\
\end{pmatrix}
\begin{pmatrix}
  I \\
  0 \\
\end{pmatrix}
\begin{pmatrix}
  \Delta t B_2 G Y^{-1} \\
  Y^{-1} \\
\end{pmatrix}
\begin{pmatrix}
  u \\ \\
  p \\
\end{pmatrix} +
\begin{pmatrix}
  -\Delta t B_3 G \hat{\phi} \\
  -\delta_1 \hat{\phi} - \delta_2 (G^T G)^{-1} G^T \left( \frac{-u}{\Delta t} + G \hat{\phi} \right) \\
\end{pmatrix} =
\begin{pmatrix}
  r \\
  z \\
\end{pmatrix} +
\begin{pmatrix}
  bc_1 \\
  bc_2 \\
\end{pmatrix} +
\begin{pmatrix}
  -\Delta t G \hat{\phi} \\
  \Delta t G^T B_3 G \hat{\phi} \\
\end{pmatrix}.
\]  

(A.31)

Multiplying matrices and bringing knowns to the right-hand side

\[
\begin{pmatrix}
  A \\
  -G^T \\
\end{pmatrix}
\begin{pmatrix}
  \Delta t A B_2 G Y^{-1} \\
  \Delta t G^T B_1 G Y^{-1} - \Delta t G^T B_2 G Y^{-1} \\
\end{pmatrix}
\begin{pmatrix}
  u \\ \\
  p \\
\end{pmatrix} =
\begin{pmatrix}
  r \\
  z \\
\end{pmatrix} +
\begin{pmatrix}
  bc_1 \\
  bc_2 \\
\end{pmatrix} +
\begin{pmatrix}
  -\Delta t G \hat{\phi} \\
  \Delta t G^T B_3 G \hat{\phi} \\
\end{pmatrix} -
\begin{pmatrix}
  A \\
  -G^T \\
\end{pmatrix}
\begin{pmatrix}
  \Delta t A B_2 G Y^{-1} \\
  \Delta t G^T B_1 G Y^{-1} - \Delta t G^T B_2 G Y^{-1} \\
\end{pmatrix}
\begin{pmatrix}
  -\Delta t B_3 G \hat{\phi} \\
  -\delta_1 \hat{\phi} - \delta_2 (G^T G)^{-1} G^T \left( \frac{-u}{\Delta t} + G \hat{\phi} \right) \\
\end{pmatrix}.
\]  

(A.32)

Rewriting the right-hand side yields

\[
\begin{pmatrix}
  A \\
  -G^T \\
\end{pmatrix}
\begin{pmatrix}
  \Delta t A B_2 G Y^{-1} \\
  \Delta t G^T B_1 G Y^{-1} - \Delta t G^T B_2 G Y^{-1} \\
\end{pmatrix}
\begin{pmatrix}
  u \\ \\
  p \\
\end{pmatrix} =
\begin{pmatrix}
  r \\
  z \\
\end{pmatrix} +
\begin{pmatrix}
  bc_1 \\
  bc_2 \\
\end{pmatrix} +
\Delta t \begin{pmatrix}
  -G + A B_3 G \\
  G^T B_3 G - G^T B_3 G \\
\end{pmatrix} \hat{\phi} +
\Delta t \begin{pmatrix}
  A B_2 G Y^{-1} \\
  G^T B_1 G Y^{-1} - G^T B_2 G Y^{-1} \\
\end{pmatrix} \delta_1 \hat{\phi} +
\Delta t \begin{pmatrix}
  A B_2 G Y^{-1} \\
  G^T B_1 G Y^{-1} - G^T B_2 G Y^{-1} \\
\end{pmatrix} \delta_2 (G^T G)^{-1} G^T \left( \frac{-u}{\Delta t} + G \hat{\phi} \right).
\]  

(A.33)

### A.4 Rotational-like Partial Commute Method

Inspired by the Brown-Cortez-Minion (rotational projection method) idea, we sought to obtain a more accurate pressure by “commuting as much as possible.” Noting that \(A_1\) commuted with \(G\) (at least with periodic boundary conditions), we
decided to try \( Y = A_1 \). Thus, we chose \( B_1 = B_2 = I, B_3 = 0, \delta_1 = 1, \delta_2 = 0, z = 0, G^T G = G^T G \), and \( Y = A_1 \) in equations (A.2)-(A.5). The set of equations these set of choices solve is

\[
\begin{pmatrix}
A & \Delta t A A_1^{-1} G \\
-G^T & 0 \\
0 & 0
\end{pmatrix}
\begin{pmatrix}
u \\
0 \\
p
\end{pmatrix}
=
\begin{pmatrix}
r \\
b c_1 \\
b c_2
\end{pmatrix}
+ \Delta t \begin{pmatrix}-G + A A_1^{-1} G \\
0
\end{pmatrix} \hat{p}. \quad \text{(A.34)}
\]

Now, since \( A = A_1 + A_2 \), we can simplify a bit to get

\[
\begin{pmatrix}
A & \Delta t G + \Delta t A_2 A_1^{-1} G \\
-G^T & 0 \\
0 & 0
\end{pmatrix}
\begin{pmatrix}
u \\
0 \\
p
\end{pmatrix}
=
\begin{pmatrix}
r \\
b c_1 \\
b c_2
\end{pmatrix}
+ \Delta t \begin{pmatrix}A_2 A_1^{-1} G \\
0
\end{pmatrix} \hat{p}. \quad \text{(A.35)}
\]

Because of the potentially large splitting error, we decided to embed this in an iterative method. We replaced \( \hat{p} \) with \( p^{n+\frac{1}{2}}q \), and took the first iteration to be \( p^{n+\frac{1}{2},0} = p^{n-1/2} \) and subsequent values of \( p^{n+\frac{1}{2},q} \) to be the value of \( p \) computed from the previous solve. Preliminary tests showed that this method would diverge in some cases (since \( \|A_2\|_p \) is much larger than either \( \|A_1\|_p \) or \( \|A_1^{-1}\|_p \)), and further that it was quite slow even for values of \( \gamma \) not very large.

## A.5 Alternative Derivation of the New Update Operator

Projection methods can be thought of in terms of having an approximation to \( p \) (namely \( \hat{p} \)) and coming up with a method for determining a velocity field from the pressure, and perhaps iterating the method. An alternative way to look at these equations is to think in terms of having an approximation to the velocity field and wanting to determine a corresponding pressure. Perhaps the simplest way of doing so is by taking the negative divergence of the momentum equation,

\[ Au + \Delta t G p = r + b c_1, \quad \text{(A.36)} \]

to obtain the following Poisson equation for \( p \)

\[ G^T A u + \Delta t G^T G p = G^T r + G^T b c_1. \quad \text{(A.37)} \]

If \( u \) is close to the correct velocity field, then solving this Poisson equation will yield a pressure that is close to the true pressure. One could therefore think of this
equation as defining a pressure that is consistent with a given approximation to the velocity field. We could think of solving this equation as an extra postprocessing step to a normal projection method that could be used to obtain a more “consistent” (and hopefully more accurate) pressure. Thinking of it in those terms, since we have expressions for \( u \) after a single step of a projection method, we can obtain an alternative expression for \( p \).

\[
\Delta t G^T G p = G^T r + G^T b c_1 - G^T A u \quad (A.38)
\]

\[
p = \frac{1}{\Delta t} (G^T G)^{-1} G^T (r + b c_1 - A u) \quad (A.39)
\]

\[
= \frac{1}{\Delta t} (G^T G)^{-1} G^T (r + b c_1 - A u^* - \Delta t B_2 G \phi) \quad (A.40)
\]

\[
= \frac{1}{\Delta t} (G^T G)^{-1} G^T (r + b c_1 - A u^* + \Delta t A B_2 G \phi) \quad (A.41)
\]

\[
= \frac{1}{\Delta t} (G^T G)^{-1} G^T (\Delta t G \hat{p} + \Delta t A B_2 G \phi) \quad (A.42)
\]

\[
= (G^T G)^{-1} G^T (G \hat{p} + A B_2 G \phi) \quad (A.43)
\]

\[
= \hat{p} + (G^T G)^{-1} G^T A B_2 G \phi \quad (A.44)
\]

If \( B_2 = I \) and \( A \) “commutes” with \( G \), then this just means \( p = \hat{p} + A \phi \), which is the Brown-Cortez-Minion choice. If \( B_2 = A^{-1} \), then this just means that \( p = \hat{p} + \phi \), which is the choice for the Uzawa method (technically, \( \hat{p} = 0 \) for the Uzawa method, but including \( \hat{p} \neq 0 \) with the Uzawa method results in a mathematically equivalent system). More generally, this pressure will be exactly the one produced by the new pressure-correction method when \( C = I \).

However, there is more than one way to define a “consistent” pressure. One could alternatively define a consistent pressure by applying \( A^{-1} \) and then the negative divergence to the momentum equation to yield the generalized Poisson equation

\[
G^T A^{-1} A u + \Delta t G^T A^{-1} G p = G^T A^{-1} r + G^T A^{-1} b c_1. \quad (A.45)
\]

If \( u \) is the true solution, solving this generalized Poisson equation yields the correct pressure. In fact, it yields the correct pressure if \( u \) is any field satisfying the motion constraints. Unfortunately, it is a computationally expensive (generalized) Poisson
problem to solve. In fact, this is the Uzawa method once again. However, this idea does suggest a range of other possibilities for defining a consistent pressure, namely generalized Poisson equations of the form

\[ G^T C A u + \Delta t G^T C G p = G^T C r + G^T C b c_1. \]  
(A.46)

If \( u \) is the correct velocity field, this generalized Poisson problem again provides the correct pressure field. The reason it provides the correct pressure field is that the true pressure, \( p^{\text{true}} \), satisfies the momentum equation,

\[ A u + \Delta t G p^{\text{true}} = r + b c_1, \]  
(A.47)

and thus (solving (A.46) for \( p \))

\[ p = \frac{1}{\Delta t} (G^T C G)^{-1} G^T C (r + b c_1 - A u) \]  
(A.48)

\[ = (G^T C G)^{-1} G^T C G p^{\text{true}} \]  
(A.49)

\[ = p^{\text{true}}. \]  
(A.50)

For approximations \( C \) sufficiently close to \( A^{-1} \) and \( u \) which satisfy the motion constraints (i.e., \( G^T u = 0 \)), solving (A.46) provides a pressure closer to the one from equation (A.45) than the one from equation (A.37), and which is thus more accurate than the one obtained by solving (A.37). Again using the fact that we have an expression for \( u \) if it is obtained via a single step of a standard projection method, we can obtain an expression for what the solution to this equation for \( p \) would be. Doing so, we obtain from

\[ \Delta t G^T C G p = G^T C (r + b c_1 - A u), \]  
(A.51)

that
\[ p = \frac{1}{\Delta t} (G^T CG)^{-1} G^T C (r + bc_1 - Au) \]
\[ = \frac{1}{\Delta t} (G^T CG)^{-1} G^T C (r + bc_1 - A(u^* - \Delta t B_2 G\phi)) \]
\[ = \frac{1}{\Delta t} (G^T CG)^{-1} G^T C (r + bc_1 - Au^* + \Delta t AB_2 G\phi) \]
\[ = \frac{1}{\Delta t} (G^T CG)^{-1} G^T C (\Delta t G\hat{\phi} + \Delta t AB_2 G\phi) \]
\[ = (G^T CG)^{-1} G^T C (G\hat{\phi} + AB_2 G\phi) \]
\[ = \hat{\phi} + (G^T CG)^{-1} G^T CAB_2 G\phi \]
\[ = \hat{\phi} + N\phi \]

Thus we see the choice \( Y = N \) appearing from a different derivation.

### A.6 Boundary Conditions

The method given by equations (A.2) through (A.5) specify how the boundary conditions on the velocity are included, but not very clearly. Also, it is not at all clear how the boundary conditions for the intermediate (generalized) Poisson equation variables are handled. (Even if you look at the problem as an algebraic system, there is still the question of what boundary conditions this algebraic system effectively enforces. Guermond [15] goes to great lengths to point out how important boundary conditions are and how all discrete systems are enforcing some kind of boundary condition whether the implementers are aware of it or not — and that the boundary conditions are crucial to overall accuracy.) We elucidate these details in this section.

The new pressure-correction method (see (A.20)-(A.25)) is

\[ Au^* = r + bc_1 - \Delta t G\hat{\phi} \]  \hspace{1cm} (A.53)
\[ \Delta t G^T B_1 G\phi_1 = G^T u^* + z + bc_2 \]  \hspace{1cm} (A.54)
\[ \hat{u} = u^* - \Delta t B_2 G\phi_1 \]  \hspace{1cm} (A.55)
\[ G^T CG\phi_2 = G^T CAB_2 G\phi_1 + bc_3 \]  \hspace{1cm} (A.56)
\[ p = \hat{\phi} + \phi_2 \]  \hspace{1cm} (A.57)
\[ u = \hat{u} + C(r + bc_1 - \Delta t Gp - Au). \]  \hspace{1cm} (A.58)
The velocity postprocessing step given by equation (A.58) may not be applicable for all problems (it depends on whether $C$ is close enough to $A^{-1}$ so that $AC - I$ is sufficiently small; if it is not, the postprocessing step is not desirable as it may increase the splitting error (compare (6.58) and (6.59)). Further, it is the combination of $\hat{u}$ and $p$ that satisfies both the momentum portion of equation (6.43) and the consistency equation (6.57). So we concentrate on the first 5 steps.

In (A.53) and (A.54), we are explicitly forcing $u^*$ (the “provisional” velocity field) to satisfy the same boundary conditions as the real velocity at timelevel $n + 1$. We expect the velocity postprocessing step to add a velocity field satisfying a homogenous boundary condition so that $\hat{u}$ should also satisfy the same boundary conditions as the velocity. Letting $n$ be the normal to the boundary at a gridpoint, these velocity boundary conditions in combination with equation (A.55) imply

$$n^T \hat{u} = n^T u^* - \Delta t n^T B_2 G \phi_1,$$

or

$$0 = n^T B_2 G \phi_1.$$

In other words, the decision to make $\hat{u}$ and $u^*$ both satisfy the same boundary conditions means that we need to apply a homogeneous Neumann-like boundary condition to $\phi_1$ (which is reflected in equation (A.54) already, as can be seen by comparing to the second equation of (A.1)). If we want $\phi_1$ to satisfy a different boundary condition, then we would need to adjust the boundary conditions on $u^*$ to compensate.

Given this Neumann-like condition, it is interesting to check whether the generalized Poisson problem in equation (A.54) is solvable. We know how to do so only in the case that $-G^T$ is a discrete divergence operator, but since that is an extremely common case it is worth checking. The discrete solvability condition for equation (A.54) is

$$\Delta t \sum_{\text{domain}} G^T B_1 G \phi_1 = \sum_{\text{domain}} G^T u^* + z + b c_2.$$
Applying a discrete form of the divergence theorem a few times and making use of the fact that $u^*$ has been forced to satisfy the same boundary conditions as the true solution $u_{true}$, we obtain

\[
\Delta t \sum_{\text{bdry}} n^T B_1 G \phi_1 = \sum_{\text{bdry}} n^T u^* + \sum_{\text{domain}} z + bc_2
\]  

(A.62)

\[
= \sum_{\text{bdry}} n^T u_{true} + \sum_{\text{domain}} z + bc_2
\]  

(A.63)

\[
= \sum_{\text{domain}} G^T u_{true} + z + bc_2
\]  

(A.64)

\[
= 0,
\]  

(A.65)

where $n$ denotes the local normal to the boundary. (Note that $G^T u_{true} + z + bc_2$ is 0 pointwise since $u_{true}$ satisfies the constraints equation). This solvability condition is satisfied automatically if $B_1 = B_2$ (due to equation (A.60)), but may present problems if that choice is not used. We could enforce the Neumann-like condition

\[
0 = n^T B_1 G \phi_1
\]  

(A.66)

instead of equation (A.60), but this may result in an inconsistency in the normal component of the computed velocity field at the boundary. (Note that projection methods in general already yield a velocity field with an incorrect tangential component along the boundary, so this would be an additional error.) To our knowledge, choosing $B_1 \neq B_2$ and the ramifications on solvability and error have not previously been discussed in the literature. An improvement would be to extrapolate the value of $n^T B_2 G \phi_1$ from previous timesteps and use it in equation (A.59) to obtain a modified boundary condition for $u^*$. Modifying the boundary conditions for $u^*$ means modifying $bc_1$ and $bc_2$ in equations (A.53) and (A.54).

The boundary condition for $\phi_2$ is somewhat different since there is no equation directly relating velocity approximations and $\phi_2$. It turns out we have some freedom in how we choose the boundary condition, and there are different ways to derive one. Our first derivation will assume that $-G^T$ is a discrete divergence operator and comes from looking at the momentum portion of equation (6.43) (which, as a
reminder, was derived for \( \hat{u} \) and \( \hat{p} \) in our present notation rather than \( u \) and \( p \).

This equation is

\[
A\hat{u} + \Delta tAB_2GN^{-1}p = r + bc_1 - \Delta tG\hat{p} + \Delta tAB_2GN^{-1}\hat{p}.
\] (A.67)

Using equations (A.55) and (A.57) to eliminate \( \hat{u} \) and \( \hat{p} \) we obtain

\[
A(u^+ - \Delta tB_2G\phi_1) + \Delta tAB_2GN^{-1}(\hat{p} + \phi_2) =
\]

\[
r + bc_1 - \Delta tG\hat{p} + \Delta tAB_2GN^{-1}\hat{p}.
\] (A.68)

Using (A.53), we obtain

\[
\begin{align*}
    r + bc_1 - \Delta tG\hat{p} - \Delta tAB_2G\phi_1 + \Delta tAB_2GN^{-1}\hat{p} + \Delta tAB_2GN^{-1}\phi_2 &=
    r + bc_1 - \Delta tG\hat{p} + \Delta tAB_2GN^{-1}\hat{p},
\end{align*}
\] (A.69)

which implies

\[
AB_2GN^{-1}\phi_2 = AB_2G\phi_1.
\] (A.70)

One way of enforcing this condition is to set

\[
N^{-1}\phi_2 = \phi_1.
\] (A.71)

Using the definition of \( N \) this is equivalent to

\[
G^TCG\phi_2 = G^TCAB_2G\phi_1,
\] (A.72)

which is (A.56), ignoring the boundary conditions. (Note that we could have substituted a different pressure update operator here instead of \( N \) if we wanted to come up with appropriate boundary conditions for other similar methods that employ a different pressure update operator.) Now we use the discrete divergence theorem on equation (A.72) to obtain a solvability condition

\[
\sum_{\text{bdry}} n^T CG\phi_2 = \sum_{\text{bdry}} n^T CAB_2G\phi_1,
\] (A.73)

which we can just apply pointwise to give us a Neumann-like boundary condition. This boundary condition can be entered into equation (A.56) using the \( bc_3 \) term.
The above derivation of the boundary condition for $\phi_2$ assumed that $-G^T$ was a discrete divergence operator. One alternative derivation that does not require such an assumption is to dot the normal vector with the momentum equation and enforce that the resulting equation hold at the boundary for $\dot{u}$ and $p$. Doing so, we obtain the equation

$$n^T A\dot{u} + \Delta t \ n^T Gp = n^T r + n^T bc_1. \quad (A.74)$$

Using equations (A.55), (A.57), and (A.53) to eliminate $\dot{u}$, $p$, and $u^*$ we obtain

$$n^T (r + bc_1 - \Delta t(G\dot{p})) - \Delta t \ n^T AB_2 G\phi_1 + \Delta t \ n^T G\dot{p} + \Delta t \ n^T G\phi_2 = n^T r + n^T bc_1. \quad (A.75)$$

This simplifies to

$$n^T G\phi_2 = n^T AB_2 G\phi_1, \quad (A.76)$$

which gives us a Neumann-like condition on $\phi_2$ (remember that $G$ may not be a discrete gradient operator, hence the usage of the terminology “Neumann-like”).

We could also repeat the above steps after multiplying the momentum equation through by $C$. In fact, this seems necessary to make the velocity postprocessing step consistent. Since the velocity postprocessing step may not always be applied, we would like both $u$ and $\dot{u}$ to satisfy the same boundary conditions. That means the correction term in equation (A.58) should have a homogeneous boundary condition, but the correction term is exactly $C$ times the residual of the momentum equation. So we multiply the momentum equation by $C$, then dot with the normal vector and force the resulting equation to hold at the boundary (which does not exactly enforce the homogeneous Dirichlet condition on the velocity since we have ignored the tangential component). The result is very similar to what we obtained before

$$n^T CG\phi_2 = n^T C AB_2 G\phi_1. \quad (A.77)$$

This is the same as applying equation (A.73) pointwise (which would also mean solvability is ensured when $-G^T$ is a discrete divergence operator).
A.7 Approximate Projections and the New Update Operator

As thoroughly discussed by Almgren et al. [2] and mentioned above in our highlights of their work, the choice of which approximate projection method to use is very important to the robustness of the overall method. Additional analysis, similar to that of Almgren et al., is needed to determine the effect of using the new pressure update operator (instead of the identity operator) in conjunction with approximate projections. This would require a full paper to explore in detail; however, we believe we have sufficient intuition about the methods to guess the likely outcome and feel confident in suggesting which scheme to use.

Perhaps the aspect of this problem that looks most challenging, at least initially, is that Almgren et al. found PM2 to be robust and PM1 to be problematic when approximate projections were used with the standard pressure update methods, while we found the opposite was true when either the Brown-Cortez-Minion or new pressure-correction scheme is used in conjunction with an exact projection. However, closer inspection reveals that the splitting errors occur in different areas, which suggests that this problem can be resolved. The differences in splitting error for either the Brown-Cortez-Minion or new pressure-correction schemes with different projection update methods affect only the momentum equation. The differences in splitting error for the three approximate projection schemes analyzed by Almgren et al. predominantly affect the constraints equation. The only overlap is in PM5 with an approximate projection which slightly modifies the momentum equation as well.

PM1 with the new update and an approximate projection (which we will refer to as N-A-PM1; note that the new update itself may contain an approximate projection; we underline it below as a reminder) solves the set of equations

\[
\begin{pmatrix}
A & \frac{\Delta t}{\rho} AB_2 G N^{-1} \\
-G^T & \frac{\Delta t}{\rho} LN^{-1} - \frac{\Delta t}{\rho} G^T G N^{-1}
\end{pmatrix}
\begin{pmatrix}
u \\
or \\
bc_1 \\
bc_2
\end{pmatrix}
= \begin{pmatrix} r \\ 0 \end{pmatrix} + \frac{\Delta t}{\rho} \begin{pmatrix} -G + AB_2 G N^{-1} \\
LN^{-1} - G^T G N^{-1}
\end{pmatrix} p^{n-1/2}.
\]

(A.78)
PM2 with the new update and an approximate projection (N-A-PM2) solves the set of equations

\[
\begin{pmatrix}
A & \frac{\Delta t}{\rho} AB_2 G N^{-1} \\
-G^T & \frac{\Delta t}{\rho} L N^{-1} - \frac{\Delta t}{\rho} G^T G N^{-1}
\end{pmatrix}
\begin{pmatrix}
u \\
p
\end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix} + \begin{pmatrix} bc_1 \\ bc_2 \end{pmatrix} + \frac{\Delta t}{\rho} \begin{pmatrix} -G + AG^T \\ 0 \end{pmatrix} p^{n-1/2}. \quad (A.79)
\]

PM5 with the new update and an approximate projection (N-A-PM5) solves the set of equations

\[
\begin{pmatrix}
A & \frac{\Delta t}{\rho} AB_2 G N^{-1} \\
-G^T & \frac{\Delta t}{\rho} L N^{-1} - \frac{\Delta t}{\rho} G^T G N^{-1}
\end{pmatrix}
\begin{pmatrix}
u \\
p
\end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix} + \begin{pmatrix} bc_1 \\ bc_2 \end{pmatrix} + \frac{\Delta t}{\rho} \begin{pmatrix} -G + AG^T \\ 0 \end{pmatrix} p^{n-1/2} + \left(\frac{\Delta t}{\rho} L N^{-1} - \frac{\Delta t}{\rho} G^T G N^{-1}\right) L^{-1} G^T \left(\frac{\rho u^n}{\Delta t} + G p^{n-1/2}\right). \quad (A.80)
\]

In the equation for N-A-PM5, note that \(L^{-1} G^T \left(\frac{\rho u^n}{\Delta t} + G p^{n-1/2}\right) = p^{n-1/2} + p_s\), where \(p_s\) should be small (e.g. \(O(h^2)\)). This is because if \(L\) is a good approximation to \(G^T G\), then \(u^n\) will approximately satisfy the constraints equation and \(L^{-1} G^T G \approx I\). So, we could rewrite the set of equations N-A-PM5 solves as

\[
\begin{pmatrix}
A & \frac{\Delta t}{\rho} AB_2 G N^{-1} \\
-G^T & \frac{\Delta t}{\rho} L N^{-1} - \frac{\Delta t}{\rho} G^T G N^{-1}
\end{pmatrix}
\begin{pmatrix}
u \\
p
\end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix} + \begin{pmatrix} bc_1 \\ bc_2 \end{pmatrix} + \frac{\Delta t}{\rho} \begin{pmatrix} -G + AB_2 G N^{-1} \\
L N^{-1} - G^T G N^{-1}
\end{pmatrix} p^{n-1/2} + \frac{\Delta t}{\rho} \begin{pmatrix} AB_2 G N^{-1} \\
L N^{-1} - G^T G N^{-1}
\end{pmatrix} p_s. \quad (A.81)
\]

Thus N-A-PM5 is very similar to N-A-PM1; the only difference is the \(p_s\) term which should be small.

The first thing to note here is that the splitting error in the momentum equation for N-A-PM1 is identical to the splitting error from the new pressure-correction method, which looks good. The splitting error in the momentum equation for N-A-PM2 looks unacceptable due to being unbalanced (see Section A.1.1). The splitting error in the momentum equation for N-A-PM5 is nearly identical to the splitting error for N-A-PM1, the difference being in the \(p_s\) term, which should be small.
Looking at the splitting error for the constraint equations for these three methods, we can predict that N-A-PM1 and N-A-PM5 should yield velocity fields that are closer to divergence-free than N-A-PM2. In fact, the constraint equations in these three methods look almost exactly the same as the constraint equations in A-PM1, A-PM2, and A-PM5 — the only difference is that $N^{-1}$ is a factor in the splitting error. Thus, being bold enough to assume that the similarity implies that the analysis from Almgren et al. on A-PM[125] carries over to N-A-PM[125], we conjecture that N-A-PM1 might go unstable but that N-A-PM5 should be stable (and N-A-PM2 is irrelevant because the splitting error in the momentum equation is unacceptable).
APPENDIX B

ENERGY CORRECTIONS

In Chapter 4 we explored some changes designed to fix the volume loss observed in the IB method. While we found we could preserve volume precisely with only a minimal amount of computational (or coding) effort, the change destabilized the system for some values of the numerical parameters. In this appendix, we present some possibilities we explored for attempting to fix these instabilities.

In Chapter 3, two things were shown: (a) the modified discretization proposed exactly conserves the sum of kinetic and potential energy when no viscosity is present, and (b) the method still suffers from volume loss when applied to a problem with a closed membrane under elastic tension (regardless of whether viscosity was present). This loss of volume results in a smaller potential energy due to the shortened distance between points. As the volume of the immersed object goes to zero, so too does the potential energy of the system. Because the numerical method conserves energy, this means that this loss of potential energy has to be compensated for by additional kinetic energy.

One can thus think of the modified discretization of Chapter 3 as a method that preserves total energy but that results in errors in both the potential and kinetic energy, with an excess of the latter and a shortage of the former. By forcing the volume to be conserved, the volume correction proposed in Chapter 4 fixes the loss of potential energy without addressing the increased kinetic energy.

Our original attempt to fix this was to try to solve

$$ KE_{\text{modified}}^{n+1} + PE_{\text{modified}}^{n+1} = KE^{n+1} + PE^{n+1} \quad (B.1) $$

for $KE_{\text{modified}}^{n+1}$, where $KE$ stands for kinetic energy (given by $\langle u, u \rangle_{\Omega_h}$) and $PE$ stands for potential energy (given by $\langle -A_f X, X \rangle_{\Gamma_h}$), and the "modified" subscript
means the value after postprocessing corrections from Chapter 4 and this appendix are applied. After getting the modified kinetic energy, we would set $u^{n+1}_{\text{modified}} = u^n + \alpha(u^{n+1} - u^n)$, where $\alpha$ was chosen so that $\langle u^{n+1}, u^{n+1} \rangle_{\Omega_h} = KE^{n+1}_{\text{modified}}$.

Unfortunately, this method can fail rather easily; consider the case of a single pressurized circular boundary at rest. A single timestep of the IB method will result in the boundary moving slightly inward (the IB method does not preserve this steady state). The volume correction will push the boundary back out uniformly so that the corrected potential energy is the same as the potential energy from the previous timestep, and then (B.1) will fail because it results in a negative kinetic energy (because viscosity has already removed some of the energy).

An alternative comes from the results of Chapter 3, noting that before postprocessing the energies between timesteps (for our implicit discretization) are related:

$$E[u, X]^{n+1} = E[u, X]^n + \text{viscous dissipation of energy}. \quad (B.2)$$

Writing this relation out gives us

$$\langle u^{n+1}, u^{n+1} \rangle_{\Omega_h} + PE^{n+1} = \langle u^n, u^n \rangle_{\Omega_h} + PE^n + \frac{\nu \Delta t}{2} \langle u^{n+1} + u^n, \Delta^h(u^{n+1} + u^n) \rangle. \quad (B.3)$$

We seek to have a similar formula hold after postprocessing, namely

$$\langle u^{n+1}_{\text{modified}}, u^{n+1}_{\text{modified}} \rangle_{\Omega_h} + PE^{n+1}_{\text{modified}} = \langle u^n, u^n \rangle_{\Omega_h} + PE^n + \frac{\nu \Delta t}{2} \langle u^{n+1}_{\text{modified}} + u^n, \Delta^h(u^{n+1}_{\text{modified}} + u^n) \rangle, \quad (B.4)$$

where $PE^{n+1}_{\text{modified}}$ comes from the volume correction of Chapter 4 and $u^{n+1}_{\text{modified}}$ is yet to be determined.

We require $u^{n+1}_{\text{modified}}$ to be of the form

$$u^{n+1}_{\text{modified}} = u^n + \alpha(u^{n+1} - u^n) \quad (B.5)$$

in an attempt to remove the excess kinetic energy added in the timestep, where it is hoped that $\alpha$ will be near 1 (in an extreme case like the single circular pressurized membrane at rest, it may be expected that $\alpha$ would be close to 0). Plugging the
corrected velocity (B.5) into the desired energy (B.4) yields a quadratic equation for $\alpha$:

$$
\langle u^n, u^n \rangle_{\Omega_h} + 2\alpha \langle u^n, u^{n+1} - u^n \rangle_{\Omega_h} + \alpha^2 \langle u^{n+1} - u^n, u^{n+1} - u^n \rangle_{\Omega_h} + PE_{\text{modified}}^{n+1} \\
= \langle u^n, u^n \rangle_{\Omega_h} + PE^n + \frac{\nu \Delta t}{2} \left( 4 \langle u^n, \Delta^h u^n \rangle_{\Omega_h} + 4\alpha \langle u^n, \Delta^h (u^{n+1} - u^n) \rangle_{\Omega_h} + \alpha^2 \langle u^{n+1} - u^n, \Delta^h (u^{n+1} - u^n) \rangle_{\Omega_h} \right)
$$

(B.6)

where we have used the symmetry of $\Delta^h$ to simplify. Collecting terms gives

$$
0 = \alpha^2 \left( \langle u^{n+1} - u^n, u^{n+1} - u^n \rangle_{\Omega_h} - \frac{\nu \Delta t}{2} \langle u^{n+1} - u^n, \Delta^h (u^{n+1} - u^n) \rangle_{\Omega_h} \right) \\
+ \alpha \left( 2 \langle u^n, u^{n+1} - u^n \rangle_{\Omega_h} - 4 \frac{\nu \Delta t}{2} \langle u^n, \Delta^h (u^{n+1} - u^n) \rangle_{\Omega_h} \right) + PE_{\text{modified}}^{n+1} - PE^n - 4 \frac{\nu \Delta t}{2} \langle u^n, \Delta^h u^n \rangle_{\Omega_h}
$$

(B.7)

If (B.7) can be solved for $\alpha$, then we can use it to define our modified velocity (B.5) and ensure that the system is stable (and that the system will exhibit the expected amount of dissipation due to viscosity). However, we found in practice that we could not always solve for $\alpha \in \mathbb{R}$ and further that the value of $\alpha$ obtained could be far from 1 in value and result in nonphysical effects (e.g., a sudden and strong reversal of motion within a single timestep).
REFERENCES


