University of Alberta

Direction-Splitting Schemes For Particulate Flows

by

John William Keating

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Applied Mathematics

Department of Mathematical and Statistical Sciences

Fall 2013
Edmonton, Alberta, Canada

Abstract

This thesis introduces a new temporally second-order accurate direction-splitting scheme for implicitly solving parabolic or elliptic partial differential equations in complex-shaped domains. While some other splitting schemes can be unstable in such domains, numerical evidence suggests that the new splitting scheme is unconditionally stable even when using non-commutative spatial operators. The new direction-splitting scheme is combined with other splitting schemes to produce an efficient numerical method for solving the incompressible Navier–Stokes equations. Finite differences using staggered grids and sharp boundary-fitting is used to achieve second-order spatial accuracy. The numerical method is extended to perform direct numerical simulations of particulate flows where each rigid particle is used as Dirichlet boundary conditions for the Navier–Stokes equations, and forces on each particle are computed by performing surface integrals of the fluid stress. The method is validated by reproducing experimental results, reproducing numerical results of other independent authors, and demonstrating second-order convergence on manufactured solutions. Particle collisions are handled using a dry viscoelastic soft-sphere model with sub-time stepping. An additional model based on lubrication theory is proposed and shown to agree with experiments of submerged collisions. The complete numerical method is suitable for parallel computing. Weak scaling results of a 3D fluidized bed simulation containing two million particles suggests that flows containing one billion particles could be computed on today's supercomputers.

Acknowledgements

I would like to thank both of my PhD supervisors, John Bowman and especially Peter Minev, for their time, direction, and financial support. I would also like to thank the Department of Mathematical and Statistical Sciences at the University of Alberta for providing me with teaching experience and financial support. Finally, I would like to thank my wife, Rebecca, for her input on my research and for looking after our trorg while I focused on writing this thesis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Fluid flows are important parts of many different scientific and engineering disciplines, from aircraft design to the study of the earth's core, and from nuclear reactors to weather prediction. In order to know the outcome of a fluid flow, experiments can sometimes be performed. However, often performing an experiment is too costly, too difficult, or simply impossible. In this case, one must use a mathematical model of some kind. One such model that agrees very well with experiments is the Navier–Stokes equations. The problem, however, is that the Navier–Stokes equations are practically impossible to solve analytically in all but the most simple situations. In fact, the well-posedness of the 3D Navier–Stokes equations is one of the most important open problems in mathematics. Therefore, with the availability of fast computers, an excellent method for predicting fluid flows is to approximate solutions of the Navier–Stokes equations numerically. The field of Computational Fluid Dynamics (CFD) includes the design and use of computer methods to solve the Navier–Stokes equations, and the field of Numerical Analysis involves the design and study of these methods.

The work in this thesis is a mixture of CFD and Numerical Analysis. We are not interested in solving a single specific fluid problem, but rather we are interested in designing improved computer methods that can be used to solve a wide range of fluid problems. Specifically, we introduce some "direction splitting" innovations to reduce computer resources while also incorporating any required boundary-fitting to increase accuracy. The work in this thesis applies not only to the Navier–Stokes equations, but to other partial differential equations as well. We will, however, focus our attention on Direct Numerical Simulations (DNS) of the incompressible Navier–Stokes equations

in a complex-shaped domain, and in particular, fluid flows containing solid particles, i.e., particulate flows. Most fluids are incompressible to high accuracy in almost all situations, and even slow moving gases can often be treated as incompressible. DNS means that when we solve the Navier–Stokes equations, any turbulence in the flow must be resolved with finer numerical resolution rather than approximated with a turbulence model. Also, DNS in the context of particulate flows means that the shape of each individual particle and surrounding fluid must be sufficiently resolved by the spatial grid, rather than approximating particles as points or treating clouds of particles using statistics. Treating each solid particle individually poses some numerical difficulties when these particles are predicted to collide, and we also spend some time on this issue.

Often the presence of solid particles in a fluid flow changes the behaviour of the flow, for example, the particles may form clumps or the flow may form plumes, etc. As a result of using DNS, the numerical methods in this thesis could be used to determine the properties such flows in order to construct non-DNS models, or in some cases, real problems could be solved using the DNS itself. Some examples of real particulate flow applications include industrial mixing or sedimentation, washing of food grains, combustion of powders, erosion, and transport of pollutants.

This thesis is organized as follows. In the remainder of this introduction, we review some literature related to direction splitting and numerical methods for particulate flows. In Chapter 2, we present the set of equations that govern particulate flows, and perform the standard nondimensionalization. In Chapter 3, we discuss all of the spatial operators that appear in the Navier–Stokes equations, and also the corresponding discretizations used for the numerical methods in the thesis. In Chapter 4, we discuss numerical methods for solving incompressible Navier–Stokes, including innovations of direction-splitting for use in complex-shaped domains. In Chapter 5, we demonstrate the stability and accuracy of the new direction-splitting methods when solving equations in complex-shaped domains. In Chapter 6, we discuss the equations of motion of submerged solid particles and present discretizations of these equations. In Chapter 7, we discuss particle collisions in more detail and present a model that could be used to resolve collisions with higher accuracy on small scales. In Chapters 8 and 9, we compare the numerical schemes in this thesis to other numerical schemes and experimental results of real fluid flow problems. In Chapter 10 we demonstrate the power of the new methods in this thesis to solve large problems using direct numerical simulation.

1.1 Literature Review

Because of their efficiency for solving high dimensional problems, direction splitting methods have been widely used in the past to solve parabolic and elliptic Partial Differential Equations (PDEs). These methods are called "alternating-direction-implicit" methods in North America, and "fractional-step" or "locally-one-dimensional" methods in Russia. The first direction-splitting method was introduced by Peaceman and Rachford [54] in 1955 for two-dimensional (2D) problems, and later in 1959 and 1962, Yanenko [72] and Douglas [20] extended these ideas for 3D problems. The Russian school of numerical analysis contributed significantly to the development of direction-splitting methods, which is summarized in the review paper by Mitchell [50] or the more recent book [59] by Samarskii, Matus, and Vabishchevich. Direction-splitting methods have recently been abandoned in favor of multigrid methods, however, Guermond and Minev [27] recently proposed a direction-splitting scheme for approximating the incompressibility constraint for the incompressible Navier–Stokes equations. When a direction-splitting scheme is also used for the Navier–Stokes momentum equation, the complete scheme involves only one-dimensional problems that can be solved using a direct solver, and this can be more efficient than multigrid methods in a parallel-computing environment, especially for complex-shaped time-dependent domains.

However, the generally accepted opinion is still that direction-splitting methods are applicable only to problems in simple-shaped domains, such as a rectangular box. This thesis contains some new modifications of direction-splitting methods which (by numerical evidence only) are applicable to domains of any shape.

The primary difficulty of solving particulate flows numerically is that the fluid domain Ω_f has a very complex time-dependent shape. This is because each solid particle is essentially a "hole" in the fluid domain. There are two major approaches for solving Navier–Stokes in such a domain. The first approach is to discretize the complex-shaped domain directly. The second approach is to extend the problem from the original complex-shaped domain Ω_f to a larger, simple-shaped domain Ω, and methods which use this approach are called Fictitious Domain Methods (FDM). This allows the problem to be discretized on the same grid at all time levels, which is computationally cheaper.

By far the most popular and easy to implement is the FDM approach. In this case, the original boundary conditions on the original boundary $\partial\Omega_f$ must somehow

be enforced on the fictitious fluid in $\Omega \setminus \Omega_f$ which is inside the larger domain Ω.

One FDM approach of imposing internal boundary conditions is to introduce a Dirac-measure force in the fluid momentum equations in the fictitious regions. Peskin [55], [56] introduced this idea in his immersed boundary method (IBM) by coupling individual grid points together using an elastic force to give the fictitious fluid viscoelastic properties. When using finite volumes on a staggered grid, Kim et al. [43] presented an IBM which has improved accuracy in complex-shaped domains by using a mass-source term. The IBM has also been used to simulate moving solid particles by approximating the solid as a fluid bound together with springs and tethers, for example, [23].

Another FDM approach of imposing these internal boundary conditions is to use penalty terms which enforce the desired constraints. For example, the Dirichlet boundary condition $\mathbf{u} = C(\mathbf{x})$ can be imposed on the boundary $\partial\Omega_f$ by adding the extra term $\frac{1}{\epsilon}(\mathbf{u} - C(\mathbf{x}))$ to the equations in $\Omega \setminus \Omega_f$, where ϵ is a small parameter. This L^2 penalty procedure generates the Brinkman equations, which are analyzed in the context of FDM by Angot [2]. Another example of a FDM using this type of penalty procedure is given in [17]. A stronger H^1 penalty procedure can be obtained by setting the fluid viscosity to a large value $\mu = \frac{1}{\epsilon}$ in the fictitious region $\Omega \setminus \Omega_f$ in order to force rigid motion there. However, this produces spatially dependent viscous terms with a huge jump discontinuity that causes severe time step restrictions for any explicit scheme. In this case, an implicit formulation is required, such as [3].

Yet another FDM approach of imposing internal boundary conditions is to use a side constraint (Lagrange multiplier) to the fluid equations in the extended domain Ω. This approach requires the solution of a saddle-point problem, and the idea has been developed by Glowinski and co-workers [24], [25]. As shown by Patankar [53], it is possible to reformulate the side constraints as a series of explicit steps applying only to the fictitious fluid $\Omega \setminus \Omega_f$ rather than the entire domain Ω. The explicit formulation is quite popular and variations of it are used in [62], [68], [70], [6], and [5], the main differences between each of these being the way in which the density jump from fluid to solid is handled, and whether or not iterations are performed in order to enforce the constraints on $\Omega \setminus \Omega_f$.

A common problem for all FDM, which generally do not perform boundary-fitting of the grid to Ω_f, is that the spatial accuracy is usually only first order. Maury explains this fact in the introduction of [10]. In order to recover second-order spatial accuracy in FDM methods, a boundary-fitted sub-problem is typically solved, as in

[75] or [10].

In recent years, a wide variety of numerical schemes based on the Lattice-Boltzmann Method (LBM) have also been used to solve particulate flow problems. Being a fully explicit scheme, the LBM is straightforward to optimize for parallel computing, and such implementations have been used to solve very large problems - for example [26], which efficiently utilized 294912 processor cores. The LBM has some advantages simulating compressible flows at very high Reynolds numbers (denoted Re), but it also has several disadvantages when it is used to approximate the unsteady incompressible Navier–Stokes equations. The LBM is a relaxation method which converges to the solution of Navier–Stokes only when the diffusion time scale is not too large compared to the advection time scale [14]. According to [22], the LBM should use a small time step $\Delta t \sim \mathrm{Re}\, h^2$ to handle viscous effects correctly, where h is the spatial grid step. For particulate flows, the local Reynolds number is not usually very large, so this is a severe time step restriction. Also, since the LBM is known to have error proportional to the square of a computational Mach number $\sqrt{3}\Delta t|\mathbf{u}|/h$, this imposes another strict CFL-type time step restriction of the form $\Delta t \ll h/|\mathbf{u}|$. Finally, the LBM does a poor job of enforcing incompressibility unless Δt is very small (see Section 4.1.5 of this thesis for more discussion).

An alternative to FDM and LBM methods is to solve the Navier–Stokes equations in the complex-shaped domain Ω_f directly using a grid that conforms to the boundary $\partial\Omega_f$ at all times. One example of this technique is the Arbitrary-Lagrangian-Eulerian method of [37], which uses a finite-element discretization on a moving mesh. Whenever the mesh becomes too distorted, it needs to be recreated from scratch and the solution needs to be projected onto the new mesh. Another example is the Element-Free-Galerkin method of [74], which uses a node-based finite-element discretization rather than a mesh-based discretization. In this case, finite-element basis functions are constructed at run time using node points, and the nodes can be moved, created, or deleted as the domain changes shape. With these methods, forces on submerged boundaries are typically computed directly using surface integrals, and grid refinement can be performed around each object. These methods are very accurate but computationally expensive when Ω_f changes significantly in time, as in particulate flows.

Another method of discretizing the complex-shaped domain Ω_f directly is to use the method of "ghost cells". In this case, a fixed Cartesian grid is used to discretize Ω_f and additional ghost cells are included outside Ω_f near the boundary such that

the smooth solution of $\mathbf{u}$ in this extended domain satisfies the boundary conditions on $\partial\Omega_f$ to high spatial accuracy. Examples of such methods are [48] and [51], both of which are second-order spatially accurate. The main challenge of these methods is choosing appropriate values for the ghost cells in the multi-dimensional case. The multi-dimensional equations are typically solved using multigrid techniques.

All numerical methods that deal with moving objects must somehow prevent the objects from overlapping. Most examples in the literature prevent overlap by introducing a short-range repulsive force to keep the objects apart, as in [24]. However, there are examples (such as [74]) where no ranged force is used. In this case, collisions are typically resolved using either a hard-sphere model such as [69], a thin-fluid layer lubrication model such as [49], or by allowing the objects to deform elastically (soft-sphere model). The review [19] compares hard and soft sphere models for the specific problem of fluidized beds.

Chapter 2

Model Derivation

We are interested in modeling the flow of many solid, rigid objects which are suspended in a viscous incompressible Newtonian fluid. For simplicity, we will assume that the solid objects are uniform density spheres and that the fluid has constant density and constant viscosity; however, the ideas in this thesis could be extended to relax these assumptions. In Cartesian coordinates, the fluid occupying a domain Ω_f is modeled by the Navier–Stokes equations, which consist of the momentum equation,

$$\underbrace{\rho_f \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right)}_{\text{density} \times \text{Eulerian acceleration}} = \underbrace{-\nabla p}_{\text{force due to pressure}} + \underbrace{\mu \nabla^2 \mathbf{u}}_{\text{force due to viscosity}} + \underbrace{\rho_f g \mathbf{e}_g}_{\text{force due to gravity}}, \quad (2.1)$$

together with the incompressibility constraint,

$$\nabla \cdot \mathbf{u} = 0, \quad (2.2)$$

where p is the total pressure, ρ_f is the fluid density, μ is the dynamic viscosity, $g = 9.8\text{m/s}^2$ is the acceleration due to gravity, and $\mathbf{e}_g = (0, 0, -1)$ is the unit vector in the direction of gravity.

The equations of motion of the i^{th} solid object occupying the domain Ω_i, $i = 1, \ldots, N$, are given by the relation between the object's velocity $\mathbf{U}_i$ and center of mass position $\mathbf{X}_i$,

$$\mathbf{U}_i = \frac{d\mathbf{X}_i}{dt}, \quad (2.3)$$

the object's momentum equation,

$$\underbrace{M_i \frac{d\mathbf{U}_i}{dt}}_{\text{mass} \times \text{acceleration}} = \underbrace{M_i g \mathbf{e}_g}_{\text{force due to gravity}} + \underbrace{\int_{\partial\Omega_i} \boldsymbol{\sigma} \cdot \mathbf{n} \, dS}_{\text{force due to fluid}}, \tag{2.4}$$

where $M_i = \rho_i V_i$ is the mass of the i^{th} object, ρ_i is its density, $V_i = \frac{4}{3}\pi r_i^3$ is its volume, r_i is its radius, $\mathbf{n}$ is the unit normal pointing out of the object, $\boldsymbol{\sigma} = -p\boldsymbol{\delta} + \mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)$ is the stress tensor of the fluid, and $\boldsymbol{\delta}$ is the Kronecker delta tensor (identity matrix). Finally, there is the object's angular momentum equation,

$$\underbrace{\mathbf{I}_i \frac{d\boldsymbol{\omega}_i}{dt}}_{\text{moment of inertia} \times \text{angular acceleration}} = \underbrace{\int_{\partial\Omega_i} (\mathbf{x} - \mathbf{X}_i) \times (\boldsymbol{\sigma} \cdot \mathbf{n}) \, dS}_{\text{torque due to fluid}}, \tag{2.5}$$

where $\mathbf{I}_i = \frac{2}{5} M_i r_i^2 \boldsymbol{\delta}$ is the inertia tensor of the i^{th} solid spherical object. The fluid stress forces in equations (2.4) and (2.5) are discussed in [37], and the equations of rigid body dynamics can be found in books on classical mechanics such as [64].

The fluid in Ω_f must satisfy homogeneous Dirichlet (no flux, no slip) boundary conditions on the boundaries of all of the object domains Ω_i. On the external boundary of Ω_f, either periodic, Dirichlet, or Neumann boundary conditions can be used, depending on the problem.

2.1 Nondimensionalization

We first choose a characteristic length L_c and velocity U_c, and we nondimensionalize the above equations by choosing new variables (indicated by a tilde):

$$\tilde{\mathbf{x}} = \frac{\mathbf{x}}{L_c}, \qquad \tilde{\mathbf{u}} = \frac{\mathbf{u}}{U_c}, \qquad \tilde{t} = \frac{U_c}{L_c} t, \qquad \tilde{p} = \frac{1}{\rho_f U_c^2} (p + \rho_f g L_c \tilde{z}) \tag{2.6}$$

With these scalings, the inverse relations are

$$\mathbf{x} = L_c \tilde{\mathbf{x}}, \qquad \mathbf{u} = U_c \tilde{\mathbf{u}}, \qquad t = \frac{L_c}{U_c} \tilde{t}, \qquad p = \rho_f U_c^2 \tilde{p} - \rho_f g L_c \tilde{z}, \tag{2.7}$$

and the derivative relations are

$$\frac{\partial}{\partial t} = \frac{U_c}{L_c}\frac{\partial}{\partial \tilde{t}} \quad \text{and} \quad \nabla := \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right) = \frac{1}{L_c}\left(\frac{\partial}{\partial \tilde{x}}, \frac{\partial}{\partial \tilde{y}}, \frac{\partial}{\partial \tilde{z}}\right) =: \frac{1}{L_c}\tilde{\nabla}. \tag{2.8}$$

Note that p is a total pressure and $\tilde{p}$ is a dynamic pressure. Writing the Navier–Stokes momentum equation (2.1) in terms of the new variables yields

$$\frac{\rho_f U_c^2}{L_c}\left(\frac{\partial \tilde{\mathbf{u}}}{\partial \tilde{t}} + \tilde{\mathbf{u}} \cdot \tilde{\nabla}\tilde{\mathbf{u}}\right) = -\frac{\rho_f U_c^2}{L_c}\tilde{\nabla}\tilde{p} - \cancel{\rho_f g \tilde{\nabla}(-\tilde{z})} + \frac{\mu U_c}{L_c^2}\tilde{\nabla}^2\tilde{\mathbf{u}} + \cancel{\rho_f g \mathbf{e}_g}. \tag{2.9}$$

Then, dividing out constants we get

$$\frac{\partial \tilde{\mathbf{u}}}{\partial \tilde{t}} + \tilde{\mathbf{u}} \cdot \tilde{\nabla}\tilde{\mathbf{u}} = -\tilde{\nabla}\tilde{p} + \frac{1}{\mathrm{Re}}\tilde{\nabla}^2\tilde{\mathbf{u}}, \tag{2.10}$$

where $\mathrm{Re} = \frac{\rho_f U_c L_c}{\mu}$ is the Reynolds number. A very high Reynolds number indicates that the flow will contain turbulence.

The incompressibility constraint (2.2) can be written in terms of the new variables as

$$\frac{U_c}{L_c}\tilde{\nabla} \cdot \tilde{\mathbf{u}} = 0, \quad \text{which is just} \quad \tilde{\nabla} \cdot \tilde{\mathbf{u}} = 0. \tag{2.11}$$

Writing the object position equation (2.3) in terms of the new variables yields

$$U_c \tilde{\mathbf{U}}_i = L_c \frac{U_c}{L_c}\frac{d\tilde{\mathbf{X}}_i}{d\tilde{t}}, \quad \text{which is just} \quad \tilde{\mathbf{U}}_i = \frac{d\tilde{\mathbf{X}}_i}{d\tilde{t}}. \tag{2.12}$$

Writing the object momentum equation (2.4) in terms of the new variables yields

$$\rho_f L_c^3 \tilde{M}_i \frac{U_c^2}{L_c}\frac{d\tilde{\mathbf{U}}_i}{d\tilde{t}} = \rho_f L_c^3 \tilde{M}_i g \mathbf{e}_g + \int_{\widetilde{\partial\Omega}_i} \tilde{\sigma} \cdot \mathbf{n} L_c^2 \, \widetilde{dS}, \tag{2.13}$$

where $\tilde{M}_i$ and $\tilde{\sigma}$ are defined from

$$M_i = \rho_i V_i = \rho_f L_c^3 \left(\frac{\rho_i}{\rho_f}\tilde{V}_i\right) = \rho_f L_c^3 \tilde{M}_i, \tag{2.14}$$

$$\tilde{\sigma} = \left(-\rho_f U_c^2 \tilde{p} + \rho_f g L_c \tilde{z}\right) \boldsymbol{\delta} + \frac{U_c}{L_c} \mu \left(\tilde{\nabla}\tilde{\mathbf{u}} + (\tilde{\nabla}\tilde{\mathbf{u}})^T\right). \tag{2.15}$$

By dividing out constants and defining the Froude number $\mathrm{Fr} = \frac{U_c^2}{gL_c}$, we can rewrite (2.13) and (2.15) as

$$\tilde{M}_i \frac{d\tilde{\mathbf{U}}_i}{d\tilde{t}} = \frac{\tilde{M}_i}{\mathrm{Fr}} \mathbf{e}_g + \int\limits_{\widetilde{\partial\Omega}_i} \left[\left(-\tilde{p} + \frac{1}{\mathrm{Fr}}\tilde{z}\right)\boldsymbol{\delta} + \frac{1}{\mathrm{Re}}\left(\tilde{\nabla}\tilde{\mathbf{u}} + (\tilde{\nabla}\tilde{\mathbf{u}})^T\right)\right] \cdot \mathbf{n}\, \widetilde{dS}. \tag{2.16}$$

Using Gauss's divergence theorem,

$$\int\limits_{\widetilde{\partial\Omega}_i} (\tilde{z}\boldsymbol{\delta}) \cdot \mathbf{n}\, \widetilde{dS} = \int\limits_{\widetilde{\Omega}_i} \tilde{\nabla} \cdot (\tilde{z}\boldsymbol{\delta})\, \widetilde{dV} = \left(0,\, 0,\, \int\limits_{\widetilde{\Omega}_i} \frac{\partial}{\partial \tilde{z}} \tilde{z}\, \widetilde{dV}\right) = \left(0, 0, \tilde{V}_i\right) = -\tilde{V}_i \mathbf{e}_g. \tag{2.17}$$

Therefore, (2.16) can finally be written as

$$\tilde{M}_i \frac{d\tilde{\mathbf{U}}_i}{d\tilde{t}} = \frac{\tilde{M}_i}{\mathrm{Fr}} \left(1 - \frac{\rho_f}{\rho_i}\right) \mathbf{e}_g + \int\limits_{\widetilde{\partial\Omega}_i} \left[-\tilde{p}\boldsymbol{\delta} + \frac{1}{\mathrm{Re}}\left(\tilde{\nabla}\tilde{\mathbf{u}} + (\tilde{\nabla}\tilde{\mathbf{u}})^T\right)\right] \cdot \mathbf{n}\, \widetilde{dS}. \tag{2.18}$$

Writing the angular momentum equation (2.5) in terms of the new variables yields

$$\rho_f L_c^5 \tilde{\mathbf{I}}_i \frac{U_c^2}{L_c^2} \frac{d\boldsymbol{\omega}_i}{d\tilde{t}} = \int\limits_{\widetilde{\partial\Omega}_i} L_c(\tilde{\mathbf{x}} - \tilde{\mathbf{X}}_i) \times (\tilde{\sigma} \cdot \mathbf{n}) L_c^2\, \widetilde{dS}, \tag{2.19}$$

where $\tilde{\sigma}$ is given by (2.15) and

$$\mathbf{I}_i = \frac{2}{5} V_i r_i^2 \rho_i \boldsymbol{\delta} = \rho_f L_c^5 \left(\frac{2}{5} \tilde{V}_i \tilde{r_i}^2 \frac{\rho_i}{\rho_f} \boldsymbol{\delta}\right) = \rho_f L_c^5 \tilde{\mathbf{I}}_i. \tag{2.20}$$

Note that for a sphere, $\tilde{\mathbf{x}} - \tilde{\mathbf{X}}_i = \tilde{r_i}\mathbf{n}$, and $\mathbf{n} \times (\boldsymbol{\delta} \cdot \mathbf{n}) = \mathbf{n} \times \mathbf{n} = 0$, so the pressure terms in the stress do not contribute to the angular momentum equation. Therefore,

after dividing out constants, (2.19) and (2.15) finally simplifies to

$$\tilde{\mathbf{I}}_i \frac{d\boldsymbol{\omega}_i}{d\tilde{t}} = \int\limits_{\widetilde{\partial\Omega}_i} (\tilde{\mathbf{x}} - \tilde{\mathbf{X}}_i) \times \left[\frac{1}{\mathrm{Re}} \left(\tilde{\nabla}\tilde{\mathbf{u}} + (\tilde{\nabla}\tilde{\mathbf{u}})^T \right) \cdot \mathbf{n} \right] \widetilde{dS}. \tag{2.21}$$

We now drop the tildes from the nondimensionalized variables and summarize the nondimensionalized equations that we will eventually solve:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} &= -\nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u}, \\ \nabla \cdot \mathbf{u} &= 0, \end{aligned} \tag{2.22}$$

$$\mathbf{U}_i = \frac{d\mathbf{X}_i}{dt}, \tag{2.23}$$

$$M_i \frac{d\mathbf{U}_i}{dt} = \frac{M_i}{\mathrm{Fr}} \left(1 - \frac{\rho_f}{\rho_i} \right) \mathbf{e}_g + \int\limits_{\partial\Omega_i} \left[-p\boldsymbol{\delta} + \frac{1}{\mathrm{Re}} \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T \right) \right] \cdot \mathbf{n} \, dS. \tag{2.24}$$

$$\mathbf{I}_i \frac{d\boldsymbol{\omega}_i}{dt} = \int\limits_{\partial\Omega_i} (\mathbf{x} - \mathbf{X}_i) \times \left[\frac{1}{\mathrm{Re}} \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T \right) \cdot \mathbf{n} \right] dS. \tag{2.25}$$

Note: In some cases we may be interested in the unsteady Stokes equations,

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} &= -\nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u}, \\ \nabla \cdot \mathbf{u} &= 0, \end{aligned} \tag{2.26}$$

which are obtained from (2.22) by omitting the advection terms $\mathbf{u} \cdot \nabla \mathbf{u}$. We will also refer to the unsteady Stokes equations in this thesis as just the Stokes equations since the time derivative will always be present.

Chapter 3

Spatial Operators

In this thesis we will use finite differences and the well known Marker And Cell (MAC) grid, which is a structured grid of rectangular cells. Since using collocated grids for the velocity and pressure is well known to produce unstable discretizations, the MAC discretization uses staggered grids for each component of the velocity and the pressure. Figure 3.1 shows what a MAC cell looks like in 2D and 3D. The velocity is always stored on the cell faces in the center if the face, and the pressure is always stored in the volumetric center of the cell. A MAC cell can have different widths in each direction, denoted h_x, h_y, h_z in 3D such that the volume of the MAC cell is $h_x h_y h_z$. If the cell is a cube, then we simply call the cell width h and it has volume h^3. A grid of MAC cells is called a "uniform grid" if all cells are identical, even if $h_x \neq h_y \neq h_z$. A uniform grid of 2D MAC cells is shown in Figure 3.2, where we can see that a curved boundary must be approximated as a "stair-step" and any given MAC cell is considered completely on one side of the boundary or the other side. However, we can also consider separately the grid points involving each velocity component (u, v, w), and the lines connecting these points are called grid lines. Curved boundaries can be intersected with these grid lines as in Figure 3.2.

Throughout this thesis, we will sometimes use the notation $\partial_x f$ and f_x to mean $\frac{\partial f}{\partial x}$. This should not be confused with the notation f_i, which means the i^{th} spatial point of the discretization of f, nor the notation f^n which means f at time step n.

In the remainder of this chapter, we will discuss several finite difference discretizations of spatial operators.

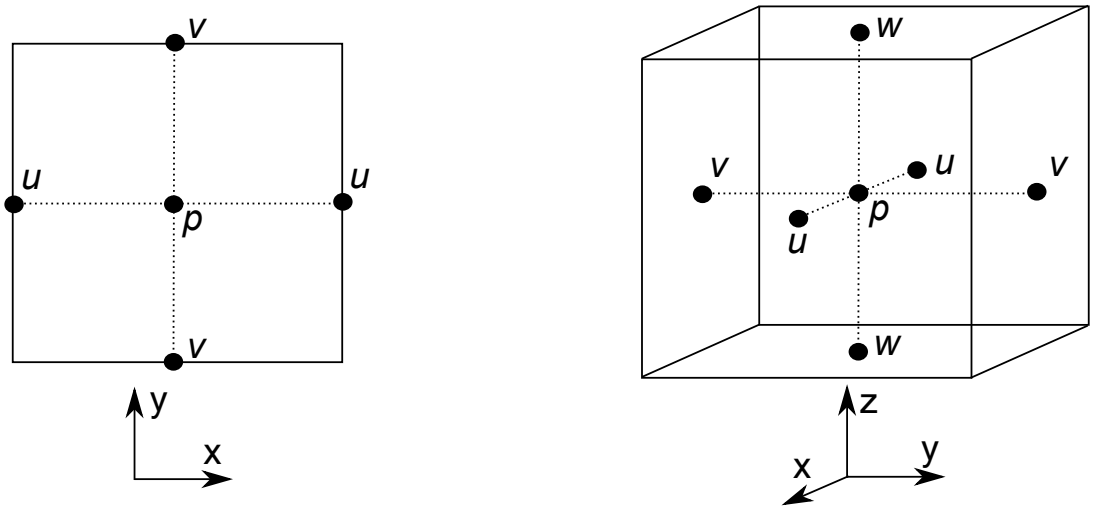

Figure 3.1: 2D MAC cell (left) where the velocity is $\mathbf{u} = (u, v)$, and a 3D MAC cell (right), where the fluid velocity is $\mathbf{u} = (u, v, w)$.

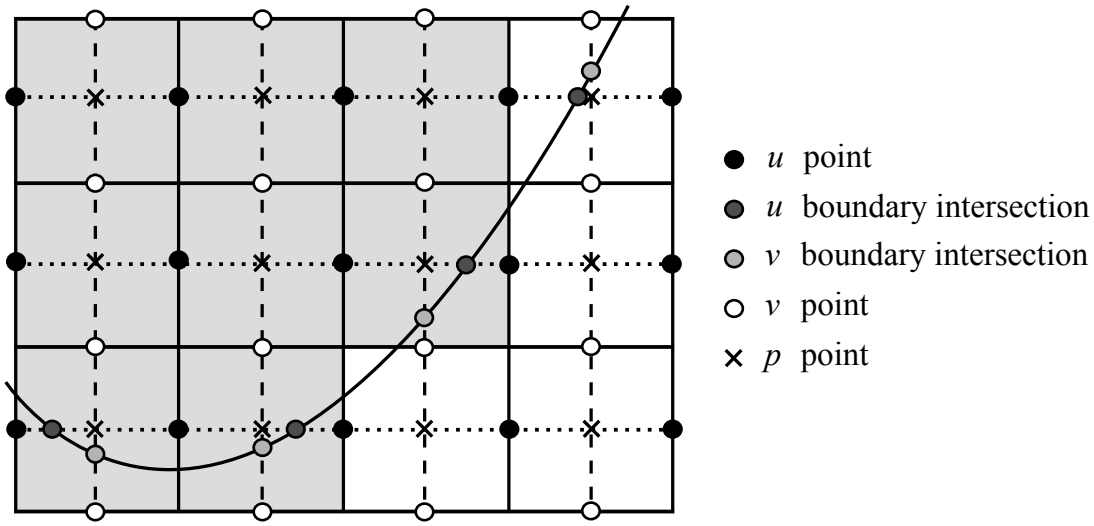

Figure 3.2: A 2D grid of $4 \times 3 = 12$ MAC cells in the presence of a curved boundary. The shaded cells are considered to be on the "solid" side of the boundary, and the white cells are considered to be on the "fluid" side. Horizontal grid lines connecting the u velocity points are shown with dotted lines, and vertical grid lines connecting the v velocity points are shown with dashed lines. Vertical grid lines connecting the u points and horizontal grid lines connecting the v points are shown with solid black lines. The solid black lines also define the MAC cell boundaries.

3.1 Laplace Operator $\nabla^2\mathbf{u}$

In this section we consider discretizations of the first and second spatial derivatives of the velocity. The first derivatives are included for completeness, but we are mostly interested in the second derivatives because the diffusion operator in Navier–Stokes (assuming constant viscosity, a Newtonian fluid, and Cartesian coordinates) is the Laplacian,

$$\nabla^2\mathbf{u} = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)\mathbf{u}. \tag{3.1}$$

Since spatial derivatives of the velocity are computed the same in all directions, we consider the x direction only. Consider three subsequent velocity grid nodes u_{i-1}, u_i, u_{i+1} located at corresponding points x_{i-1}, x_i, x_{i+1} on a grid line as in Figure 3.3a. Let $h_{i-1} = x_i - x_{i-1}$ and $h_i = x_{i+1} - x_i$. The standard three-point finite difference approximations of first and second derivatives at the point x_i are

$$\left(\frac{\partial u}{\partial x}\right)_i \approx \frac{1}{h_{i-1}+h_i}\left[\frac{h_i}{h_{i-1}}\left(u_i - u_{i-1}\right) + \frac{h_{i-1}}{h_i}\left(u_{i+1} - u_i\right)\right], \tag{3.2}$$

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i \approx \frac{2}{h_{i-1}+h_i}\left[\frac{u_{i+1}-u_i}{h_i} - \frac{u_i - u_{i-1}}{h_{i-1}}\right]. \tag{3.3}$$

The first derivative stencil is second-order spatially accurate on any grid, and the second derivative stencil is second-order spatially accurate if the grid is at least quasi-uniform (see [47], Theorem 2.1).

Now, consider Figure 3.3b where point x_{i-1} is a solid point, i.e., a "ghost point" outside the computational domain of the fluid. In this case, denote the point where the grid line intersects the solid boundary by x_{bL}, denote the velocity value at this point by u_{bL}, and let $\gamma_L = x_i - x_{bL}$. Using the standard stencil (3.2) or (3.3) in this situation effectively moves the domain boundary from the point x_{bL} to the point x_{i-1}, resulting in a loss of accuracy. In order to maintain second-order spatial accuracy of the stencil, we adjust the ghost point of the stencil such that it correctly approximates the boundary condition. More precisely, in Figure 3.3b, u_{bL} can be approximated by the linear interpolation

$$u_{bL} = \frac{\gamma_L}{h_{i-1}}u_{i-1} + \frac{h_{i-1}-\gamma_L}{h_{i-1}}u_i. \tag{3.4}$$

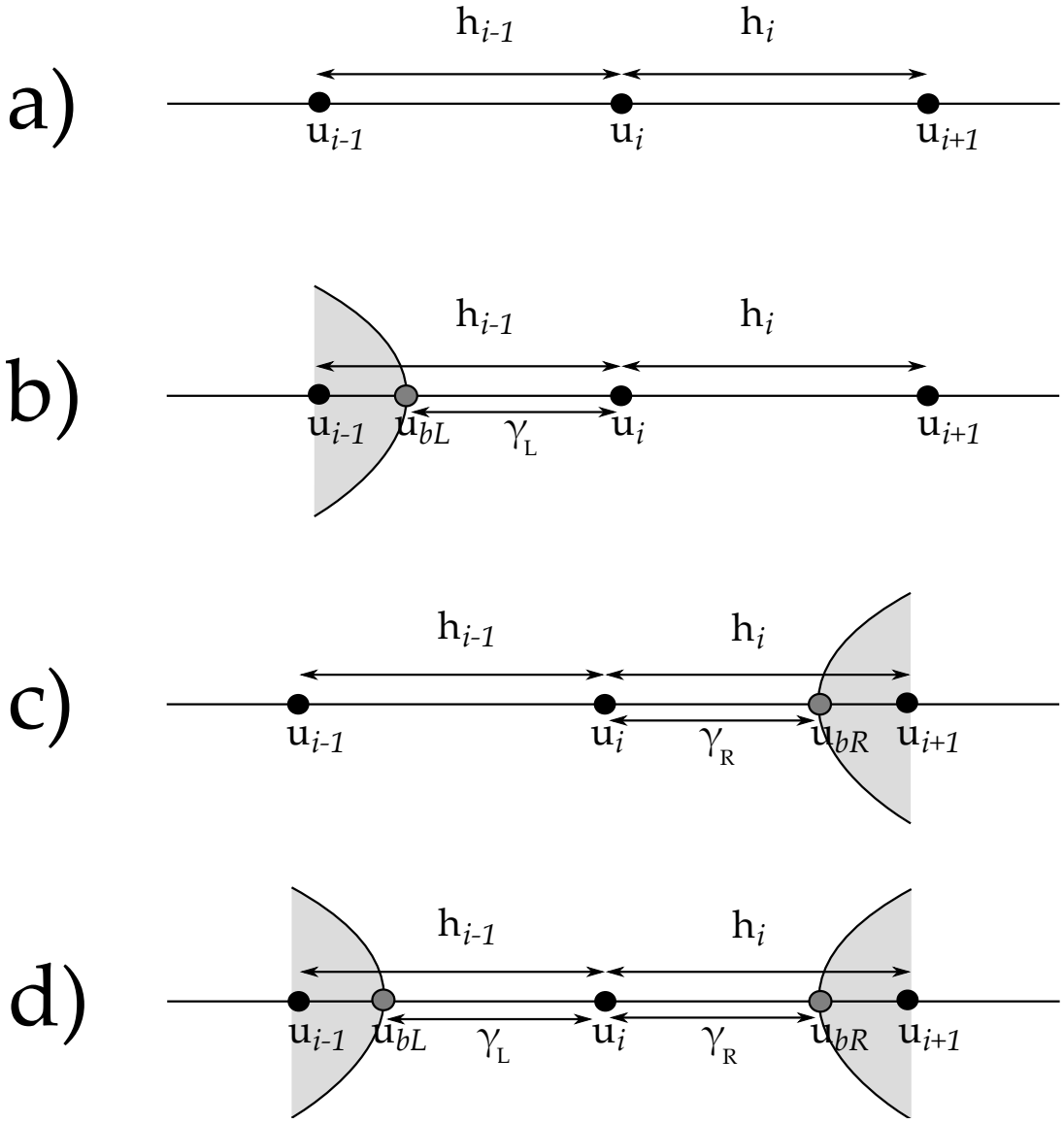

Figure 3.3: Boundary intersections of a three-point finite difference stencil. The shaded zones are solid and the white zones are fluid.

Substituting u_{i-1} from (3.4) into (3.2) and (3.3) gives the boundary fitted stencils

$$\left(\frac{\partial u}{\partial x}\right)_i \approx \frac{1}{h_{i-1}+h_i}\left[\frac{-h_i}{\gamma_L}u_{bL} + \left(\frac{h_i}{\gamma_L} - \frac{h_{i-1}}{h_i}\right)u_i + \frac{h_{i-1}}{h_i}u_{i+1}\right], \tag{3.5}$$

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i \approx \frac{2}{h_{i-1}+h_i}\left[\frac{1}{\gamma_L}u_{bL} - \left(\frac{1}{\gamma_L} + \frac{1}{h_i}\right)u_i + \frac{1}{h_i}u_{i+1}\right]. \tag{3.6}$$

Similarly, in Figure 3.3c, u_{bR} can be approximated by the linear interpolation

$$u_{bR} = \frac{\gamma_R}{h_i}u_{i+1} + \frac{h_i - \gamma_R}{h_i}u_i, \tag{3.7}$$

and substituting u_{i+1} from (3.7) into (3.2) and (3.3) gives the boundary fitted stencils

$$\left(\frac{\partial u}{\partial x}\right)_i \approx \frac{1}{h_{i-1}+h_i}\left[\frac{-h_i}{h_{i-1}}u_{i-1} + \left(\frac{h_i}{h_{i-1}} - \frac{h_{i-1}}{\gamma_R}\right)u_i + \frac{h_{i-1}}{\gamma_R}u_{bR}\right], \tag{3.8}$$

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i \approx \frac{2}{h_{i-1}+h_i}\left[\frac{1}{h_{i-1}}u_{i-1} - \left(\frac{1}{h_{i-1}} + \frac{1}{\gamma_R}\right)u_i + \frac{1}{\gamma_R}u_{bR}\right]. \tag{3.9}$$

Finally, in Figure 3.3d, the boundary fitted first and second derivative stencils are

$$\left(\frac{\partial u}{\partial x}\right)_i \approx \frac{1}{h_{i-1}+h_i}\left[\frac{-h_i}{\gamma_L}u_{bL} + \left(\frac{h_i}{\gamma_L} - \frac{h_{i-1}}{\gamma_R}\right)u_i + \frac{h_{i-1}}{\gamma_R}u_{bR}\right], \tag{3.10}$$

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i \approx \frac{2}{h_{i-1}+h_i}\left[\frac{1}{\gamma_L}u_{bL} - \left(\frac{1}{\gamma_L} + \frac{1}{\gamma_R}\right)u_i + \frac{1}{\gamma_R}u_{bR}\right]. \tag{3.11}$$

Since the interpolations (3.4) and (3.7) are spatially second-order accurate and (3.2) and (3.3) are spatially second-order accurate stencils, the above boundary fitted stencils are also spatially second-order accurate. To avoid division by zero in practice, we consider the point x_i to be inside the solid if γ_L or γ_R is smaller than $h/10000$.

The discrete boundary fitted operators (3.6), (3.9), (3.11) are non-commutative in general, i.e., $\delta_{xx}\delta_{yy}u \neq \delta_{yy}\delta_{xx}u$, where δ_{xx} represents the discrete operator for $\frac{\partial^2}{\partial x^2}$. However, the negative of the discrete boundary fitted operators $(-\delta_{xx}u)$ are still positive definite (see Lemma 3.2 of [4]), which is important for stability. There are also other ways to construct boundary fitted stencils (see Section 2.3 of [4]); however, the above method achieves the best accuracy and stability of all the stencils we tested.

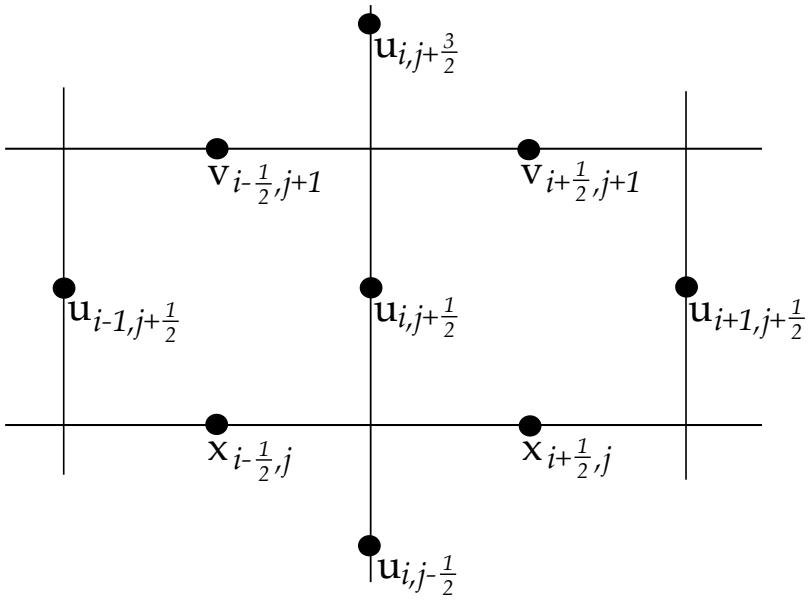

Figure 3.4: Layout of velocity grid nodes centered around $u_{i,j+\frac{1}{2}}$.

3.2 Advection Operator $\mathbf{u} \cdot \nabla \mathbf{u}$

The advection operator in Cartesian coordinates is defined as

$$\begin{aligned} \mathbf{u} \cdot \nabla \mathbf{u} &= \big((u,v,w) \cdot (\partial_x, \partial_y, \partial_z) \big) (u,v,w) \\ &= (\ uu_x + vu_y + wu_z, \quad uv_x + vv_y + wv_z, \quad uw_x + vw_y + ww_z\ ) . \end{aligned} \tag{3.12}$$

By examining Figure 3.4, we can determine a second-order finite difference stencil for this operator. The term uu_x can be approximated by

$$(uu_x)_{i,j+\frac{1}{2}} \approx u_{i,j+\frac{1}{2}} \left(\frac{\partial u}{\partial x} \right)_{i,j+\frac{1}{2}}, \tag{3.13}$$

where $\left(\frac{\partial u}{\partial x}\right)_{i,j+\frac{1}{2}}$ is given by (3.2). The term vu_y can be approximated similarly by

$$(vu_y)_{i,j+\frac{1}{2}} \approx (v_{\text{interpolated}}) \left(\frac{\partial u}{\partial y} \right)_{i,j+\frac{1}{2}}, \tag{3.14}$$

but in this case v is not available at the point $x_i, y_{j+\frac{1}{2}}$ so we need to interpolate the surrounding points $v_{i-\frac{1}{2},j}$, $v_{i-\frac{1}{2},j+1}$, $v_{i+\frac{1}{2},j}$, $v_{i+\frac{1}{2},j+1}$ in 2D in order to get $v_{\text{interpolated}}$ at the point $x_i, y_{j+\frac{1}{2}}$. In 3D, the interpolation involves eight points instead of four. The other advection terms are computed similarly.

All numerical results (see Section 5.3) indicate that second-order accuracy is achieved even without boundary fitting the advection operator. It is not entirely clear why the ∂_{xx} operators in Section 3.1 require boundary fitting but the advection operator does not, however, we can speculate why this is the case. One reason could be that the advection operator is only a first derivative, so it requires less regularity of the field to which it is applied. Therefore, a continuous extension of the velocity field beyond the domain boundary (which is the case for all problems we are interested in) may be sufficient regularity to control the error associated with using the discrete advection operator without boundary fitting. Also, the advection operator acts along characteristics of the flow (in the upstream direction) and the characteristics do not intersect the boundary due to the no-flux solid boundary condition. Another reason could be that very close to the boundary, the ∂_{xx} operators (which represent viscous effects) become completely dominant and therefore the error in the advection term becomes insignificant. It is possible that there exists problems for which the advection operator must be fitted to the domain boundary, but we have not encountered such problems. We do not perform boundary fitting of the advection operator in this thesis, however, it could be done by using the boundary fitted first derivative approximations in Section 3.1 together with a boundary fitted interpolation as in Section 6.3.

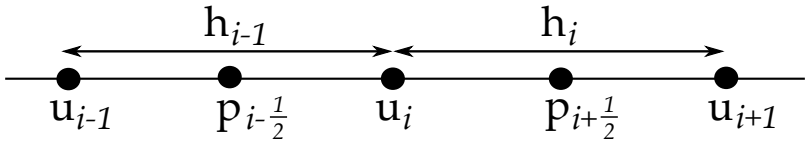

Figure 3.5: A grid line through the u and p points of the MAC grid.

3.3 Pressure Gradient ∇p

For the MAC grid, we are interested in computing $\nabla p = \left(\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial z} \right)$ at points of the grid where the velocity is stored. In particular, p_x is computed at the u grid points, p_y at v grid points, and p_z at w grid points. By examining Figure 3.5, we can determine a stencil representing the first derivative of the pressure p evaluated at the u_i velocity point,

$$\left(\frac{\partial p}{\partial x} \right)_i \approx \frac{p_{i+\frac{1}{2}} - p_{i-\frac{1}{2}}}{\frac{1}{2}\left(h_{i-1} + h_i\right)}. \tag{3.15}$$

If the MAC grid is uniform then $h_{i-1} = h_i$, and the difference is second-order accurate. The other components of the pressure gradient are computed similarly.

3.3.1 Explicit Pressure Extrapolation

For complex-shaped fluid domains Ω_f embedded in a larger box domain Ω, there are two problems with the pressure at the boundary $\partial\Omega_f$. First, consider the case of a time-dependent complex-shaped Ω_f, for example, a particulate flow where the moving solid particles are not part of Ω_f. Since the pressure is not generally meaningful outside Ω_f, meaningless pressure values at some grid points outside Ω_f may become part of Ω_f on the next time step when Ω_f changes shape. Second, even if Ω_f does not change in time, there are velocity grid points in Ω_f that are close enough to $\partial\Omega_f$ that the discrete pressure gradient (3.15) will reference a pressure point which is outside Ω_f. Therefore, we must make sure that the pressure at grid points just outside Ω_f has meaningful values. A simple solution is to extrapolate the pressure from inside Ω_f to a few grid points outside Ω_f. Using a constant extrapolation is roughly equivalent to a homogeneous Neumann boundary condition $\frac{\partial p}{\partial \mathbf{n}} = \nabla p \cdot \mathbf{n}|_{\partial\Omega_f} = 0$ for the pressure. Using a linear extrapolation is roughly equivalent to a "free" boundary condition $\frac{\partial^2 p}{\partial \mathbf{n}^2} = 0$ for the pressure.

For simplicity, we use a constant extrapolation, which can be implemented in the

following way for MAC grids. Call a MAC cell a "solid cell" if its pressure point is outside Ω_f and a "fluid cell" if its pressure point is inside Ω_f. Figure 3.7 on page 22 shows a disc in 2D with surrounding cells labeled as either solid or fluid using this definition. If a solid cell has at least one fluid cell neighbour, we average the pressure in all surrounding fluid cells and prescribe this average pressure value at the given solid cell. This extends the fluid pressure into the first layer of solid cells outside Ω_f. A similar procedure is repeated once more to extrapolate one layer deeper into the solid, and this seems to be sufficient. Since the above constant extrapolation is $O(h)$, it could be better to use a linear extrapolation (which would be $O(h^2)$), however, we did not investigate this. Figure 9.5 on page 137 shows this explicit pressure extrapolation procedure in use when solving for the flow behind a 2D disc.

3.4 Divergence Operator $\nabla \cdot \mathbf{u}$

The divergence operator is defined as $\nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}$. One advantage of the MAC grid is that the divergence evaluated at pressure points can be computed using a second-order accurate stencil even when the grid is non-uniform. By inspection of Figure 3.6a, we can determine the following "standard" finite difference stencil for the divergence in 2D,

$$(\nabla \cdot \mathbf{u})_{i+\frac{1}{2},j+\frac{1}{2}} \approx \frac{u_{i+1,j+\frac{1}{2}} - u_{i,j+\frac{1}{2}}}{x_{i+1} - x_i} + \frac{v_{i+\frac{1}{2},j+1} - v_{i+\frac{1}{2},j}}{y_{j+1} - y_j}. \tag{3.16}$$

The divergence operator (3.16) is $O(h^2)$ accurate in the maximum norm for any domain that aligns with the MAC cell boundaries. However, for complex-shaped domains that do not align with cell boundaries, (3.16) reduces to $O(1)$ in cells which are intersected by the fluid domain boundary $\partial\Omega_f$. This is because the velocity field is only guaranteed to be continuous across $\partial\Omega_f$. In this case, numerical results (Section 5.2.1) indicate that the divergence must be boundary-fitted in order to maintain second-order spatial accuracy of the velocity field. Because the pressure gradient and the divergence are closely related (see the beginning of Chapter 4), it has proven difficult to obtain a boundary-fitted discrete divergence operator that works well in all situations. Therefore, we consider in this section two different options, each with their own advantages and disadvantages.

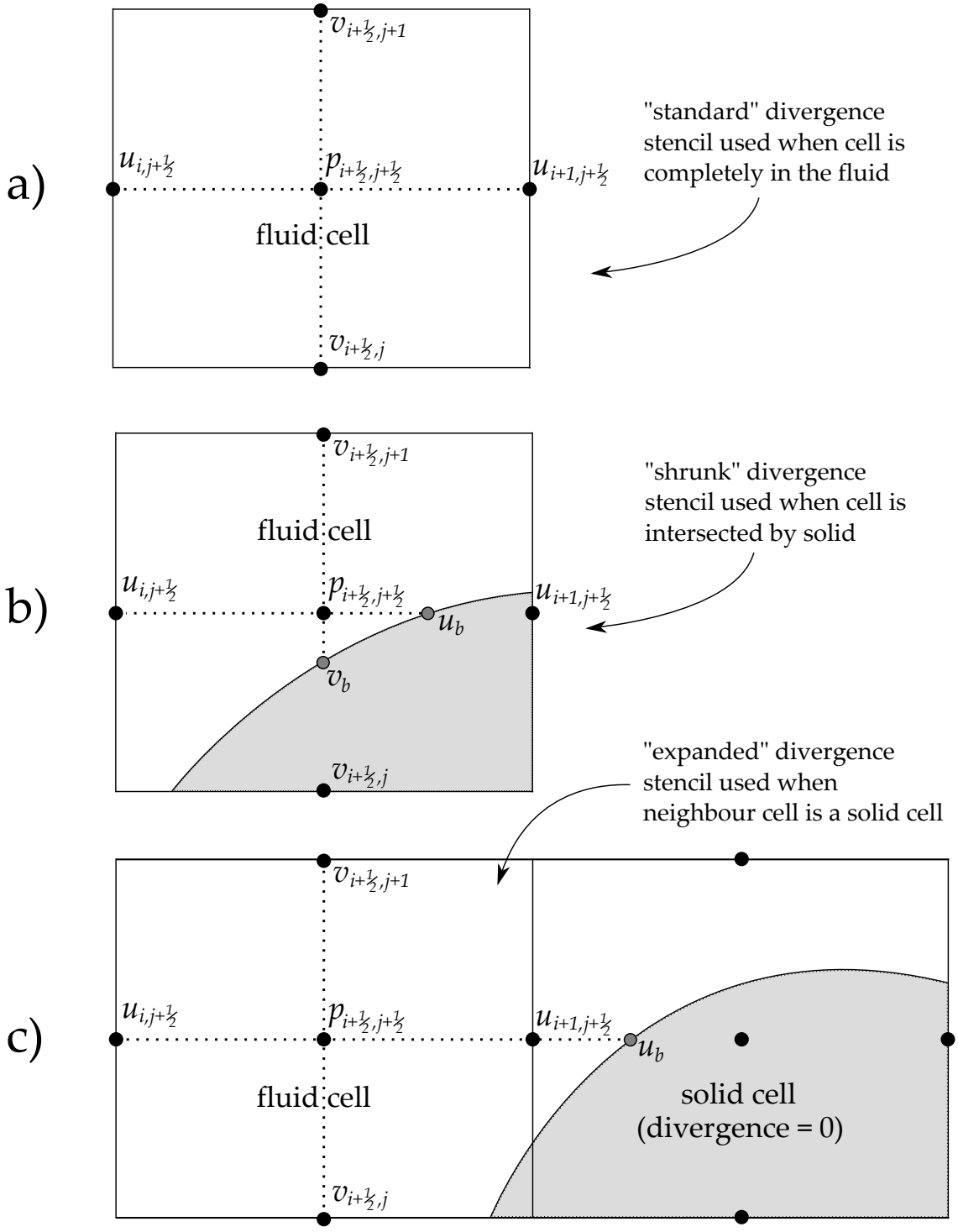

Figure 3.6: Different situations for computing the divergence with finite differences.

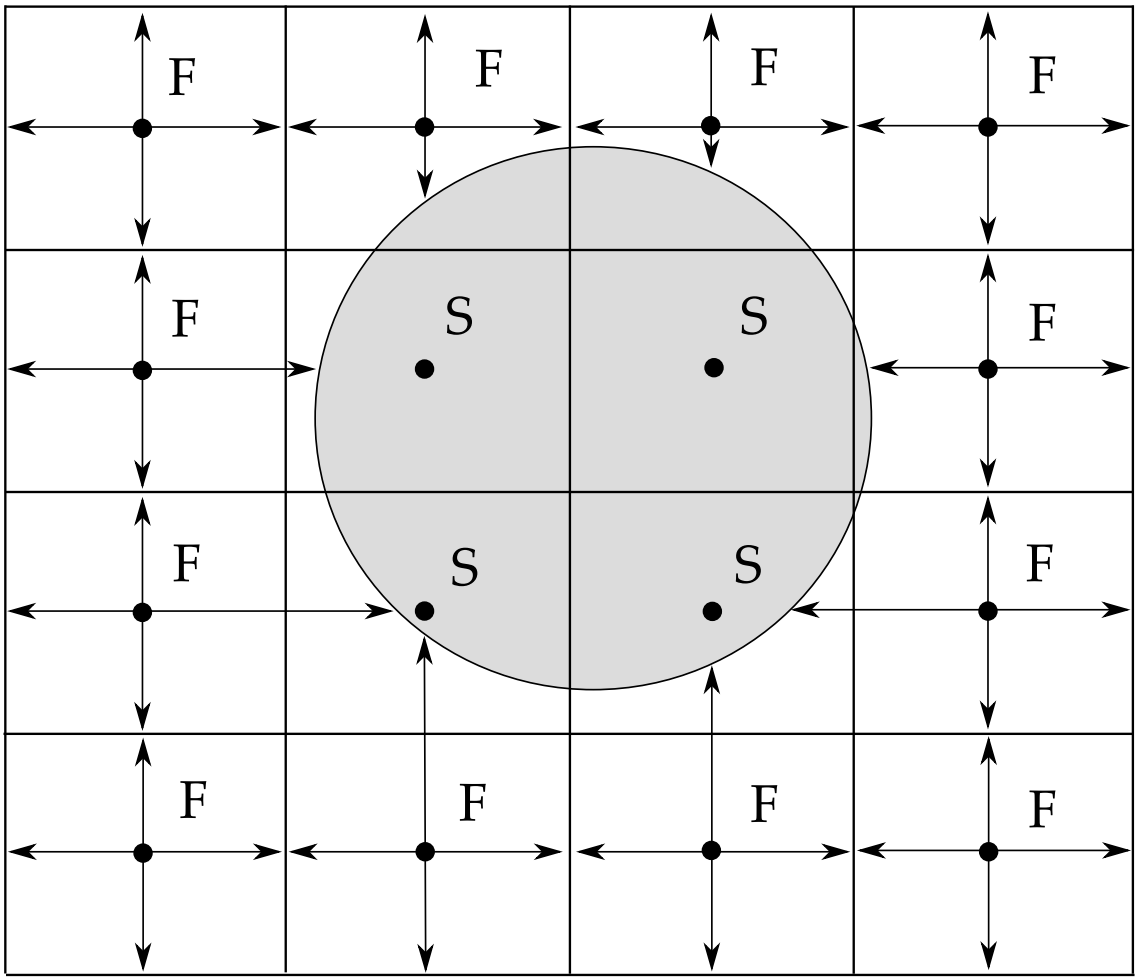

Figure 3.7: A 2D sample grid around a disc showing the divergence stencils in each MAC cell. Each box is a MAC cell, and the dots are the pressure points at the centroids of each cell. Each cell is labeled as F for "fluid cell" or S for "solid cell" based on whether the pressure point in the middle of the cell is inside the fluid or the solid, respectively.

3.4.1 Boundary Fitted Finite Difference Approximation

In this section, we consider fitting the standard divergence stencil (3.16) to the domain Ω_f in the spirit of finite differences. We again use the notation of "fluid cell" and "solid cell" as in Section 3.3.1. First, the divergence is set to zero in all "solid cells" because the rigid body velocity field inside a solid object has zero divergence. Second, in each "fluid cell" that does not share a face with any "solid cell", the standard difference (3.16) is used. In all remaining "fluid cells" that share one or more faces with a "solid cell", the divergence stencil is either shrunk or extended to fit the boundary in the direction orthogonal to these faces. Figure 3.7 shows how the divergence stencils would be fitted to a circular disc in 2D, and examples of shrunk and expanded stencils are illustrated in Figure 3.6b and Figure 3.6c respectively. The resulting approximation using the shrunk stencil in Figure 3.6b is given by

$$(\nabla \cdot \mathbf{u})_{i+\frac{1}{2},j+\frac{1}{2}} \approx \left(\frac{u_b - u_{i,j+\frac{1}{2}}}{x_b - x_i} + \frac{v_{i+\frac{1}{2},j+1} - v_b}{y_{j+1} - y_b} \right) \beta, \tag{3.17}$$

and the resulting approximation using the extended stencil in Figure 3.6c is given by

$$(\nabla \cdot \mathbf{u})_{i+\frac{1}{2},j+\frac{1}{2}} \approx \left(\frac{u_b - u_{i,j+\frac{1}{2}}}{x_b - x_i} + \frac{v_{i+\frac{1}{2},j+1} - v_{i+\frac{1}{2},j}}{y_{j+1} - y_j} \right) \beta, \tag{3.18}$$

where u_b and v_b are the known values of the velocity on the solid surface where the grid line intersects it, x_b and y_b are the coordinates of the points corresponding to u_b and v_b, and $\beta \leq 1$ is a scaling factor required for stability. The need for such a scaling can be understood in two ways. First, it makes sense that the divergence in the MAC cell should be scaled by the fluid volume fraction of the cell. In particular, this scaling would exist if we computed the divergence by subdividing into many smaller cells, since many of the smaller cells would be completely inside the solid and would have zero divergence. Second, since a solid obeys rigid motion (A.4), it is always true for the example in Figure 3.6b that $u_b = u_{i+1,j+\frac{1}{2}}$ and $v_b = v_{i+\frac{1}{2},j}$ (similarly for w in 3D). If we now consider a square MAC cell where where both x_b and y_b are equidistant from $x_{i+\frac{1}{2},j+\frac{1}{2}}$, then (3.17) can be written as

$$(\nabla \cdot \mathbf{u})_{i+\frac{1}{2},j+\frac{1}{2}} \approx \left(\frac{u_{i+1,j+\frac{1}{2}} - u_{i,j+\frac{1}{2}}}{h} + \frac{v_{i+\frac{1}{2},j+1} - v_{i+\frac{1}{2},j}}{h} \right) \frac{h}{d} \beta, \tag{3.19}$$

where $h = x_{i+1} - x_i = y_{j+1} - y_j$ and $d = x_b - x_i = y_{j+1} - y_b$. Equation (3.19) is exactly the standard divergence (3.16) multiplied by $\frac{h}{d}\beta$. Since we know that scaling the standard divergence by a value > 1 does not typically produce a stable scheme, this suggests the need for $\beta \leq \frac{d}{h}$, at least in this case. Inspired by this example, we choose the cell-dependent scaling factors

$$\beta = \min\left\{1, \frac{1}{2}\left(\frac{\Delta x}{h_x} + \frac{\Delta y}{h_y}\right)\right\}, \qquad \beta = \min\left\{1, \frac{1}{3}\left(\frac{\Delta x}{h_x} + \frac{\Delta y}{h_y} + \frac{\Delta z}{h_z}\right)\right\}, \quad (3.20)$$

in 2D and 3D respectively, where Δx, Δy, Δz are the divergence stencil widths used in that cell, and h_x, h_y, h_z are the MAC cell dimensions. For example, both stencils (3.17) and (3.18) have $\Delta x = x_b - x_i$ and $h_x = x_{i+1} - x_i$. Numerical evidence suggests that this choice of β is always stable. While scaling the divergence by a somewhat arbitrary factor β appears questionable, it is valid to scale the divergence by any constant ≤ 1 because we will show later, for example equation (4.49), that the divergence is already scaled by a penalty parameter and thus β simply changes this penalty parameter.

The divergence as computed above sometimes "skips over" a velocity node as in Figure 3.6c where $u_{i+1,j+\frac{1}{2}}$ is skipped when computing $u_b - u_{i,j+\frac{1}{2}}$. In some rare cases this can destabilize the scheme, so the skipped velocity node $u_{i+1,j+\frac{1}{2}}$ needs to be set to the linear interpolation between u_b and $u_{i,j+\frac{1}{2}}$. Alternatively, a variant of the divergence operator can be used that does not skip over nodes, and in this case $u_b - u_{i,j+\frac{1}{2}}$ is replaced by $u_{i+1,j+\frac{1}{2}} - u_{i,j+\frac{1}{2}}$.

The discrete divergence operator in this section does not satisfy the discrete integration by parts formula (i.e, summation by parts), so summing the divergence over all cells in the domain does not in general give zero. Therefore, the pressure extrapolation in Section 3.3.1 should be used in order to remove any artificial pressure gradient across the domain boundary caused by a drift of the average pressure in the fluid domain.

The divergence operator in this section was tested numerically to be $O(h^2)$ accurate in the maximum norm for any domain that aligns with the MAC cell boundaries, and otherwise it reduces to $O(h)$ in the maximum norm and $O(h^{\frac{3}{2}})$ in the L^2 norm.

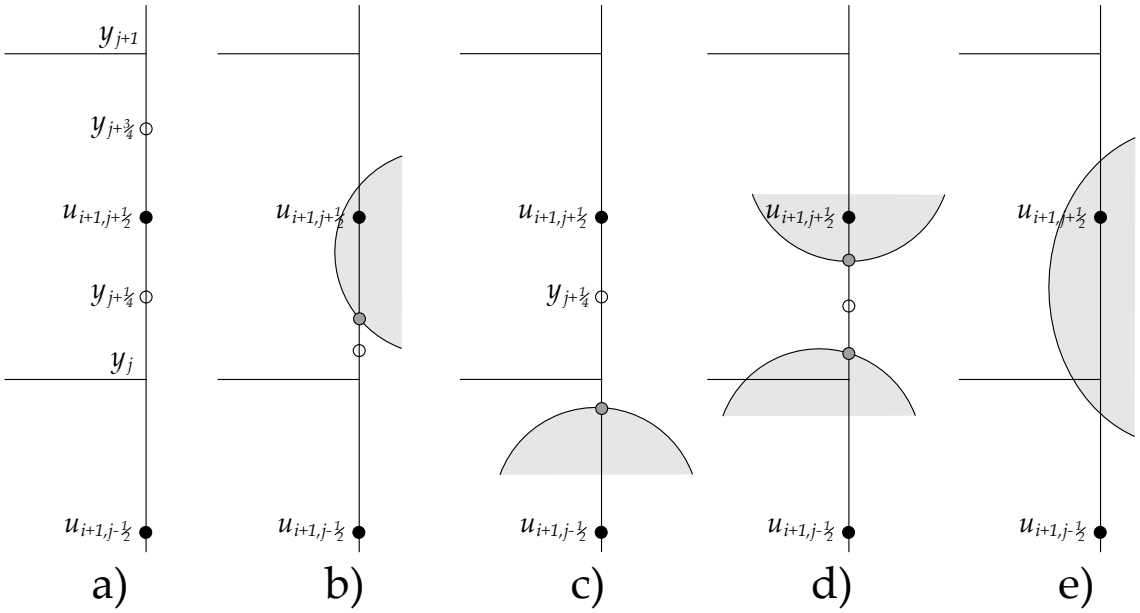

Figure 3.8: Different types of intersections that can occur when computing the flux through a cell face. The velocity is always interpolated at the midpoints (white dots).

3.4.2 Boundary Fitted Finite Volume Approximation

In this section, we consider fitting the standard divergence stencil (3.16) to the domain Ω_f by using a finite volume formulation. We will only consider the 2D operator here, but the 3D operator can be constructed similarly. Consider the 2D MAC cell in Figure 3.6a, and assume that the entire cell is in Ω_f. The finite volume formulation of an operator is obtained by integrating the operator over the cell and then dividing by the cell volume,

$$\nabla \cdot \mathbf{u} \approx \frac{1}{h_x h_y} \int \nabla \cdot \mathbf{u} \, dV = \frac{1}{h_x h_y} \int \mathbf{u} \cdot \mathbf{n} \, dA, \tag{3.21}$$

after using Gauss's divergence theorem. Thus, we need to compute fluxes $\mathbf{u}\cdot\mathbf{n}$ through all four faces of the cell. Since the flux through one face can be approximated by the velocity at the midpoint on that face multiplied by the face area, we have

$$\frac{1}{h_x h_y} \int \mathbf{u} \cdot \mathbf{n} \, dA \approx \frac{1}{h_x h_y} \left(u_{i+1,j+\frac{1}{2}} h_y - u_{i,j+\frac{1}{2}} h_y + v_{i+\frac{1}{2},j+1} h_x - v_{i+\frac{1}{2},j} h_x \right), \tag{3.22}$$

which simplifies to the standard finite difference stencil (3.16). We now consider a similar approximation of the flux through one cell face which works more naturally when the cell face is intersected by $\partial\Omega_f$. Consider Figure 3.8a which shows one MAC cell face. Previously, we computed the flux through this face as $u_{i+1,j+\frac{1}{2}}h_y$, but this time we will compute the flux through the face in two parts — the lower half of the face from y_j to $y_{j+\frac{1}{2}}$, and the upper half of the face from $y_{j+\frac{1}{2}}$ to y_{j+1}. This is done to ensure that the face-center velocity $u_{i+1,j+\frac{1}{2}}$ maintains a dominant role in the flux, which is required for stability. The flux through both lower and upper halves are computed similarly, so we will consider only the lower half. If there is no boundary intersection of the lower half face as in Figure 3.8a, then the flux through the lower half is computed as $\frac{1}{2}u_{i+1,j+\frac{1}{4}}h_y$, where $\frac{1}{2}h_y$ is the length of the half face and $u_{i+1,j+\frac{1}{4}}$ is the velocity at the midpoint $y_{j+\frac{1}{4}}$ of the half face. This midpoint velocity is obtained as the linear interpolation $u_{i+1,j+\frac{1}{4}} = \frac{1}{4}u_{i+1,j-\frac{1}{2}} + \frac{3}{4}u_{i+1,j+\frac{1}{2}}$. Therefore, the flux through the entire cell face (both halves) is $\left(\frac{1}{8}u_{i+1,j-\frac{1}{2}} + \frac{6}{8}u_{i+1,j+\frac{1}{2}} + \frac{1}{8}u_{i+1,j+\frac{3}{2}}\right)h_y$ instead of $u_{i+1,j+\frac{1}{2}}h_y$. If one or more solid objects intersects the lower half face as in Figure 3.8 b,c,d,e, then several fluxes may need to be computed, but in every case the velocity at the midpoint of the line segments are used, and these velocities are obtained by interpolating between either the known velocity on the object boundaries (gray dots) or the velocities of the background grid (black dots), whichever are closer. Also, if the solid objects are moving then their velocity (but not their angular velocity) contributes to the flux.

The divergence operator in this section was tested numerically to be $O(h^2)$ accurate in the maximum norm for box domains using uniform grids, and for complex-shaped domains it reduces to $O(h)$ in the maximum norm and $O(h^{\frac{3}{2}})$ in the L^2 norm. This divergence operator is quite a bit more complicated (and costs more CPU time) than the finite difference divergence in Section 3.4.1, however it has some advantages. First, by construction it always satisfies the summation by parts formula, and this may enhance stability and other energy-related properties. Second, it is a "smooth" operator, meaning that when Ω_f changes shape by a small amount, the operator only changes a small amount. This also enhances stability and tends to improve accuracy in practice. However, one disadvantage is that the discrete divergence will in general be nonzero inside MAC cells that have all of their velocity nodes inside a solid, and this can create some difficulty with the pressure. In order to avoid these difficulties, the pressure extrapolation in Section 3.3.1 should be used. Alternatively, the pres-

sure gradient operator could be modified in other ways, but we do not explore other ways in this thesis.

3.5 Spatial Operators Using Ghost Cells

Similar to how the one-dimensional diffusion operators were fitted to the boundary in Section 3.1 by choosing an appropriate value for the "ghost point", it may also be possible to fit multi-dimensional operators (like the divergence) to the boundary in a similar way. However, when considering the divergence operator, computing first derivatives by using a boundary-condition-satisfying ghost point in one dimension as in Section 3.1 produces exactly the divergence stencil of Section 3.4.1 but without the β factor. The "ghost cell" method used in [51] appears to be stable and second-order accurate, however, the method cannot directly be applied in our case because we perform direction splitting of the pressure equation (discussed later in Section 4.1.6), and this does not easily allow Neumann boundary conditions to be imposed on surfaces that are not parallel to a coordinate plane. Nonetheless, extending the velocity field can be shown to work on the Taylor–Couette flow in Appendix A by extending the analytical solution in Ω_f to all points outside Ω_f. In this case, all of the non-boundary-fitted operators are second-order accurate everywhere and the resulting solution is also second-order accurate. However, when the analytical solution is unknown, it is difficult to obtain a smooth extension of the velocity field that produces a stable and consistent numerical scheme overall.

Chapter 4

Discretizations For Incompressible Navier–Stokes Equations

In this chapter, we consider solving the incompressible Navier–Stokes equations (2.22). There are three terms of interest in the Navier–Stokes equations: the advection term $\mathbf{u} \cdot \nabla \mathbf{u}$, the pressure gradient term ∇p, and the diffusion term $\frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u}$. The advection and diffusion terms may either be treated explicitly (at time level n) or implicitly (at time level $n+1$). Explicit terms are simple and efficient to compute but they impose time step restrictions to maintain numerical stability. In contrast, implicit terms are more difficult and costly to compute but they allow larger time steps.

Treating the advection terms implicitly is very difficult because one must solve for $\mathbf{u}^{n+1}$ in an equation of the form

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} + \mathbf{u}^{n+1} \cdot \nabla \mathbf{u}^{n+1} = \mathbf{f}, \tag{4.1}$$

which is nonlinear in $\mathbf{u}^{n+1}$. Therefore, one must resort to root finding methods such as Newton-Raphson, which is difficult for multi-dimensional problems. Treating advection terms explicitly imposes the well known Courant-Friedrichs-Lewy (CFL) condition which restricts the time step $\Delta t \leq Ch/|\mathbf{u}|$, where h is the MAC cell width and C is a constant usually on the order of 1. Note, however, that (4.1) is a transport equation and therefore it doesn't make a lot of sense to use time steps so large that fluid skips over multiple grid cells in a single time step. Thus, advection terms are

usually treated explicitly when solving the Navier–Stokes equations.

If the Reynolds number is very large, then the flow has structure at extremely small spatial scales that cannot be resolved, and in this case turbulence modeling must be used. However, in this thesis we are interested in flows that can be fully resolved by direct numerical simulation (DNS). Therefore, the diffusion terms must be treated implicitly in order to avoid the potentially severe $\Delta t \sim \mathrm{Re}\, h^2$ time step restriction.

The pressure gradient term is somewhat different than the other terms. In particular, for the incompressible Navier–Stokes equations, the pressure is just a Lagrange multiplier that enforces the side constraint $\nabla \cdot \mathbf{u} = 0$ on the momentum equation. The pressure still has physical meaning as a dynamic pressure, but since there is no time-derivative of the pressure in any of the equations, the pressure cannot simply be advanced in time like the velocity. One can always introduce a fictitious time derivative of the pressure, as in the "artificial compressibility" approximation

$$\nabla \cdot \mathbf{u} = -\delta \frac{\partial p}{\partial t}, \tag{4.2}$$

where the artificial compressibility parameter δ should be small to correctly approximate unsteady incompressible flows. For stability, the time step must satisfy $\Delta t \leq C\delta^{\frac{1}{2}} h$ due to high frequency sound waves created by the scheme (see [15]). The stability restriction with $\delta \ll 1$ is more severe than the CFL condition. In particular, if one chooses $\delta = \Delta t$ then $\Delta t \leq C\Delta t^{\frac{1}{2}} h \Rightarrow \Delta t \leq C^2 h^2$. Thus, we will not use explicit methods for computing the pressure in incompressible flows.

With explicit advection, implicit diffusion, and implicit pressure, we are thus interested in solving the incompressible Navier–Stokes equations in a form such as

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} + \mathbf{u}^n \cdot \nabla \mathbf{u}^n = -\nabla p^{n+1} + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u}^{n+1}, \qquad \nabla \cdot \mathbf{u}^{n+1} = 0. \tag{4.3}$$

To solve (4.3) directly, one can write it in matrix form,

$$\left[\begin{array}{c|c} I - \frac{1}{\mathrm{Re}}\nabla^2 & \nabla \\ \hline \nabla\cdot & 0 \end{array}\right] \left[\begin{array}{c} \mathbf{u}^{n+1} \\ \hline p^{n+1} \end{array}\right] = \left[\begin{array}{c} -\mathbf{u}^n \cdot \nabla \mathbf{u}^n \\ \hline 0 \end{array}\right], \tag{4.4}$$

where $\mathbf{u}$ and p are huge vectors containing all of the spatially discrete unknowns, I is the identity matrix, ∇^2 is the matrix form of the Laplacian operator, ∇ is the matrix

form of the gradient operator, $\nabla\cdot$ is the matrix form of the divergence operator, and 0 is the zero matrix (or the zero vector). Since $\nabla\cdot$ is the transpose of the ∇ operator, the above matrix equation can be written as

$$\left[\begin{array}{c|c} A & B^T \\ \hline B & 0 \end{array}\right] \left[\begin{array}{c} \mathbf{u} \\ \hline p \end{array}\right] = \left[\begin{array}{c} f \\ \hline g \end{array}\right], \tag{4.5}$$

where $A = I - \frac{1}{Re}\nabla^2$ is symmetric and positive definite. Matrix equations of this form are called saddle point problems because the solution is always a saddle point of the Lagrangian

$$\mathcal{L}(u,p) = \frac{1}{2}u^T A u - f^T u + (Bu - g)^T p. \tag{4.6}$$

An overview of saddle point problems can be found in [9]. Saddle-point problems are difficult to solve efficiently because a saddle point is intuitively more difficult to find than a minimum. More concretely, the zero in the lower right corner of the matrix prevents direct methods (Gaussian Elimination) due to pivoting and memory requirements, and the matrix typically has poor spectral properties which makes it difficult for iterative solvers. Therefore, we need a different approach to solve equation (4.3).

4.1 Pressure-Correction Projection Methods

Pressure-correction projection methods aim to approximate the original incompressible Navier–Stokes equations (2.22) by using operator splitting to decouple the incompressibility constraint from the momentum equation. The momentum equation is solved first without the incompressibility constraint (or with an explicitly predicted pressure), and then the resulting velocity field is "projected" onto a function space which satisfies the incompressibility condition. Projection schemes are computationally cheaper than solving the full saddle-point problem (4.4). An excellent overview of projection methods can be found in [31]. For the discussion of pressure projection methods in this section, we include the advection terms and the time level at which they should be discretized, but we will not actually discretize the advection terms since this will be discussed in Section 4.3. We begin by discussing the original Chorin–Temam projection scheme, which is a first-order accurate scheme.

We then discuss second-order projection schemes called "incremental" schemes. We also include discussion of a "rotational" formulation which improves accuracy of all projection schemes. We then discuss a more efficient "computational" formulation of the projection schemes, and we perform an analysis of the accuracy in this context. Finally, we discuss a recently introduced direction-factorized scheme, which is what we will ultimately use in this thesis.

4.1.1 Chorin–Temam Scheme

The simplest projection method was originally proposed by Chorin [16] and also independently by Temam [65]:

$$\frac{\tilde{\mathbf{u}}^{n+1} - \mathbf{u}^n}{\Delta t} + \tilde{\mathbf{u}}^n \cdot \nabla \tilde{\mathbf{u}}^n = \frac{1}{\mathrm{Re}} \nabla^2 \tilde{\mathbf{u}}^{n+1}, \qquad \tilde{\mathbf{u}}^{n+1}\big|_{\partial\Omega} = 0, \tag{4.7}$$

$$\frac{\mathbf{u}^{n+1} - \tilde{\mathbf{u}}^{n+1}}{\Delta t} = -\nabla p^{n+1}, \tag{4.8}$$

$$\nabla \cdot \mathbf{u}^{n+1} = 0, \qquad \mathbf{u}^{n+1} \cdot \mathbf{n}\big|_{\partial\Omega} = 0. \tag{4.9}$$

The above scheme loosely resembles the original Navier–Stokes equations if one considers the sum of (4.7) and (4.8). The intermediate velocity $\tilde{\mathbf{u}}^{n+1}$ is the solution to the momentum equation (4.7) neglecting the pressure contribution, which means that $\tilde{\mathbf{u}}^{n+1}$ is not divergence free, but it has the correct "no flux, no slip" boundary condition. Equations (4.8) and (4.9) can be regarded as the projection step, which can be understood as decomposing $\tilde{\mathbf{u}}^{n+1} \in L_2$ into a divergence free part $\mathbf{u}^{n+1}$ and an irrotational part ∇p^{n+1} as in the Helmholtz decomposition theorem. The final velocity $\mathbf{u}^{n+1}$ obtained from (4.8) is divergence free, but it may have nonzero tangential component on $\partial\Omega$. The projection step can be reformulated by taking the divergence of (4.8) and using (4.9) to give

$$\frac{0 - \nabla \cdot \tilde{\mathbf{u}}^{n+1}}{\Delta t} = -\nabla \cdot \nabla p^{n+1}, \qquad \Rightarrow \qquad \frac{\nabla \cdot \tilde{\mathbf{u}}^{n+1}}{\Delta t} = \nabla^2 p^{n+1}. \tag{4.10}$$

To use the Chorin–Temam scheme in practice, one solves (4.7) implicitly to obtain $\tilde{\mathbf{u}}^{n+1}$, then one solves the Poisson problem (4.10) for p^{n+1} using boundary conditions $\nabla p^{n+1} \cdot \mathbf{n}\big|_{\partial\Omega} = 0$, and finally one uses (4.8) to obtain $\mathbf{u}^{n+1}$. The artificially imposed boundary condition $\nabla p^{n+1} \cdot \mathbf{n}\big|_{\partial\Omega} = 0$ creates increased error in an exponen-

tially decaying boundary layer around $\partial\Omega$ (see [58]). The Chorin–Temam scheme as presented above has the following time step error estimates according to [31]:

$$\begin{aligned} \|\tilde{\mathbf{u}} - \mathbf{u}_{\text{exact}}\|_{L^\infty(0,T;L^2(\Omega))} + \|\mathbf{u} - \mathbf{u}_{\text{exact}}\|_{L^\infty(0,T;L^2(\Omega))} &\leq C\Delta t, \\ \|\tilde{\mathbf{u}} - \mathbf{u}_{\text{exact}}\|_{L^\infty(0,T;H^1(\Omega))} + \|p - p_{\text{exact}}\|_{L^\infty(0,T;L^2(\Omega))} &\leq C\Delta t^{\frac{1}{2}}, \end{aligned} \tag{4.11}$$

where $\|\cdot\|_{L^\infty(0,T;L^2(\Omega))}$ means the maximum over all time steps of the L^2 spatial error, $\|\cdot\|_{L^\infty(0,T;H^1(\Omega))}$ means the maximum over all time steps of the H^1 spatial error, and $\|\cdot\|_{L^2(0,T;L^2(\Omega))}$ means the discrete L^2 norm over all time steps of the L^2 spatial error:

$$\|f\|_{L^2(0,T;L^2(\Omega))} = \sqrt{\Delta t \sum_{n=0}^{N} \|f^n\|_{L^2}^2}, \tag{4.12}$$

where f^n is a field at time step n.

4.1.2 Incremental Scheme

One can increase the time accuracy of the Chorin–Temam scheme in Section 4.1.1 by including an explicitly predicted pressure in the momentum equation to reduce the splitting error. The Crank–Nicolson incremental scheme is

$$\frac{\tilde{\mathbf{u}}^{n+1} - \mathbf{u}^n}{\Delta t} + \tilde{\mathbf{u}}^{n+\frac{1}{2}} \cdot \nabla \tilde{\mathbf{u}}^{n+\frac{1}{2}} = \frac{1}{\text{Re}} \nabla^2 \frac{\tilde{\mathbf{u}}^{n+1} + \tilde{\mathbf{u}}^n}{2} - \nabla p^{n-\frac{1}{2}}, \tag{4.13}$$

$$\frac{\mathbf{u}^{n+1} - \tilde{\mathbf{u}}^{n+1}}{\Delta t} = -\nabla \left(p^{n+\frac{1}{2}} - p^{n-\frac{1}{2}} \right), \tag{4.14}$$

$$\nabla \cdot \mathbf{u}^{n+1} = 0, \tag{4.15}$$

where if we take the sum of (4.13) and (4.14), the pressure predictor $p^{n-\frac{1}{2}}$ cancels out and all terms are approximated at the $n + \frac{1}{2}$ time level so that the time derivative centered around $n + \frac{1}{2}$ is formally second order accurate. The incremental scheme is named because the projection step (4.14) includes the time-increment of the pressure rather than the pressure itself, as in the Chorin–Temam scheme. A non-incremental scheme in Crank–Nicolson form can be obtained by setting $p^{n-\frac{1}{2}}$ to zero in (4.13), (4.14). The boundary conditions and solution procedure are identical for

the incremental scheme as for the Chorin–Temam scheme, except that the Poisson problem (similar to (4.10)) involves the pressure increment rather than the pressure. In particular, the incremental scheme artificially imposes the boundary condition $\nabla\left(p^{n+\frac{1}{2}}-p^{n-\frac{1}{2}}\right)\cdot\mathbf{n}\big|_{\partial\Omega}=0$, which implies $\nabla p^{n+\frac{1}{2}}\cdot\mathbf{n}\big|_{\partial\Omega}$ is constant in time. The incremental scheme as presented above has the following time step error estimates according to [31]:

$$\begin{aligned}\|\tilde{\mathbf{u}}-\mathbf{u}_{\text{exact}}\|_{L^2(0,T;L^2(\Omega))}+\|\mathbf{u}-\mathbf{u}_{\text{exact}}\|_{L^2(0,T;L^2(\Omega))}&\leq C\Delta t^2,\\ \|\tilde{\mathbf{u}}-\mathbf{u}_{\text{exact}}\|_{L^\infty(0,T;H^1(\Omega))}+\|p-p_{\text{exact}}\|_{L^\infty(0,T;L^2(\Omega))}&\leq C\Delta t,\end{aligned}\tag{4.16}$$

where the above norms are explained in Section 4.1.1.

4.1.3 Rotational Scheme

In order to reduce the error associated with the artificial pressure boundary condition $\nabla p\cdot\mathbf{n}\big|_{\partial\Omega}=0$ as explained in Section 4.1.1, one can reformulate any projection scheme into a so-called "rotational" form by including an additional term in the projection step as originally proposed in [66]. The rotational version of the incremental scheme of Section 4.1.2 is:

$$\frac{\tilde{\mathbf{u}}^{n+1}-\mathbf{u}^n}{\Delta t}+\tilde{\mathbf{u}}^{n+\frac{1}{2}}\cdot\nabla\tilde{\mathbf{u}}^{n+\frac{1}{2}}=\frac{1}{\mathrm{Re}}\nabla^2\frac{\tilde{\mathbf{u}}^{n+1}+\tilde{\mathbf{u}}^n}{2}-\nabla p^{n-\frac{1}{2}},\tag{4.17}$$

$$\frac{\mathbf{u}^{n+1}-\tilde{\mathbf{u}}^{n+1}}{\Delta t}=-\nabla\left(p^{n+\frac{1}{2}}-p^{n-\frac{1}{2}}+\frac{\chi}{\mathrm{Re}}\nabla\cdot\frac{\tilde{\mathbf{u}}^{n+1}+\tilde{\mathbf{u}}^n}{2}\right),\tag{4.18}$$

$$\nabla\cdot\mathbf{u}^{n+1}=0,\tag{4.19}$$

where (4.18) contains the additional term $\frac{\chi}{\mathrm{Re}}\nabla\left(\nabla\cdot\frac{\tilde{\mathbf{u}}^{n+1}+\tilde{\mathbf{u}}^n}{2}\right)$, and $\chi=1$. The rotational scheme is named as such because the vector identity

$$\nabla^2\mathbf{F}=\nabla(\nabla\cdot\mathbf{F})-\nabla\times(\nabla\times\mathbf{F})\tag{4.20}$$

is used to decompose the diffusion term, and the $\nabla(\nabla\cdot\mathbf{F})$ component is the additional term that appears in the projection step, where $\mathbf{F}=\frac{1}{2}\left(\tilde{\mathbf{u}}^{n+1}+\tilde{\mathbf{u}}^n\right)$. The rotational scheme can be intuitively understood as follows. First, take $\nabla\times(\nabla\times)$ of equation

(4.18) and make use of the vector identity $\nabla \times (\nabla f) = 0 \; \forall f$ to obtain

$$-\nabla \times \left(\nabla \times \tilde{\mathbf{u}}^{n+1}\right) = -\nabla \times \left(\nabla \times \mathbf{u}^{n+1}\right). \tag{4.21}$$

Then add $\nabla \left(\nabla \cdot \mathbf{u}^{n+1}\right)$ (which is zero by (4.19)) to the right-hand side of (4.21) and use (4.20) to get

$$-\nabla \times \left(\nabla \times \tilde{\mathbf{u}}^{n+1}\right) = \nabla^2 \mathbf{u}^{n+1}. \tag{4.22}$$

Next, taking the sum of (4.17) and (4.18), canceling out $\tilde{\mathbf{u}}^{n+1}$ and $p^{n-\frac{1}{2}}$, and omitting the advection terms to be concise, we get

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \frac{1}{\mathrm{Re}} \nabla^2 \frac{\tilde{\mathbf{u}}^{n+1} + \tilde{\mathbf{u}}^n}{2} - \nabla \left(p^{n+\frac{1}{2}} + \frac{1}{\mathrm{Re}} \nabla \cdot \frac{\tilde{\mathbf{u}}^{n+1} + \tilde{\mathbf{u}}^n}{2} \right). \tag{4.23}$$

Using (4.20) to decompose the Laplacian term in (4.23) yields

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = -\frac{1}{\mathrm{Re}} \nabla \times \left(\nabla \times \frac{\tilde{\mathbf{u}}^{n+1} + \tilde{\mathbf{u}}^n}{2} \right) - \nabla \left(p^{n+\frac{1}{2}} \right), \tag{4.24}$$

and then we can use (4.22) (which is true at time level n as well) to get

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \frac{1}{\mathrm{Re}} \nabla^2 \frac{\mathbf{u}^{n+1} + \mathbf{u}^n}{2} - \nabla p^{n+\frac{1}{2}}, \tag{4.25}$$

which, together (4.19), is the Crank–Nicolson discretization for the incompressible unsteady Stokes equations without operator splitting the pressure term. Thus, the rotational pressure projection scheme (4.17)–(4.19) implies the equivalent scheme without operator splitting for the pressure, but with slightly different boundary conditions. The advantage of the rotational scheme vs. the standard incremental scheme can be seen by taking $() \cdot \mathbf{n}\big|_{\partial\Omega}$ of (4.24) and applying the velocity boundary condition $\mathbf{u} \cdot \mathbf{n}\big|_{\partial\Omega} = 0$ to show that the artificial pressure boundary condition is

$$\nabla p^{n+\frac{1}{2}} \cdot \mathbf{n}\big|_{\partial\Omega} = -\frac{1}{\mathrm{Re}} \nabla \times \left(\nabla \times \frac{\tilde{\mathbf{u}}^{n+1} + \tilde{\mathbf{u}}^n}{2} \right) \cdot \mathbf{n}\big|_{\partial\Omega}. \tag{4.26}$$

This artificial boundary condition for the pressure contains the velocity as well, so it doesn't directly constrain the pressure unlike the artificial boundary condition $\nabla p^{n+\frac{1}{2}} \cdot \mathbf{n}\big|_{\partial\Omega} = 0$ imposed by the standard incremental scheme in Section 4.1.2.

The rotational incremental scheme (4.17)–(4.19) has the following time step error estimates according to [31]:

$$\begin{aligned}
&\|\tilde{\mathbf{u}} - \mathbf{u}_{\text{exact}}\|_{L^2(0,T;L^2(\Omega))} + \|\mathbf{u} - \mathbf{u}_{\text{exact}}\|_{L^2(0,T;L^2(\Omega))} \leq C\Delta t^2, \\
&\|\tilde{\mathbf{u}} - \mathbf{u}_{\text{exact}}\|_{L^2(0,T;H^1(\Omega))} + \|\mathbf{u} - \mathbf{u}_{\text{exact}}\|_{L^2(0,T;H^1(\Omega))} + \|p - p_{\text{exact}}\|_{L^2(0,T;L^2(\Omega))} \leq C\Delta t^{\frac{3}{2}},
\end{aligned} \tag{4.27}$$

where the above norms are explained in Section 4.1.1. Compared to the non-rotational schemes, the rotational scheme has a higher-order error estimate for the pressure. The rotational scheme error estimates in [31] are proven for a BDF2 formulation rather than a Crank–Nicolson formulation, but a similar estimate can most likely be proven for the Crank–Nicolson formulation.

Note: The parameter χ in (4.18) can be set to any real number between 0 and 1, where $\chi = 0$ gives the incremental scheme in standard form, $\chi = 1$ gives the the incremental scheme in rotational form, and $\chi \in (0, 1)$ gives a combination of the two.

4.1.4 Computational Formulation

As discussed in Section 4.1.1 and also in [58] and [31], the end-of-step velocity $\mathbf{u}^{n+1}$ given by (4.18) is divergence free, but in general it has a nonzero tangential component on the domain boundary, i.e., some error in the form of "slip" at the boundary. The intermediate-step velocity $\tilde{\mathbf{u}}^{n+1}$ given by (4.17) is not divergence free but it satisfies the correct no-slip boundary conditions. While there are some arguments for preferring the divergence free solution $\mathbf{u}$, this solution is only *discrete* divergence free, and the discrete divergence is usually only $O(h^2)$ at best. Finally, we can see from (4.27) that both $\mathbf{u}$ and $\tilde{\mathbf{u}}$ satisfy the same accuracy estimate. Therefore, it is still correct to view $\tilde{\mathbf{u}}$ as "the velocity" rather than $\mathbf{u}$. We will adopt this viewpoint, which allows a computationally cheaper formulation where the velocity $\mathbf{u}$ can be completely eliminated from the equations (4.17)–(4.19), as follows. First, define

$$\phi^{n+\frac{1}{2}} = p^{n+\frac{1}{2}} - p^{n-\frac{1}{2}} + \frac{\chi}{\text{Re}} \nabla \cdot \frac{\tilde{\mathbf{u}}^{n+1} + \tilde{\mathbf{u}}^n}{2}, \tag{4.28}$$

and write (4.18) at time level n to get

$$\frac{\mathbf{u}^n - \tilde{\mathbf{u}}^n}{\Delta t} = -\nabla \phi^{n-\frac{1}{2}}. \tag{4.29}$$

Then add (4.29) to (4.17) to get

$$\frac{\tilde{\mathbf{u}}^{n+1}-\tilde{\mathbf{u}}^{n}}{\Delta t}+\tilde{\mathbf{u}}^{n+\frac{1}{2}}\cdot\nabla\tilde{\mathbf{u}}^{n+\frac{1}{2}}=\frac{1}{\mathrm{Re}}\nabla^{2}\frac{\tilde{\mathbf{u}}^{n+1}+\tilde{\mathbf{u}}^{n}}{2}-\nabla\left(p^{n-\frac{1}{2}}+\phi^{n-\frac{1}{2}}\right). \tag{4.30}$$

Finally, taking the divergence of (4.18) and substituting equations (4.19) and (4.28) yields

$$\frac{\nabla\cdot\tilde{\mathbf{u}}^{n+1}}{\Delta t}=\nabla^{2}\phi^{n+\frac{1}{2}}. \tag{4.31}$$

The computational formulation consists of the two equations (4.30) and (4.31) together with the definition of ϕ from (4.28). Given $\tilde{\mathbf{u}}^{n}$, $\tilde{\mathbf{u}}^{n-1}$, $p^{n-\frac{1}{2}}$, $p^{n-\frac{3}{2}}$, one can obtain $\phi^{n-\frac{1}{2}}$ from (4.28). Advancing to the next time step involves computing $\tilde{\mathbf{u}}^{n+1}$, $\phi^{n+\frac{1}{2}}$, $p^{n+\frac{1}{2}}$ by first solving for $\tilde{\mathbf{u}}^{n+1}$ in (4.30), then solving the Poisson problem (4.31) for $\phi^{n+\frac{1}{2}}$, and finally obtaining $p^{n+\frac{1}{2}}$ from (4.28).

4.1.5 Perturbation Analysis Of The Stokes Equations

Starting from (4.28), (4.30), (4.31), we omit the advection terms and consider the time-discretized unsteady Stokes equations:

$$\frac{\tilde{\mathbf{u}}^{n+1}-\tilde{\mathbf{u}}^{n}}{\Delta t}=\frac{1}{\mathrm{Re}}\nabla^{2}\frac{\tilde{\mathbf{u}}^{n+1}+\tilde{\mathbf{u}}^{n}}{2}-\nabla\left(p^{n-\frac{1}{2}}+\phi^{n-\frac{1}{2}}\right), \tag{4.32}$$

$$\frac{\nabla\cdot\tilde{\mathbf{u}}^{n+1}}{\Delta t}=\nabla^{2}\phi^{n+\frac{1}{2}}, \tag{4.33}$$

$$\phi^{n+\frac{1}{2}}=p^{n+\frac{1}{2}}-p^{n-\frac{1}{2}}+\frac{\chi}{\mathrm{Re}}\nabla\cdot\frac{\tilde{\mathbf{u}}^{n+1}+\tilde{\mathbf{u}}^{n}}{2}. \tag{4.34}$$

We now take the average of (4.33) and itself written at the previous time level, and we write each variable as a Taylor expansion about the $n+\frac{1}{2}$ time level,

$$\nabla\cdot\frac{\tilde{\mathbf{u}}^{n+1}+\tilde{\mathbf{u}}^{n}}{2}=\frac{1}{2}\Delta t\nabla^{2}\left(\phi^{n+\frac{1}{2}}+\phi^{n-\frac{1}{2}}\right), \tag{4.35}$$

$$\nabla\cdot\left(\tilde{\mathbf{u}}^{n+\frac{1}{2}}+O(\Delta t^{2})\right)=\frac{1}{2}\Delta t\nabla^{2}\left(\phi^{n+\frac{1}{2}}+\phi^{n+\frac{1}{2}}-\Delta t\frac{\partial\phi}{\partial t}^{n+\frac{1}{2}}+O(\Delta t^{2})\right), \tag{4.36}$$

$$\nabla\cdot\tilde{\mathbf{u}}^{n+\frac{1}{2}}=\Delta t\nabla^{2}\phi^{n+\frac{1}{2}}+O(\Delta t^{2}). \tag{4.37}$$

Again writing each variable as a Taylor expansion about the $n+\frac{1}{2}$ time level, (4.34) can be written as

$$\begin{aligned}\phi^{n+\frac{1}{2}} &= \Delta t \frac{\partial p}{\partial t}^{n+\frac{1}{2}} + O(\Delta t^2) + \frac{\chi}{\mathrm{Re}} \nabla \cdot \left(\tilde{\mathbf{u}}^{n+\frac{1}{2}} + O(\Delta t^2) \right) \\ &= \Delta t \frac{\partial p}{\partial t}^{n+\frac{1}{2}} + \frac{\chi}{\mathrm{Re}} \nabla \cdot \tilde{\mathbf{u}}^{n+\frac{1}{2}} + O(\Delta t^2). \end{aligned} \tag{4.38}$$

We can also Taylor expand the pressure derivative in (4.38) about the $n+\frac{3}{2}$ time level, and substitute equation (4.37) to replace the divergence term,

$$\begin{aligned}\phi^{n+\frac{1}{2}} &= \Delta t \left(\frac{\partial p}{\partial t}^{n+\frac{3}{2}} + O(\Delta t) \right) + \frac{\chi}{\mathrm{Re}} \left(\Delta t \nabla^2 \phi^{n+\frac{1}{2}} + O(\Delta t^2) \right) + O(\Delta t^2) \\ &= \Delta t \frac{\partial p}{\partial t}^{n+\frac{3}{2}} + \frac{\chi}{\mathrm{Re}} \Delta t \nabla^2 \phi^{n+\frac{1}{2}} + O(\Delta t^2). \end{aligned} \tag{4.39}$$

Equation (4.39) written at the previous time level is

$$\phi^{n-\frac{1}{2}} = \Delta t \frac{\partial p}{\partial t}^{n+\frac{1}{2}} + \frac{\chi}{\mathrm{Re}} \Delta t \nabla^2 \phi^{n-\frac{1}{2}} + O(\Delta t^2). \tag{4.40}$$

Substituting (4.40) into itself yields

$$\begin{aligned}\phi^{n-\frac{1}{2}} &= \Delta t \frac{\partial p}{\partial t}^{n+\frac{1}{2}} + \frac{\chi}{\mathrm{Re}} \Delta t \nabla^2 \left(\Delta t \frac{\partial p}{\partial t}^{n+\frac{1}{2}} + \frac{\chi}{\mathrm{Re}} \Delta t \nabla^2 \phi^{n-\frac{1}{2}} + O(\Delta t^2) \right) + O(\Delta t^2) \\ &= \Delta t \frac{\partial p}{\partial t}^{n+\frac{1}{2}} + O(\Delta t^2). \end{aligned} \tag{4.41}$$

We now consider (4.32) and again write each variable (except $\phi^{n-\frac{1}{2}}$) as a Taylor expansion about the $n+\frac{1}{2}$ time level,

$$\frac{\partial \tilde{\mathbf{u}}}{\partial t}^{n+\frac{1}{2}} = \frac{1}{\mathrm{Re}} \nabla^2 \tilde{\mathbf{u}}^{n+\frac{1}{2}} - \nabla \left(p^{n+\frac{1}{2}} - \Delta t \frac{\partial p}{\partial t}^{n+\frac{1}{2}} + \phi^{n-\frac{1}{2}} \right) + O(\Delta t^2) \tag{4.42}$$

Substituting $\phi^{n-\frac{1}{2}}$ from (4.41) into (4.42) cancels out the pressure derivative to yield

$$\frac{\partial \tilde{\mathbf{u}}}{\partial t}^{n+\frac{1}{2}} = \frac{1}{\mathrm{Re}}\nabla^2 \tilde{\mathbf{u}}^{n+\frac{1}{2}} - \nabla p^{n+\frac{1}{2}} + O(\Delta t^2). \tag{4.43}$$

If we now define $\epsilon := \Delta t$ and drop all terms containing ϵ^2 or higher while taking the limit $\epsilon \to 0$ in equations (4.43), (4.37), (4.38), we obtain the following singularly perturbed form of the unsteady Stokes equations:

$$\frac{\partial \mathbf{u}_\epsilon}{\partial t} = \frac{1}{\mathrm{Re}}\nabla^2 \mathbf{u}_\epsilon - \nabla p_\epsilon, \tag{4.44}$$

$$\nabla \cdot \mathbf{u}_\epsilon = \epsilon \nabla^2 \phi_\epsilon, \tag{4.45}$$

$$\phi_\epsilon = \epsilon \frac{\partial p_\epsilon}{\partial t} + \frac{\chi}{\mathrm{Re}} \nabla \cdot \mathbf{u}_\epsilon. \tag{4.46}$$

According to [34], the compressibility error of (4.44)–(4.46) satisfies the estimate

$$\|\nabla \cdot \mathbf{u}_\epsilon\|_{L^\infty(0,T;L^2(\Omega))} \leq C \Delta t^{\frac{3}{2}}, \tag{4.47}$$

where the above norm is explained in Section 4.1.1. Equations (4.44)–(4.46) are an $O(\Delta t^2)$ perturbation of the incompressibility constraint $\nabla \cdot \mathbf{u} = 0$. This is easier to see if $\chi = 0$, because then we can eliminate ϕ_ϵ from (4.44)–(4.46) (remembering that $\epsilon = \Delta t$) to get

$$\frac{\partial \mathbf{u}_\epsilon}{\partial t} = \frac{1}{\mathrm{Re}}\nabla^2 \mathbf{u}_\epsilon - \nabla p_\epsilon, \tag{4.48}$$

$$\nabla \cdot \mathbf{u}_\epsilon = \Delta t^2 \nabla^2 \frac{\partial p_\epsilon}{\partial t}. \tag{4.49}$$

This $O(\Delta t^2)$ perturbation is much better than the penalty formulation $\nabla \cdot \mathbf{u} = -\Delta t\, p$ or the artificial compressibility formulation $\nabla \cdot \mathbf{u} = -\Delta t \frac{\partial p}{\partial t}$, which are both only $O(\Delta t)$ perturbations. All of these formulations are better than the Lattice-Boltzmann method (LBM) which is inherently a method for compressible flow. Even when the LBM is improved to approximate incompressible flow, the incompressibility constraint is still only approximated as $\nabla \cdot \mathbf{u} = -\frac{\Delta t}{h}\frac{\partial p}{\partial t}$ (see [36]). As an illustration of the huge difference between the LBM and our pressure projection method (4.49), suppose that we want to refine the resolution of a given simulation by halving Δt and halving h. The LBM would not gain any improvement to the incompressibility

constraint due to the factor $\Delta t/h$, however the pressure projection method (4.49) would become more incompressible by a factor of four due to the factor Δt^2.

4.1.6 Direction Factorized Pressure "Projection"

A new deviation from the traditional projection paradigm was proposed in [29], in which ∇^2 in (4.45) is replaced by a more general operator. As shown in [29], if an operator A with domain $D(A)$ and bilinear form $a(p,q) := \int_\Omega qAp\,d\mathbf{x}$ satisfies the following properties:

$$\begin{aligned} &a(p,q) = a(q,p), \qquad \forall\ p,q \in D(A) \\ &\|\nabla \cdot q\|_{L^2}^2 \le a(q,q), \quad \forall\ q \in D(A), \end{aligned} \tag{4.50}$$

then (4.44)–(4.46), with ∇^2 in (4.45) replaced by $(-A)$, is stable and consistent with the unsteady Stokes equations. A relatively efficient choice for A is given by $A := (1-\partial_{xx})(1-\partial_{yy})(1-\partial_{zz})$, where A has homogeneous Neumann boundary conditions if the velocity has Dirichlet boundary conditions, and vise-versa. This choice of A satisfies properties (4.50) provided that the domain Ω is a parallelepiped (see Lemma 2.2 in [29]). With this choice, (4.45) becomes

$$\nabla \cdot \mathbf{u} = -\Delta t(1-\partial_{xx})(1-\partial_{yy})(1-\partial_{zz})\phi, \tag{4.51}$$

which is identical to

$$\begin{aligned} (1-\partial_{xx})\theta &= -\frac{1}{\Delta t}\nabla \cdot \mathbf{u}, \\ (1-\partial_{yy})\psi &= \theta, \\ (1-\partial_{zz})\phi &= \psi, \end{aligned} \tag{4.52}$$

where θ and ψ are temporary intermediate variables. Equation (4.52) is a computationally cheap set of one-dimensional problems where ϕ can be solved for directly, for example, using the Thomas algorithm for tridiagonal matrices. This allows for a parallel implementation of the incompressibility constraint that uses reduced communication between CPU cores (see [28] for details). Without the direction factorization, one must solve the computationally expensive three-dimensional Poisson problem (4.45) for ϕ, which is typically done using iterative solvers for sparse matrices.

The unsteady Stokes equations (4.32)–(4.34) with the direction factorized pres-

sure "projection" as explained above are:

$$\frac{\tilde{\mathbf{u}}^{n+1}-\tilde{\mathbf{u}}^{n}}{\Delta t}=\frac{1}{\mathrm{Re}}\nabla^2\frac{\tilde{\mathbf{u}}^{n+1}+\tilde{\mathbf{u}}^{n}}{2}-\nabla\left(p^{n-\frac{1}{2}}+\phi^{n-\frac{1}{2}}\right), \tag{4.53}$$

$$\frac{\nabla\cdot\tilde{\mathbf{u}}^{n+1}}{\Delta t}=-(1-\partial_{xx})(1-\partial_{yy})(1-\partial_{zz})\phi^{n+\frac{1}{2}}, \tag{4.54}$$

$$\phi^{n+\frac{1}{2}}=p^{n+\frac{1}{2}}-p^{n-\frac{1}{2}}+\frac{\chi}{\mathrm{Re}}\nabla\cdot\frac{\tilde{\mathbf{u}}^{n+1}+\tilde{\mathbf{u}}^{n}}{2}, \tag{4.55}$$

and (4.53)–(4.55) have been proven to be unconditionally stable for $\chi=0$ in simple-shaped domains (see Theorem 4.2 of [29]). According to [30], these equations are proven to be at least $O(\Delta t^{3/2})$ accurate in time for the velocity in the $L^2(\Omega)$ norm. Several other error bounds are included in [30], but a proof of $O(\Delta t^2)$ accuracy does not currently exist, despite the fact that all numerical results indicate it is the case. Also in [30], it is demonstrated numerically that the scheme (4.53)–(4.55) has error no worse than twice that of the same scheme where the pressure Poisson problem (4.45) is solved without direction splitting.

This concludes the discussion of pressure projection schemes. The scheme that we use in this thesis is given by (4.53)–(4.55), however we will also perform direction-splitting of the momentum equation (4.53), as explained in the following sections.

4.2 Discretizations of the Momentum Equation

In this section, we will consider solving for $\mathbf{u}^{n+1}$ in the momentum equation

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\Delta t}+\mathbf{u}\cdot\nabla\mathbf{u}=-\nabla p+\frac{1}{\mathrm{Re}}\nabla^2\mathbf{u}^{n+1}, \tag{4.56}$$

where the terms p and $\mathbf{u}\cdot\nabla\mathbf{u}$ are assumed to be known from previous time levels, as in (4.30). The problem then boils down to solving the implicit diffusion equation

$$\left(1-\frac{\Delta t}{\mathrm{Re}}\nabla^2\right)\frac{\mathbf{u}^{n+1}}{\Delta t}=\frac{\mathbf{u}^{n}}{\Delta t}+\mathbf{f}^{n+\frac{1}{2}}, \tag{4.57}$$

where $\mathbf{f}^{n+\frac{1}{2}}=-\mathbf{u}\cdot\nabla\mathbf{u}-\nabla p$ is some known right-hand side vector evaluated at the $n+\frac{1}{2}$ time level. Solving equation (4.57) involves computing $\left(1-\frac{\Delta t}{\mathrm{Re}}\nabla^2\right)^{-1}\left(\frac{\mathbf{u}^n}{\Delta t}+\mathbf{f}^{n+\frac{1}{2}}\right)$, which is computationally expensive for three-dimensional problems, and one must

typically use iterative solvers for sparse matrices. An alternative is to use direction splitting, or ADI (Alternating Direction Implicit) methods. We now present four different splitting methods: A simple splitting based on the first-order backward difference (BDF1), a second-order splitting based on the Crank–Nicolson formulation called Douglas splitting, a new second-order accurate "modified Douglas" splitting, and a splitting based on the second-order backward difference (BDF2). The new "modified Douglas" splitting is what we will use in this thesis.

4.2.1 BDF1 Splitting

A simple splitting scheme that approximates (4.57) is

$$\left(1-\frac{\Delta t}{\mathrm{Re}}\partial_{xx}\right)\left(1-\frac{\Delta t}{\mathrm{Re}}\partial_{yy}\right)\left(1-\frac{\Delta t}{\mathrm{Re}}\partial_{zz}\right)\frac{\mathbf{u}^{n+1}}{\Delta t}=\frac{\mathbf{u}^{n}}{\Delta t}+\mathbf{f}^{n+\frac{1}{2}}, \tag{4.58}$$

which when expanded yields

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\Delta t}=\frac{1}{\mathrm{Re}}\nabla^2\mathbf{u}^{n+1}+\mathbf{f}^{n+\frac{1}{2}}-\frac{\Delta t}{\mathrm{Re}^2}\left(\mathbf{u}^{n+1}_{xxyy}+\mathbf{u}^{n+1}_{xxzz}+\mathbf{u}^{n+1}_{yyzz}\right)+\frac{\Delta t^2}{\mathrm{Re}^3}\mathbf{u}^{n+1}_{xxyyzz}. \tag{4.59}$$

We can see that the factorized operator approximates the momentum equation with two error terms, $\frac{\Delta t}{\mathrm{Re}^2}\left(\mathbf{u}^{n+1}_{xxyy}+\mathbf{u}^{n+1}_{xxzz}+\mathbf{u}^{n+1}_{yyzz}\right)$ and $\frac{\Delta t^2}{\mathrm{Re}^3}\mathbf{u}^{n+1}_{xxyyzz}$. Since we are considering viscous flow, it is reasonable to assume that the spatial derivatives of $\mathbf{u}$ are bounded in the fluid domain Ω_f. Therefore, the leading error term is of order Δt, and this method cannot be better than first-order accurate in time. Note: all splitting schemes discussed in this thesis work in both two and three dimensions, where the 2D version can be obtained in the obvious way by omitting the last factor $\left(1-\frac{\Delta t}{\mathrm{Re}}\partial_{zz}\right)$ in (4.58).

4.2.2 Douglas Splitting

An $O(\Delta t^2)$ accurate splitting scheme due to Douglas [20] is

$$\left(1-\frac{\Delta t}{2\mathrm{Re}}\partial_{xx}\right)\left(1-\frac{\Delta t}{2\mathrm{Re}}\partial_{yy}\right)\left(1-\frac{\Delta t}{2\mathrm{Re}}\partial_{zz}\right)\left(\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}\right)=\frac{1}{\mathrm{Re}}\nabla^2\mathbf{u}^n+\mathbf{f}^{n+\frac{1}{2}}, \tag{4.60}$$

which when expanded yields

$$\begin{aligned}\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}&=\frac{1}{\mathrm{Re}}\nabla^2\left(\frac{\mathbf{u}^{n+1}+\mathbf{u}^n}{2}\right)+\mathbf{f}^{n+\frac{1}{2}}\\&-\frac{\Delta t^2}{4\mathrm{Re}^2}\left(\partial_{xxyy}+\partial_{xxzz}+\partial_{yyzz}\right)\left(\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}\right)+\frac{\Delta t^3}{8\mathrm{Re}^3}\partial_{xxyyzz}\left(\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}\right).\end{aligned} \tag{4.61}$$

The Douglas splitting scheme is an approximation to the Crank–Nicolson form of the diffusion equation,

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}=\frac{1}{\mathrm{Re}}\nabla^2\left(\frac{\mathbf{u}^{n+1}+\mathbf{u}^n}{2}\right)+\mathbf{f}^{n+\frac{1}{2}}, \tag{4.62}$$

which we also use in (4.13), (4.30). If we assume bounded spatial derivatives, then the leading error term in (4.61) is of order Δt^2 because the term $\left(\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}\right)$ approximates $\frac{\partial \mathbf{u}}{\partial t}$ which is an acceleration and therefore must be bounded for any realistic flow. Furthermore, on steady flows, $\frac{\partial \mathbf{u}}{\partial t}\to 0$ and the splitting error eventually vanishes completely. Similar to (4.52), the Douglas splitting (4.60) can be written as:

$$\begin{aligned}\left(\frac{\boldsymbol{\xi}^{n+1}-\mathbf{u}^n}{\Delta t}\right)&=\frac{1}{\mathrm{Re}}\nabla^2\mathbf{u}^n+\mathbf{f}^{n+\frac{1}{2}},\\\left(1-\frac{\Delta t}{2\mathrm{Re}}\partial_{xx}\right)\left(\frac{\boldsymbol{\eta}^{n+1}-\mathbf{u}^n}{\Delta t}\right)&=\left(\frac{\boldsymbol{\xi}^{n+1}-\mathbf{u}^n}{\Delta t}\right),\\\left(1-\frac{\Delta t}{2\mathrm{Re}}\partial_{yy}\right)\left(\frac{\boldsymbol{\zeta}^{n+1}-\mathbf{u}^n}{\Delta t}\right)&=\left(\frac{\boldsymbol{\eta}^{n+1}-\mathbf{u}^n}{\Delta t}\right),\\\left(1-\frac{\Delta t}{2\mathrm{Re}}\partial_{zz}\right)\left(\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}\right)&=\left(\frac{\boldsymbol{\zeta}^{n+1}-\mathbf{u}^n}{\Delta t}\right),\end{aligned} \tag{4.63}$$

where $\left(\frac{\boldsymbol{\xi}^{n+1}-\mathbf{u}^n}{\Delta t}\right)$, $\left(\frac{\boldsymbol{\eta}^{n+1}-\mathbf{u}^n}{\Delta t}\right)$, $\left(\frac{\boldsymbol{\zeta}^{n+1}-\mathbf{u}^n}{\Delta t}\right)$ are intermediate expressions defined by introducing temporary variables $\boldsymbol{\xi}^{n+1},\boldsymbol{\eta}^{n+1},\boldsymbol{\zeta}^{n+1}$. By rearranging some terms, (4.63)

can also be written as

$$\begin{aligned}
\frac{\boldsymbol{\xi}^{n+1}-\mathbf{u}^n}{\Delta t} &= \frac{1}{\mathrm{Re}}\nabla^2\mathbf{u}^n + \mathbf{f}^{n+\frac{1}{2}},\\
\frac{\boldsymbol{\eta}^{n+1}-\boldsymbol{\xi}^{n+1}}{\Delta t} &= \frac{1}{2\mathrm{Re}}\partial_{xx}\left(\boldsymbol{\eta}^{n+1}-\mathbf{u}^n\right),\\
\frac{\boldsymbol{\zeta}^{n+1}-\boldsymbol{\eta}^{n+1}}{\Delta t} &= \frac{1}{2\mathrm{Re}}\partial_{yy}\left(\boldsymbol{\zeta}^{n+1}-\mathbf{u}^n\right),\\
\frac{\mathbf{u}^{n+1}-\boldsymbol{\zeta}^{n+1}}{\Delta t} &= \frac{1}{2\mathrm{Re}}\partial_{zz}\left(\mathbf{u}^{n+1}-\mathbf{u}^n\right).
\end{aligned} \tag{4.64}$$

However, the previous form (4.63) is more efficient numerically because the solution of each one-dimensional problem is exactly the right-hand side of the next problem.

Proof of stability in 2D

As given in Section 2.2.3 of [60], there is a proof that the Douglas splitting (4.60) (in 2D only) is unconditionally stable for the diffusion equation even if the spatial operators in the diffusion terms do not commute. For reference, we present here the same proof as in [60]. Define the operators

$$\begin{aligned}
&A_1 = -\tfrac{1}{\mathrm{Re}}\partial_{xx}, && A_2 = -\tfrac{1}{\mathrm{Re}}\partial_{yy}, && A = A_1 + A_2,\\
&B_1 = \left(1+\tfrac{\Delta t}{2}A_1\right), && B_2 = \left(1+\tfrac{\Delta t}{2}A_2\right), &&
\end{aligned} \tag{4.65}$$

which we will assume are positive operators (and thus have inverses) but not necessarily commutative operators. The 2D Douglas splitting with a forcing term $\mathbf{f}$ is

$$B_1B_2\left(\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}\right) = -A\mathbf{u}^n + \mathbf{f}^{n+\frac{1}{2}}, \tag{4.66}$$

or equivalently,

$$B_1B_2\mathbf{u}^{n+1} = B_1B_2\mathbf{u}^n - \Delta t A\mathbf{u}^n + \Delta t\mathbf{f}^{n+\frac{1}{2}}. \tag{4.67}$$

Multiplying by the inverse of B_1 yields

$$B_2\mathbf{u}^{n+1} = B_1^{-1}\left(B_1B_2 - \Delta t A\right)\mathbf{u}^n + \Delta t B_1^{-1}\mathbf{f}^{n+\frac{1}{2}}, \tag{4.68}$$

and the operator $(B_1B_2 - \Delta tA)$ can be expanded as

$$\begin{aligned}(B_1B_2 - \Delta tA) &= \left[\left(1 + \tfrac{\Delta t}{2}A_1 + \tfrac{\Delta t}{2}A_2 + \tfrac{\Delta t^2}{4}A_1A_2\right) - \Delta t\,(A_1 + A_2)\right] \\ &= \left[1 - \tfrac{\Delta t}{2}A_1 - \tfrac{\Delta t}{2}A_2 + \tfrac{\Delta t^2}{4}A_1A_2\right] \\ &= \left(1 - \tfrac{\Delta t}{2}A_1\right)\left(1 - \tfrac{\Delta t}{2}A_2\right). \end{aligned} \tag{4.69}$$

Combining (4.68) and (4.69), we get

$$\begin{aligned}B_2\mathbf{u}^{n+1} &= B_1^{-1}\left(1 - \tfrac{\Delta t}{2}A_1\right)\left(1 - \tfrac{\Delta t}{2}A_2\right)\mathbf{u}^n + \Delta tB_1^{-1}\mathbf{f}^{n+\frac{1}{2}} \\ &= \underbrace{B_1^{-1}\left(1 - \tfrac{\Delta t}{2}A_1\right)}_{C_1}\underbrace{\left(1 - \tfrac{\Delta t}{2}A_2\right)B_2^{-1}}_{C_2}B_2\mathbf{u}^n + \Delta tB_1^{-1}\mathbf{f}^{n+\frac{1}{2}}. \end{aligned} \tag{4.70}$$

Taking a norm (e.g. L^2 norm) of (4.70) and using the property of operator norms $\|Lx\| \le \|L\|\|x\|$ for a linear operator L, we get

$$\|B_2\mathbf{u}^{n+1}\| \le \|C_1\|\|C_2\|\|B_2\mathbf{u}^n\| + \Delta t\|B_1^{-1}\mathbf{f}^{n+\frac{1}{2}}\|. \tag{4.71}$$

We can now use Kellogg's lemma, which says that for any nonnegative operator L,

$$\|\,(1 + L)^{-1}\,(1 - L)\,\| \le 1. \tag{4.72}$$

Kellogg's lemma is the operator equivalent of the property $\left|\frac{1-x}{1+x}\right| \le 1$ for all scalar $x \ge 0$. The operators C_1 and C_2 defined by (4.70) satisfy Kellogg's lemma (noting that A_2 and B_2 commute). Therefore, (4.71) implies

$$\|B_2\mathbf{u}^{n+1}\| \le \|B_2\mathbf{u}^n\| + \Delta t\|B_1^{-1}\mathbf{f}^{n+\frac{1}{2}}\|, \tag{4.73}$$

and summing (4.73) from $n = 0, ..., N$ causes telescoping cancellation and gives

$$\|B_2\mathbf{u}^{N+1}\| \le \|B_2\mathbf{u}^0\| + \Delta t\sum_{n=0}^{N}\|B_1^{-1}\mathbf{f}^{n+\frac{1}{2}}\|, \tag{4.74}$$

which is proof of stability in the norm $\|B_2(\cdot)\| = \|\left(1 + \frac{\Delta t}{2}A_2\right)(\cdot)\|$, and also in the standard norm $\|(\cdot)\|$ since A_2 is a positive operator and therefore $\|\mathbf{u}^{N+1}\| \le \|\left(1 + \frac{\Delta t}{2}A_2\right)\mathbf{u}^{N+1}\|$.

The above proof guarantees that the Douglas scheme is stable for any domain geometry and for any positive diffusion operators. However, the proof does not work in 3D, and our numerical testing (see Section 5.1.3) indicates that the Douglas splitting can be unstable when using non-commutative operators such as (3.6) in 3D. The Douglas scheme has been proven stable in 3D only in simple-shaped (parallelepiped) domains when at least two of the diffusion operators commute.

4.2.3 Modified Douglas Splitting

In order to address the stability issue of the Douglas splitting scheme in 3D, we invented a new "modified Douglas" splitting scheme,

$$\begin{aligned}
\frac{\boldsymbol{\xi}^{n+1}-\mathbf{u}^n}{\Delta t} &= \frac{1}{\mathrm{Re}}\left(\partial_{xx}\boxed{\boldsymbol{\eta}^n}+\partial_{yy}\boxed{\boldsymbol{\zeta}^n}+\partial_{zz}\mathbf{u}^n\right)+\mathbf{f}^{n+\frac{1}{2}},\\
\frac{\boldsymbol{\eta}^{n+1}-\boldsymbol{\xi}^{n+1}}{\Delta t} &= \frac{1}{2\mathrm{Re}}\partial_{xx}\left(\boldsymbol{\eta}^{n+1}-\boxed{\boldsymbol{\eta}^n}\right),\\
\frac{\boldsymbol{\zeta}^{n+1}-\boldsymbol{\eta}^{n+1}}{\Delta t} &= \frac{1}{2\mathrm{Re}}\partial_{yy}\left(\boldsymbol{\zeta}^{n+1}-\boxed{\boldsymbol{\zeta}^n}\right),\\
\frac{\mathbf{u}^{n+1}-\boldsymbol{\zeta}^{n+1}}{\Delta t} &= \frac{1}{2\mathrm{Re}}\partial_{zz}\left(\mathbf{u}^{n+1}-\mathbf{u}^n\right).
\end{aligned} \tag{4.75}$$

where if the boxed variables are replaced by $\mathbf{u}^n$ then the original Douglas scheme (4.64) is recovered. The motivation for making the above modification is the following. Operators such as (3.6) can be non-commutative and lopsided because of the term $(u_{bL}-u_i)/\gamma_L$ when γ_L is small. In the original Douglas scheme (4.64), the end-of-step field $\mathbf{u}^{n+1}$ is obtained by inverting the potentially non-commutative and lopsided operator $\left(1-\frac{\Delta t}{2\mathrm{Re}}\partial_{zz}\right)$, so the $\mathbf{u}^{n+1}$ field is guaranteed to be "smooth" with respect to this operator in the z direction, i.e., $(u_{bL}-u_i)/\gamma_L$ is controlled in some sense. However, $\mathbf{u}^{n+1}$ does not in general have the same degree of discrete regularity in the x and y directions because these directions are not treated implicitly at the last step when solving the z directional problem. If γ_L is small, then applying an explicit operator containing the term $(u_{bL}-u_i)/\gamma_L$ to a field where u_{bL} and u_i differ by moderate amount will generate an extremely large value. To prevent this situation, the modified Douglas scheme (4.75) pairs each explicit time level n operator with the sub-step field that was computed with the matching implicit operator. While we could not obtain a proof of unconditionally stability of the modified Douglas scheme,

numerical evidence suggests that it is always stable for the diffusion equation (see Section 5.1.3). More discussion of the modified Douglas scheme, and how it relates to a class of schemes known in the Russian literature as "vectorial additive schemes", can be found in [4].

The modified Douglas scheme is second order consistent in time, which can be seen by eliminating the intermediate variables $\boldsymbol{\xi}, \boldsymbol{\eta}, \boldsymbol{\zeta}$ as follows. First, sum together all four equations of (4.75) to obtain

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t} = \frac{1}{2\mathrm{Re}}\left[\partial_{xx}\left(\boldsymbol{\eta}^{n+1}+\boxed{\boldsymbol{\eta}^n}\right)+\partial_{yy}\left(\boldsymbol{\zeta}^{n+1}+\boxed{\boldsymbol{\zeta}^n}\right)+\partial_{zz}\left(\mathbf{u}^{n+1}+\mathbf{u}^n\right)\right]+\mathbf{f}^{n+\frac{1}{2}}. \tag{4.76}$$

For the remainder of this algebra we drop the box notation because it will cease to have meaning. Solving the third equation of (4.75) for $\boldsymbol{\eta}^{n+1}$ yields

$$\boldsymbol{\eta}^{n+1} = \boldsymbol{\zeta}^{n+1} - \frac{\Delta t}{2\mathrm{Re}}\partial_{yy}\left(\boldsymbol{\zeta}^{n+1}-\boldsymbol{\zeta}^n\right). \tag{4.77}$$

Adding (4.77) to itself written at the previous time level is

$$\boldsymbol{\eta}^{n+1}+\boldsymbol{\eta}^n = \boldsymbol{\zeta}^{n+1}+\boldsymbol{\zeta}^n - \frac{\Delta t}{2\mathrm{Re}}\partial_{yy}\left(\boldsymbol{\zeta}^{n+1}-\boldsymbol{\zeta}^{n-1}\right), \tag{4.78}$$

which can be substituted into (4.76) to eliminate $\boldsymbol{\eta}$, giving

$$\begin{aligned}\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t} = \frac{1}{2\mathrm{Re}}\Bigg[&\partial_{xx}\left(\boldsymbol{\zeta}^{n+1}+\boldsymbol{\zeta}^n\right)-\frac{\Delta t}{2\mathrm{Re}}\partial_{xxyy}\left(\boldsymbol{\zeta}^{n+1}-\boldsymbol{\zeta}^{n-1}\right)\\ &+\partial_{yy}\left(\boldsymbol{\zeta}^{n+1}+\boldsymbol{\zeta}^n\right)+\partial_{zz}\left(\mathbf{u}^{n+1}+\mathbf{u}^n\right)\Bigg]+\mathbf{f}^{n+\frac{1}{2}}.\end{aligned} \tag{4.79}$$

Solving the fourth equation of (4.75) for $\boldsymbol{\zeta}^{n+1}$ yields

$$\boldsymbol{\zeta}^{n+1} = \mathbf{u}^{n+1} - \frac{\Delta t}{2\mathrm{Re}}\partial_{zz}\left(\mathbf{u}^{n+1}-\mathbf{u}^n\right), \tag{4.80}$$

and (4.80) together with itself written at two previous time levels can be used to

produce the following two equations:

$$\begin{aligned}\boldsymbol{\zeta}^{n+1} + \boldsymbol{\zeta}^{n} &= \mathbf{u}^{n+1} + \mathbf{u}^{n} - \frac{\Delta t}{2\mathrm{Re}}\partial_{zz}\left(\mathbf{u}^{n+1} - \mathbf{u}^{n-1}\right), \\ \boldsymbol{\zeta}^{n+1} - \boldsymbol{\zeta}^{n-1} &= \mathbf{u}^{n+1} - \mathbf{u}^{n-1} - \frac{\Delta t}{2\mathrm{Re}}\partial_{zz}\left(\mathbf{u}^{n+1} - \mathbf{u}^{n-1} - \mathbf{u}^{n} + \mathbf{u}^{n-2}\right).\end{aligned} \tag{4.81}$$

Finally, we can eliminate $\boldsymbol{\zeta}$ by substituting (4.81) into (4.79) to get

$$\begin{aligned}\frac{\mathbf{u}^{n+1} - \mathbf{u}^{n}}{\Delta t} = \frac{1}{2\mathrm{Re}}\Bigg[&\partial_{xx}\left(\mathbf{u}^{n+1} + \mathbf{u}^{n} - \frac{\Delta t}{2\mathrm{Re}}\partial_{zz}\left(\mathbf{u}^{n+1} - \mathbf{u}^{n-1}\right)\right) \\ &- \frac{\Delta t}{2\mathrm{Re}}\partial_{xxyy}\left(\mathbf{u}^{n+1} - \mathbf{u}^{n-1} - \frac{\Delta t}{2\mathrm{Re}}\partial_{zz}\left(\mathbf{u}^{n+1} - \mathbf{u}^{n-1} - \mathbf{u}^{n} + \mathbf{u}^{n-2}\right)\right) \\ &+ \partial_{yy}\left(\mathbf{u}^{n+1} + \mathbf{u}^{n} - \frac{\Delta t}{2\mathrm{Re}}\partial_{zz}\left(\mathbf{u}^{n+1} - \mathbf{u}^{n-1}\right)\right) \\ &+ \partial_{zz}\left(\mathbf{u}^{n+1} + \mathbf{u}^{n}\right)\Bigg] + \mathbf{f}^{n+\frac{1}{2}},\end{aligned} \tag{4.82}$$

which simplifies to

$$\begin{aligned}\frac{\mathbf{u}^{n+1} - \mathbf{u}^{n}}{\Delta t} = &\frac{1}{\mathrm{Re}}\nabla^2\left(\frac{\mathbf{u}^{n+1} + \mathbf{u}^{n}}{2}\right) - \frac{\Delta t^2}{2\mathrm{Re}^2}\left(\partial_{xxyy} + \partial_{xxzz} + \partial_{yyzz}\right)\left(\frac{\mathbf{u}^{n+1} - \mathbf{u}^{n-1}}{2\Delta t}\right) \\ &+ \frac{\Delta t^4}{4\mathrm{Re}^3}\partial_{xxyyzz}\left[\frac{1}{2\Delta t}\left(\frac{\mathbf{u}^{n+1} - \mathbf{u}^{n}}{\Delta t} - \frac{\mathbf{u}^{n-1} - \mathbf{u}^{n-2}}{\Delta t}\right)\right] + \mathbf{f}^{n+\frac{1}{2}},\end{aligned} \tag{4.83}$$

which we can see is an approximation to the Crank–Nicolson form of the diffusion equation (4.62). If we assume bounded spatial derivatives, then the leading error term is of order Δt^2 since the term $\left(\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n-1}}{2\Delta t}\right)$ is a "leapfrog" approximation to $\frac{\partial \mathbf{u}}{\partial t}$, which is an acceleration and therefore must be bounded for any realistic flow. The next error term is of order Δt^4 since the term $\left[\frac{1}{2\Delta t}\left(\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\Delta t} - \frac{\mathbf{u}^{n-1}-\mathbf{u}^{n-2}}{\Delta t}\right)\right]$ is an approximation to $\frac{\partial^2 \mathbf{u}}{\partial t^2}$, which is again a physical quantity that is typically bounded. Comparing (4.83) to (4.61), we can see that the leading error term of the modified Douglas scheme is twice as big as the leading error term of the original Douglas scheme, but they are both $O(\Delta t^2)$.

By re-arranging some terms in (4.75), we can get the computational form of the

modified Douglas scheme,

$$\begin{aligned}
\boldsymbol{\xi}^{n+1} &= \mathbf{u}^n + \Delta t \left[\frac{1}{\mathrm{Re}} \left(\partial_{xx} \boxed{\boldsymbol{\eta}^n} + \partial_{yy} \boxed{\boldsymbol{\zeta}^n} + \partial_{zz} \mathbf{u}^n \right) + \mathbf{f}^{n+\frac{1}{2}} \right], \\
\left(1 - \frac{\Delta t}{2\mathrm{Re}} \partial_{xx} \right) \boldsymbol{\eta}^{n+1} &= \boldsymbol{\xi}^{n+1} - \frac{\Delta t}{2\mathrm{Re}} \partial_{xx} \boxed{\boldsymbol{\eta}^n}, \\
\left(1 - \frac{\Delta t}{2\mathrm{Re}} \partial_{yy} \right) \boldsymbol{\zeta}^{n+1} &= \boldsymbol{\eta}^{n+1} - \frac{\Delta t}{2\mathrm{Re}} \partial_{yy} \boxed{\boldsymbol{\zeta}^n}, \\
\left(1 - \frac{\Delta t}{2\mathrm{Re}} \partial_{zz} \right) \mathbf{u}^{n+1} &= \boldsymbol{\zeta}^{n+1} - \frac{\Delta t}{2\mathrm{Re}} \partial_{zz} \mathbf{u}^n.
\end{aligned} \tag{4.84}$$

Compared to the original Douglas scheme in its computational form (4.63), the computational form of the modified Douglas scheme is about 5% less efficient due to the additional loops required for the right-hand side. Also, six additional storage vectors are required in 3D for the terms $\boldsymbol{\eta}^n_{xx}, \boldsymbol{\zeta}^n_{yy}$. However, unlike (4.63) which computes $\mathbf{u}^{n+1} - \mathbf{u}^n$, the computational form (4.84) computes $\mathbf{u}^{n+1}$ directly, which has several advantages. The first is that $\mathbf{u}^n$ is not needed to recover $\mathbf{u}^{n+1}$ at the final step, so the memory of $\mathbf{u}^n$ can be used to store temporary variables such as the tri-diagonal matrix associated with solving the implicit one-dimensional problems. This allows a memory savings equivalent to three vectors, which means that the modified Douglas scheme really only requires three additional vectors of storage, not six. Also, the whole point of the modified Douglas scheme is to be able to use boundary fitted stencils, and the number of modified stencils that need to be stored at a time is significantly reduced if $\boldsymbol{\eta}^n_{xx}, \boldsymbol{\zeta}^n_{yy}, \mathbf{u}^n_{zz}$ are stored. In fact, when grid stencils are stored rather than recomputed (which can be expensive), the formulation (4.84) is actually *more* memory efficient than the formulation (4.63). The second advantage of the computational form (4.84) is that for problems where the shape of the fluid domain changes in time (which is the case for particulate flows), boundary conditions for the implicit one-dimensional problems are *much* easier to specify for the Eulerian velocity $\mathbf{u}^{n+1}$ than for the Eulerian velocity time-difference $\mathbf{u}^{n+1} - \mathbf{u}^n$. Even for a boundary that moves with constant velocity, we cannot simply impose the Eulerian condition $\mathbf{u}^{n+1} - \mathbf{u}^n = 0$ because each grid point may be outside Ω_f at time level n but inside Ω_f at time level $n+1$ (and vise versa).

The factorization order in which the operators ∂_{xx}, ∂_{yy}, ∂_{zz} appear in (4.84) is arbitrary — it is equally valid to solve the one directional problems in the y direction

first, then the x direction, then the z direction, for example. In practice, it is best to solve for $\mathbf{u} = (u, v, w)$ such that the x direction problem for u is solved last, the y direction problem for v is solved last, and the z direction problem for w is solved last. This has two benefits: First, because the implicit problem for u in the x direction is solved last, u will be smoother in the x direction for computing u_x, which is required for the divergence. A similar property is also true for the terms v_y and w_z in the divergence. Second, in the context of the MAC grid, it allows easier enforcement of most boundary conditions on the end-of-step velocity field. For example, solving for w in the z direction last allows direct enforcement of boundary conditions of the form $w\big|_{z=0,L} = 0$ or $\frac{\partial w}{\partial z}\big|_{z=0,L} = 0$ on the final solution.

When the associated spatial operators are time-dependent, the modified Douglas scheme is stable (by extensive numerical evidence only) in the following form,

$$\begin{aligned}
\boldsymbol{\xi}^{n+1} &= \mathbf{u}^n + \Delta t \left[\frac{1}{\mathrm{Re}} \left(\partial_{xx}^n \boxed{\boldsymbol{\eta}^n} + \partial_{yy}^n \boxed{\boldsymbol{\zeta}^n} + \partial_{zz}^n \mathbf{u}^n \right) + \mathbf{f}^{n+\frac{1}{2}} \right], \\
\left(1 - \frac{\Delta t}{2\mathrm{Re}} \partial_{xx}^{n+1} \right) \boldsymbol{\eta}^{n+1} &= \boldsymbol{\xi}^{n+1} - \frac{\Delta t}{2\mathrm{Re}} \partial_{xx}^n \boxed{\boldsymbol{\eta}^n}, \\
\left(1 - \frac{\Delta t}{2\mathrm{Re}} \partial_{yy}^{n+1} \right) \boldsymbol{\zeta}^{n+1} &= \boldsymbol{\eta}^{n+1} - \frac{\Delta t}{2\mathrm{Re}} \partial_{yy}^n \boxed{\boldsymbol{\zeta}^n}, \\
\left(1 - \frac{\Delta t}{2\mathrm{Re}} \partial_{zz}^{n+1} \right) \mathbf{u}^{n+1} &= \boldsymbol{\zeta}^{n+1} - \frac{\Delta t}{2\mathrm{Re}} \partial_{zz}^n \mathbf{u}^n,
\end{aligned} \tag{4.85}$$

where ∂^n indicates the spatial operator at time level n, and again the boxed variables can be replaced by $\mathbf{u}^n$ to obtain the original Douglas scheme. For time-dependent spatial operators, it becomes difficult to argue rigorously that the Douglas or modified Douglas splitting schemes maintain $O(\Delta t^2)$ accuracy because (4.85) cannot be put into the form (4.83) without involving terms such as $\partial_{zz}^{n+1} \mathbf{u}^n$ where the time level of the spatial operator does not match the time level of the field, and the error associated with these terms is unclear. The BDF1 splitting (4.58) does not suffer from the same issue since spatial operators are only applied to fields at time level $n + 1$. Therefore, the modified Douglas splitting (4.85) can be shown to be at least consistent by comparing numerical solutions to those obtained using the BDF1 splitting. Numerical results in Section 9.1 confirm that the modified Douglas and BDF1 splitting converge to the same solution. In fact, the modified Douglas scheme converges faster, suggesting better than $O(\Delta t)$ even when Ω_f is time dependent.

Stability of Modified Douglas Splitting

No proof of stability exists for either the modified or original Douglas schemes in 3D if the diffusion operators do not commute. In this section, we prove stability of the modified Douglas splitting (4.83) in the special case where $\Omega_f = \Omega$ is a 2D or 3D box domain with homogeneous Dirichlet boundary conditions on $\partial\Omega$, zero right-hand side forcing term $\mathbf{f}$, and we will assume that the diffusion operators commute. We use the standard notation $\langle a, b\rangle = \int_\Omega ab\, d\mathbf{x}$ for the L^2 inner product with corresponding norm $\|a\| = \sqrt{\langle a, a\rangle}$. We will make use of the identity

$$2\langle a, a-b\rangle = \|a\|^2 + \|a-b\|^2 - \|b\|^2, \tag{4.86}$$

which can be proven as follows:

$$\begin{aligned}
2\langle a, a-b\rangle &= \int (2a)(a-b)\, d\mathbf{x} \\
&= \int (a-b)^2\, d\mathbf{x} - \int (a-b)^2\, d\mathbf{x} + \int (2a)(a-b)\, d\mathbf{x} \\
&= \int (a-b)^2\, d\mathbf{x} + \int (a+b)(a-b)\, d\mathbf{x} \\
&= \int (a-b)^2\, d\mathbf{x} + \int a^2\, d\mathbf{x} - \int b^2\, d\mathbf{x} \\
&= \|a-b\|^2 + \|a\|^2 - \|b\|^2 \qquad \square
\end{aligned} \tag{4.87}$$

Since all vector components of (4.83) are decoupled, it is sufficient to prove stability for u instead of $\mathbf{u}$. To be concise in writing down many terms, we will denote derivatives using subscript notation. For example, we will allow the derivative $\frac{\partial u}{\partial y}$ to be denoted by either u_y or u_{x_2}, where $(x, y, z) = (x_1, x_2, x_3)$. We will also use the notation x_i, where $x_{i=0 \bmod 3} = x$, $x_{i=1 \bmod 3} = y$, and $x_{i=2 \bmod 3} = z$.

A useful property of box domains $\Omega = [0,1]^3$ with $u = 0$ on the boundary is that the tangential derivatives on the boundary are zero, i.e., $u_y = u_z = 0$ whenever $x = 0, 1$, $u_x = u_z = 0$ whenever $y = 0, 1$, and $u_x = u_y = 0$ whenever $z = 0, 1$. Combined with the assumption that all spatial derivatives commute, the following integration-by-parts formulas hold:

Integration-by-parts formula 1:

$$\langle u, v_{x_i x_i} \rangle = u v_{x_i} \big|_{x_i=0,1} - \langle u_{x_i}, v_{x_i} \rangle = -\langle u_{x_i}, v_{x_i} \rangle, \tag{4.88}$$

Integration-by-parts formula 2:

$$\begin{aligned} \langle u, v_{x_i x_i x_{i+1} x_{i+1}} \rangle &= \langle u, v_{x_i x_{i+1} x_{i+1} x_i} \rangle = u v_{x_i x_{i+1} x_{i+1}} \big|_{x_i=0,1} - \langle u_{x_i}, v_{x_i x_{i+1} x_{i+1}} \rangle \\ &= -\langle u_{x_i}, v_{x_i x_{i+1} x_{i+1}} \rangle = -u_{x_i} v_{x_i x_{i+1}} \big|_{x_{i+1}=0,1} + \langle u_{x_i x_{i+1}}, v_{x_i x_{i+1}} \rangle \\ &= \langle u_{x_i x_{i+1}}, v_{x_i x_{i+1}} \rangle, \end{aligned} \tag{4.89}$$

Integration-by-parts formula 3:

$$\begin{aligned} \langle u, v_{xxyyzz} \rangle &= \langle u, v_{xyzzyx} \rangle = u v_{xyzzy} \big|_{x=0,1} - \langle u_x, v_{xyzzy} \rangle = -\langle u_x, v_{xyzzy} \rangle \\ &= -u_x v_{xyzz} \big|_{y=0,1} + \langle u_{xy}, v_{xyzz} \rangle = \langle u_{xy}, v_{xyzz} \rangle \\ &= u_{xy} v_{xyz} \big|_{z=0,1} - \langle u_{xyz}, v_{xyz} \rangle = -\langle u_{xyz}, v_{xyz} \rangle. \end{aligned} \tag{4.90}$$

The modified Douglas scheme (4.83) multiplied by Δt and written using the u_{x_i} notation is

$$\begin{aligned} u^{n+1} - u^n =& \frac{\Delta t}{2\mathrm{Re}} \sum_{i=1}^{3} \left(u^{n+1}_{x_i x_i} + u^n_{x_i x_i} \right) - \frac{\Delta t^2}{4\mathrm{Re}^2} \sum_{i=1}^{3} \left(u^{n+1} - u^{n-1} \right)_{x_i x_i x_{i+1} x_{i+1}} \\ &+ \frac{\Delta t^3}{8\mathrm{Re}^3} \left[\left(u^{n+1} - u^n \right) - \left(u^{n-1} - u^{n-2} \right) \right]_{xxyyzz}. \end{aligned} \tag{4.91}$$

Applying the inner product $\langle u^{n+1} - u^n, (\cdot) \rangle$ to (4.91) yields

$$\|u^{n+1} - u^n\|^2 = \frac{\Delta t}{2\mathrm{Re}} \sum_{i=1}^{3} \langle u^{n+1} - u^n, \left(u^{n+1} + u^n \right)_{x_i x_i} \rangle \tag{4.92}$$

$$- \frac{\Delta t^2}{4\mathrm{Re}^2} \sum_{i=1}^{3} \langle u^{n+1} - u^n, \left(u^{n+1} - u^n + u^n - u^{n-1} \right)_{x_i x_i x_{i+1} x_{i+1}} \rangle \tag{4.93}$$

$$+ \frac{\Delta t^3}{8\mathrm{Re}^3} \langle u^{n+1} - u^n, \left[\left(u^{n+1} - u^n \right) - \left(u^{n-1} - u^{n-2} \right) \right]_{xxyyzz} \rangle. \tag{4.94}$$

Applying the integrations by parts formulas (4.88)–(4.90) to lines (4.92)–(4.94) respectively, we get

$$\begin{aligned}\|u^{n+1}-u^n\|^2 = &-\frac{\Delta t}{2\mathrm{Re}}\sum_{i=1}^{3}\langle u_{x_i}^{n+1}-u_{x_i}^n, u_{x_i}^{n+1}+u_{x_i}^n\rangle \\ &-\frac{\Delta t^2}{4\mathrm{Re}^2}\sum_{i=1}^{3}\langle u_{x_ix_{i+1}}^{n+1}-u_{x_ix_{i+1}}^n, \left(u_{x_ix_{i+1}}^{n+1}-u_{x_ix_{i+1}}^n\right)+\left(u_{x_ix_{i+1}}^n-u_{x_ix_{i+1}}^{n-1}\right)\rangle \\ &-\frac{\Delta t^3}{8\mathrm{Re}^3}\langle u_{xyz}^{n+1}-u_{xyz}^n, \left(u_{xyz}^{n+1}-u_{xyz}^n\right)-\left(u_{xyz}^{n-1}-u_{xyz}^{n-2}\right)\rangle. \end{aligned} \tag{4.95}$$

Then, using $\langle a-b, a+b\rangle = \|a\|^2-\|b\|^2$ and the identity (4.86), we have

$$\begin{aligned}\|u^{n+1}-u^n\|^2 = &-\frac{\Delta t}{2\mathrm{Re}}\sum_{i=1}^{3}\left(\|u_{x_i}^{n+1}\|^2-\|u_{x_i}^n\|^2\right) \\ &-\frac{\Delta t^2}{8\mathrm{Re}^2}\sum_{i=1}^{3}\left(\|u_{x_ix_{i+1}}^{n+1}-u_{x_ix_{i+1}}^n\|^2+\|u_{x_ix_{i+1}}^{n+1}-u_{x_ix_{i+1}}^{n-1}\|^2-\|u_{x_ix_{i+1}}^n-u_{x_ix_{i+1}}^{n-1}\|^2\right) \\ &-\frac{\Delta t^3}{16\mathrm{Re}^3}\left(\|u_{xyz}^{n+1}-u_{xyz}^n\|^2+\|\left(u_{xyz}^{n+1}-u_{xyz}^n\right)-\left(u_{xyz}^{n-1}-u_{xyz}^{n-2}\right)\|^2-\|u_{xyz}^{n-1}-u_{xyz}^{n-2}\|^2\right). \end{aligned} \tag{4.96}$$

If we sum equation (4.96) from $n=0,...,N$, several terms disappear in a telescoping sum and we get

$$\begin{aligned}&\sum_{n=0}^{N}\|u^{n+1}-u^n\|^2+\frac{\Delta t}{2\mathrm{Re}}\sum_{i=1}^{3}\|u_{x_i}^{N+1}\|^2 \\ &+\frac{\Delta t^2}{8\mathrm{Re}^2}\sum_{i=1}^{3}\left(\|u_{x_ix_{i+1}}^{N+1}-u_{x_ix_{i+1}}^N\|^2+\sum_{n=0}^{N}\|u_{x_ix_{i+1}}^{n+1}-u_{x_ix_{i+1}}^{n-1}\|^2\right) \\ &+\frac{\Delta t^3}{16\mathrm{Re}^3}\left(\|u_{xyz}^{N+1}-u_{xyz}^N\|^2+\|u_{xyz}^N-u_{xyz}^{N-1}\|^2+\sum_{n=0}^{N}\|\left(u_{xyz}^{n+1}-u_{xyz}^n\right)-\left(u_{xyz}^{n-1}-u_{xyz}^{n-2}\right)\|^2\right) \\ &=\frac{\Delta t}{2\mathrm{Re}}\sum_{i=1}^{3}\|u_{x_i}^0\|^2+\frac{\Delta t^2}{8\mathrm{Re}^2}\sum_{i=1}^{3}\|u_{x_ix_{i+1}}^0-u_{x_ix_{i+1}}^{-1}\|^2 \\ &+\frac{\Delta t^3}{16\mathrm{Re}^3}\left(\|u_{xyz}^0-u_{xyz}^{-1}\|^2+\|u_{xyz}^{-1}-u_{xyz}^{-2}\|^2\right). \end{aligned} \tag{4.97}$$

The stability estimate (4.97) bounds several norms of u on the left hand side by the initial conditions u^0, u^{-1}, u^{-2} on the right-hand side. Although (4.97) does not explicitly bound $\|u^{N+1}\|$, $\|u^{N+1}\|$ is still bounded by initial conditions because $\|u^{N+1}\|^2 - \|u^0\|^2 \lesssim \sum_{n=0}^{N} \|u^{n+1} - u^n\|^2$ for the same reason that $|a^{N+1}| - |a^0| \leq \sum_{n=0}^{N} |a^{n+1} - a^n|$ for a sequence of real numbers a^n. Also, the definition of stability requires only that for all time steps N, there exists a constant C_N (which can depend on N) such that $\|u^N\| \leq C_N \|u^0\|$. Since the modified Douglas scheme is both stable and consistent, the Lax equivalence theorem implies that it is also convergent.

Note: The original Douglas splitting (4.61) can be proven stable with the same assumptions as above using a similar proof. The only significant differences are that one applies the inner product $\langle u^{n+1}, (\cdot)\rangle$ instead of $\langle u^{n+1} - u^n, (\cdot)\rangle$, and as a result one also obtains an explicit bound on $\|u^{N+1}\|$.

4.2.4 BDF2 Splitting

It is straightforward to change the original Douglas splitting from Crank–Nicolson to some other linear multistep method. For example,

$$\left(1 - \frac{2\Delta t}{3\mathrm{Re}}\partial_{xx}\right)\left(1 - \frac{2\Delta t}{3\mathrm{Re}}\partial_{yy}\right)\left(1 - \frac{2\Delta t}{3\mathrm{Re}}\partial_{zz}\right)\left(\frac{\frac{3}{2}\mathbf{u}^{n+1} - \frac{4}{2}\mathbf{u}^n + \frac{1}{2}\mathbf{u}^{n-1}}{\Delta t}\right) = \frac{1}{\mathrm{Re}}\nabla^2\left(\frac{4}{3}\mathbf{u}^n - \frac{1}{3}\mathbf{u}^{n-1}\right) + \mathbf{f}^{n+1} \tag{4.98}$$

is an $O(\Delta t^2)$ splitting based on the BDF2 (second-order backward difference) time discretization. Just as the original Douglas scheme was modified in Section 4.2.3, so to could the BDF2 splitting be modified in the same way. However, we do not use this splitting because it requires an extra velocity time level to be stored and differentiated, which is a significant cost. If one is considering the BDF2 formulation, then one should also consider the "vectorial additive schemes" referred to in [4], since these schemes are similar in complexity to the BDF2 splitting and some of them have been proven to be stable in complex-shaped domains with non-commutative operators.

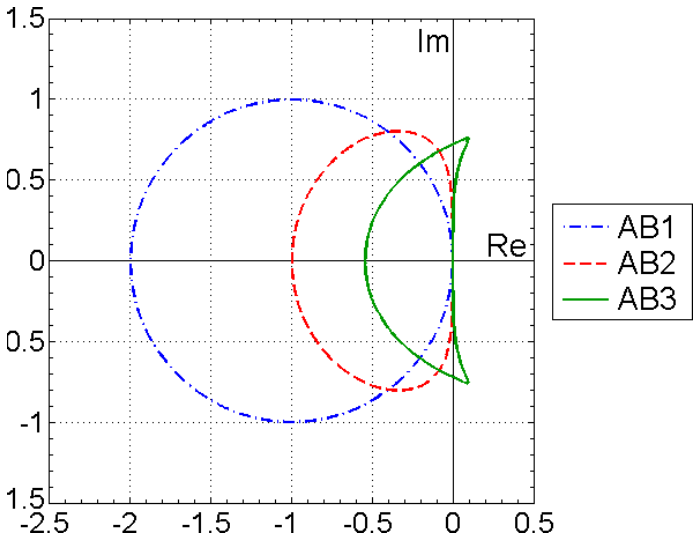

Figure 4.1: Stability regions in the complex plane for the 1st, 2nd, and 3rd order Adams–Bashforth methods when applied to the linear advection problem $u' = \lambda u$ for complex valued λ. The 1st order scheme has the largest stability region, $|1+\Delta t\lambda| \leq 1$. The 3rd order scheme has the smallest stability region. The maximum CFL number is approximately half the width of the stability region measured along the real axis.

4.3 Advection Terms

In this section, we consider time discretizations of the advection terms that appears in the Crank–Nicolson form of the Navier–Stokes equations (4.30). In particular, for time derivative $\frac{\mathbf{u}^{n+1}-\mathbf{u}^n}{\Delta t}$, we wish to approximate $\mathbf{u}^{n+\frac{1}{2}} \cdot \nabla \mathbf{u}^{n+\frac{1}{2}}$. Usually a fully explicit approximation based only on the velocities $\mathbf{u}^n, \mathbf{u}^{n-1}, \mathbf{u}^{n-2}$, etc. is used for the advection terms, as explained in the beginning of this chapter. Such methods are called Adams–Bashforth methods, which are a subset of linear multistep methods. Since the diffusion and pressure terms are only accurate to second order in time, it makes sense to use the second order Adams–Bashforth method, which uses the second order extrapolation

$$\mathbf{u}^{n+\frac{1}{2}} \cdot \nabla \mathbf{u}^{n+\frac{1}{2}} \approx \frac{3}{2}\mathbf{u}^n \cdot \nabla \mathbf{u}^n - \frac{1}{2}\mathbf{u}^{n-1} \cdot \nabla \mathbf{u}^{n-1}. \tag{4.99}$$

However, the first or third order Adams–Bashforth method could also be used. The first-, second-, and third-order methods impose the approximate CFL time step

restrictions $\Delta t \leq 1.0h/|\mathbf{u}|$, $\Delta t \leq 0.5h/|\mathbf{u}|$, and $\Delta t \leq 0.3h/|\mathbf{u}|$ respectively. This can be seen by examining the well known stability regions of the Adams–Bashforth methods for the linear advection equation in Figure 4.1. For a given velocity field $\mathbf{u} = (u, v, w)$, we will sometimes refer to an approximate "CFL number", which we define as

$$\text{CFL number} = \max_{\Omega_f} \left\{ u\frac{\Delta t}{h_x}, v\frac{\Delta t}{h_y}, w\frac{\Delta t}{h_z} \right\}. \tag{4.100}$$

For the second order Adams–Bashforth discretization, the scheme may become unstable if the CFL number rises significantly above 0.5.

Instead of a fully explicit scheme for the advection, a partially implicit discretization can be used, such as the second-order approximation

$$\mathbf{u}^{n+\frac{1}{2}} \cdot \nabla \mathbf{u}^{n+\frac{1}{2}} \approx \frac{1}{2}\mathbf{u}^{n} \cdot \nabla \mathbf{u}^{n+1} + \frac{1}{2}\mathbf{u}^{n+1} \cdot \nabla \mathbf{u}^{n}. \tag{4.101}$$

This can be proven to be second-order accurate in time by using a Taylor expansion about the $n + \frac{1}{2}$ time level:

$$\begin{aligned} \mathbf{u}^{n} \cdot \nabla \mathbf{u}^{n+1} &= \left(\mathbf{u}^{n+\frac{1}{2}} - \frac{\Delta t}{2}\mathbf{u}_t^{n+\frac{1}{2}} + O(\Delta t^2) \right) \cdot \nabla \left(\mathbf{u}^{n+\frac{1}{2}} + \frac{\Delta t}{2}\mathbf{u}_t^{n+\frac{1}{2}} + O(\Delta t^2) \right) \\ &= \mathbf{u}^{n+\frac{1}{2}} \cdot \nabla \mathbf{u}^{n+\frac{1}{2}} + \frac{\Delta t}{2}\mathbf{u}^{n+\frac{1}{2}} \cdot \nabla \mathbf{u}_t^{n+\frac{1}{2}} - \frac{\Delta t}{2}\mathbf{u}_t^{n+\frac{1}{2}} \cdot \nabla \mathbf{u}^{n+\frac{1}{2}} + O(\Delta t^2), \end{aligned} \tag{4.102}$$

$$\begin{aligned} \mathbf{u}^{n+1} \cdot \nabla \mathbf{u}^{n} &= \left(\mathbf{u}^{n+\frac{1}{2}} + \frac{\Delta t}{2}\mathbf{u}_t^{n+\frac{1}{2}} + O(\Delta t^2) \right) \cdot \nabla \left(\mathbf{u}^{n+\frac{1}{2}} - \frac{\Delta t}{2}\mathbf{u}_t^{n+\frac{1}{2}} + O(\Delta t^2) \right) \\ &= \mathbf{u}^{n+\frac{1}{2}} \cdot \nabla \mathbf{u}^{n+\frac{1}{2}} - \frac{\Delta t}{2}\mathbf{u}^{n+\frac{1}{2}} \cdot \nabla \mathbf{u}_t^{n+\frac{1}{2}} + \frac{\Delta t}{2}\mathbf{u}_t^{n+\frac{1}{2}} \cdot \nabla \mathbf{u}^{n+\frac{1}{2}} + O(\Delta t^2). \end{aligned} \tag{4.103}$$

We can now see that all terms of order Δt cancel out when taking the sum of (4.102) and (4.103). The partially implicit formulation (4.101) has two advantages. The first is that compared to (4.99), we can save memory by not needing to store $\mathbf{u}^{n-1}$. The second is that numerical testing indicates (4.101) maintains stability for CFL numbers several times larger than (4.99). Also, since the partially implicit terms are linear in $\mathbf{u}^{n+1}$, the operator for $\mathbf{u}^{n+1}$ can be inverted by solving a variable-coefficient tridiagonal linear system in each dimension. This can be incorporated naturally into our numerical scheme since we are already solving variable-coefficient tridiagonal

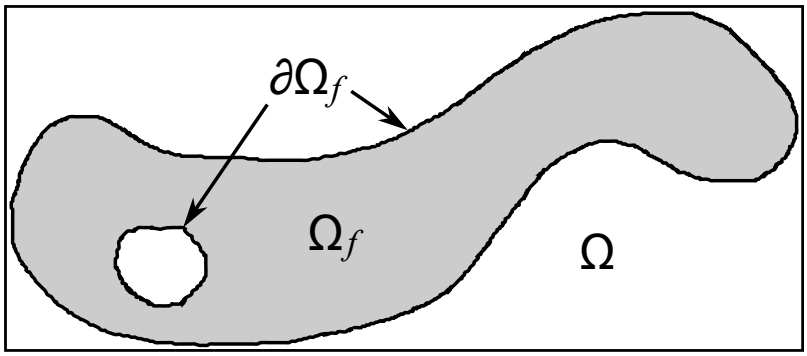

Figure 4.2: Complex-shaped fluid domain Ω_f (gray region) embedded in a simple box domain Ω (gray region $\cup$ white regions). The boundary $\partial\Omega_f$ is indicated.

linear systems for the boundary-fitted diffusion operators. Therefore, the implicit advection operator (4.101) should be preferred whenever boundary fitted diffusion operators are used. For all of the numerical results in this thesis, however, we use the second-order Adams–Bashforth scheme.

4.4 Complete Navier–Stokes Scheme

In this section, we combine the splitting methods discussed in this chapter to produce a complete scheme for solving the incompressible Navier–Stokes equations (2.22) in a domain of a complex time-dependent shape. For numerical calculation, we embed the complex-shaped domain Ω_f within a larger rectangular domain $\Omega = [0, X] \times [0, Y] \times [0, Z]$, as in Figure 4.2. We use the direction-factorized pressure "projection" scheme (4.53)–(4.55) together with the modified Douglas splitting (4.85), and we use the explicit second-order Adams–Bashforth time-discretization of the advection terms given by (4.99). The momentum equation is solved only in the fluid domain Ω_f using boundary conditions on $\partial\Omega_f$, but the pressure is extended and solved in the entire box domain Ω in order to guarantee that the properties (4.50) are satisfied. The scheme can use either the standard or boundary-fitted spatial operators in Chapter 3, and this makes the scheme either $O(h)$ or $O(h^2)$ accurate respectively. The complete scheme is given as follows:

Initial Conditions: Given an initial velocity field $\mathbf{u}_0$ and initial pressure field p_0, set $\mathbf{u}^{-1} = \mathbf{u}^0 = \mathbf{u}_0$, set $p^{-\frac{1}{2}} = p_0$, and set $\phi^{-\frac{1}{2}} = 0$. Then for all $n = 0..N$,
Pressure predictor: The pressure at time $n + \frac{1}{2}$ is predicted in Ω as

$$p^{*,n+\frac{1}{2}} = p^{n-\frac{1}{2}} + \phi^{n-\frac{1}{2}}. \tag{4.104}$$

Velocity update: Solve for $\mathbf{u}^{n+1}$ in Ω_f^{n+1} using

$$\begin{aligned}
\boldsymbol{\xi}^{n+1} = \mathbf{u}^n + \Delta t \Bigg[& -\nabla p^{*,n+\frac{1}{2}} - \left(\frac{3}{2}\mathbf{u}^n \cdot \nabla \mathbf{u}^n - \frac{1}{2}\mathbf{u}^{n-1} \cdot \nabla \mathbf{u}^{n-1} \right) \\
& + \frac{1}{\mathrm{Re}} \left(\partial_{xx}^n \boxed{\boldsymbol{\eta}^n} + \partial_{yy}^n \boxed{\boldsymbol{\zeta}^n} + \partial_{zz}^n \mathbf{u}^n \right) + \mathbf{f}^{n+\frac{1}{2}} \Bigg], \\
\left(1 - \frac{\Delta t}{2\mathrm{Re}} \partial_{xx}^{n+1} \right) \boldsymbol{\eta}^{n+1} &= \boldsymbol{\xi}^{n+1} - \frac{\Delta t}{2\mathrm{Re}} \partial_{xx}^n \boxed{\boldsymbol{\eta}^n}, \\
\left(1 - \frac{\Delta t}{2\mathrm{Re}} \partial_{yy}^{n+1} \right) \boldsymbol{\zeta}^{n+1} &= \boldsymbol{\eta}^{n+1} - \frac{\Delta t}{2\mathrm{Re}} \partial_{yy}^n \boxed{\boldsymbol{\zeta}^n}, \\
\left(1 - \frac{\Delta t}{2\mathrm{Re}} \partial_{zz}^{n+1} \right) \mathbf{u}^{n+1} &= \boldsymbol{\zeta}^{n+1} - \frac{\Delta t}{2\mathrm{Re}} \partial_{zz}^n \mathbf{u}^n,
\end{aligned} \tag{4.105}$$

where boundary conditions on $\partial\Omega_f^{n+1}$ are used for each one-dimensional problem.
Incompressibility penalty step: Solve for $\phi^{n+\frac{1}{2}}$ in Ω using

$$\begin{aligned}
(1 - \partial_{xx})\theta &= -\frac{1}{\Delta t} \nabla \cdot \mathbf{u}^{n+1}, \\
(1 - \partial_{yy})\psi &= \theta, \\
(1 - \partial_{zz})\phi^{n+\frac{1}{2}} &= \psi,
\end{aligned} \tag{4.106}$$

where Neumann/Dirichlet boundary conditions are used on $\partial\Omega$ whenever the velocity has Dirichlet/Neumann boundary conditions, respectively.
Pressure update: The pressure at time $n + \frac{1}{2}$ is corrected in Ω to be

$$p^{n+\frac{1}{2}} = p^{n-\frac{1}{2}} + \phi^{n+\frac{1}{2}} - \frac{\chi}{2\mathrm{Re}} \nabla \cdot (\mathbf{u}^{n+1} + \mathbf{u}^n). \tag{4.107}$$

The above scheme has the usual CFL time step restriction for the Navier–Stokes equations, and if $\chi = 0$ then numerical evidence suggests that the scheme is unconditionally stable for the Stokes equations.

Chapter 5

Convergence Rates Of The Numerical Schemes

In this chapter, we use the following discrete spatial norms based on the cell centers $\mathbf{x}_{i+\frac{1}{2},\ j+\frac{1}{2},\ k+\frac{1}{2}}$ for uniform MAC grids:

$$\|p\|_{L^\infty(\Omega)} = \max_{i,j,k} \left| p_{i+\frac{1}{2},\ j+\frac{1}{2},\ k+\frac{1}{2}} \right|, \qquad \|p\|_{L^2(\Omega)} = \sqrt{\sum_{i,j,k} V_{ijk} p^2_{i+\frac{1}{2},\ j+\frac{1}{2},\ k+\frac{1}{2}}}, \tag{5.1}$$

$$\|\mathbf{u}\|_{L^\infty(\Omega)} = \max_{i,j,k} \|\mathbf{u}_{i+\frac{1}{2},\ j+\frac{1}{2},\ k+\frac{1}{2}}\|, \qquad \|\mathbf{u}\|_{L^2(\Omega)} = \sqrt{\sum_{i,j,k} V_{ijk} \|\mathbf{u}_{i+\frac{1}{2},\ j+\frac{1}{2},\ k+\frac{1}{2}}\|^2}, \tag{5.2}$$

where V_{ijk} is the volume of the MAC cell with centroid $\mathbf{x}_{i+\frac{1}{2},\ j+\frac{1}{2},\ k+\frac{1}{2}}$ and $\|\cdot\|$ is a vector norm at the cell centroid. We define $\|\cdot\|$ by

$$\|\mathbf{u}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}\| = \sqrt{\left(\frac{|u_-| + |u_+|}{2}\right)^2 + \left(\frac{|v_-| + |v_+|}{2}\right)^2 + \left(\frac{|w_-| + |w_+|}{2}\right)^2}, \tag{5.3}$$

$$\begin{aligned} u_- &= u_{i,\ j+\frac{1}{2},\ k+\frac{1}{2}}, \quad & u_+ &= u_{i+1,\ j+\frac{1}{2},\ k+\frac{1}{2}}, \\ v_- &= v_{i+\frac{1}{2},\ j,\ k+\frac{1}{2}}, \quad & v_+ &= v_{i+\frac{1}{2},\ j+1,\ k+\frac{1}{2}}, \\ w_- &= w_{i+\frac{1}{2},\ j+\frac{1}{2},\ k}, \quad & w_+ &= w_{i+\frac{1}{2},\ j+\frac{1}{2},\ k+1}. \end{aligned} \tag{5.4}$$

The above norms are taken over the entire extended domain Ω, i.e., the set of discrete points over which the norm is computed is exactly the set of all MAC cell centers. We also define similar norms on a subset of these points:

Define $\|p\|_{L^\infty(\Omega_f)}$ and $\|p\|_{L^2(\Omega_f)}$ to be identical to (5.1) except that if a given MAC cell has *all* of its velocity nodes (four in 2D, six in 3D) outside Ω_f, then the point at the center of that MAC cell is excluded when calculating the norm.

Define $\|p\|_{L^\infty(\Omega_f \setminus \partial\Omega_f)}$ and $\|p\|_{L^2(\Omega_f \setminus \partial\Omega_f)}$ to be identical to (5.1) except that if a given MAC cell has *any* of its velocity nodes outside Ω_f, then the point at the center of that MAC cell is excluded when calculating the norm.

Note: Since the discrete divergence is stored at pressure points, it uses the same norm as the pressure. Also, any norm involving a pressure field is computed only after subtracting off the average value from each field involved.

While it could be useful to compute time-based norms analogous to the semi-discrete norms used at the end of Section 4.1.1, we do not do this. In particular, we simply compute the spatial error "instantaneously" at a particular time step rather than an average spatial error over a specified time interval.

5.1 Diffusion Equation

In this section, we present numerical results that show that the modified Douglas scheme (4.84) is second-order accurate in both time and space when solving the diffusion equation (heat equation),

$$\frac{\partial u}{\partial t} = \nabla^2 u + f, \tag{5.5}$$

in a complex-shaped domain where the discrete non-commutative boundary-fitted spatial operators (3.6) are used. All tests use a uniform MAC grid where h is the width of each MAC cell.

5.1.1 Spatial Convergence

To show spatial accuracy, we choose the domain Ω_f to be a cube $[0,1] \times [0,1] \times [0,1]$ with a spherical hole of radius 0.25 removed from the center of the cube. We use a manufactured solution,

$$\begin{aligned} u_{\text{exact}} &= \sin(x)\sin(y+z), \\ f &= 3\sin(x)\sin(y+z), \end{aligned} \tag{5.6}$$

which is an exact solution of the 3D diffusion equation (5.5). The numerical solution u is initialized to u_{exact} at $t = 0$, and u_{exact} is used as boundary conditions throughout the simulation. The solver is run with $\Delta t = 0.1$ for $t \in [0, 100]$, and the error at $t = 100$ using various grid resolutions are presented in Table 5.1. The results clearly indicate a second-order spatial convergence rate. The convergence rate between two grid resolutions h_1, h_2 with associated errors e_1, e_2 is defined as $\log(e_1/e_2)/\log(h_1/h_2)$, where $h_2 < h_1$.

h	$\|u - u_{\text{exact}}\|_{L^2(\Omega)}$	rate	$\|u - u_{\text{exact}}\|_{L^\infty(\Omega)}$	rate
1/10	3.172e-04		7.608e-04	
1/20	8.560e-05	1.89	2.113e-04	1.85
1/40	2.131e-05	2.01	5.547e-05	1.93
1/80	5.347e-06	1.99	1.420e-05	1.97
1/160	1.336e-06	2.00	3.714e-06	1.94

Table 5.1: Spatial convergence for the 3D diffusion equation.

Since the manufactured solution (5.6) is smooth across the domain boundary, we also check spatial accuracy on a problem whose solution is not smooth across the boundary of the domain. One such problem is the 2D Taylor–Couette laminar flow (See Appendix A). If we prescribe the pressure p to be identically zero everywhere for all time, then the Stokes equations (2.26) become a vectorial diffusion equation with analytical solution $\mathbf{u}_{\text{exact}}$ given in Appendix A, and the source term f in (5.5) is zero. We solve this problem with $R_1 = 0.25$, $R_2 = 0.5$, $\omega_1 = 1$, $\omega_2 = -1$, $\text{Re} = 1$, and both cylinders centered in the domain $[0,1] \times [0,1]$. Boundary fitted stencils are used to fit the fluid domain to the inner and outer cylinders. The numerical solution $\mathbf{u}$ is initialized to $\mathbf{u}_{\text{exact}}$ at $t = 0$, and the solver is run using $\Delta t = 0.1$ for $t \in [0, 100]$. The error at $t = 100$ using various grid resolutions are presented in Table 5.2, and the results clearly indicate second-order spatial convergence.

h	$\|\mathbf{u}-\mathbf{u}_{\text{exact}}\|_{L^2(\Omega)}$	rate	$\|\mathbf{u}-\mathbf{u}_{\text{exact}}\|_{L^\infty(\Omega)}$	rate
1/10	3.159e-03		7.624e-03	
1/20	1.020e-03	1.63	3.298e-03	1.21
1/40	2.556e-04	2.00	1.144e-03	1.53
1/80	6.312e-05	2.02	3.352e-04	1.77
1/160	1.559e-05	2.02	9.032e-05	1.89
1/320	3.847e-06	2.02	2.317e-05	1.96
1/640	9.545e-07	2.01	5.672e-06	2.03

Table 5.2: Spatial convergence for Taylor–Couette $p = 0$ diffusion equation.

5.1.2 Temporal Convergence

To show time accuracy, we solve the problem (5.5) in the same domain (cube with spherical hole) as in the previous section, and we use the manufactured solution $u_{\text{exact}} = \sin(x+t)\sin(y+z)$ with source term $f = [\cos(x+t) + 3\sin(x+t)]\sin(y+z)$. A uniform grid resolution $h = 1/128$ and various time step sizes are used to solve the problem for $t \in [0, 10]$. Table 5.3 shows the error at $t = 10$, and the results indicate at least second-order temporal convergence.

Δt	$\|u-u_{\text{exact}}\|_{L^2(\Omega)}$	rate	$\|u-u_{\text{exact}}\|_{L^\infty(\Omega)}$	rate
1/2	5.480e-02		1.293e-01	
1/4	1.879e-02	1.54	5.609e-02	1.20
1/8	4.006e-03	2.23	1.345e-02	2.06
1/16	9.047e-04	2.15	3.396e-03	1.99
1/32	2.135e-04	2.08	8.383e-04	2.02
1/64	4.998e-05	2.10	1.953e-04	2.10
1/128	1.018e-05	2.30	4.289e-05	2.19
1/256	1.812e-06	2.49	7.539e-06	2.51

Table 5.3: Temporal convergence for the 3D diffusion equation.

One property of the modified Douglas scheme is a time step scale oscillatory behaviour of very small magnitude. Table 5.4 shows how the error typically changes each time step when using the modified Douglas and original Douglas schemes to resolve the manufactured solution (5.6) without boundary fitted operators. Though

the actual error is the same for both schemes, the oscillatory behaviour of the modified Douglas scheme significantly increases the time taken to reach a strict numerical steady state. However, the oscillatory behaviour goes away if one uses a sufficiently small time step, and then the modified Douglas scheme reaches a steady state at the exact same rate as the original Douglas scheme.

	Modified Douglas		Original Douglas	
t	$\|u - u_{\text{exact}}\|_{L^2(\Omega)}$	change	$\|u - u_{\text{exact}}\|_{L^2(\Omega)}$	change
1.0	2.94948e-06	-7.6362e-07	3.33613e-06	7.1455e-09
1.1	3.69249e-06	7.4301e-07	3.34202e-06	5.8883e-09
1.2	3.00163e-06	-6.9086e-07	3.34697e-06	4.9485e-09
1.3	3.67540e-06	6.7376e-07	3.35120e-06	4.2263e-09
...	...		...	
10.0	3.28438e-06	-2.4287e-07	3.39872e-06	7.7024e-11
10.1	3.52611e-06	2.4172e-07	3.39879e-06	7.5509e-11
10.2	3.28574e-06	-2.4036e-07	3.39887e-06	7.4038e-11
10.3	3.52499e-06	2.3924e-07	3.39894e-06	7.2609e-11
...	...		...	
100.0	3.37154e-06	-7.2026e-08	3.40537e-06	6.1098e-13
100.1	3.44352e-06	7.1982e-08	3.40537e-06	6.0961e-13
100.2	3.37158e-06	-7.1943e-08	3.40537e-06	6.0825e-13
100.3	3.44348e-06	7.1899e-08	3.40537e-06	6.0689e-13

Table 5.4: Evolution of the error using $\Delta t = 0.1$ for the modified Douglas and original Douglas schemes. The oscillatory behaviour of the modified Douglas scheme still exists if one uses $\Delta t = 0.001$ but dissapears for $\Delta t \leq 0.0001$. The actual error for both schemes is the same.

5.1.3 Stability

By extensive numerical testing, the modified Douglas scheme (4.84) seems to be unconditionally stable in both 2D and 3D when solving the diffusion equation (5.5) when using boundary-fitted operators (3.6) in a domain of any shape. However, there are situations where the original Douglas scheme (4.63) is unstable even though the modified Douglas scheme is stable. For example, consider (5.5), (5.6) in the domain with spherical hole as above. If we choose $h = 1/162$, $\Delta t = 0.1$, then the

original Douglas scheme blows up exponentially (error magnified by 1.04 per time step starting after only 15 time steps) but the modified Douglas scheme shows no signs of instability even after thousands of time steps and even when using a wide variety of time steps. It is not completely clear what causes the instability of the original Douglas scheme since using $\Delta t = 0.01$ or $\Delta t = 10$ both stabilize the scheme but using $\Delta t = 1$ still magnifies the error by 1.01 every time step. Also, changing the grid resolution to $h = 1/161$ stabilizes the original Douglas scheme even though the minimum stencil γ in (3.6) is 10^{-5} for $h = 1/161$ and 2×10^{-5} for $h = 1/162$. The instabilities that we found suggest that the commutativity of spatial operators is necessary for the original Douglas scheme to be unconditionally stable in 3D when solving the diffusion equation. No numerical evidence was found that the original Douglas scheme is unstable in 2D, even when using stencils with $\gamma \sim 10^{-10}$. This makes sense because the original Douglas scheme has been proven to be stable in 2D even with non-commutative operators (see Section 4.2.2). Also, no instability was found for the original Douglas splitting when solving the diffusion equation in complex-shaped domains that align perfectly with the MAC cell walls. In other words, it appears from numerical evidence that the original Douglas scheme is stable unless boundary-fitted spatial operators are used.

5.2 Unsteady Stokes Equations

5.2.1 Spatial Convergence

In this section, we use the scheme of Section 4.4 (with $\chi = 0$) to solve the unsteady Stokes equations (2.26) for the same complex-shaped domain in Section 5.1.1 (cube with spherical hole in the middle). We use the boundary fitted spatial operators (3.6), but we use the standard divergence stencil (3.16).

To show spatial accuracy, we use the 3D manufactured solution

$$\begin{aligned} u &= \sin(x)\sin(y+z), & f_u &= \cos(x+y+z) + 3\sin(x)\sin(y+z) \\ v &= -\cos(x)\cos(y+z), & f_v &= \cos(x+y+z) - 3\cos(x)\cos(y+z) \\ w &= 2\cos(x)\cos(y+z), & f_w &= \cos(x+y+z) + 6\cos(x)\cos(y+z) \\ p &= \sin(x+y+z), \end{aligned} \tag{5.7}$$

which is an exact solution of the Stokes equations (2.26) with right-hand-side source

term $\mathbf{f} = (f_u,\, f_v,\, f_w)$. We use time step $\Delta t = 0.001$ and run simulations for $t \in [0, 1]$. The error at $t = 1$ for the velocity and pressure are presented in Table 5.5. Similar convergence results using a larger time step are shown in Tables 6,7 of [4]. The spatial accuracy of the velocity and pressure are clearly second-order accurate.

h	$\|\mathbf{u} - \mathbf{u}_{\text{exact}}\|_{L^2(\Omega)}$	rate	$\|\mathbf{u} - \mathbf{u}_{\text{exact}}\|_{L^\infty(\Omega)}$	rate
1/8	7.817e-04		2.419e-03	
1/16	2.362e-04	1.73	8.148e-04	1.57
1/32	6.347e-05	1.90	2.275e-04	1.84
1/64	1.620e-05	1.97	5.997e-05	1.92
1/128	4.080e-06	1.99	1.581e-05	1.92

h	$\|\nabla \cdot \mathbf{u}\|_{L^2(\Omega)}$	rate	$\|\nabla \cdot \mathbf{u}\|_{L^\infty(\Omega)}$	rate
1/8	6.925e-04		3.510e-03	
1/16	7.341e-04	-0.08	7.338e-03	-1.06
1/32	3.466e-04	1.08	5.098e-03	0.53
1/64	1.281e-04	1.44	3.002e-03	0.76
1/128	4.644e-05	1.46	1.507e-03	0.99

h	$\|p - p_{\text{exact}}\|_{L^2(\Omega_f)}$	rate	$\|p - p_{\text{exact}}\|_{L^\infty(\Omega_f)}$	rate
1/8	4.291e-02		1.415e-01	
1/16	1.303e-02	1.72	6.929e-02	1.03
1/32	3.541e-03	1.88	2.866e-02	1.27
1/64	9.126e-04	1.96	9.080e-03	1.66
1/128	2.294e-04	1.99	2.486e-03	1.87

Table 5.5: Spatial convergence for manufactured solution of 3D Stokes equations.

Again, we also test spatial convergence on a non-smooth field using the Taylor–Couette solution to the unsteady Stokes equations given in Appendix A. We use $\text{Re} = 1$, $R_1 = 0.25$, $R_2 = 0.5$, $\omega_1 = 1$, $\omega_2 = -1$, and both cylinders are centered in the domain $[0, 1] \times [0, 1]$. We use the boundary fitted spatial operators (3.6), but we use the standard divergence stencil (3.16). Using initial conditions equal to the exact solution, we use the scheme of Section 4.4 (with $\chi = 0$) to run simulations for $t \in [0, 1]$ using $\Delta t = 0.001$. Table 5.6 shows the spatial error for the velocity and pressure at $t = 1$. The velocity shows second-order spatial convergence, and the pressure seems to converge at an even higher rate, probably because the exact

solution of the pressure is $p = 0$, which is not spatially dependent. The divergence in Ω does not converge in the maximum norm because the standard divergence stencil (3.16) is $O(1)$ spatially accurate on $\partial\Omega_f$ due to the jump discontinuity of the velocity derivative there (see Appendix A). However, the divergence in Ω does converge in the L^2 norm because the set of MAC cells which are intersected by $\partial\Omega_f$ is small compared to all of the cells in the domain. Also, we can see that the divergence converges in the maximum norm in the domain $\Omega_f \setminus \partial\Omega_f$.

The abnormal error values for the $h = 1/10$ grid in Table 5.6 occur because the smoothing behaviour of the pressure "projection" step (4.106) with $\chi = 0$ is reduced when $\Delta t \ll h$ (or for any Δt if $\chi > 0$). In this case, the discrete divergence is forced to zero on a cell-by-cell basis by the incremental scheme (Section 4.1.2), and any error in the divergence is transferred directly to the pressure. This raises the question of whether it is even meaningful to talk about the divergence on time-independent flows, since for $\Delta t \ll h$ the divergence approaches zero to floating point precision. However, for $\chi = 0$ and $\Delta t \sim h$, the divergence does not approach zero to floating point precision even after 100 million time steps (this is true when using the original Douglas splitting as well). Therefore, it is still meaningful to talk about the divergence on steady flows.

Table 5.7 shows the result of running the identical simulation as in Table 5.6 but using $\Delta t \ll h$. In this case, the discrete divergence approaches floating point zero, but the pressure no longer converges in the maximum norm. Also, because of the $O(1)$ pressure error on $\partial\Omega_f$, the velocity convergence rate is reduced to about $O(h^{\frac{3}{4}})$ in the maximum norm and $O(h^{\frac{5}{4}})$ in the L^2 norm (higher resolution tests also agree). This can be fixed by using a boundary-fitted divergence operator such as the one described in Sections 3.4.1 or 3.4.2. Table 5.8 shows the same simulation as that of Table 5.7, but using the boundary fitted divergence of Section 3.4.1 instead of the standard divergence stencil (3.16). We can see that the error for all variables in all norms decreases significantly, and the velocity is restored to $O(h^2)$ accuracy (also confirmed by higher resolution tests).

5.2.2 Temporal Convergence

To show temporal accuracy of the scheme of Section 4.4 (with $\chi = 0$), we use the 3D manufactured solution

$$\begin{aligned}
u &= \sin(x+t)\sin(y+z), \\
v &= -\cos(x+t)\cos(y+z), \\
w &= 2\cos(x+t)\cos(y+z), \\
p &= \sin[(x+y+z)\cos(t)], \\
f_u &= \sin(y+z)[\cos(x+t) + 3\sin(x+t)] + \cos[(x+y+z)\cos(t)]\cos(t), \\
f_v &= \cos(y+z)[\sin(x+t) - 3\cos(x+t)] + \cos[(x+y+z)\cos(t)]\cos(t), \\
f_w &= \cos(y+z)[-2\sin(x+t) + 6\cos(x+t)] + \cos[(x+y+z)\cos(t)]\cos(t),
\end{aligned} \tag{5.8}$$

which is an exact solution of the unsteady Stokes equations (2.26) with right-hand-side source term $\mathbf{f} = (f_u,\, f_v,\, f_w)$. The domain is again the cube with spherical hole as in Section 5.1.1, and we use the boundary fitted spatial operators (3.6) together with the standard divergence stencil (3.16). We use grid resolution $h = 1/128$ and run simulations for $t \in [0, 1]$ with various time step sizes. The error at $t = 1$ for the velocity and pressure are presented in Table 5.9. The temporal convergence of the velocity is approximately 1.8 order and the pressure is approximately 1.5 order. Similar convergence results are shown in Tables 8,9 of [4], though there the manufactured pressure solution is time independent.

5.2.3 Stability

The stability of the scheme of Section 4.4 (using $\chi = 0$, boundary fitted diffusion operators (3.6), and the standard divergence stencil (3.16)) was tested extensively on both 2D and 3D solutions in various complex-shaped domains with time steps ranging from $\Delta t = 10^{-8}$ to $\Delta t = 10^{6}$, grids ranging from 3×3 to 1000×1000 in 2D and $3\times3\times3$ to $200\times200\times200$ in 3D, and Reynolds numbers from $\mathrm{Re} = 10^{-6}$ to $\mathrm{Re} = 10^{8}$. Even after many thousands of time steps, the scheme always remained stable. This suggests that the scheme of Section 4.4 (using $\chi = 0$) is unconditionally stable for the unsteady Stokes equations in 2D and 3D in any shape of domain. However, when resolving time-dependent flows where $\frac{\Delta t}{\mathrm{Re}}$ is large, the accuracy is certainly questionable since the error terms of (4.83) could be dominant. Nonetheless, the

scheme is still stable. For example, using a ridiculous $\mathrm{Re} = 10^{-12}$ together with $\Delta t = 10^{-6}$ is stable even after 100 million time steps, however the pressure error is very large.

5.2.4 Comparing Values Of Rotational Parameter χ

In this section, we investigate using different values of the rotational parameter χ as explained in Section 4.1.3. With the combination of MAC spatial discretization, Crank–Nicolson formulation, solving for the pressure in the entire extended domain Ω, and discrete divergence that we use in this thesis, the scheme of Section 4.4 is only conditionally stable for $\chi > 0$ in both 2D and 3D. Extensive numerical testing indicates that for the Stokes equations in 2D or 3D, the scheme is stable whenever

$$\Delta t \leq C \frac{\mathrm{Re}}{\chi} h^2, \tag{5.9}$$

where C is a constant that depends on the shape of the domain but does not depend on the size of the domain nor the velocity or pressure fields (magnitude, time-dependence, spatial derivatives, etc.). Violating (5.9) produces an instability that begins with the pressure (sometimes a "checkerboard" and sometimes a domain-scale oscillation). The restriction (5.9) is true regardless of the shape of the domain, regardless of whether boundary-fitted spatial operators are used or not, and regardless of whether the original Douglas (4.63) or modified Douglas (4.84) splitting is used. The only exceptions are that if Ω_f is a 2D box domain then the scheme appears stable for any Δt, and if Ω_f is a 3D box domain and the original Douglas splitting is used, then again the scheme is stable for any Δt. One interesting example is that for $\chi = 1$, $\Delta t = 10$, $h = 1/100$, the scheme is unstable on the Taylor–Couette geometry even when using the original Douglas splitting without any boundary-fitted spatial operators. This is interesting because the original Douglas splitting is proven (see Section 4.2.2) to be unconditionally stable in any shaped domain for the diffusion equation. This suggests that modifying the relationship between the discrete divergence and pressure could stabilize the scheme for any Δt and $\chi > 0$.

Typical values of the constant C in (5.9) are $C = 3000$ in 2D and $C = 2$ in 3D. Changing the domain shape, spatial operators, or momentum splitting method improves the constant by no more than a factor of 10 or so. For complex-shaped domains, the discrete domain shape depends on h and thus the constant C changes

with h, and the h^2 dependence in (5.9) is not clear until the domain shape is well resolved. While a time step restriction like $\Delta t \leq h^2$ is usually very bad, (5.9) is not so bad because one can always choose χ such that the scheme is stable for any time step size. In particular, choosing $\chi = 0$ is always stable.

Despite the stability restriction (5.9), it is still worthwhile to choose the largest χ that is stable. This is because the error for the pressure and divergence of the rotational incremental scheme (4.27) is $O(\Delta t^{\frac{3}{2}})$, but the error for the standard incremental scheme (4.16) is $O(\Delta t)$. The rotational scheme also performs better in practice, which can be demonstrated via an example where we solve the 3D Stokes equations for time-dependent manufactured solution (5.8) adjusted to $\mathrm{Re} = 100$. We use a complex-shaped fluid domain involving a cube, a pipe, and 3 holes. Specifically, Ω_f is defined as

$$\begin{aligned}&([0,1]\times[0,1]\times[0,1]) \;\cap\; \left(\|(x,y)-(\tfrac{1}{2},\tfrac{1}{2})\| \leq \tfrac{11}{20}\right) \;\cap\; \left(\|(x,y,z)-(\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2})\| \geq \tfrac{1}{5}\right)\\ &\cap\; \left(\|(x,y,z)-(\tfrac{1}{4},\tfrac{1}{4},\tfrac{11}{20})\| \geq \tfrac{1}{10}\right) \;\cap\; \left(\|(x,y,z)-(\tfrac{1}{2},\tfrac{17}{20},\tfrac{1}{4})\| \geq \tfrac{1}{10}\right).\end{aligned} \tag{5.10}$$

The boundary fitted diffusion operators (3.6) and boundary fitted divergence operators of Section (3.4.1) are used. Ω_f is embedded in the unit cube Ω and a uniform spatial grid with $h = 1/64$ is used. Time-representative errors at $t = 1$ when using various values of χ are presented in Table 5.10. Also in the table are results for the same situations but in a simple unit box domain using two different time steps. The table shows that choosing a larger χ reduces the error in situations where reducing the time step also reduces the error, i.e., if the spatial error is not dominant.

h	$\|\mathbf{u}-\mathbf{u}_{\text{exact}}\|_{L^2(\Omega)}$	rate	$\|\mathbf{u}-\mathbf{u}_{\text{exact}}\|_{L^\infty(\Omega)}$	rate
1/10	9.419e-03		2.137e-02	
1/20	2.005e-03	2.23	5.664e-03	1.92
1/40	4.200e-04	2.25	1.645e-03	1.78
1/80	8.841e-05	2.25	4.202e-04	1.97
1/160	2.222e-05	1.99	9.699e-05	2.12
1/320	5.031e-06	2.14	2.281e-05	2.09
1/640	1.166e-06	2.11	5.713e-06	2.00

h	$\|\nabla\cdot\mathbf{u}\|_{L^2(\Omega)}$	rate	$\|\nabla\cdot\mathbf{u}\|_{L^\infty(\Omega)}$	rate
1/10	2.782e-04		6.066e-04	
1/20	6.067e-02	-7.77	2.701e-01	-8.80
1/40	4.911e-02	0.31	4.292e-01	-0.67
1/80	2.306e-02	1.09	3.116e-01	0.46
1/160	2.233e-02	0.05	5.016e-01	-0.69
1/320	1.579e-02	0.50	4.423e-01	0.18
1/640	1.093e-02	0.53	5.524e-01	-0.32

h	$\|\nabla\cdot\mathbf{u}\|_{L^2(\Omega_f\setminus\partial\Omega_f)}$	rate	$\|\nabla\cdot\mathbf{u}\|_{L^\infty(\Omega_f\setminus\partial\Omega_f)}$	rate
1/10	3.665e-04		6.066e-04	
1/20	4.970e-02	-7.08	1.725e-01	-8.15
1/40	2.293e-02	1.12	9.670e-02	0.83
1/80	6.540e-03	1.81	2.380e-02	2.02
1/160	2.640e-03	1.31	7.872e-03	1.60
1/320	5.946e-04	2.15	2.146e-03	1.88
1/640	1.175e-04	2.34	1.284e-03	0.74

h	$\|p-p_{\text{exact}}\|_{L^2(\Omega_f)}$	rate	$\|p-p_{\text{exact}}\|_{L^\infty(\Omega_f)}$	rate
1/10	5.601e-01		1.850e+00	
1/20	2.040e-01	1.46	1.128e+00	0.71
1/40	3.560e-02	2.52	2.426e-01	2.22
1/80	6.733e-03	2.40	4.497e-02	2.43
1/160	2.269e-03	1.57	1.103e-02	2.03
1/320	4.841e-04	2.23	2.513e-03	2.13
1/640	9.233e-05	2.39	5.405e-04	2.22

Table 5.6: Spatial convergence when solving the Stokes equations for the Taylor–Couette flow. The standard divergence stencil and $\Delta t = 0.001$ are used. This table shows second-order spatial accuracy of the velocity and pressure.

h	$\|\mathbf{u}-\mathbf{u}_{\text{exact}}\|_{L^2(\Omega)}$	rate	$\|\mathbf{u}-\mathbf{u}_{\text{exact}}\|_{L^\infty(\Omega)}$	rate	$\|\nabla\cdot\mathbf{u}\|_{L^\infty(\Omega)}$
1/10	9.425e-03		2.138e-02		1.998e-15
1/20	3.542e-03	1.41	1.372e-02	0.64	6.106e-15
1/40	1.201e-03	1.56	8.024e-03	0.77	3.053e-14
1/80	2.713e-04	2.15	2.831e-03	1.50	9.104e-14
1/160	1.394e-04	0.96	1.759e-03	0.69	2.741e-05

h	$\|p-p_{\text{exact}}\|_{L^2(\Omega_f)}$	rate	$\|p-p_{\text{exact}}\|_{L^\infty(\Omega_f)}$	rate
1/10	5.605e-01		1.852e+00	
1/20	4.929e-01	0.19	3.170e+00	-0.78
1/40	3.699e-01	0.41	3.676e+00	-0.21
1/80	1.449e-01	1.35	2.541e+00	0.53
1/160	1.819e-01	-0.33	5.254e+00	-1.05

Table 5.7: Spatial convergence when solving the Stokes equations for the Taylor–Couette flow. The standard divergence stencil and $\Delta t = 0.000001$ are used, showing that the small time step produces zero divergence to floating point precision, but the $O(1)$ error in the divergence at the boundary Ω_f is transferred to the pressure.

h	$\|\mathbf{u}-\mathbf{u}_{\text{exact}}\|_{L^2(\Omega)}$	rate	$\|\mathbf{u}-\mathbf{u}_{\text{exact}}\|_{L^\infty(\Omega)}$	rate	$\|\nabla\cdot\mathbf{u}\|_{L^\infty(\Omega)}$
1/10	2.874e-03		6.003e-03		1.332e-15
1/20	1.546e-03	0.90	5.128e-03	0.23	3.220e-15
1/40	3.534e-04	2.13	1.088e-03	2.24	1.132e-14
1/80	1.139e-04	1.63	3.820e-04	1.51	6.206e-14
1/160	2.081e-05	2.45	9.997e-05	1.93	2.428e-13

h	$\|p-p_{\text{exact}}\|_{L^2(\Omega_f)}$	rate	$\|p-p_{\text{exact}}\|_{L^\infty(\Omega_f)}$	rate
1/10	1.219e-01		4.018e-01	
1/20	6.141e-02	0.99	2.630e-01	0.61
1/40	2.113e-02	1.54	1.379e-01	0.93
1/80	8.556e-03	1.30	1.265e-01	0.12
1/160	2.364e-03	1.86	5.678e-02	1.16

Table 5.8: Spatial convergence when solving the Stokes equations for the Taylor–Couette flow using $\Delta t = 0.000001$ and the boundary-fitted divergence of Section 3.4.1. The accuracy is improved compared to when non-boundary-fitted divergence was used in Table 5.7. Using the divergence of Section 3.4.2 produces similar results.

Δt	$\|\mathbf{u}-\mathbf{u}_{\text{exact}}\|_{L^2(\Omega)}$	rate	$\|\mathbf{u}-\mathbf{u}_{\text{exact}}\|_{L^\infty(\Omega)}$	rate
1/4	3.166e-02		8.403e-02	
1/8	1.008e-02	1.65	2.812e-02	1.58
1/16	4.610e-03	1.13	1.379e-02	1.03
1/32	2.661e-03	0.79	7.025e-03	0.97
1/64	1.079e-03	1.30	2.328e-03	1.59
1/128	3.243e-04	1.73	6.526e-04	1.84
1/256	9.504e-05	1.77	2.346e-04	1.48
1/512	2.666e-05	1.83	9.349e-05	1.33
1/1024	7.947e-06	1.75	4.034e-05	1.21

Δt	$\|\nabla\cdot\mathbf{u}\|_{L^2(\Omega)}$	rate	$\|\nabla\cdot\mathbf{u}\|_{L^\infty(\Omega)}$	rate
1/4	1.039e-01		5.663e-01	
1/8	4.425e-02	1.23	3.273e-01	0.79
1/16	2.607e-02	0.76	1.641e-01	1.00
1/32	1.531e-02	0.77	8.373e-02	0.97
1/64	7.272e-03	1.07	3.245e-02	1.37
1/128	2.912e-03	1.32	1.726e-02	0.91
1/256	1.135e-03	1.36	9.356e-03	0.88
1/512	4.335e-04	1.39	4.556e-03	1.04
1/1024	1.614e-04	1.43	1.965e-03	1.21

Δt	$\|p-p_{\text{exact}}\|_{L^2(\Omega_f)}$	rate	$\|p-p_{\text{exact}}\|_{L^\infty(\Omega_f)}$	rate
1/4	1.812e-001		8.505e-001	
1/8	1.760e-001	0.04	8.437e-001	0.01
1/16	1.225e-001	0.52	6.476e-001	0.38
1/32	6.907e-002	0.83	3.812e-001	0.76
1/64	2.781e-002	1.31	1.769e-001	1.11
1/128	8.715e-003	1.67	6.273e-002	1.50
1/256	2.912e-003	1.58	2.193e-002	1.52
1/512	1.019e-003	1.52	1.186e-002	0.89
1/1024	3.993e-004	1.35	7.012e-003	0.76

Table 5.9: Temporal convergence when solving the 3D Stokes equations ($\chi = 0$) using a manufactured solution in a complex-shaped domain. For $\Delta t \leq 1/1024$, spatial error and possibly $\Delta t \ll h$ "standard divergence stencil" error shows up. Otherwise, this table shows that the velocity convergence rate is approximately $O(\Delta t^{1.8})$ and the pressure convergence rate is approximately $O(\Delta t^{1.5})$. The theoretical estimates (4.16) are $O(\Delta t^2)$ and $O(\Delta t)$ for the velocity and pressure.

Unit Box Domain $\Delta t = 0.01$:

χ	$\|\mathbf{u} - \mathbf{u}_{\text{exact}}\|_{L^2(\Omega)}$	$\|\mathbf{u} - \mathbf{u}_{\text{exact}}\|_{L^\infty(\Omega)}$	$\|p - p_{\text{exact}}\|_{L^2(\Omega_f)}$	$\|p - p_{\text{exact}}\|_{L^\infty(\Omega_f)}$
0.0	8.58e-04	4.68e-03	3.76e-04	6.30e-03
0.1	6.93e-04	2.75e-03	2.92e-04	3.68e-03
1.0	5.28e-04	2.23e-03	1.90e-04	5.91e-03

χ	$\|\nabla \cdot \mathbf{u}\|_{L^2(\Omega)}$	$\|\nabla \cdot \mathbf{u}\|_{L^\infty(\Omega)}$
0.0	1.53e-02	1.17e-01
0.1	8.92e-03	9.18e-02
1.0	4.27e-03	7.54e-02

Unit Box Domain $\Delta t = 0.001$:

χ	$\|\mathbf{u} - \mathbf{u}_{\text{exact}}\|_{L^2(\Omega)}$	$\|\mathbf{u} - \mathbf{u}_{\text{exact}}\|_{L^\infty(\Omega)}$	$\|p - p_{\text{exact}}\|_{L^2(\Omega_f)}$	$\|p - p_{\text{exact}}\|_{L^\infty(\Omega_f)}$
0.0	1.53e-05	2.49e-04	1.10e-05	7.86e-04
0.1	1.39e-05	6.40e-05	6.77e-06	2.65e-04
1.0	1.34e-05	3.85e-05	5.82e-06	1.94e-04

χ	$\|\nabla \cdot \mathbf{u}\|_{L^2(\Omega)}$	$\|\nabla \cdot \mathbf{u}\|_{L^\infty(\Omega)}$
0.0	3.55e-04	1.59e-02
0.1	1.29e-04	3.10e-03
1.0	7.55e-05	1.94e-03

Complex Domain (5.10) $\Delta t = 0.01$:

χ	$\|\mathbf{u} - \mathbf{u}_{\text{exact}}\|_{L^2(\Omega)}$	$\|\mathbf{u} - \mathbf{u}_{\text{exact}}\|_{L^\infty(\Omega)}$	$\|p - p_{\text{exact}}\|_{L^2(\Omega_f)}$	$\|p - p_{\text{exact}}\|_{L^\infty(\Omega_f)}$
0.0	9.03e-04	4.09e-03	6.89e-04	1.06e-02
0.1	7.01e-04	2.65e-03	6.10e-04	8.86e-03
1.0	5.38e-04	1.92e-03	5.64e-04	8.20e-03

χ	$\|\nabla \cdot \mathbf{u}\|_{L^2(\Omega)}$	$\|\nabla \cdot \mathbf{u}\|_{L^\infty(\Omega)}$	$\|\nabla \cdot \mathbf{u}\|_{L^2(\Omega_f \setminus \partial\Omega_f)}$	$\|\nabla \cdot \mathbf{u}\|_{L^\infty(\Omega_f \setminus \partial\Omega_f)}$
0.0	2.04e-02	1.78e-01	2.05e-02	1.78e-01
0.1	9.89e-03	1.09e-01	9.98e-03	1.09e-01
1.0	4.37e-03	6.16e-02	4.54e-03	5.88e-02

Table 5.10: This table shows the error when solving the 3D Stokes equations with different values of the rotational parameter χ. Choosing a larger χ always reduces the error unless the time step is small and the error is mostly spatial. In this table, the accuracy improvement between $\chi = 0$ and $\chi = 0.1$ is almost the same as the accuracy improvement between $\chi = 0.1$ and $\chi = 1$. Thus, even a small nonzero χ is worthwhile for improving the accuracy of the numerical scheme.

5.3 Navier–Stokes Equations

In this section, we solve the full Navier–Stokes equations (2.22) using the scheme of Section 4.4. We choose $\chi = 1$ and use the boundary fitted diffusion (3.6) and divergence (Section 3.4.1) operators.

We start by demonstrating that even for Reynolds number 1000, the advection operator does not need to be boundary fitted to produce an $O(h^2)$ accurate velocity in complex-shaped domains. We consider again the Taylor–Couette flow $R_1 = 0.25$, $R_2 = 0.5$, $\omega_1 = 1$, $\omega_2 = -1$ as in Appendix A. The time step $\Delta t = 0.001$ is used and each simulation is run from $t = 0$ until a numerical steady state is reached, which is on the order of $t = 100$. Table 5.11 shows that the velocity convergence rate is clearly $O(h^2)$ even in the maximum norm.

Next we show that if the domain shape Ω_f is time-dependent, the scheme of Section 4.4 (again with $\chi = 1$ and boundary fitted spatial operators) solves the 3D Navier–Stokes equations with second-order accuracy in both time and space. We use the manufactured Stokes solution (5.8) adjusted for $\mathrm{Re} = 100$, and we add an additional right-hand-side source term

$$\begin{aligned} g_u &= \sin(x+t)\cos(x+t), \\ g_v &= \sin(y+z)\cos(y+z), \\ g_w &= -2\sin(y+z)\cos(y+z) \end{aligned} \tag{5.11}$$

in order to obtain an exact solution for the Navier–Stokes equations. We use a domain Ω_f which is defined as the unit cube with a *moving* spherical hole,

$$\Omega_f = [0,1]^3 \cap \left(\|(x,y,z) - (\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2}) - \tfrac{1}{5}(\sin(t),\cos(t),\sin(t))\| \geq \tfrac{1}{4} \right). \tag{5.12}$$

The analytical solution is used as boundary conditions on $\partial\Omega_f$, i.e., on the walls of the cube and for all points inside the spherical hole. When the CFL condition is present, it is difficult to choose situations that have insignificant spatial error. Therefore, we show convergence by refining both Δt and h together. Table 5.12 shows representative error at $t = 1$. We can see that in the L^2 norm, the velocity converges as $O(h^2+\Delta t^2)$ and the pressure converges as $O(h^{\frac{3}{2}}+\Delta t^{\frac{3}{2}})$. The convergence rates are only slightly less in the maximum norm.

h	$\|\mathbf{u}-\mathbf{u}_{\text{exact}}\|_{L^2(\Omega)}$	rate	$\|\mathbf{u}-\mathbf{u}_{\text{exact}}\|_{L^\infty(\Omega)}$	rate
1/20	7.804e-02		2.254e-01	
1/40	4.574e-03	4.09	2.582e-02	3.13
1/80	1.104e-03	2.05	7.473e-03	1.79
1/160	2.335e-04	2.24	2.881e-03	1.38
1/320	5.517e-05	2.08	7.370e-04	1.97
1/640	1.354e-05	2.03	1.902e-04	1.95
1/1280	3.246e-06	2.06	4.793e-05	1.99

h	$\|\nabla\cdot\mathbf{u}\|_{L^2(\Omega)}$	rate	$\|\nabla\cdot\mathbf{u}\|_{L^\infty(\Omega)}$	rate
1/20	1.102e-02		3.903e-02	
1/40	3.475e-03	1.67	1.970e-02	0.99
1/80	1.248e-03	1.48	9.094e-03	1.12
1/160	4.227e-04	1.56	3.243e-03	1.49
1/320	1.058e-04	2.00	8.754e-04	1.89
1/640	2.940e-05	1.85	2.451e-04	1.84
1/1280	6.911e-06	2.09	5.784e-05	2.08

h	$\|p-p_{\text{exact}}\|_{L^2(\Omega_f)}$	rate	$\|p-p_{\text{exact}}\|_{L^\infty(\Omega_f)}$	rate
1/20	7.525e-03		3.109e-02	
1/40	1.291e-03	2.54	8.467e-03	1.88
1/80	4.656e-04	1.47	4.570e-03	0.89
1/160	1.673e-04	1.48	2.505e-03	0.87
1/320	5.556e-05	1.59	1.208e-03	1.05
1/640	2.015e-05	1.46	6.219e-04	0.96
1/1280	7.299e-06	1.46	3.066e-04	1.02

Table 5.11: Spatial convergence when solving the Navier–Stokes equations ($\chi = 1$) on the Taylor–Couette flow. A 10x10 grid was insufficient for stability at $\text{Re} = 1000$. The CFL number on the $h = 1/1280$ grid is about 0.64. The velocity is clearly second-order accurate in space even in the maximum norm. The pressure convergence rate is $O(h^{3/2})$ in the L^2 norm and $O(h)$ in the maximum norm. Since the analytical pressure solution (A.2) for Taylor–Couette is spatially dependent for Navier–Stokes (unlike for Stokes), this explains why the pressure is $O(h^{\frac{3}{2}})$ in this table but appears to be higher order for the Stokes equations in Table 5.6.

h	Δt	$\|\mathbf{u}-\mathbf{u}_{\text{exact}}\|_{L^2(\Omega)}$	rate	$\|\mathbf{u}-\mathbf{u}_{\text{exact}}\|_{L^\infty(\Omega)}$	rate
1/5	1/25	1.791e-02		6.252e-02	
1/10	1/50	4.128e-03	2.12	2.429e-02	1.36
1/20	1/100	8.952e-04	2.20	8.984e-03	1.44
1/40	1/200	1.943e-04	2.20	2.380e-03	1.92
1/80	1/400	4.807e-05	2.02	8.441e-04	1.50
1/160	1/800	1.236e-05	1.96	2.103e-04	2.01
1/320	1/1600	3.638e-06	1.76	5.557e-05	1.92

h	Δt	$\|\nabla\cdot\mathbf{u}\|_{L^2(\Omega)}$	rate	$\|\nabla\cdot\mathbf{u}\|_{L^\infty(\Omega)}$	rate
1/5	1/25	1.052e-01		6.943e-01	
1/10	1/50	6.126e-02	0.78	8.518e-01	-0.29
1/20	1/100	1.346e-02	2.19	2.498e-01	1.77
1/40	1/200	4.439e-03	1.60	3.194e-01	-0.35
1/80	1/400	1.423e-03	1.64	1.819e-01	0.81
1/160	1/800	4.589e-04	1.63	9.680e-02	0.91
1/320	1/1600	1.781e-04	1.37	5.642e-02	0.78

h	Δt	$\|p-p_{\text{exact}}\|_{L^2(\Omega_f)}$	rate	$\|p-p_{\text{exact}}\|_{L^\infty(\Omega_f)}$	rate
1/5	1/25	1.433e-02		7.571e-02	
1/10	1/50	4.262e-03	1.75	4.137e-02	0.87
1/20	1/100	1.489e-03	1.52	2.061e-02	1.01
1/40	1/200	4.852e-04	1.62	9.670e-03	1.09
1/80	1/400	1.700e-04	1.51	5.231e-03	0.89
1/160	1/800	6.096e-05	1.48	2.531e-03	1.05
1/320	1/1600	2.130e-05	1.52	1.258e-03	1.01

Table 5.12: Convergence rates in both time and space when solving the Navier–Stokes equations ($\chi = 1$) in a time-dependent complex-shaped domain using boundary-fitted spatial operators. In the L^2 norm, the velocity converges as $O(h^2 + \Delta t^2)$ and the pressure converges as $O(h^{\frac{3}{2}} + \Delta t^{\frac{3}{2}})$. The convergence rates are only slightly less in the maximum norm. The CFL number ranges from 0.2 to 0.4.

Chapter 6

Discretization Of ODEs For Rigid Objects

In this chapter, we consider discretizing the equations of motion of the rigid objects given by the ordinary differential equations (ODEs) in equations (2.23), (2.24), (2.25). The discretization of the object ODEs works together with the Navier–Stokes scheme of Section 4.4 in order to solve the coupled problem of a fluid containing rigid objects, (2.22)–(2.25). The complete scheme is referred to in this thesis as the "direct scheme with boundary fitting", because the rigid objects are treated *directly* as rigid objects, and the fluid equations are solved with Dirichlet boundary conditions on the object surfaces, using boundary-fitted spatial operators.

6.1 Time Discretization

Equations (2.23), (2.24), (2.25) can be discretized in time in the following way. First, we assume that we know the position $\mathbf{X}_i^n$, velocity $\mathbf{U}_i^n$, and angular velocity $\boldsymbol{\omega}_i^n$ of the i^{th} object at time level n, and we also know the fluid velocity $\mathbf{u}^n$ and pressure $p^{n-\frac{1}{2}}$ as output from the Navier–Stokes solver in Section 4.4. The goal is to obtain the object positions $\mathbf{X}_i^{n+1}$, velocities $\mathbf{U}_i^{n+1}$, and angular velocities $\boldsymbol{\omega}_i^{n+1}$ at time level $n+1$. We start by approximating the fluid stress tensor at time level n as

$$\boldsymbol{\sigma}^n = -\frac{1}{2}\left(p^{n-\frac{1}{2}} + p^{*,n+\frac{1}{2}}\right)\boldsymbol{\delta} + \frac{1}{\mathrm{Re}}(\nabla\mathbf{u}^n + (\nabla\mathbf{u}^n)^T), \tag{6.1}$$

where the pressure predictor $p^{*,n+\frac{1}{2}}$ is computed from (4.104) and p^n is the midpoint approximation $\frac{1}{2}\left(p^{n-\frac{1}{2}} + p^{*,n+\frac{1}{2}}\right)$. The force on the i^{th} object at time level n is defined as

$$\mathbf{F}_i^n = \int\limits_{\partial\Omega_i^n} \boldsymbol{\sigma}^n \cdot \mathbf{n}\, dA + \mathbf{F}_{\text{col},i,j}, \tag{6.2}$$

which is the sum of the force due to the fluid stress and an object-on-object collision force $\mathbf{F}_{\text{col},i,j}$, which will be discussed later. Similarly, the torque on the i^{th} object due to the fluid stress at time level n is defined as

$$\mathbf{T}_i^n = \int\limits_{\partial\Omega_i^n} (\mathbf{x} - \mathbf{X}_i^n) \times (\boldsymbol{\sigma}^n \cdot \mathbf{n})\, dA, \tag{6.3}$$

however no object-on-object collision torque is included (also discussed later). The force and torque require calculation of surface integrals, but this will be discussed in the next section. Assuming for now that the force and torque are known at time levels n and $n-1$, the linear momentum ODE (2.24) can be approximated as

$$M_i \left(\frac{\mathbf{U}_i^{n+1} - \mathbf{U}_i^n}{\Delta t}\right) = \frac{M_i}{\text{Fr}} \left(1 - \frac{\rho_f}{\rho_i}\right) \mathbf{e}_g + \frac{3}{2}\mathbf{F}_i^n - \frac{1}{2}\mathbf{F}_i^{n-1}, \tag{6.4}$$

and the angular momentum ODE (2.25) can be approximated as

$$\mathbf{I}_i \left(\frac{\boldsymbol{\omega}_i^{n+1} - \boldsymbol{\omega}_i^n}{\Delta t}\right) = \frac{3}{2}\mathbf{T}_i^n - \frac{1}{2}\mathbf{T}_i^{n-1}, \tag{6.5}$$

both of which are second-order Adams–Bashforth discretizations similar to (4.99). Finally, the object position equation (2.23) can be approximated by

$$\frac{\mathbf{X}_i^{n+1} - \mathbf{X}_i^n}{\Delta t} = \frac{1}{2}(\mathbf{U}_i^{n+1} + \mathbf{U}_i^n). \tag{6.6}$$

The discretizations (6.4) and (6.5) are both fully explicit steps, and they impose a stability restriction on the time step which depends on the minimum object radius $r_{\min}$. For the explicit step (6.4), we require $\Delta t \leq C r_{\min} h \text{Re}$ and for the explicit step (6.5), we require $\Delta t \leq C r_{\min}^2 h \text{Re}$. The $r_{\min}^2$ restriction occurs because the moment of inertia $\mathbf{I}$ and the torque $\mathbf{T}$ are respectively proportional to r_i^5 and r_i^3 in equa-

tion (6.5), which means that (6.5) is similar to solving an ODE of the form $\frac{d\boldsymbol{\omega}}{dt} \sim \frac{1}{r^2}$. This restriction can be severe if the Reynolds number or the object radius is very small. In such cases one may have to perform iterations to simulate an implicit discretization, or use a small time step. An alternative is to use a fictitious domain method (FDM) such as in Chapter 8, which does not have such a stability restriction because (6.4) and (6.5) are incorporated implicitly in the momentum equation for the fluid. However, at small Reynolds numbers, the accuracy of the FDM suffers severely, even for objects with large radius.

Using a pressure projection scheme for solving Navier–Stokes together with the explicit discretizations (6.4), (6.5) for the object ODEs can be unconditionally *unstable* if any of the objects are less dense than the fluid. This is explained in [37, p. 433] using a heuristic argument based on the "added mass effect". After the publication of this instability, explicit discretizations of the rigid object ODEs were widely abandoned. However, a flow that contains solid objects can be interpreted as a variable density flow where there is a density discontinuity at the object boundaries. The Navier–Stokes equations in this situation are discussed in Appendix B. Following an idea proposed in [32] for modification of pressure projection schemes for flows with variable density, we discovered how to stabilize our scheme for any object density. In particular, the factor

$$\eta_{\min} = \min\left\{1, \min_i \left\{\frac{\rho_i}{\rho_f}\right\}\right\} \tag{6.7}$$

simply needs to be included as a scaling for the divergence in the pressure projection step. This is valid to do because as described in equation (4.49), the divergence is already scaled by a penalty parameter so $\eta_{\min}$ just changes the penalty parameter. The stability and optimality of such formulations is analyzed in [33]. Performing this scaling avoids the need to solve the difficult pressure Poisson problem (B.6) that contains a discontinuous density coefficient. To be concrete, the "incompressibility penalty step" (4.106) needs to be replaced with

$$\begin{aligned}
(1 - \partial_{xx})\theta &= -\frac{\eta_{\min}}{\Delta t}\nabla \cdot \mathbf{u}^{n+1}, \\
(1 - \partial_{yy})\psi &= \theta, \\
(1 - \partial_{zz})\phi^{n+\frac{1}{2}} &= \psi,
\end{aligned} \tag{6.8}$$

but nothing else from Section 4.4 is changed at all. The numerical results in Section 9.5 confirm that the numerical scheme with this modification is stable even when objects are lighter than the fluid.

This completes the time discretization of the object ODEs. The newly computed time level $n+1$ object positions $\mathbf{X}_i^{n+1}$ define the fluid domain Ω_f^{n+1}, and when using the Navier–Stokes solver in Section 4.4 to solve for the fluid velocity $\mathbf{u}^{n+1}$, the object velocities $\mathbf{U}_i^{n+1}$, and angular velocities $\boldsymbol{\omega}_i^{n+1}$ are imposed as Dirichlet boundary conditions on the fluid domain Ω_f^{n+1}.

6.2 Object Orientations

Since all the rigid objects in this thesis are spheres, the orientation of each object is irrelevant. However, it is still useful to compute the orientation of each object for visualization purposes, for example, as shown in Figure 10.3. Therefore, we keep track of the local coordinate system $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, $\hat{\mathbf{z}}$, that defines the orientation of each object. In practice, $\hat{\mathbf{z}}$ is not stored since $\hat{\mathbf{z}} = \hat{\mathbf{x}} \times \hat{\mathbf{y}}$.

After obtaining $\boldsymbol{\omega}_i^{n+1}$ from (6.5), the local coordinate system of each object is updated by computing a rotation matrix representing a rotation of $\Delta t \|\boldsymbol{\omega}_i^{n+1}\|$ radians around the axis $\boldsymbol{\omega}_i^{n+1}/\|\boldsymbol{\omega}_i^{n+1}\|$, and multiplying $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, $\hat{\mathbf{z}}$ by this matrix every time step. It is well known that the vectors $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, $\hat{\mathbf{z}}$ need to be orthonormalized every so often due to buildup of floating point errors.

6.3 Surface Integral Discretization

In this section, we describe how to compute the surface integrals of the fluid stress that appear in (6.2) and (6.3). The integration is discretized on the sphere using a surface grid formed between parallels and meridians such that the surface areas of all surface grid cells are approximately equal and each octant of the sphere contains an equal number of cells (see Figure 6.1). These symmetry properties seem to drastically improve the accuracy of the integral, since the integrand typically contains terms with large magnitudes that cancel each other out. A midpoint quadrature is used to approximate the integral on each surface grid cell.

Since the fluid pressure has been extended into the solid sphere using the procedure in Section 3.3.1, the pressure at the midpoint of each surface grid cell is simply

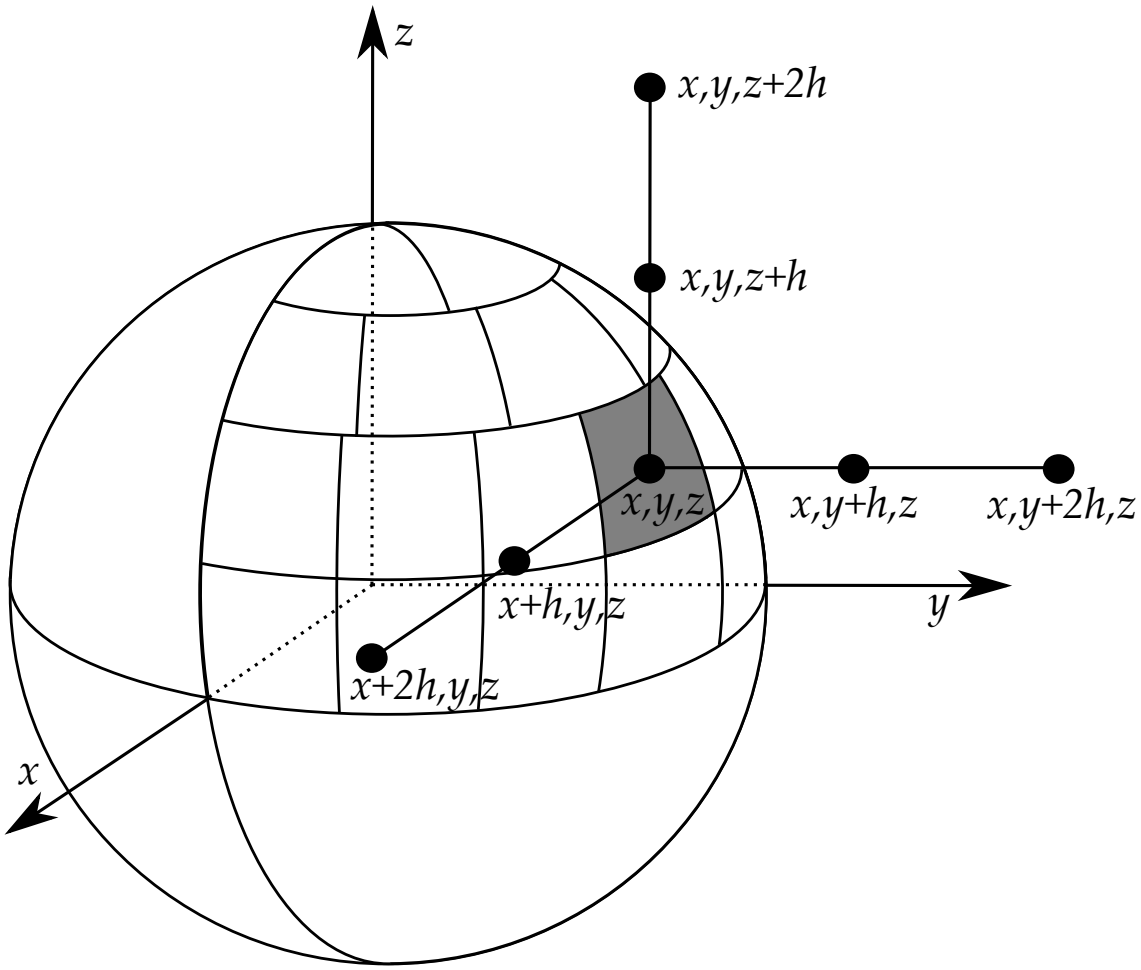

Figure 6.1: Sample surface grid on one octant of a sphere. Also shown is the midpoint of a spherical grid cell and the one-sided three-point finite difference stencils that would be used to approximate velocity derivatives in each of the x, y, and z directions.

approximated using the standard trilinear interpolation of the MAC grid pressure points. However, computing the velocity derivatives that appear in the stress tensor (6.1) at the midpoint of the surface grid is non-trivial. In particular, using any points of the MAC grid that lie outside Ω_f to compute these derivatives leads to an $O(1)$ approximation of the stress. This is because the velocity field in the extended domain Ω is only continuous across the boundary $\partial\Omega_f$, so points outside Ω_f are only $O(h)$ accurate extrapolations of the velocity in Ω_f, i.e., $u_i = u_{i,\text{exact}} + O(h)$. If these $O(h)$ accurate points are then used in a first-order finite difference with a perfectly accurate point $u_{i+1,\text{exact}}$ in the fluid,

$$\frac{\partial u}{\partial x} \approx \frac{u_{i+1,\text{exact}} - (u_{i,\text{exact}} + O(h))}{h} = \frac{u_{i+1,\text{exact}} - u_{i,\text{exact}}}{h} + O(1). \tag{6.9}$$

Numerical testing verifies that this error is indeed $O(1)$, and also that it significantly impacts the overall accuracy of the surface integral.
Note: This $O(1)$ error argument also applies to the advection terms which are essentially first derivatives of the velocity, however, numerical results (Section 5.3) show that advection error of this type has minimal impact on the accuracy of the scheme as a whole.

In order to use only Ω_f points for calculating velocity derivatives, we use *one-sided* finite differences, which means that three points must be used to obtain an $O(h^2)$ accurate approximation of the first derivative. As shown in Figure 6.1, these three points consist of the known surface velocity at (x, y, z) and two additional points inside the fluid. The velocity at these two fluid points is unknown since these two points are not necessarily nodes of the MAC grid. Therefore, we need to approximate the velocity at these points using an interpolation which must not reference any points outside Ω_f. Ideally, we would like an $O(h^3)$ (quadratic) interpolation so that using the interpolated points in a first-order difference would be $O(h^2)$ accurate. However, it is difficult to construct a stable interpolation of this type because quadratic interpolations can produce unbounded values.

We instead construct a boundary-fitted linear interpolation, which is explained in Figure 6.2. This boundary-fitted interpolation has been numerically verified as $O(h^2)$ accurate. By using an $O(h^2)$ accurate interpolation for the velocity, the overall local accuracy of the stress approximation cannot be higher than $O(h)$ due to (6.9). Therefore, a two-point $O(h)$ finite difference stencil could be used instead of the

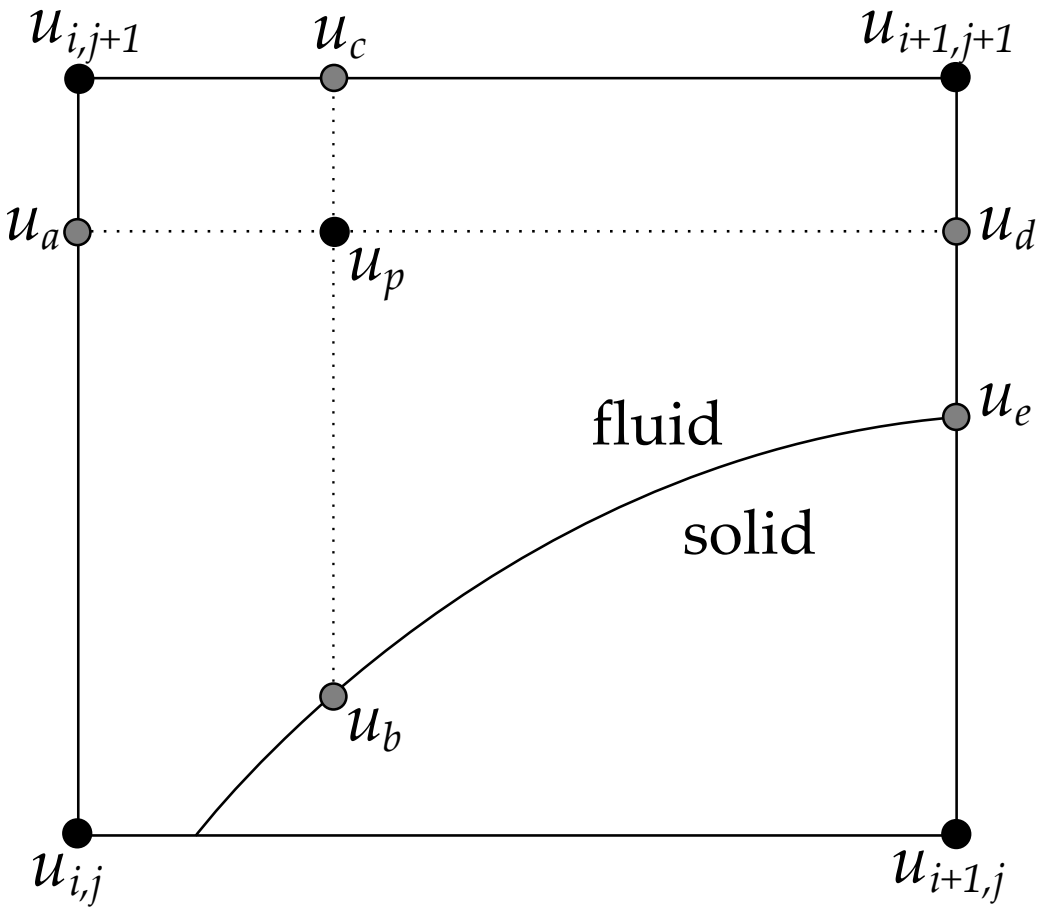

Figure 6.2: An example boundary fitted interpolation in 2D. Suppose u_p is the velocity at a point which requires interpolation. u_b and u_e are known values of u on the object surface, u_a is the linear interpolation between $u_{i,j}$ and $u_{i,j+1}$, u_c is the linear interpolation between $u_{i,j+1}$ and $u_{i+1,j+1}$, and u_d is the linear interpolation between u_e and $u_{i+1,j+1}$. The final approximation for u_p is computed by averaging the values of the horizontal linear interpolation between u_a and u_d and the vertical linear interpolation between u_b and u_c. A similar procedure is used for linearly interpolating the velocity near an object boundary in 3D. In this case, the 2D procedure can be used to interpolate the velocity on cell faces and then the final result is obtained by interpolating these face values.

three-point stencil when approximating the velocity derivatives in the stress, and the asymptotic order would not change. Table 6.1 shows the error when computing the force and torque on the surface of a sphere in 3D when the velocity and pressure fields everywhere in Ω are given by

$$\begin{aligned} u &= x + x^2 y + y^4 z^3, \\ v &= x^4 y^3 z^2, \\ w &= (x+y)^2 (x+z)^2 (y+z)^2, \\ p &= z + xyz + x^2 y, \end{aligned} \tag{6.10}$$

and the Reynolds number is 100. We can see from the table that the surface integral is $O(h^2)$ accurate when computing the pressure contribution to the force, however it is only $O(h)$ accurate when computing the velocity contributions to the force and torque, as we expect from the above discussion. However, in real fluid flows, the flow field tends to have some symmetry properties around the objects. In this case, the three-point stencils often yield more accurate results because the symmetric fields allows some error cancellation. Table 6.2 compares the two-point and three-point stencils by computing the torque on the surface of a 2D disc in a Taylor–Couette flow as in Appendix A. In this highly symmetric flow, the three-point stencils produce significantly less error, beginning with with almost 20 times less error on coarse grids and also converging faster than $O(h)$. For this reason, we always use the three-point stencils for the velocity derivatives when computing the fluid stress.

	Pressure Force		Velocity Force		Torque	
h	error	conv. rate	error	conv. rate	error	conv. rate
1/24	1.783e-04		3.602e-03		1.672e-04	
1/48	4.925e-05	1.86	1.753e-03	1.04	8.424e-05	0.99
1/96	1.229e-05	2.00	8.709e-04	1.01	4.287e-05	0.97
1/192	3.073e-06	2.00	4.348e-04	1.00	2.184e-05	0.97
1/384	7.785e-07	1.98	2.158e-04	1.01	1.082e-05	1.01

Table 6.1: Error when computing the surface force and torque on a sphere in 3D using three-point stencils for the velocity derivatives. This table shows the error contributions in the force vector due to the velocity and pressure integrals separately. Each row of this table contains the average error when placing the sphere at 100 random locations in the domain. This shows that symmetry of the grid is not relevant.

	two-point stencils		three-point stencils	
h	error	conv. rate	error	conv. rate
1/24	2.216e-03		1.378e-04	
1/48	1.112e-03	0.99	2.496e-05	2.46
1/96	5.585e-04	0.99	9.044e-06	1.46
1/192	2.788e-04	1.00	2.716e-06	1.74
1/384	1.394e-04	1.00	1.458e-06	0.90
1/768	6.967e-05	1.00	5.518e-07	1.40
1/1536	3.484e-05	1.00	2.355e-07	1.23
1/3072	1.742e-05	1.00	1.126e-07	1.06
1/6144	8.709e-06	1.00	5.438e-08	1.05

Table 6.2: Error when computing the surface torque on the inner cylinder of a Taylor–Couette flow with $R_1 = 0.25$, $R_2 = 0.5$, $\omega_1 = 1$, $\omega_2 = -1$, $\mathrm{Re} = 100$. Each row contains the average error when placing the cylinders at 1000 random locations in the domain to demonstrate that symmetry of the grid is not relevant. The torque error converges as $O(h^{1.4})$ when using the three-point stencils and it converges as $O(h)$ when using the two-point stencils. The force error is not shown in this table, but the two-point and three-point stencils produce nearly identical error for the force, which converges consistently as $O(h^{1.5})$ in both cases.

6.4 Interpolation-Free Surface Integral

In this section, we consider an alternative surface integral discretization that can be used instead of the discretization in Section 6.3. Since the surface integral of Section 6.3 suffers reduced accuracy due to finite differencing an interpolated field, here we construct a method which does not use any interpolation at all. Consider the x component of the fluid force $\boldsymbol{\sigma} \cdot \mathbf{n}$ in 3D, which is given by

$$F_x = \left[\frac{1}{\mathrm{Re}}\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial x}\right) - p\right] n_x + \left[\frac{1}{\mathrm{Re}}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right] n_y + \left[\frac{1}{\mathrm{Re}}\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right)\right] n_z,$$

where $\mathbf{n} = (n_x, n_y, n_z)$ is the unit normal pointing out of the sphere, and $\mathbf{u} = (u, v, w)$ is the fluid velocity. The surface integral of each term must be computed separately because each term uses different grid points of the MAC grid.

Consider the term $\frac{\partial v}{\partial x}$, for example. In this case, we consider only the v MAC grid points, and we intersect the object with the lines of this grid that are parallel to the x axis. By dividing the sphere first into "belts" and then into cells, each grid line intersection point on the sphere is placed in the approximate middle of an irregularly-shaped surface grid cell, as in Figure 6.3. Figure 6.4 shows how the x direction grid lines intersect one belt. At each point where these grid lines intersect the object, the velocity derivative $\frac{\partial v}{\partial x}$ is computed using a one-sided three-point finite difference using points *along a single grid line*, and therefore no interpolation is needed. As an example, consider calculation of $\frac{\partial v}{\partial x}$ evaluated at the "right intersection point" in Figure 6.4. In this case, the three-point finite difference stencil is computed using the "right intersection point" and the points $i+5$ and $i+6$ along the second to bottom grid line in the figure. For the same derivative evaluated at the "left intersection point", the finite difference stencil would use points i, $i+1$, and the "left intersection point". The point $i+2$ is "too close" to the boundary so it needs to be skipped in order to maintain stability. Numerical testing indicates that skipping points closer to the boundary than $h/4$ is sufficient for stability.

Finally, the area of each surface grid cell is computed analytically and the integral is approximated using either the midpoint rule or Simpson's rule. A similar procedure is done for each term of $\boldsymbol{\sigma} \cdot \mathbf{n}$ to obtain the complete surface force.

Because the surface grid is determined by where each grid line happens to intersect the object, the size of the largest surface grid cell (and thus the accuracy of the integral) is influenced by the shape of the object's surface. For a line integral around

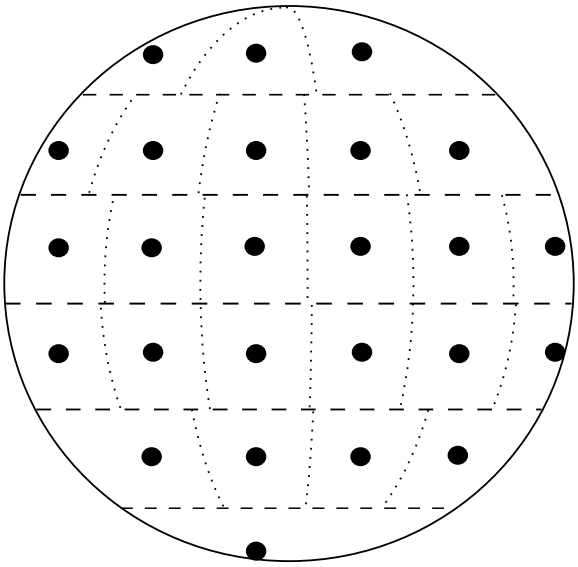

Figure 6.3: An irregularly shaped surface grid on a sphere. The dashed lines divide the sphere into belts, and the dotted lines divide the belts into cells. The x direction grid lines are perpendicular to the plane of the page and appear as solid dots.

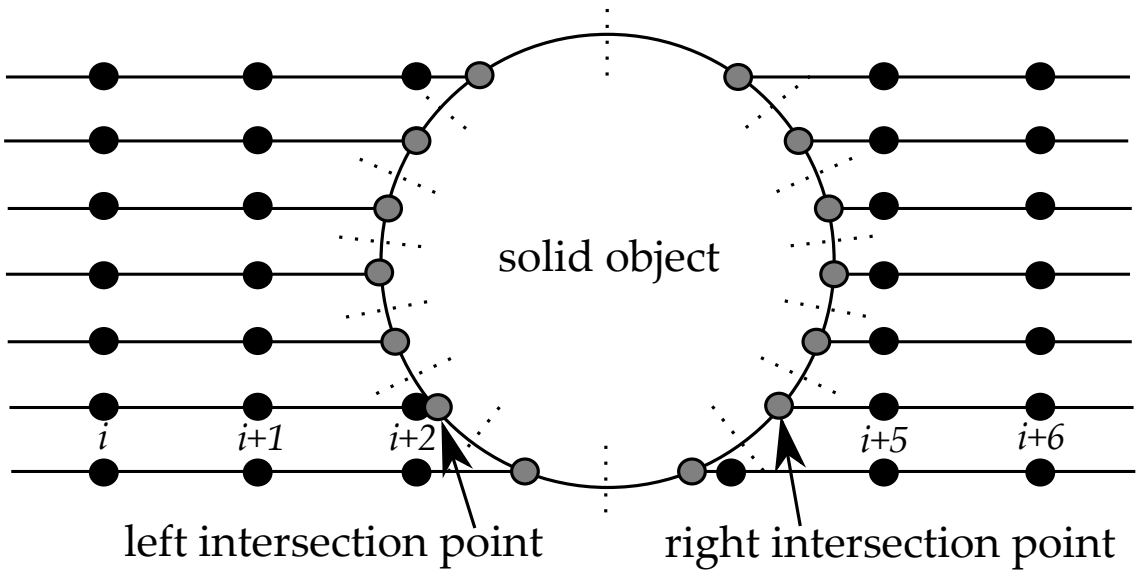

Figure 6.4: x direction grid lines intersecting a single belt (or a disc in 2D). The grid lines run left to right and intersect the belt at the gray dots. The black dots are the points of the MAC grid along the grid lines where the relevant variable is stored (for example, the v velocity component). The short dotted lines denote surface cell boundaries such that each gray dot is near the middle of each cell.

a disc in 2D, it can be shown that reducing h by a factor of 2 will on average reduce the arc length of the largest segment on the circumference of the disc (denoted by $s_{\max}$) by a factor of only $\sqrt{2}$. Since the error when integrating along a curve of length s is $O(s^3)$ using the midpoint rule and $O(s^5)$ using Simpson's rule, this suggests the integration is $O(h^{\frac{3}{2}})$ accurate using the midpoint rule and $O(h^{\frac{5}{2}})$ accurate using Simpson's rule. Table 6.3 confirms the $O(h^{\frac{3}{2}})$ and $O(h^{\frac{5}{2}})$ convergence rates in 2D. Note: The error of the midpoint and Simpson's rules are respectively $O(s^2)$ and $O(s^4)$ if the curve length is constant.

Table 6.4 shows the results of re-calculating the force and torque on a sphere in 3D as originally done in Table 6.1, but this time using the interpolation-free surface integral discretization of this section with three-point stencils and the midpoint rule. The table confirms $O(h^{\frac{3}{2}})$ convergence, as we expect. This strongly suggests that using the interpolation-free surface integral with Simpson's rule would produce a fully $O(h^2)$ discretization.

Despite the increased asymptotic order, however, the interpolation-free method almost always performs worse in practice. This likely occurs for several reasons. First, since surface integrals only use a few grid points near the surface boundary, it is possible that increased error around boundaries (particularly for the non-rotational $\chi = 0$ scheme) can reduce accuracy when calculating the surface stress. The interpolation method in Section 6.3 may reduce the effects of this localized boundary error by smoothing out local bumps, while the "interpolation-free" calculation does not have this property. Also, the method in Section 6.3 has a high degree of spatial symmetry in the operator, while the interpolation-free method lacks symmetry in the layout of the surface grid, and also each term of the stress is computed on a completely different grid (because the MAC grid is staggered). The torque is particularly vulnerable to these undesirable properties, which can be seen by comparing the torque error in Tables 6.1 and 6.4. Even though the interpolation-free method converges at a faster rate, the initial error on coarse grids is much larger, and it takes too many grid refinements for the interpolation-free method to actually become better. However, for numerical schemes that use a non-staggered grid, the interpolation-free surface integral could be significantly better, especially when used with Simpson's rule.

			Midpoint rule		Simpson's rule	
h	$s_{\max}$	rate	error	rate	error	rate
1	3.1416		2.07e+00		5.22e-01	
1/2	1.6029	0.971	1.21e+00	0.778	2.47e-01	1.081
1/4	1.0610	0.595	5.14e-01	1.236	5.22e-02	2.242
1/8	0.7309	0.538	1.98e-01	1.372	9.75e-03	2.419
1/16	0.5107	0.517	7.33e-02	1.436	1.76e-03	2.468
1/32	0.3591	0.508	2.67e-02	1.460	3.16e-04	2.481
1/64	0.2532	0.504	9.57e-03	1.477	5.63e-05	2.488
1/128	0.1788	0.502	3.43e-03	1.483	9.97e-06	2.496
1/256	0.1263	0.501	1.21e-03	1.496	1.77e-06	2.492
1/512	0.0893	0.500	4.32e-04	1.492	3.12e-07	2.505
1/1024	0.0631	0.500	1.53e-04	1.494	5.55e-08	2.492
1/2048	0.0446	0.500	5.43e-05	1.498	9.90e-09	2.487
1/4096	0.0316	0.500	1.92e-05	1.497	1.71e-09	2.538
1/8192	0.0223	0.500	6.80e-06	1.499	3.02e-10	2.500

Table 6.3: Spatial convergence of midpoint rule vs. Simpson's rule for a line integral of the scalar field $f(x, y) = 1 + x + x^2$ around a disc of diameter 2 in 2D. The number of grid points across the disc diameter in each direction is $N = 2/h$, and $s_{\max}$ is the maximum arc segment on the circumference. Each row of this table is the average of 100 tests using small perturbations of disc position and diameter in order to eliminate any grid alignment effects.

	force vector (p)		force vector ($\mathbf{u}$)		torque vector	
h	error	conv. rate	error	conv. rate	error	conv. rate
1/24	1.783e-04		3.790e-03		1.015e-03	
1/48	4.925e-05	1.86	1.450e-03	1.39	3.984e-04	1.35
1/96	1.229e-05	2.00	5.388e-04	1.43	1.472e-04	1.44
1/192	3.073e-06	2.00	1.981e-04	1.44	5.483e-05	1.42
1/384	7.785e-07	1.98	7.113e-05	1.48	1.979e-05	1.47

Table 6.4: Error when computing the surface force and torque on a sphere in 3D using three-point stencils for the velocity derivatives exactly as in Table 6.1, except here we use the "interpolation free" discretization with the midpoint rule. Again, each table row contains the average error when placing the sphere at the same 100 random locations in the domain.

6.5 A simple collision model

Whenever there is at least one moving rigid object, the object may collide with the domain walls or other objects. When finite space and time resolutions are used, the object position update step (6.6) will occasionally predict object positions $\mathbf{X}_i^{n+1}$ such that one or more objects overlaps the domain walls or other objects. This is clearly unphysical, and can also introduce numerical instability in some cases. The difficulty with object collisions is that they typically take place over extremely small time and spatial scales. It is completely impractical to fully resolve these small scales when the scales over which the simulation must be performed are much larger in comparison. Therefore, a collision "subgrid" model of some kind is required to prevent objects from overlapping. In Chapter 7, we present an accurate subgrid model for submerged collisions based on lubrication theory. In this section, we consider a simple "dry" collision model that does not directly incorporate fluid effects at small scales.

Here we use a "soft-sphere" collision model, which means that the objects are allowed to compress slightly when they collide. With this assumption, the time-scale of an object collision is increased, however, it is still small compared to the time scale of the overall fluid flow. A summary of various soft-sphere impact models (in the absence of fluid) is given in [63]. Two good models for modeling dry collisions are the "viscoelastic model" and the "elastic-plastic" model. The viscoelastic model is essentially a spring-dashpot model adjusted for Hertzian elasticity of spheres instead of linear springs. Elastic-plastic models are similar but instead of including a damping term, the spring stiffness changes in time (see Section 7.4). Denote the normal-direction repulsive acceleration acting on object i due to collision with object j by $A_{i,j}$. We use the viscoelastic model, which is given by

$$A_{i,j} = \left(\frac{M_j}{M_i + M_j}\right)\sqrt{x}\left(kx + \gamma\frac{dx}{dt}\right), \tag{6.11}$$

where x is the "overlap" distance of spheres i and j measured along the line connecting the sphere centroids and assuming both spheres are uncompressed. The parameter k is an elastic stiffness parameter of the spheres, and the parameter γ is a damping parameter which affects energy loss during the collision. Note that if object i has infinite mass M_i (such as a wall), then the collision acceleration (6.11) is zero.

The above collision model is incorporated into the "direct scheme with boundary fitting" by using sub-time steps to resolve the collisions. Let $\Delta\tau$ be the sub-step

size, Δt be the regular time step size, and denote a variable f at time $n\Delta t + m\Delta\tau$ by $f^{n+m\frac{\Delta\tau}{\Delta t}}$. The linear object momentum equation (6.4) is advanced in time using sub-stepping as

$$M_i \left(\frac{\mathbf{U}_i^{n+(m+1)\frac{\Delta\tau}{\Delta t}} - \mathbf{U}_i^{n+m\frac{\Delta\tau}{\Delta t}}}{\Delta\tau} \right) = \frac{M_i}{\mathrm{Fr}} \left(1 - \frac{\rho_f}{\rho_i} \right) \mathbf{e}_g + \frac{3}{2}\mathbf{F}_i^n - \frac{1}{2}\mathbf{F}_i^{n-1} + \sum_{j=1}^{N} M_i A_{i,j}, \tag{6.12}$$

where $\mathbf{U}_i^{n+1}$ is obtained when $m = \frac{\Delta t}{\Delta\tau}$. In other words, between each completely coupled "fluid + rigid objects" time step Δt, we solve the rigid object ODEs with a much smaller time step in order to resolve "dry" collisions. This is consistent because by reducing Δt, one eventually recovers all fluid effects of the collision. However, for large Δt, some fluid effects may be missed by resolving the "dry" collisions only. In this case, one can combine this "dry" collision model with a "wet" model as explained in Chapter 7.

Since each sphere using the soft-sphere model is allowed to compress, we enforce a strict "no-overlap" rule for the uncompressed objects by treating each object (for the purpose of computing the overlap distance only) as if it has a radius $R+\epsilon$, where we set ϵ to be the MAC cell width h. This is required because the surface integral methods do not work if the object is not completely surrounded by fluid. Again, this is consistent because the true size of the objects is recovered when $h \to 0$.

Since a thin gap of fluid is always maintained between colliding objects, the surface integral procedure resolves significant torque transfer during a collision and therefore we do not use sub-stepping of the object angular momentum equation 6.5. However, this could easily be incorporated into the scheme.

Note: Using about 100 sub-steps seems sufficient to maintain stability with a large $k \approx 10^6$ so that the objects are significantly rigid and do not overlap even when colliding with large velocities. However, it becomes important to have an efficient parallel code implementation of this sub-stepping that uses non-blocking communication, otherwise the sub-stepping may be expensive in a parallel computing environment (see Section 10.1).

6.5.1 Galilean Cannon

In this section, we check the robustness of the collision model in Section 6.5 by simulating a "Galilean Cannon", which is similar to a "Newton's Cradle". A "Galilean Cannon" is a situation where several balls of different sizes are dropped in a stacked configuration with the smallest ball on top. When the balls hit the floor, most of the rebound momentum of the group of balls is transferred to the smallest ball, and it appears to be fired upwards like a cannon ball. The purpose of this test is not to quantitatively verify correctness of the numerical scheme, but rather to demonstrate prevention of object overlap even in high-impact collision situations. A Galilean Cannon is often used as a physics toy to demonstrate the principles of conservation of momentum, and the "Galilean Cannon" Wikipedia article describes its qualitative behaviour.

Figure 6.5 shows a simulation of a 2D Galilean cannon in a domain $[0, 20] \times [0, 40]$ using three discs of radius 1/2, 1, 2, each with a relative density $\rho_{\text{disc}}/\rho_f = 7.78$, which is the equivalent of steel in water. We use Re = 10, Fr = 0.1, $\Delta t = 1/4000$, $h = 1/10$, modified Douglas splitting with boundary-fitted spatial operators, $\chi = 1$, and viscoelastic collision parameters $k = 10^7$, $\gamma = 0$, using 100 sub-time steps. The maximum CFL number is 0.33 which occurs at impact when fluid is squished out to the sides. Figure 6.6 shows a zoomed-in snapshot of the three discs immediately after collision with the floor. The simulation is successful in preventing any overlap of the discs and maintaining stability in this high-impact collision. Also, qualitatively correct results are produced, for example, the small disc is fired upwards as expected.

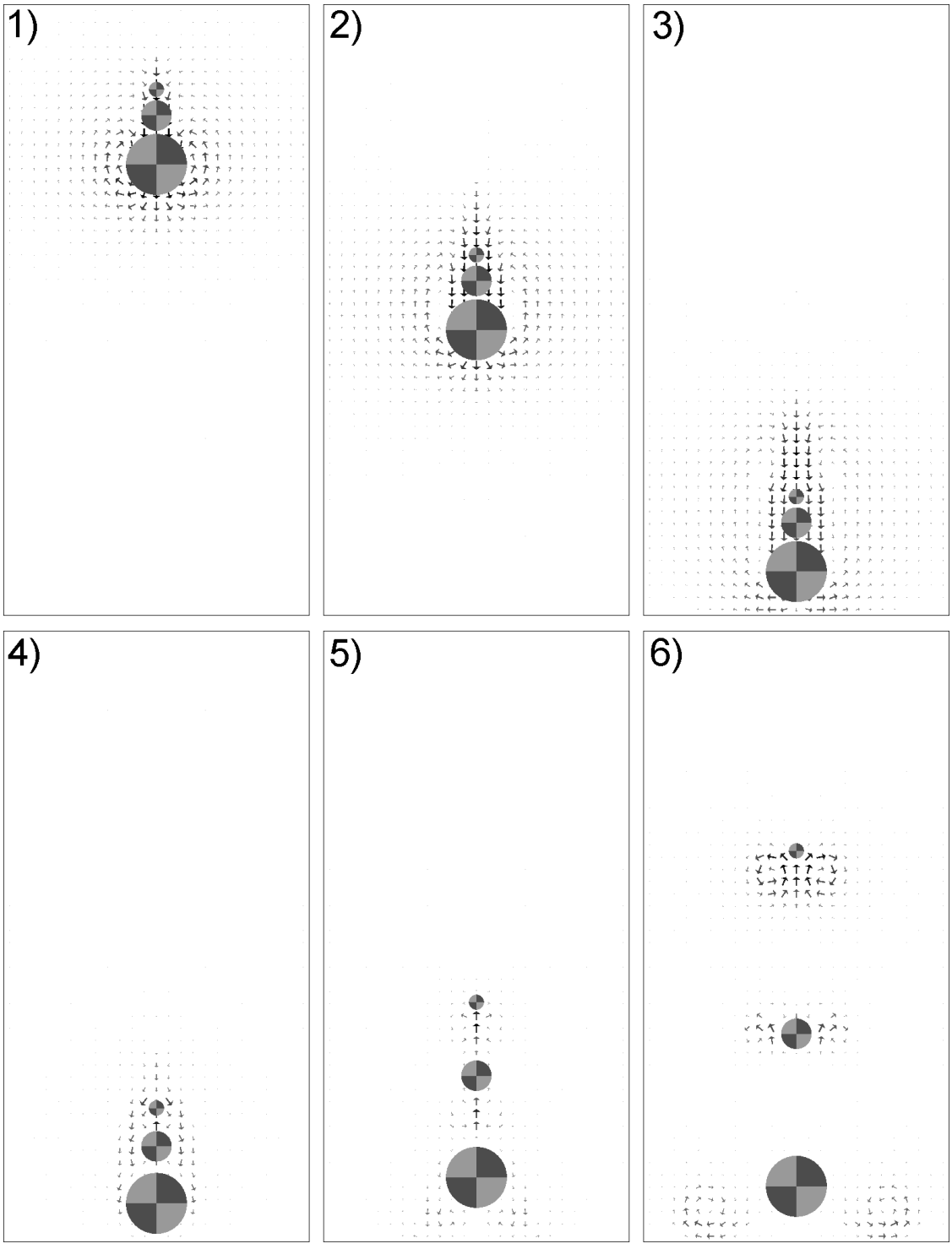

Figure 6.5: Sequence of pictures showing the Galilean cannon simulation.

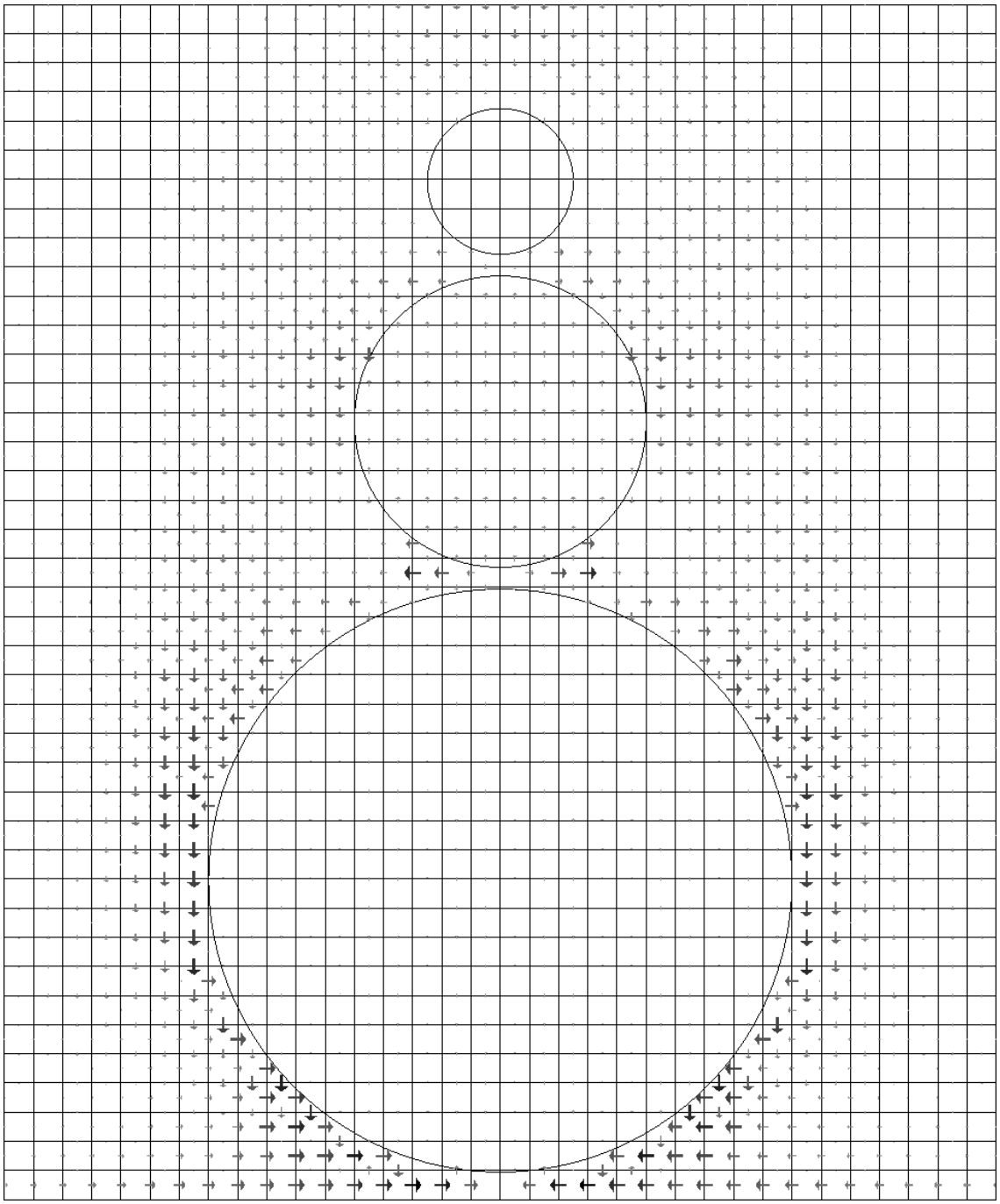

Figure 6.6: Three discs of the Galilean cannon simulation immediately after hitting the floor. In this figure, the u and v velocity components are shown on the MAC cell faces as they are represented numerically rather than showing a cell-center average velocity. The rigid body velocity field inside each object is also shown.

Chapter 7

A Lubrication-Based Collision Model

The collision model in Section 6.5 is sufficient for preventing object overlap and approximating submerged object collisions. However, the model in Section 6.5 uses sub time steps which assume that the fluid stress is constant over these sub-steps. Also, there are some dynamics of submerged collisions that only appear when using finely resolved numerical grids and small time step sizes. Since it is impractical to use such fine grids and small time steps on a large problem, we investigate in this chapter a model that could be incorporated to "fill in" any missing dynamics of submerged collisions when the time and spatial scales between objects are not fully resolved. In particular, we design a high-accuracy method for resolving submerged collisions over small time and spatial scales. The intent is that the high-accuracy method could be used by itself to determine the outcome of a wide range of different collision situations, and these results could all be stored as data in a table. When a collision is predicted using a more simplistic model such as the model of Section 6.5, we could "look up" results of the closest matching collision situation in this data table and incorporate the outcome of the collision rather than trying to fully resolve the collision at run-time. The most simple example of tabulated data that could be generated is a mapping from approach velocity to rebound velocity, computed over the exact distance which the simple collision model does not resolve. This is an improvement over using experimentally measured rebound ratios (coefficients of restitution) because the simple collision model and the associated Navier–Stokes solver already resolves some of the viscous loss upon approach and rebound, so using

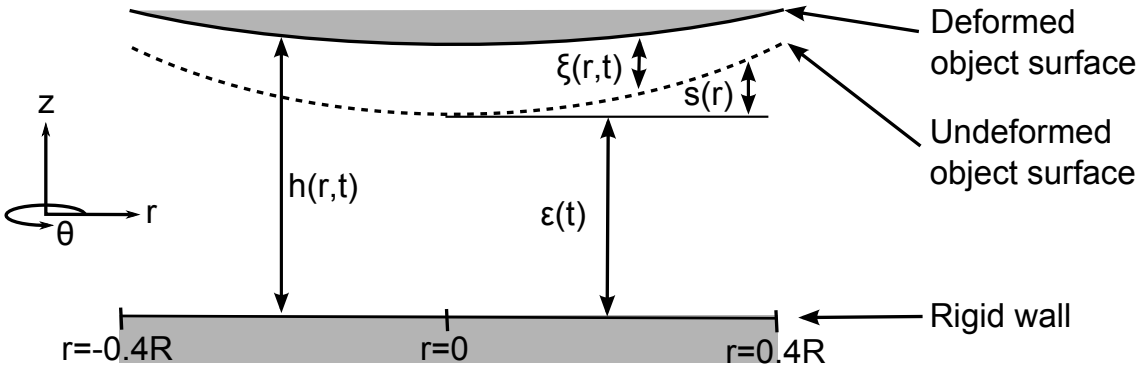

Figure 7.1: Object colliding with wall

the experimental rebound ratio would count these losses twice. Combining the high-accuracy collision model in this chapter with the numerical scheme in the rest of this thesis is a topic for future work, and all the numerical results in this thesis do not use the model in this chapter, except of course for the results in this chapter itself. Readers who are not interested in accurate object collisions can skip this chapter entirely.

We now discuss a high-accuracy method for resolving submerged collisions over small time and spatial scales. Consider Figure 7.1, where a solid, soft sphere of radius R collides with a rigid wall while everything is submerged in a fluid. It is useful to use a cylindrical coordinate system where z is the direction normal to the wall along which the sphere approaches, and r, θ are the coordinates in the plane of the wall. Since both the sphere and the wall have cylindrical symmetry, it is reasonable to assume that if no variable depends on the θ coordinate initially, then it will remain so for all time. Although making this assumption rules out the possibility of waves or turbulence in the θ direction, it greatly simplifies the model. With these assumptions, the three-dimensional fluid velocity has the form

$$\mathbf{u}(r, z, t) = u(r, z, t)\hat{\mathbf{r}}(\theta) + w(r, z, t)\hat{\mathbf{z}}. \tag{7.1}$$

The shape of the sphere is given by

$$s(r) = R - \sqrt{R^2 - r^2}, \tag{7.2}$$

where R is the radius of the sphere. The position of the sphere as it moves toward or away from the wall is given by $\epsilon(t)$ such that the center of the sphere is located at $\epsilon + R$. The sphere may deform as it approaches the wall, and we denote the local deformation of the sphere by $\xi(r,t)$. The thickness of the gap between the sphere and the wall, $h(r,t)$, is completely determined by existing variables,

$$h(r,t) = \epsilon(t) + s(r) + \xi(r,t). \tag{7.3}$$

Since this collision model is intended to be used as a subgrid model, we are only interested in what happens when the sphere is very close to the wall. In particular, we assume a small aspect ratio $h/R \ll 1$. In order for this assumption to be true for all r, we also need to restrict the domain of r. If we choose $r \in [0, 0.4R]$ then the variation in h due to curvature of the sphere $s(r)$ is restricted to about $0.083R$ according to (7.2). This somewhat arbitrary choice of $0.4R$ will be shown to be reasonable in Section 7.12.

7.1 Lubricating Fluid

Between the sphere and the wall (see Figure 7.1) is a thin film of viscous Newtonian fluid. Since collisions can involve large pressure changes, the fluid is treated as compressible using a realistically small amount of compressibility for fluids such as water. In cylindrical polar coordinates with the assumption (7.1), the fluid is governed by the continuity equation,

$$\frac{\partial \rho}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}\left(r\rho u\right) + \frac{\partial}{\partial z}\left(\rho w\right) = 0,$$

the $\hat{\mathbf{z}}$ component of the Navier–Stokes equation,

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial r} + w\frac{\partial w}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial z} + \frac{1}{\rho}\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\mu\frac{\partial w}{\partial r}\right) + \frac{\partial}{\partial z}\left(\mu\frac{\partial w}{\partial z}\right)\right),$$

the $\hat{\mathbf{r}}$ component of the Navier–Stokes equation,

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial r} + w\frac{\partial u}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial r} + \frac{1}{\rho}\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\mu\frac{\partial u}{\partial r}\right) + \frac{\partial}{\partial z}\left(\mu\frac{\partial u}{\partial z}\right) - \frac{\mu u}{r^2}\right),$$

and an equation of state, which we will discuss in Section 7.2. In the above, ρ is the fluid density, μ is the dynamic viscosity of the fluid, and p is the fluid pressure. Since the gap thickness h is small, the Reynolds number based on this length scale is small. It is also reasonable to assume that fluid properties p, μ, ρ, do not change significantly across the gap, since the gap is small. The three approximations $h/R \ll 1$, small Reynolds number, and $\frac{\partial p}{\partial z} = \frac{\partial \rho}{\partial z} = \frac{\partial \mu}{\partial z} = 0$ are collectively called the "lubrication approximation". With these assumptions, one can integrate the continuity and Navier–Stokes equations from $z = 0$ to $z = h$, and this removes the z coordinate from the equations and introduces the variable h. After performing this integration (see [35, ch.7] or [7]), one obtains a two-dimensional equation for the transport of mass, which is called the Reynolds lubrication equation. According to [45], the Reynolds lubrication equation (without the assumption (7.1)) is

$$\frac{\partial}{\partial t}(h\rho) = \nabla \cdot \left(\frac{\rho h^3}{12\mu} \nabla p - \frac{1}{2} \rho h \left(\mathbf{v}_{\text{obj}} + \mathbf{v}_{\text{wall}} \right) \right), \tag{7.4}$$

where $\mathbf{v}_{\text{obj}}$ is the boundary condition shear velocity at the surface of the spherical object and $\mathbf{v}_{\text{wall}}$ is the boundary condition shear velocity at the surface of the wall. For simplicity and to be consistent with our assumption (7.1), we set $\mathbf{v}_{\text{wall}} = \mathbf{v}_{\text{obj}} = 0$ (the nonzero case is a task for future work). After expanding the divergence operator in (7.4) in plane polar coordinates and setting the θ derivatives to zero, we get

$$\frac{\partial}{\partial t}(h\rho) = \frac{1}{r} \frac{\partial}{\partial r} \left(\frac{r \rho h^3}{12\mu} \frac{\partial p}{\partial r} \right), \tag{7.5}$$

which is the basis of the collision model in this chapter.

7.2 Equation Of State

The thermodynamic variables of a *single phase* pure substance are the pressure p, density ρ, temperature T, internal energy per unit mass e, enthalpy per unit mass h, and entropy per unit mass s. Given any two independent pieces of information involving these six variables, the values of all six variables can be uniquely determined. For example, if p and T are known, then we can determine ρ by a relation of the form $\rho = \rho(p, T)$. Since the only thermodynamic variables that appear in our equation (7.5) are the pressure p and density ρ, we will choose a simplified equation

of state,

$$\rho = \rho(p, X) \tag{7.6}$$

where X is an expression involving the four thermodynamic variables $\{T, e, h, s\}$, and X is assumed to be constant. By making an assumption of this form, our equation of state reduces to $\rho = \rho(p)$ for a particular choice of the expression X, and we avoid having to include temperature and heat transfer in our equations. This seems reasonable because an object collision occurs over very short times and therefore heat transfer is not likely important.

Our lubricating fluid will initially be liquid water at normal atmospheric conditions. However, during the process of a collision, the liquid is first exposed to extremely high pressures on impact, but also extremely low pressures on rebound. The possibility that a submerged collision may cause cavitation in the surrounding liquid has been experimentally demonstrated by [44]. Therefore, we allow our lubricating fluid to be one of these two possibilities:

1. Compressed liquid, which means a liquid that isn't about to vaporize.

2. Saturated liquid-vapor mixture, which means an equilibrium of liquid at its vaporization (boiling) point mixed with vapor at its condensation point.

For our purposes, it is unnecessary to include other possibilities such as superheated vapor, saturated liquid-vapor-ice mixture, or saturated vapor-ice mixture. We also ignore any surface tension effects which could exist in cavitation bubbles if some of the fluid vaporizes. Accurate experimental data for the thermodynamic properties of water (see [38], [13]) exists for each separate phase, and this data can be interpreted as a function using interpolation. The experimental data exists in the form $\rho_L(p, X)$ for compressed liquid, and for saturated liquid and saturated vapor, the data exists in the two-part form $\rho_{\text{satL}}(p)$, $X_{\text{satL}}(p)$ and $\rho_{\text{satV}}(p)$, $X_{\text{satV}}(p)$ respectively. Given pressure p and an additional thermodynamic property X, we define our equation of state (7.6) as

$$\rho = \begin{cases} \rho_L(p, X) & \text{if } p, X \text{ is in the domain of } \rho_L(p, X) \\ q\rho_{\text{satV}}(p) + (1-q)\rho_{\text{satL}}(p) & \text{if } p \text{ is in the domain of } \rho_{\text{satL}}(p) \text{ and} \\ & \exists \text{ ``quality'' } q \in [0,1] \text{ such that} \\ & X = qX_{\text{satV}}(p) + (1-q)X_{\text{satL}}(p) \end{cases}$$

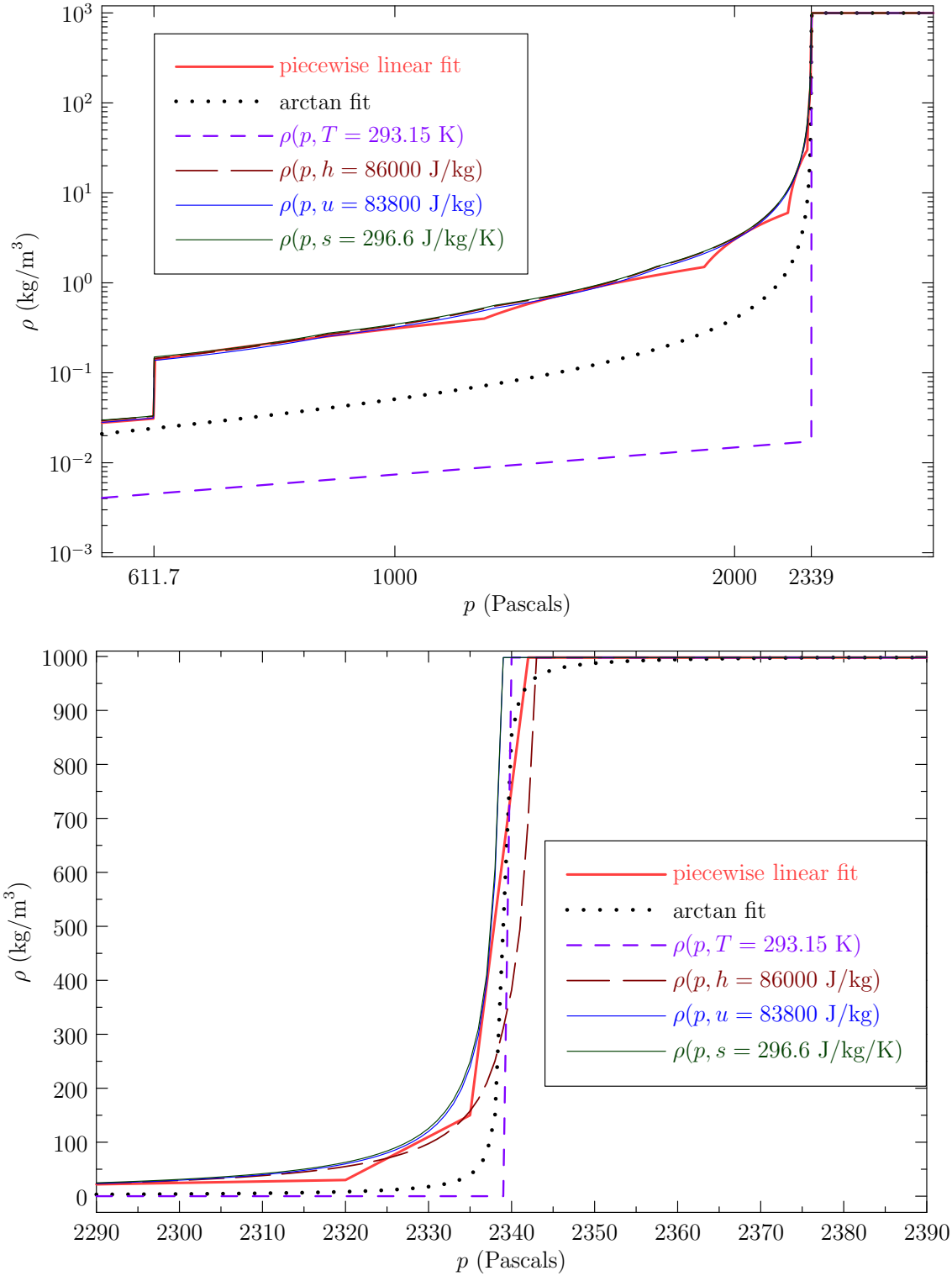

Figure 7.2: Equations of State $\rho = \rho(p)$. The bottom figure is zoomed in around $p = 2339$, the saturation pressure when cavitation occurs. For higher p values, the fluid is in compressed liquid state and ρ changes very slowly with increasing p.

In practice, the above definition is well defined and chooses a unique value of ρ given p and X, with two known exceptions. First, if X is chosen to be temperature, i.e., $\rho = \rho(p, T)$, then when $p = p_{\text{sat}}$ (the saturation pressure at temperature T), there is insufficient information to uniquely determine ρ. However, the temperature is certainly *not* constant during a collision so we do not choose X to be temperature. Second, if $p = p_T$ (the triple point pressure), then p and X are again insufficient to uniquely determine ρ. However, the triple point pressure of water is 611.7 Pascals, and the pressure never drops this low even in high-impact collisions. Both of these cases can be seen as jump discontinuities in Figure 7.2 which shows four different equations of state $\rho = \rho(p, X)$ with the same atmospheric conditions but different thermodynamic properties X chosen to be constant. We can see in the figure that $\rho = \rho(p, T)$ has a jump discontinuity at $p = 2339$ Pascals (the saturation pressure when $T = 293.15°K$), and the other three equations of state have a jump discontinuity at the triple point $p = 611.7$ Pascals.

It turns out that choosing $\rho = \rho(p, h)$, $\rho = \rho(p, e)$, $\rho = \rho(p, s)$, or even one of the two approximate curve fits in Figure 7.2 all give the same macroscopic collision result. However, choosing something completely different (for example, an ideal gas equation of state) changes the outcome of the collision. Therefore, we define our equation of state

$$\rho = \rho(p) \tag{7.7}$$

as the "piecewise linear fit" in Figure 7.2. The *exact* form of this curve fit is not important provided it qualitatively matches the data.

7.3 Fluid Viscosity

During the process of a collision, the pressure may change by a large amount and therefore the fluid viscosity μ is not necessarily constant. Accurate experimental data [39] for the viscosity of water exists in the form $\mu = \mu(\rho, T)$. Figure 7.3 shows the experimental data $\mu = \mu(\rho)$ for liquid water at three different constant temperatures. For $\rho < 998.0$ kg/m^3, water exists as a mixture of vapor and liquid, and it is difficult to find the viscosity of such a mixture in the literature. Also, the experimental data does not exist for extremely high pressures and densities. We must therefore approximate and extrapolate the data. Figure 7.3 shows two piecewise linear approximations/extrapolations, "Approx 1" and "Approx 2". It

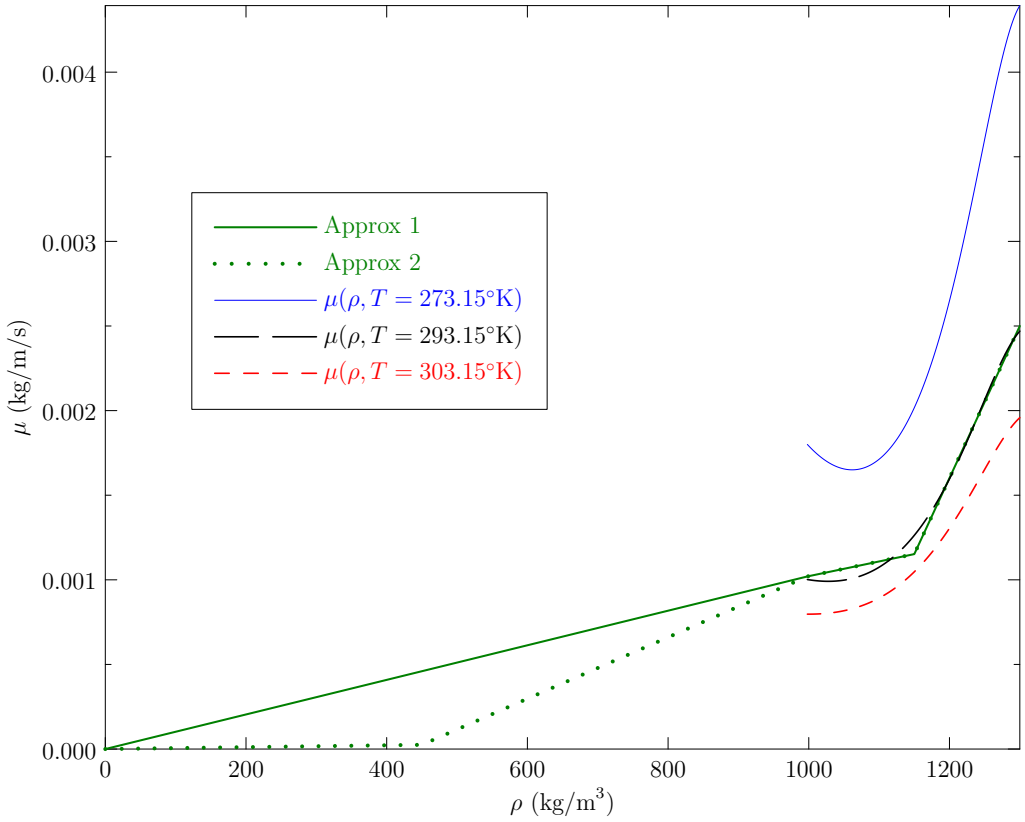

Figure 7.3: Experimental data for $\mu = \mu(\rho)$ for liquid water at three different constant temperatures. Also included are two piecewise linear curve fits "Approx 1" and "Approx 2".

is probably more realistic for μ to drop quickly when the fluid starts to cavitate for $\rho < 998.0$ kg/m^3, however, this presents a more CPU time consuming problem because μ appears in the denominator in (7.5). Testing indicates that both "Approx 1" and "Approx 2" produce the same macroscopic collision result, but "Approx 1" allows larger time steps. We therefore choose

$$\mu = \mu(\rho) \tag{7.8}$$

to be "Approx 1" in Figure 7.3. Again, the *exact* form of this curve fit is not important provided it qualitatively matches the data.

7.4 Solid Compressibility

Although the solid sphere should be very rigid, under large pressures a small amount of compressibility will occur. We model the compressibility of the sphere by allowing the position of the sphere's surface to compress like a spring in the z direction by an amount $\xi(r,t)$ (see Figure 7.1). We wish to choose a spring model that has a physical basis, leads to efficient numerical implementation, and contains parameters for which there exist experimental data. A good choice is an elastic-plastic piecewise linear spring law similar to [46, p. 138]. The model is essentially Hooke's law "$p = k\xi$" but with two different spring constants depending on the time history of the spring's state:

$$p = \begin{cases} k_1 \xi & \text{if } \xi \geq \xi_{\max} \\ k_2 \xi + \xi_{\max}\,(k_1 - k_2) & \text{if } \xi < \xi_{\max} \end{cases}, \tag{7.9}$$

where

$$\xi_{\max}(r,t) = \max_{\tau<t}\ \xi(r,\tau)$$

is the maximum spring compression to-date. Note that if $k_1 = k_2$, then we get Hooke's law which is perfectly elastic. With $k_2 > k_1$, however, the spring does not rebound all the way to its original position. This models a partially plastic compression. Consider Figure 7.4 where, as in a typical collision, the spring is compressed using spring constant k_1 from $\xi = 0$ to its maximum compression at $\xi = \xi_1$. On rebound, $\xi < \xi_{\max} = \xi_1$ and spring constant k_2 is used. Finally, the spring settles at its new equilibrium state, $\xi = \xi_2$. The total energy absorbed by the spring during this process can be computed as the area bounded by the solid lines in Figure 7.4,

$$\begin{aligned} E_{\text{lost}} &= \int p\, d\xi = \int_0^{\xi_1} k_1 \xi\, d\xi + \int_{\xi_1}^{\xi_2} k_2 \xi + \xi_1\,(k_1 - k_2)\, d\xi \\ &= \cdots = \frac{1}{2} k_1 \xi_1^2 \left(1 - \frac{k_1}{k_2}\right) \end{aligned} \tag{7.10}$$

where in the above calculation, we have omitted some algebra and made use of the fact that $\xi_2 = \xi_1 \left(1 - \frac{k_1}{k_2}\right)$, which can be derived from Figure 7.4. If there is no fluid, all of the incoming object's kinetic energy will be transferred to the spring when

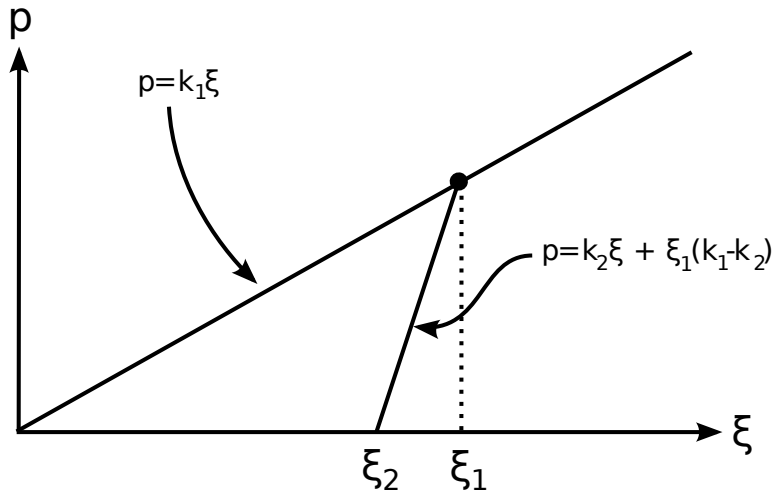

Figure 7.4: Spring force (p) vs. compression distance (ξ)

$\xi = \xi_1$, so the object's initial kinetic energy is

$$E_{\text{in}} = \frac{1}{2} k_1 \xi_1^2. \tag{7.11}$$

We can now relate the parameters k_1, k_2 in our spring model to experimentally measured parameters. Young's modulus E is defined by $F = \frac{E}{L}\Delta L$ for a material of length L with force F compressing it a distance ΔL. Equation (7.9) suggests that we take

$$k_1 = \frac{E}{R}, \tag{7.12}$$

where the radius of the sphere R takes the role of L and ξ takes the role of ΔL. When there is no fluid, the dry coefficient of restitution e_{dry} is defined by $e_{\text{dry}} = \frac{V_{\text{out}}}{V_{\text{in}}}$ where V_{in} is the magnitude of the incoming, pre-collision velocity of the sphere and V_{out} is the magnitude of the outgoing, post-collision velocity of the sphere. Thus,

$$e_{\text{dry}}^2 = \frac{V_{\text{out}}^2}{V_{\text{in}}^2} = \frac{\frac{1}{2}MV_{\text{out}}^2}{\frac{1}{2}MV_{\text{in}}^2} = \frac{E_{\text{in}} - E_{\text{lost}}}{E_{\text{in}}} = \frac{\frac{1}{2}k_1\xi_1^2 - \frac{1}{2}k_1\xi_1^2\left(1 - \frac{k_1}{k_2}\right)}{\frac{1}{2}k_1\xi_1^2} = \frac{k_1}{k_2}, \tag{7.13}$$

where E_{in} is given by (7.11) and E_{lost} is given by (7.10). Therefore, we choose

$$k_2 = \frac{k_1}{e_{\text{dry}}^2}. \tag{7.14}$$

Our spring model is now defined by Young's modulus and the coefficient of restitution for dry collisions, for which experimental data exists (see [42], [41]). It turns out to be more useful to express ξ as a function of p, so we finally invert (7.9) to get

$$\xi = \begin{cases} \frac{p}{k_1} & \text{if } p \geq p_{\max} \\ \frac{p - p_{\max}}{k_2} + \frac{p_{\max}}{k_1} & \text{if } p < p_{\max} \end{cases}, \tag{7.15}$$

where k_1, k_2 are given by (7.12), (7.14), and

$$p_{\max}(r,t) = \max_{\tau < t} \; p(r,\tau).$$

7.5 Object Motion

Since the sphere is solid and we expect it not to deform a large amount compared to its radius, the center of mass of the sphere can be taken to be $\epsilon + R$ (see Figure 7.1). The motion of the sphere obeys Newton's second law,

$$\frac{\partial^2 \epsilon}{\partial t^2} = \frac{1}{M} \iint (p - p_0) \; dA, \tag{7.16}$$

where M is the mass of the sphere, p is the fluid pressure, p_0 is the "ambient pressure", and the integral is carried out over the surface of the sphere. Since the pressure p in the domain does not depend on z, it is sufficient to perform the integration over the 2-D disc beneath the sphere.

7.6 Nondimensionalization

Our full set of seven equations for the seven variables $p, \rho, \mu, \xi, h, s, \epsilon$ are (7.2), (7.3), (7.5), (7.7), (7.8), (7.15), and (7.16), together with the initial and boundary condi-

tions

$$\epsilon(t=0)=\epsilon_0, \qquad \frac{d\epsilon}{dt}(t=0)=-\dot{\epsilon_0}, \qquad p(r,t=0)=p_0,$$

$$\frac{\partial p}{\partial r}(r=0,t)=0, \qquad p(r=0.4R,t)=p_0,$$

where ϵ_0 is the initial separation distance between the sphere and the wall, and $\dot{\epsilon_0}$ is the initial speed of the sphere.

Let the dimensional variables and constants in our equations thus far be denoted by a $\hat{\text{hat}}$. We choose new nondimensional variables using the scalings

$$(\hat{r},\hat{h},\hat{\epsilon},\hat{\xi})=\hat{R}(r,h,\epsilon,\xi)$$

$$\hat{\rho}=\hat{\rho_0}\rho, \qquad \hat{\mu}=\hat{\mu_0}\mu,$$

$$\hat{t}=\frac{\hat{R}}{\hat{\dot{\epsilon}_0}}t,$$

$$(\hat{p},\hat{p_0},\hat{p_{\max}})=\frac{12\hat{\dot{\epsilon}_0}\hat{\mu_0}}{\hat{R}}(p,p_0,p_{\max}),$$

where $\hat{\rho_0}$ and $\hat{\mu_0}$ are the ambient density and viscosity associated with the ambient pressure $\hat{p_0}$ through equations (7.7) and (7.8). Using these scalings, our seven equations transform into

$$s(r)=1-\sqrt{1-r^2}, \qquad h(r,t)=\epsilon(t)+s(r)+\xi(r,t), \tag{7.17}$$

$$\xi=\begin{cases}\frac{p}{k_1} & \text{if } p\geq p_{\max}\\ \frac{p-p_{\max}}{k_2}+\frac{p_{\max}}{k_1} & \text{if } p<p_{\max}\end{cases}, \tag{7.18}$$

$$\rho=\rho(p), \qquad \mu=\mu(\rho), \tag{7.19}$$

$$\frac{\partial}{\partial t}(h\rho)=\frac{1}{r}\frac{\partial}{\partial r}\left(\frac{r\rho h^3}{\mu}\frac{\partial p}{\partial r}\right), \tag{7.20}$$

$$\frac{\partial^2\epsilon}{\partial t^2}=\frac{1}{M}\iint(p-p_0)\,dA, \tag{7.21}$$

where the new nondimensional parameters M, k_1, k_2 are

$$M = \frac{\hat{M}\hat{\dot{\epsilon}}_0}{12\hat{R}^2\hat{\mu}_0}, \qquad k_1 = \frac{E\hat{R}}{12\hat{\dot{\epsilon}}_0\hat{\mu}_0}, \qquad k_2 = \frac{k_1}{e_{\text{dry}}^2}, \tag{7.22}$$

and our nondimensional initial and boundary conditions are

$$\epsilon(t=0) = \epsilon_0, \qquad \frac{d\epsilon}{dt}(t=0) = -1, \qquad p(r, t=0) = p_0,$$

$$\frac{\partial p}{\partial r}(r=0, t) = 0, \qquad p(r=0.4, t) = p_0.$$

7.7 Solution Method

We are interested in solving the equations (7.17)–(7.21) numerically. However, a more elegant formulation can be obtained if we first perform some manipulations. Let $\dot{\epsilon}(t) = \frac{d\epsilon}{dt}(t)$ and differentiate (7.17) with respect to t to get

$$\frac{\partial h}{\partial t} = \dot{\epsilon} + \frac{\partial \xi}{\partial t}. \tag{7.23}$$

Also, differentiate (7.18) with respect to t to get

$$\frac{\partial \xi}{\partial t} = \begin{cases} \frac{1}{k_1}\frac{\partial p}{\partial t} & \text{if } p \geq p_{\max} \\ \frac{1}{k_2}\frac{\partial p}{\partial t} & \text{if } p < p_{\max} \end{cases},$$

which for simplicity we will denote by

$$\frac{\partial \xi}{\partial t} = \frac{1}{k_p}\frac{\partial p}{\partial t}, \tag{7.24}$$

where k_p is either k_1 or k_2 as appropriate. Combining (7.23) and (7.24) gives

$$\frac{\partial h}{\partial t} = \dot{\epsilon} + \frac{1}{k_p}\frac{\partial p}{\partial t}. \tag{7.25}$$

We also need to differentiate (7.19) with respect to t using the chain rule to get

$$\frac{\partial \rho}{\partial t} = \frac{d\rho}{dp}\frac{\partial p}{\partial t}. \tag{7.26}$$

We then write (7.20) using the product rule as

$$\rho\frac{\partial h}{\partial t} + h\frac{\partial \rho}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(\frac{r\rho h^3}{\mu}\frac{\partial p}{\partial r}\right), \tag{7.27}$$

and substitute (7.25) and (7.26) to get

$$\rho\left(\dot{\epsilon} + \frac{1}{k_p}\frac{\partial p}{\partial t}\right) + h\frac{d\rho}{dp}\frac{\partial p}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(\frac{r\rho h^3}{\mu}\frac{\partial p}{\partial r}\right),$$

which we finally re-arrange as

$$\frac{\partial p}{\partial t} = \frac{\frac{1}{r}\frac{\partial}{\partial r}\left(\frac{r\rho h^3}{\mu}\frac{\partial p}{\partial r}\right) - \rho\dot{\epsilon}}{\frac{\rho}{k_p} + h\frac{d\rho}{dp}}.$$

Our system of equations (7.17)–(7.21) can now be solved in the much more natural format,

$$\frac{\partial}{\partial t}\begin{bmatrix} p \\ \epsilon \\ \dot{\epsilon} \end{bmatrix} = \begin{bmatrix} \frac{\frac{1}{r}\frac{\partial}{\partial r}\left(\frac{r\rho h^3}{\mu}\frac{\partial p}{\partial r}\right) - \rho\dot{\epsilon}}{\left(\frac{\rho}{k_p} + h\frac{d\rho}{dp}\right)} \\ \dot{\epsilon} \\ \frac{1}{M}\iint (p - p_0)\, dA \end{bmatrix}, \tag{7.28}$$

together with the relations (7.17)–(7.19). Given ϵ, $\dot{\epsilon}$, p, $p_{\max}$ at time t (where $p_{\max} = p$ at $t = 0$), we can get ϵ, $\dot{\epsilon}$, p, $p_{\max}$ at time $t + \Delta t$ as follows:

1. Use (7.18) to get ξ at time t from p, $p_{\max}$ at time t.
2. Use (7.17) to get h at time t from ξ, ϵ at time t.
3. Use (7.19) to get ρ at time t from p at time t.
4. Use (7.19) to get μ at time t from ρ at time t.
5. Integrate (7.28) forward in time by Δt to get ϵ, $\dot{\epsilon}$, p at time $t + \Delta t$.
6. Update $p_{\max}$ at each grid point as appropriate.

7.8 Can Solid On Solid Contact Occur?

One result from ODE theory is that "fixed points" can split the domain into invariant regions. More precisely, suppose $\mathbf{x}(t) = (x_1(t), .., x_n(t))$ obeys the system of ODEs $\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, t)$, where $\mathbf{f}(\mathbf{x}, t) = (f_1(\mathbf{x}, t), .., f_n(\mathbf{x}, t))$ is Lipshitz continuous. Then

$$x_i(t = 0) > 0 \text{ and } f_i(x_1, .., x_{i-1}, 0, x_{i+1}, .., x_n, t) = 0 \quad \Rightarrow \quad x_i > 0 \quad \forall \quad t. \tag{7.29}$$

Thus, to show that h will remain positive for all time (no solid on solid contact), it is sufficient to show that $\frac{\partial h}{\partial t}\big|_{h=0} = 0$. If the fluid is incompressible (ρ = constant), then the lubrication equation (7.20) becomes

$$\frac{\partial h}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(\frac{rh^3}{\mu}\frac{\partial p}{\partial r}\right),$$

which can be re-written using the product rule as

$$\frac{\partial h}{\partial t} = h^3\frac{1}{r}\frac{\partial}{\partial r}\left(\frac{r}{\mu}\frac{\partial p}{\partial r}\right) + 3h^2\frac{\partial h}{\partial r}\left(\frac{1}{\mu}\frac{\partial p}{\partial r}\right).$$

We can see that as long as $\mu \neq 0$ and all spatial derivatives are bounded, $\frac{\partial h}{\partial t}\big|_{h=0} = 0$, and from (7.29) that means h remains positive for all time. This means that contact cannot occur for an incompressible fluid (with bounded spatial derivatives) regardless of elasticity model or forcing on the sphere.

We now consider the time derivative of h in our model if ρ is not constant, i.e., a compressible fluid. Using (7.25) and (7.26), we can write

$$\frac{\partial \rho}{\partial t} = k_p\frac{d\rho}{dp}\left(\frac{\partial h}{\partial t} - \dot{\epsilon}\right),$$

which we then sub into (7.27) and re-arrange to get

$$\frac{\partial h}{\partial t} = \frac{\frac{1}{r}\frac{\partial}{\partial r}\left(\frac{r\rho h^3}{\mu}\frac{\partial p}{\partial r}\right) + hk_p\dot{\epsilon}\frac{d\rho}{dp}}{\rho + hk_p\frac{d\rho}{dp}}.$$

Then using the product rule, we have

$$\frac{\partial h}{\partial t} = \frac{h^3 \frac{1}{r}\frac{\partial}{\partial r}\left(\frac{r\rho}{\mu}\frac{\partial p}{\partial r}\right) + 3h^2\frac{\partial h}{\partial r}\left(\frac{\rho}{\mu}\frac{\partial p}{\partial r}\right) + hk_p\dot{\epsilon}\frac{d\rho}{dp}}{\rho + hk_p\frac{d\rho}{dp}}.$$

As long as $\mu \neq 0$, $\rho \neq 0$, $\frac{d\rho}{dp}$ is bounded, and all spatial derivatives are bounded, it follows that $\frac{\partial h}{\partial t}\big|_{h=0} = 0$. We can again conclude from (7.29) that h will remain positive (no solid on solid contact) for all time regardless of what forcing function $\epsilon(t)$ is given. Numerical testing confirms this "no contact" result (provided sufficient spatial resolution is used), even for heavy objects traveling very fast toward the wall. However, [7] indicates that when the sphere and wall are considered to be *completely* rigid, solid on solid contact can occur.

7.9 Spatial Discretization

We discretize (7.28) spatially in the radial coordinate r by using a uniform grid of one-dimensional cells where the values of the variables p, ρ, h are stored at the center of each grid cell. The first grid point is located at $r = \frac{1}{2}\Delta r$, where Δr is the width of the one-dimensional cells.

Let subscript i denote evaluation at $r = i\Delta r$, and let $f = \frac{\rho h^3}{\mu}$. A finite volume discretization for the expression $\frac{1}{r}\frac{\partial}{\partial r}\left(rf\frac{\partial p}{\partial r}\right)$ that appears in (7.28) can be obtained by integrating the expression over the area of a ring and dividing by the area of that ring:

$$\begin{aligned}
\left[\frac{1}{r}\frac{\partial}{\partial r}\left(rf\frac{\partial p}{\partial r}\right)\right]_{i+\frac{1}{2}} &\approx \frac{\int_0^{2\pi}\int_{i\Delta r}^{(i+1)\Delta r}\frac{1}{r}\frac{\partial}{\partial r}\left(rf\frac{\partial p}{\partial r}\right)\, r\, dr\, d\theta}{\pi r_{i+1}^2 - \pi r_i^2} \\
&\approx \frac{2\pi\int_{i\Delta r}^{(i+1)\Delta r}\frac{\partial}{\partial r}\left(rf\frac{\partial p}{\partial r}\right)\, dr}{\pi r_{i+1}^2 - \pi r_i^2} \\
&\approx \frac{2\left[rf\frac{\partial p}{\partial r}\right]_{i+1} - 2\left[rf\frac{\partial p}{\partial r}\right]_i}{r_{i+1}^2 - r_i^2}.
\end{aligned} \tag{7.30}$$

For the expression $\left[rf\frac{\partial p}{\partial r}\right]$, we use the midpoint approximation and a centered finite

difference,

$$\left[rf\frac{\partial p}{\partial r}\right]_i \approx r_i \left(\frac{f_{i+\frac{1}{2}} + f_{i-\frac{1}{2}}}{2}\right)\left(\frac{p_{i+\frac{1}{2}} - p_{i-\frac{1}{2}}}{\Delta r}\right), \tag{7.31}$$

which is second-order accurate. The combination of (7.30) and (7.31) produces the discretization

$$\left[\frac{1}{r}\frac{\partial}{\partial r}\left(rf\frac{\partial p}{\partial r}\right)\right]_{i+\frac{1}{2}} \approx \tag{7.32}$$
$$\frac{r_{i+1}\left(f_{i+\frac{3}{2}} + f_{i+\frac{1}{2}}\right)\left(p_{i+\frac{3}{2}} - p_{i+\frac{1}{2}}\right) - r_i\left(f_{i+\frac{1}{2}} + f_{i-\frac{1}{2}}\right)\left(p_{i+\frac{1}{2}} - p_{i-\frac{1}{2}}\right)}{\Delta r\left(r_{i+1}^2 - r_i^2\right)},$$

which is second-order accurate. The total mass in the domain is $\int(\rho h)\,dA$ integrated over the 2D domain area $r \le 0.4R$. The stencil (7.32) conserves total mass in the domain in the discrete sense because

$$\begin{aligned}
\frac{d}{dt}\sum(\rho h)\,\Delta A &= \sum_{i=0}^{N-1}\frac{\partial}{\partial t}(\rho h)_{i+\frac{1}{2}}\left(\pi r_{i+1}^2 - \pi r_i^2\right) \\
\text{using equation (7.20)} \quad &= \sum_{i=0}^{N-1}\left[\frac{1}{r}\frac{\partial}{\partial r}\left(rf\frac{\partial p}{\partial r}\right)\right]_{i+\frac{1}{2}}\left(\pi r_{i+1}^2 - \pi r_i^2\right) \\
\text{using equation (7.30)} \quad &= \sum_{i=0}^{N-1} 2\pi\left(\left[rf\frac{\partial p}{\partial r}\right]_{i+1} - \left[rf\frac{\partial p}{\partial r}\right]_i\right) \\
&= 2\pi\left(\left[rf\frac{\partial p}{\partial r}\right]_N - \left[rf\frac{\partial p}{\partial r}\right]_0\right) \\
&= 2\pi\left[rf\frac{\partial p}{\partial r}\right]_N,
\end{aligned}$$

which is a telescoping sum where the only term remaining represents mass flux through the boundary of the domain at $r = 0.4R$.

The area integral appearing as the last component in (7.28) is computed using a Riemann sum based on ring areas $\pi r_{i+1}^2 - \pi r_i^2$:

$$\iint (p - p_0)\,dA \approx \sum_{i=0}^{N-1}(p - p_0)_{i+\frac{1}{2}}\left(\pi r_{i+1}^2 - \pi r_i^2\right).$$

7.10 Time Discretization

Let superscripts denote time, and let $\mathbf{u} = (\mathbf{p}, \epsilon, \dot{\epsilon}) = (p_0, p_1, ..., p_{N-1}, \epsilon, \dot{\epsilon})$ be the vector of variables that we need to solve for in (7.28). Equation (7.28) is stiff because $\mathbf{p}$ changes very rapidly compared to ϵ. Therefore, we must use an implicit scheme to avoid being forced to use an unreasonably small time step for the entire simulation. A fully implicit scheme could be used to solve for $\mathbf{u}^{n+1}$ in

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \mathbf{F}(\mathbf{u}^{n+1}),$$

where $\mathbf{F}$ is the (nonlinear) function of $\mathbf{u} = (\mathbf{p}, \epsilon, \dot{\epsilon})$ given by (7.28). Therefore, iterative root-finding methods like Newton-Raphson would need to be used. Instead, we choose a semi-implicit scheme, which means that $\mathbf{F}$ is linearized about time level n so that solving for $\mathbf{u}^{n+1}$ becomes a linear algebra problem,

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \mathbf{F}(\mathbf{u}^n) + J\left(\mathbf{u}^{n+1} - \mathbf{u}^n\right),$$

where $J = \frac{\partial \mathbf{F}}{\partial \mathbf{u}}(\mathbf{u}^n)$ is the Jacobian matrix of $\mathbf{F}$ evaluated at $\mathbf{u}^n$. As our time stepping scheme, we use an embedded fourth order A-stable semi-implicit Rosenbrock method with an adaptive time step, as in [57]. A-stable is sometimes called "stiffly stable" and means that the scheme is stable for any time step size (provided the ODE being solved is stable). By using a semi-implicit scheme, we need to compute the Jacobian of the right-hand side of (7.28), and then at each time step we need to solve several algebraic systems of the form $(I - J)\mathbf{u^{n+1}} = b$. For the $\mathbf{F}$ given by (7.28), the Jacobian has the form

$$J = \begin{bmatrix} X & X & & & & & X & X \\ X & X & X & & & & X & X \\ & X & X & X & & & X & X \\ & & X & X & X & & X & X \\ & & & X & X & X & X & X \\ & & & & X & X & X & X \\ & & & & & & X & X \\ X & X & X & X & X & X & X & X \end{bmatrix},$$

and these algebraic systems can be solved in $O(N)$ time using Gaussian elimination. However, since the above matrix is a strange form, a custom Gaussian elimination routine was written in C++ specifically for solving linear systems where the matrix has the above form. For our problem, it is possible to encounter a zero pivot. In this case, the offending row can be swapped with the row immediately below it, which doesn't spoil the $O(N)$ solution time.

The error introduced in $\mathbf{u}$ from one time step to the next is computed by comparing the fourth-order solution to the third-order solution which is also obtained from the embedded scheme. If the error is too large or the solution is unphysical ($p < 0$ for example), then the time step is reduced and the step is recalculated.

Figure 7.5 shows numerical results of a high-impact collision where a 20cm diameter steel ball traveling at 1m/s collides with a rigid wall while submerged in water at atmospheric conditions. The nondimensional parameters of this simulation are $M = 266250,\ k_1 = 1.55 \times 10^{12},\ e_{\text{dry}} = 0.97$. In order to resolve this high-impact collision, 10000 grid cells were needed. The maximum pressure during impact is 8143 atmospheres, and the minimum pressure during rebound is 0.022 atmospheres (2229.6 Pascals), which is below the cavitation pressure (2339 Pascals) of the fluid. We can see from the figure that the smallest time steps are required during object rebound - in particular, the smallest time step used was 3.5×10^{-16}, which is extremely small compared to the entire simulation length of about 0.04 time units. Adaptive time stepping is therefore essential.

7.11 Comparison To Experiments

When comparing macroscopic collision results, it is convenient to compare the wet (submerged collision) coefficient of restitution e_{wet} to the Stokes number St. Here, e_{wet} is defined to be the ratio of the post-collision velocity to the pre-collision velocity. The Stokes number represents the ratio of the sphere's inertia to the viscous strength of the fluid, and it is defined (see [18]) by

$$\text{St} = \frac{MV}{6\pi\mu R^2},$$

where M is the mass of the sphere, V is the initial velocity of the sphere, R is the radius of the sphere, and μ is the fluid viscosity at ambient conditions. In

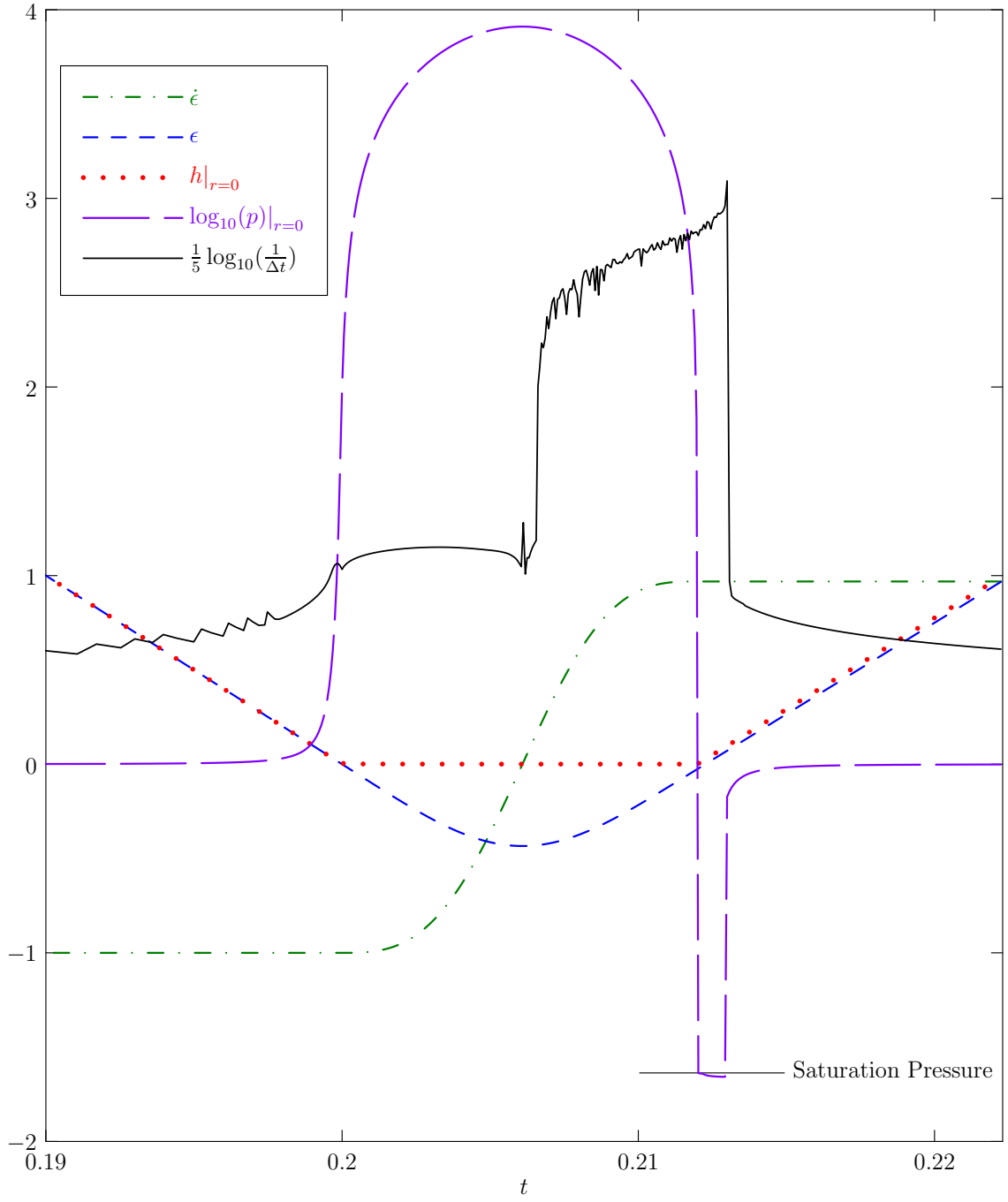

Figure 7.5: High-impact collision involving a 20cm diameter steel ball traveling underwater at 1m/s towards a rigid wall. The distances ϵ and h are measured in units of $R/100$. The pressure at $r = 0$ is measured as multiples of the atmospheric pressure, but the logarithm of this value is drawn in the graph. The saturation pressure below which the fluid vaporizes is indicated. The time step size is also indicated using a logarithm.

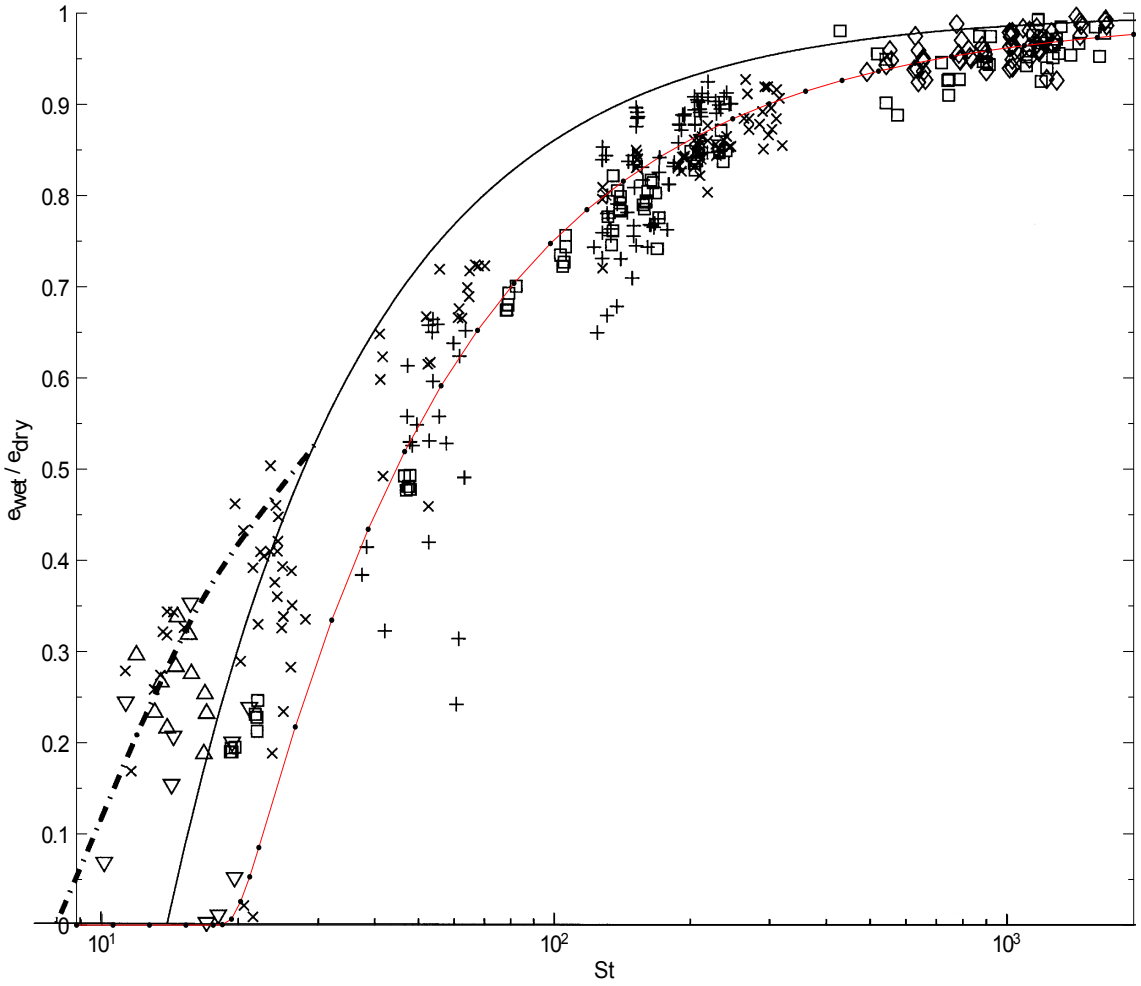

Figure 7.6: Comparison of our numerical solutions (solid red curve connecting tiny points) to experimental data for spheres of various materials impacting various walls. Also shown are two theoretical predictions by [18] (dashed line) and [8] (solid line). In our simulations, we use steel spheres with $e_{\text{dry}} = 0.97$, density 7780 kg/m^3, Young's modulus of 190 GPa, radius $R = 6.35$ mm, and water at atmospheric conditions as lubricating fluid. In order to produce a range of Stokes numbers, we vary the initial velocity of the spheres from 0.0008 to 0.2m/s^2.

our nondimensionalization from Section 7.6, the parameter M in equation (7.22) is related to the Stokes number by $\mathrm{St} = \frac{2}{\pi}M$.

Figure 7.6 is a comparison of our model predictions with all sorts of experimental results from [41, Fig. 12]. We can see that our model agrees with the experiments very well except at low Stokes numbers, where the experiments show on average a higher rebound velocity than our model predicts. One explanation for this difference is that at low Stokes number, the wall has an effect on the colliding object at larger distances, and the lubrication assumption of our model isn't valid at large distances. It is also possible that a more rigorous comparison of results should consider more parameters than just coefficient of restitution vs. Stokes number.

7.12 Comparison Of Arbitrary Parameters

There are two parameters in our model that are chosen somewhat arbitrarily – the initial distance of the object from the wall ϵ_0 (which we have been arbitrarily taking to be $0.2R$), and the domain size as a fraction of the sphere's radius (which we have been arbitrarily taking to be $0.4R$). It is therefore important to know if these two parameters affect our solution in a large way.

First we check the effect of domain size. An inspection of the pressure field during a typical numerical simulation (not shown here) indicates that all fluid properties beyond $0.3R$ are very close to ambient conditions at all times, so restricting our domain to $0.4R$ seems reasonable. Figure 7.7 shows the coefficient of restitution vs Stokes number for identical collisions but using different domain sizes. We can see that there is some effect, but not a lot. On one hand we want to choose a small domain size so the sphere's curvature doesn't violate the lubrication approximation (as explained at the beginning of this chapter), but on the other hand we want to include as much of the sphere as possible. Figure 7.7 indicates that $0.4R$ seems to be a reasonable balance between $0.3R$ and R.

Next, we check the effect of initial gap thickness ϵ_0. Figure 7.7 shows the coefficient of restitution vs Stokes number for identical collisions but using different values of ϵ_0. We can see that again, there is some effect, but not a lot. On one hand, we should choose a small ϵ_0 so the lubrication approximation is valid, but on the other hand, choosing ϵ_0 too small is surely leaving out some drag on the sphere that would happen farther away than ϵ_0. Figure 7.7 indicates that choosing $\epsilon_0 = 0.2R$

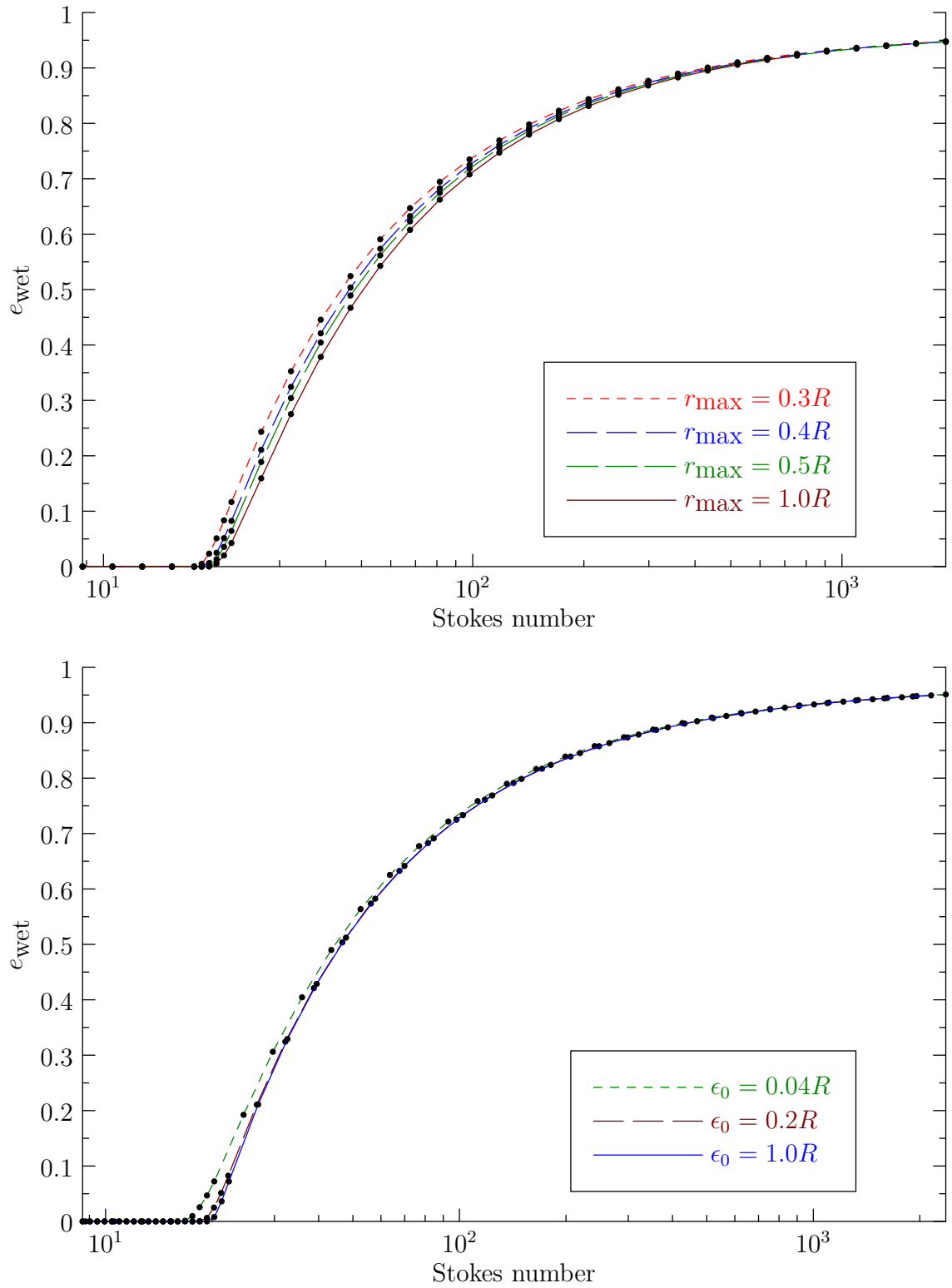

Figure 7.7: Comparison of different domain lengths (top), and different initial gap thicknesses (bottom).

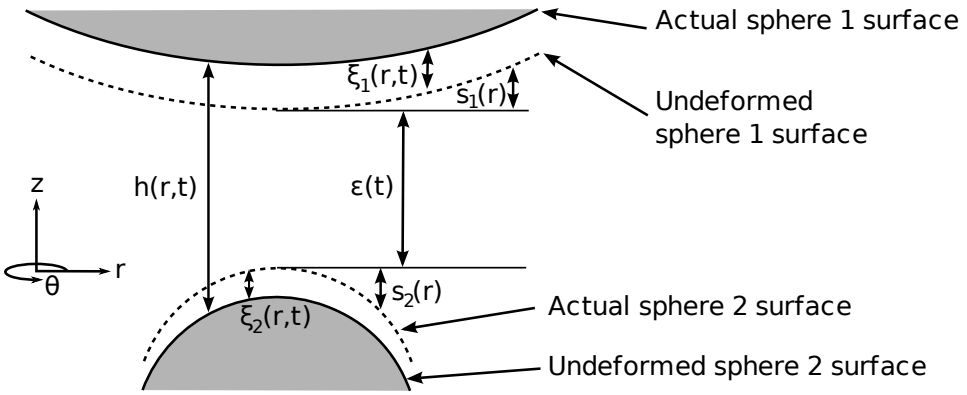

Figure 7.8: Two spheres colliding

does not leave out considerable drag, even though [41, p. 339] provides experimental evidence that a colliding sphere is affected by the wall as far away as $0.5R$. Also, if this collision model is used as a subgrid model in a larger scale code, the larger scale code would already resolve the effects that occur at large distances.

7.13 Two Sphere Collision

The lubrication collision model in this chapter can easily be extend for two spheres colliding with each other instead of a single sphere colliding with a wall. Consider Figure 7.8, where we now have two spheres with masses M_1, M_2, radii R_1, R_2, center of mass positions x_1, x_2, and spring laws ξ_1, ξ_2. In this case we have

$$s(r) = s_1(r) + s_2(r) = R_1 - \sqrt{R_1^2 - r^2} + R_2 - \sqrt{R_2^2 - r^2},$$

$$\xi = \xi_1 + \xi_2 = \begin{cases} \frac{p}{k_{1,1}} + \frac{p}{k_{1,2}} & \text{if } p \geq p_{\max} \\ \left(\frac{p - p_{\max}}{k_{2,1}} + \frac{p_{\max}}{k_{1,1}}\right) + \left(\frac{p - p_{\max}}{k_{2,2}} + \frac{p_{\max}}{k_{1,2}}\right) & \text{if } p < p_{\max} \end{cases},$$

where $k_{1,i} = \frac{E_i}{R_i}$, $k_{2,i} = \frac{k_{1,i}}{e_{\text{dry},i}^2}$, E_i is Young's modulus for the material in sphere i, and $e_{\text{dry},i}$ is the experimental coefficient of restitution for sphere i bouncing off a rigid wall. With these definitions, $h(r,t) = \epsilon(t) + s(r) + \xi(r,t)$ as before. Newton's second

law for two spheres traveling directly toward or away from each other is

$$\frac{\partial^2 x_1}{\partial t^2} = \frac{1}{M_1} \iint (p - p_0)\, dA, \qquad \frac{\partial^2 x_2}{\partial t^2} = -\frac{1}{M_2} \iint (p - p_0)\, dA.$$

Since $\epsilon = |x_1 - x_2| - R_1 - R_2$, this gives us

$$\frac{\partial^2 \epsilon}{\partial t^2} = \left(\frac{1}{M_1} + \frac{1}{M_2} \right) \iint (p - p_0)\, dA.$$

The lubrication equation (7.5) and the two equations of state $\rho = \rho(p)$, $\mu = \mu(\rho)$ are the same as in the sphere-wall case. Our initial conditions for two spheres are also the same, ϵ_0 being the initial distance from one sphere's surface to the other sphere's surface, and $\dot{\epsilon_0} = |V_1 - V_2|$ being the relative speed at which the two spheres are approaching each other along the line connecting them. We can see that if $M_2 \to \infty$, $R_2 \to \infty$, and $\xi_2 \to 0$ then we recover our original sphere-wall equations.

Chapter 8

A Direction Splitting Fictitious Domain Method

In Chapter 6, we provided a discretization of the rigid object ODEs that when coupled with the Navier–Stokes solver in Section 4.4, forms a complete scheme for solving the coupled problem of fluid flows containing rigid objects. In this section, we provide an alternative Fictitious Domain Method (FDM) for solving the coupled problem that we will compare with the more "direct" method. The FDM that we construct below is similar to the FDM of [68], however, here we use the direction splitting Navier–Stokes solver in Section 4.4, and we use finite differences instead of the finite element approach of [68].

The idea behind all FDM is to embed a complicated fluid domain Ω_f inside a simple box domain Ω, and then treat the entire domain Ω as fluid subject to rigid motion constraints in the "solid" parts of the domain $\Omega \setminus \Omega_f$. This can be justified briefly in the following way. Using Gauss's divergence theorem to rewrite the surface integral in the object linear momentum equation (2.24) as a volume integral, we obtain

$$\frac{d\mathbf{U}_i}{dt} = \frac{1}{\mathrm{Fr}}\left(1 - \frac{\rho_f}{\rho_i}\right)\mathbf{e}_g + \frac{1}{V_i}\frac{\rho_f}{\rho_i}\int_{\Omega_i} \nabla \cdot \boldsymbol{\sigma}\, d\mathbf{x}, \tag{8.1}$$

where we have used the definition $M_i = V_i\frac{\rho_i}{\rho_f}$, where V_i is the volume of Ω_i. Note: M_i is $\tilde{M}_i$ from (2.14), where we have since dropped the tilde. The divergence of the

stress can be written as

$$\begin{aligned}\nabla \cdot \boldsymbol{\sigma} &= \nabla \cdot \left[-p\boldsymbol{\delta} + \frac{1}{\mathrm{Re}} \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T \right) \right] \\ &= -\nabla p \cdot \boldsymbol{\delta} - p \nabla \cdot \boldsymbol{\delta} + \frac{1}{\mathrm{Re}} \Big(\nabla \cdot (\nabla \mathbf{u}) + \nabla \left(\nabla \cdot \mathbf{u} \right) \Big) \\ &= -\nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u}, \end{aligned} \tag{8.2}$$

since $\boldsymbol{\delta}$ is a constant and $\nabla \cdot \mathbf{u} = 0$ from the incompressibility constraint. In the Lagrangian (object-following) reference frame, the object velocity $\mathbf{U}_i$ is always equal to the volume-averaged pointwise velocity in the object domain Ω_i. Similarly, in the Eulerian (non-object following) reference frame, the velocity field $\mathbf{u}$ inside the object domain Ω_i must satisfy

$$\frac{1}{V_i} \int\limits_{\Omega_i} \frac{D\mathbf{u}}{Dt} \, d\mathbf{x} = \frac{d\mathbf{U}_i}{dt}, \quad \text{where} \quad \frac{D\mathbf{u}}{Dt} = \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}. \tag{8.3}$$

By combining (8.1), (8.2), (8.3), we have

$$\frac{1}{V_i} \int\limits_{\Omega_i} \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) \, d\mathbf{x} = \frac{1}{\mathrm{Fr}} \left(1 - \frac{\rho_f}{\rho_i} \right) \mathbf{e}_g + \frac{1}{V_i} \frac{\rho_f}{\rho_i} \int\limits_{\Omega_i} -\nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u} \, d\mathbf{x}, \tag{8.4}$$

which is exactly the volume averaged Navier–Stokes equations (2.22) with a density scaling factor on the force terms and an additional buoyancy term. A similar volume-averaged equation holds for the angular momentum of the object. Therefore, the Navier–Stokes equations can be solved everywhere (even in $\Omega \setminus \Omega_f$), and then volume averages can be used to recover the original object motion equations. For a more complete derivation of the FDM, see [68].

The FDM numerical scheme (assuming no object collisions) can be summarized as follows:

Step 1: Update object positions according to

$$\frac{\mathbf{X}_i^{n+1} - \mathbf{X}_i^n}{\Delta t} = \frac{3}{2} \mathbf{U}_i^n - \frac{1}{2} \mathbf{U}_i^{n-1}, \tag{8.5}$$

where $\mathbf{X}_i$ is the centroid position of the i^{th} object and $\mathbf{U}_i$ is its velocity.

Step 2: Predict the pressure in Ω using (4.104).
Step 3: Solve the Navier–Stokes momentum equation in Ω using (4.105) or its original Douglas splitting equivalent. No boundary fitted spatial operators are used since Ω_f is a box domain. Since the Navier–Stokes equations are solved with constant density and viscosity, the original Douglas splitting (4.63) will be stable and is actually preferred over the modified Douglas splitting (4.84) in this specialized case due to reduced memory requirements, reduced CPU usage, and reduced splitting error. However, the modified Douglas splitting also works.
Step 4: Compute the average linear and angular velocities $\mathbf{u}_{\text{avg}}$, $\boldsymbol{\omega}_{\text{avg}}$ of the fluid inside each Ω_i,

$$V_i \mathbf{u}_{\text{avg},i}^{n+1} = \int_{\Omega_i} \mathbf{u}^{n+1} \, d\mathbf{x}, \tag{8.6}$$

$$\mathbf{I}_{v,i} \, \boldsymbol{\omega}_{\text{avg},i}^{n+1} = \int_{\Omega_i} \left(\mathbf{x} - \mathbf{X}_i^{n+1}\right) \times \left(\mathbf{u}^{n+1} - \mathbf{u}_{\text{avg},i}^{n+1}\right) \, d\mathbf{x}, \tag{8.7}$$

where V_i is the volume of Ω_i and $\mathbf{I}_{v,i} = \frac{2}{5} V_i r_i^2 \boldsymbol{\delta}$ is the volume moment. On a uniform MAC grid, an example discretization of the x component of (8.6) and of the z component of (8.7) is

$$\left(\sum_{\mathbf{x}_{n,i}} 1 \right) \; u_{\text{avg},i} = \sum_{\mathbf{x}_{n,i}} u, \tag{8.8}$$

$$\left(\sum_{\mathbf{x}_{m,i}} x^2 + \sum_{\mathbf{x}_{n,i}} y^2 \right) \; \omega_{\text{avg},i} = \sum_{\mathbf{x}_{m,i}} x(v - v_{\text{avg},i}) - \sum_{\mathbf{x}_{n,i}} y(u - u_{\text{avg},i}), \tag{8.9}$$

where $\mathbf{x}_{n,i}$ and $\mathbf{x}_{m,i}$ are respectively all u and v MAC grid points that lie inside Ω_i, and the coordinates x, y are measured relative to the object's centroid $\mathbf{X}_i^{n+1}$.
Step 5: Update each object velocity $\mathbf{U}_i$ and angular velocity $\boldsymbol{\omega}_i$ such that object i receives a net change in momentum equal to the net change in average fluid momentum in Ω_i over the same time interval. This is accomplished using

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n + \frac{\rho_f}{\rho_i} \left(\mathbf{u}_{\text{avg},i}^{n+1} - \mathbf{U}_i^n\right) + \frac{\Delta t}{\text{Fr}} \left(1 - \frac{\rho_f}{\rho_i}\right) \mathbf{e}_g, \tag{8.10}$$

$$\boldsymbol{\omega}_i^{n+1} = \boldsymbol{\omega}_i^n + \frac{\rho_f}{\rho_i}\left(\boldsymbol{\omega}_{\text{avg},i}^{n+1} - \boldsymbol{\omega}_i^n\right). \tag{8.11}$$

Step 6: Prescribe the fluid velocity field inside each Ω_i to be the rigid motion defined by $\mathbf{U}_i^{n+1}$ and $\boldsymbol{\omega}_i^{n+1}$,

$$\mathbf{u}^{n+1} = \mathbf{U}_i^{n+1} + \boldsymbol{\omega}_i^{n+1} \times \left(\mathbf{x} - \mathbf{X}_i^{n+1}\right) \text{ for all } \mathbf{x} \in \Omega_i. \tag{8.12}$$

In the context of the MAC grid, this means that whenever a u grid node lies inside Ω_i, the u component of (8.12) is prescribed there. A similar procedure is done for the v and w grid nodes.

Step 7: Solve the incompressibility penalty step exactly as in (4.106).

Step 8: Perform the pressure update exactly as in (4.107).

This completes the loop of time discretization. The FDM presented above (with $\chi = 0$ and using the divergence scaling factor $\eta_{\min}$ given by (6.7)) is unconditionally stable for the unsteady Stokes equations. If the original Douglas splitting is used, then the FDM also appears to be unconditionally stable for $\chi = 1$. In either case, the rigid motion constraint (Step 6) must be performed after solving the momentum equation (Step 3) but before computing the divergence in Step 7. Enforcing Step 6 after computing the divergence in Step 7 produces an unconditionally unstable scheme.

As explained by Patankar [53], a FDM which imposes the rigid constraint explicitly (as in Steps 4-6 above) should ideally perform iterations of Steps 2–8 in order to avoid error in the form of "slip" at the fluid-solid interface, particularly when ρ_f/ρ_i is not close to unity. This error appears because the end-of-step velocity field is the result of applying the discontinuous operation (8.12) to selected areas of an otherwise smooth solution that comes from solving the momentum equation. To perform iterations that converge to the proper "no-slip" condition, one needs to store the correction $\mathbf{u}_{\text{Step 6}} - \mathbf{u}_{\text{Step 3}}$, where $\mathbf{u}_{\text{Step 6}}$ is the "end-of-step" velocity field produced after Step 6 and $\mathbf{u}_{\text{Step 3}}$ is the "intermediate" velocity field produced after Step 3. This correction then needs to be applied in the momentum equation on the next iteration, with the appropriate density scaling. Since we do not perform iterations of this type, a small time step is required in order to reduce this "slip" error. In the limit as $\Delta t \to 0$, this error disappears completely. Because the divergence is

computed on the end-of-step field which contains the "slip" error, the magnitude of the discrete divergence is poorly controlled at the fluid-solid boundary. Numerical results (see Section 9.5) confirm this effect.

8.1 FDM Convergence Rates

In this section, we test the spatial and temporal convergence rates of the FDM by solving Navier–Stokes for the Taylor–Couette problem in Appendix A with $R_1 = 0.25$, $R_2 = 0.5$, $\omega_1 = 1$, $\omega_2 = -1$, $\mathrm{Re} = 1$, $\chi = 1$. This means that the FDM must enforce the rigid motion constraint in both the inner and outer cylinders. To test spatial convergence, we use various uniform grid resolutions with $\Delta t = 10^{-7}$, and we run each simulation until a numerical steady state has been reached. The error presented in Table 8.1 shows that the velocity is $O(h^{0.8})$ accurate in the L^2 norm and $O(h^{0.64})$ accurate in the maximum norm, and the pressure is $O(h^{0.6})$ accurate in the L^2 norm but does not converge at all in the maximum norm (due to "slip" error transferred from the divergence, as explained in the previous section). The temporal convergence is shown in Table 8.2 using the same Taylor–Couette simulation with $h = 1/320$ and various time step sizes, again running each simulation until a steady state has been reached. The velocity is $O(\Delta t^{0.55})$ accurate in the L^2 norm and $O(\Delta t^{0.75})$ accurate in the maximum norm. The pressure is $O(\Delta t^{0.85})$ accurate in both norms. Since all of these convergence rates are relatively poor, it could be worthwhile to improve the spatial accuracy of the volume integrals (8.8), (8.9) by using local grid refinement as in in [62], and also reduce the "slip" error by performing iterations as in [6]. However, the FDM as it exists here is still consistent and can be used for comparison with other methods.

8.2 Momentum Conservation

Numerical testing indicates that the FDM performs significantly better when momentum is accurately conserved by Steps 4,5,6 (given by equations (8.6)–(8.12)). We now describe what we mean by this "momentum conservation" property. First, assume for simplicity that $\rho_f = \rho_i$ so that Step 5 simplifies to $\mathbf{U}_i^{n+1} = \mathbf{u}_{\mathrm{avg},i}^{n+1}$, $\boldsymbol{\omega}_i^{n+1} = \boldsymbol{\omega}_{\mathrm{avg},i}^{n+1}$,

h	$\|u - u_{\text{exact}}\|_{L^2(\Omega)}$	rate	$\|u - u_{\text{exact}}\|_{L^\infty(\Omega)}$	rate	$\|\nabla \cdot \mathbf{u}\|_{L^\infty(\Omega)}$
1/10	2.465e-02		6.569e-02		2.498e-15
1/20	1.722e-02	0.52	4.967e-02	0.40	6.217e-15
1/40	9.435e-03	0.87	3.357e-02	0.56	1.749e-14
1/80	6.519e-03	0.53	2.527e-02	0.41	1.332e-14
1/160	2.855e-03	1.19	1.254e-02	1.01	1.776e-14
1/320	1.683e-03	0.76	7.041e-03	0.83	4.139e-12

h	$\|p - p_{\text{exact}}\|_{L^2(\Omega_f)}$	rate	$\|p - p_{\text{exact}}\|_{L^\infty(\Omega_f)}$	rate
1/10	1.3150e+00		5.777e+00	
1/20	7.5131e-01	0.81	4.563e+00	0.34
1/40	5.8171e-01	0.37	5.190e+00	-0.19
1/80	3.1941e-01	0.86	4.651e+00	0.16
1/160	3.1910e-01	0.00	8.051e+00	-0.79
1/320	1.8079e-01	0.82	6.750e+00	0.25

Table 8.1: Spatial convergence of the FDM when solving the Navier–Stokes equations for the Taylor–Couette flow using $\text{Re} = 1$ and $\Delta t = 10^{-7}$. Each simulation was run long enough to reach a strict numerical steady state. Since $\chi = 1$, the divergence is zero to floating point precision at steady state.

and Step 6 simplifies to

$$\mathbf{u}_{\text{Step 6}}^{n+1} = \mathbf{u}_{\text{avg},i}^{n+1} + \boldsymbol{\omega}_{\text{avg},i}^{n+1} \times \left(\mathbf{x} - \mathbf{X}_i^{n+1}\right) \text{ for all } \mathbf{x} \in \Omega_i. \tag{8.13}$$

Again we distinguish $\mathbf{u}_{\text{Step 6}}^{n+1}$ to be the final velocity field after Step 6 is applied and $\mathbf{u}_{\text{Step 3}}^{n+1}$ is the intermediate velocity field before Step 3 is applied. Define $\left(\overline{\mathbf{u}}_i, \overline{\boldsymbol{\omega}}_i\right)(\mathbf{u})$ to be the "Step 4" operator which accepts a velocity field $\mathbf{u}$ as input and computes average the momentum values $\mathbf{u}_{\text{avg},i}$, $\boldsymbol{\omega}_{\text{avg},i}$ as output. Then the "momentum conservation" property is the following:

$$\overline{\mathbf{u}}_i(\mathbf{u}_{\text{Step 3}}^{n+1}) = \overline{\mathbf{u}}_i(\mathbf{u}_{\text{Step 6}}^{n+1}) \tag{8.14}$$

$$\overline{\boldsymbol{\omega}}_i(\mathbf{u}_{\text{Step 3}}^{n+1}) = \overline{\boldsymbol{\omega}}_i(\mathbf{u}_{\text{Step 6}}^{n+1}) \tag{8.15}$$

In other words, the application of Step 6 to the velocity field should not change the volume averaged momentum as computed by Step 4. This property is always true

in the limit $h \to 0$, however, the overall accuracy of the FDM can be significantly improved by simply improving the accuracy to which this property is satisfied when the spatial resolution is finite.

It can be shown that grid-pointwise imposition of (8.12) paired with (8.8) satisfies (8.14) to floating point precision. However, (8.9) satisfies (8.15) exactly only when the object's centroid $\mathbf{X}_i$ is also the discrete centroid of all grid points contained within the object. For a uniform MAC grid, this is only true when each coordinate of $\mathbf{X}_i$ independently lies on MAC cell faces or MAC cell centers.

To test the "discrete momentum conservation" idea of this section, we can discretize the *position* of the objects so that

$$\mathbf{X}_{\text{discrete}} \in \left\{ (x, y, z) \;\middle|\; x = N_1 \frac{h}{2},\ y = N_2 \frac{h}{2},\ z = N_3 \frac{h}{2} \right\}, \tag{8.16}$$

where N_1, N_2, N_3 are integers and h is the MAC cell width. For the equation of motion (8.5), we still use the real (non-discrete) position $\mathbf{X}_i$. However, for imposition of the rigid constraints (8.6)–(8.12), we use the discretized position $\mathbf{X}_{\text{discrete}}$, which is defined to be the closest discretized position to the real position. Discretizing the position may sound weird, but many numerical schemes do this already without calling it as such. For example, any scheme which uses a uniform grid and does not perform boundary-fitting essentially discretizes the object boundary to a set of grid-aligned positions. In either case, the schemes are consistent because when $h \to 0$, the discrete position approaches a real-valued position.

In this thesis, the FDM which is used by default is the regular (not position-discretized) version. The position-discretized FDM is only used in Section 9.3 where it is compared to the regular FDM.

Δt	$\|u - u_{\text{exact}}\|_{L^2(\Omega)}$	rate	$\|u - u_{\text{exact}}\|_{L^\infty(\Omega)}$	rate
1/200	1.237e-01		1.659e+00	
1/400	8.033e-02	0.62	9.027e-01	0.88
1/800	5.351e-02	0.59	4.973e-01	0.86
1/1600	3.635e-02	0.56	2.748e-01	0.86
1/3200	2.496e-02	0.54	1.546e-01	0.83
1/6400	1.721e-02	0.54	9.170e-02	0.75
1/12800	1.186e-02	0.54	5.673e-02	0.69
1/25600	8.184e-03	0.54	3.587e-02	0.66
1/51200	5.684e-03	0.53	2.321e-02	0.63
1/102400	4.036e-03	0.49	1.566e-02	0.57
1/204800	2.997e-03	0.43	1.129e-02	0.47

Δt	$\|p - p_{\text{exact}}\|_{L^2(\Omega_f)}$	rate	$\|p - p_{\text{exact}}\|_{L^\infty(\Omega_f)}$	rate
1/200	6.603e+01		1.901e+03	
1/400	3.572e+01	0.89	1.000e+03	0.93
1/800	1.903e+01	0.91	5.310e+02	0.91
1/1600	1.005e+01	0.92	2.770e+02	0.94
1/3200	5.290e+00	0.93	1.456e+02	0.93
1/6400	2.804e+00	0.92	7.805e+01	0.90
1/12800	1.519e+00	0.88	4.301e+01	0.86
1/25600	8.600e-01	0.82	2.609e+01	0.72
1/51200	5.238e-01	0.72	1.723e+01	0.60
1/102400	3.530e-01	0.57	1.235e+01	0.48
1/204800	2.666e-01	0.41	9.684e+00	0.35

Table 8.2: Temporal convergence of the FDM when solving Navier–Stokes for the Taylor–Couette flow using $\text{Re} = 1$, $\chi = 1$, and $h = 1/320$. Each simulation was run long enough to reach a strict numerical steady state. In this case, the divergence is zero to floating point precision so it is not shown.

Chapter 9

Validation Of Schemes On Realistic Flows

In this chapter, we validate and compare the two numerical schemes in this thesis by applying them to realistic fluid problems that contain solid objects in the flow. The first scheme is the "direct scheme with boundary fitting" which solves Navier–Stokes only in the fluid domain Ω_f using boundary-fitted spatial operators and Dirichlet boundary conditions on $\partial\Omega_f$, and the fluid forces on each solid object are computed directly using surface integrals as in Chapter 6. The second scheme is the FDM of Chapter 8, which solves Navier–Stokes in the entire extended domain Ω and imposes rigid motion within the "solid" domains Ω_i by using volume averages of the fluid quantities. Both schemes use the direction splitting Navier–Stokes solver in Section 4.4. Since the convergence rates of the boundary fitting scheme in Section 5.3 are much better than the FDM convergence rates in Section 8.1, we expect the boundary fitting scheme to also outperform the FDM on realistic flow problems. As a benchmark, numerical and experimental results of other independent authors are also included.

9.1 Sedimentation Of A Single 3D Ball, Re=4.1

In this section, we use the numerical schemes in this thesis to compute the terminal velocity of a single 3D ball as it falls in a rectangular tank. The results of such a situation are well documented both experimentally and numerically in the literature

(see [12]). The parameters that we use are exactly the same as in [12]: $\mathrm{Re} = 4.1, \mathrm{Fr} = 0.024$, and the ball has a density ratio $\rho_{\mathrm{ball}}/\rho_f = 1.16$. Using the diameter of the ball as a characteristic length, the size of the container is $20/3 \times 20/3 \times 32/3$ and the initial position of the ball is (10/3, 10/3, 8). The terminal velocity of the ball in an infinite fluid is used as a characteristic velocity.

In addition to the two numerical methods in this thesis, we also compare against results computed with a finite element fictitious domain method (FE-FDM) which does not use any direction splitting. The FE-FDM (as described in [68]) employs $P_2 - P_1$ finite elements to discretize the flow equations, so for a pure flow problem (without solid objects), the FE-FDM is expected to be third-order accurate in space. This accuracy comes at a cost in both CPU time and memory requirements, and we were not able to easily run the FE-FDM code on spatial grids that required more than 24GB of memory (Note the "N/A" entries in Table 9.1). The volume integrals of the FE-FDM are performed using high-order Gauss quadratures. The FE-FDM uses a mesh of tetrahedral elements that is formed by subdividing a uniform grid of hexahedral cubes into five tetrahedra per cube. Therefore, the number of grid nodes for the velocity in a given direction is equal to twice the number of hexahedral cubes in this direction, plus one. We consider h to be the minimum distance between two grid nodes when performing comparisons with MAC discretizations.

The numerical results for the maximum instantaneous velocity of the falling ball using all three methods are presented in table 9.1. The "direct scheme with boundary fitting" is the clear winner for fast convergence, particularly when three-point stencils are used for the velocity derivatives in the fluid stress (see Section 6.3). Even on the coarsest grid which has only 3.75 MAC cells across the ball diameter, the "direct scheme with boundary fitting" still correctly computes the terminal velocity to within 5% error. The various FDM schemes all perform similarly, though the FE-FDM of [68] does not suffer as much as the FDM in this thesis when Δt is large. In terms of CPU and memory efficiency, the "direct scheme with boundary fitting" requires about twice as much CPU and memory as the FDM of this thesis, however it performs significantly better than the FDM even if the FDM uses half the time step size. In either case, both of the methods in this thesis require less CPU and less memory than the FE-FDM because of direction splitting. The experimentally measured value according to [12] is 0.953, which is only 1.8% different from 0.97, our most accurate numerical result. Since the velocity we report is the maximum *instantaneous* velocity over the entire simulation, this is very good agreement.

Since the rotational scheme (see Section 4.1.3) primarily corrects error around fluid domain boundaries, the value of χ has a significant effect on the accuracy of the surface integral and therefore the velocity of the falling ball. All of the simulations in table 9.1 use $\chi = 1$ except for the FE-FDM. Consider the entry in table 9.1 under column MD3 and row $h = 8/60$. If we run the identical simulation again with $\chi = 0$, we get 0.943 instead of 0.990. However, running yet again with $\chi = 0$ but using a smaller $\Delta t = 1/600$ instead of $\Delta t = 1/60$, we get 0.985 and recover almost all of the error from using $\chi = 0$.

h	Δt	FE-FDM	FDM	FDM$^{\Delta t}$	BDF1	BDF1$^{\Delta t}$	MD2	MD3
16/60	1/30	1.167	1.308	1.212	0.988	1.003	1.140	1.000
8/60	1/60	1.084	1.132	1.059	0.981	0.993	1.057	0.990
4/60	1/120	1.057	1.072	1.018	0.962	0.975	1.011	0.974
2/60	1/240	N/A	1.038	0.995	0.961	0.970	0.990	0.971
1/60	1/480	N/A	1.017	0.984	0.960	0.969	0.980	0.970

Table 9.1: Maximum vertical speed of a 3D sedimenting ball.
The FE-FDM column presents the results of the FE-FDM of [68].
The FDM column presents the results of the FDM in Chapter 8 of this thesis.
The FDM$^{\Delta t}$ column uses the identical method as the FDM column but the time step is 16 times smaller than the listed value. For example, $\Delta t = 1/7680$ in the last row.
The BDF1 and BDF1$^{\Delta t}$ columns present the results of the "direct scheme with boundary fitting" using BDF1 splitting as in Section 4.2.1 and using three-point stencils for the velocity derivatives in the fluid stress as in Section 6.3.
The BDF1$^{\Delta t}$ column uses a time step 10 times smaller than the listed value.
The MD2 and MD3 columns present the results of the "direct scheme with boundary fitting" using modified Douglas splitting as in Section 4.2.3 and either two-point stencils (MD2) or three-point stencils (MD3) for the fluid stress.
The experimentally measured velocity according to [12] is 0.953.

9.2 Migration Of A Ball In A 3D Pipe, Re $\approx$ 26.7

As discovered experimentally by [61], a neutrally buoyant ball in a 3D Poiseuille flow (see Figure 9.1) tends to migrate to an equilibrium position approximately 0.6 cylinder radii away from the axis of the cylinder where the fluid forces on the ball

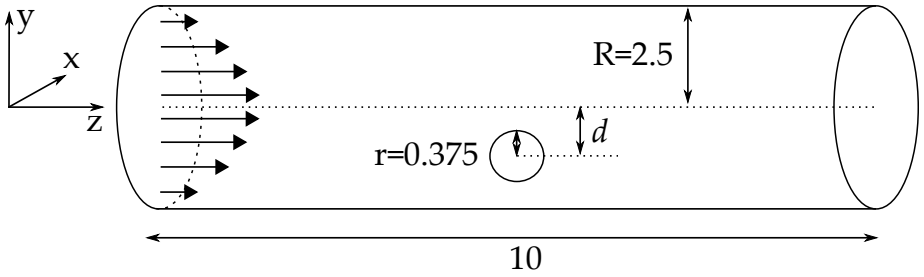

Figure 9.1: Ball in a 3D Poiseuille pipe flow.

are balanced. This flow is also well documented numerically (see [52], [73], [68]) and therefore we use it as another validation test in 3D. We consider the same setting as in [52]: a cylindrical pipe of radius 2.5 and length 10 contains a fluid with viscosity $\mu = 1$ and density $\rho_f = 1$, and the neutrally buoyant ball in the pipe has radius 0.375, as in Figure 9.1. A parabolic profile with maximum velocity of 20 is prescribed at the inlet of the pipe, and the approximately stress free condition $p = 0$ and $\frac{\partial \mathbf{u}}{\partial z} = 0$ is prescribed at the outlet. Based on the diameter of the ball and the maximum flow velocity, the Reynolds number is 80/3. In order to avoid the ball exiting the computation domain, we use a "ball following" coordinate system given by

$$\tilde{\mathbf{x}} = \mathbf{x} - \left(0, 0, \int_0^t C(s)\, ds\right), \tag{9.1}$$

where $C(t) = \frac{dz}{dt}$ is the z component of the velocity of the ball. In the new coordinate system,

$$\frac{d\tilde{z}}{dt} = \frac{dz}{dt} - \frac{d}{dt}\int_0^t C(s)\, ds = C(t) - C(t) = 0 \tag{9.2}$$

so the ball has fixed $\tilde{z}$ coordinate. The velocity time derivative appearing in the Navier–Stokes momentum equation (2.22) must be transformed according to

$$\left.\frac{\partial \mathbf{u}(\mathbf{x},t)}{\partial t}\right|_{\mathbf{x}=\text{const}} = \left.\frac{\partial \tilde{\mathbf{u}}(\tilde{\mathbf{x}},t)}{\partial t}\right|_{\mathbf{x}=\text{const}} = \left.\frac{\partial \tilde{\mathbf{u}}(\tilde{\mathbf{x}},t)}{\partial t}\right|_{\tilde{\mathbf{x}}=\text{const}} + \left(\left.\frac{\partial \tilde{\mathbf{x}}}{\partial t}\right|_{\mathbf{x}=\text{const}} \cdot \tilde{\nabla}\right) \tilde{\mathbf{u}}(\tilde{\mathbf{x}},t),$$

where $\frac{\partial \tilde{\mathbf{x}}}{\partial t}\big|_{\mathbf{x}=\text{const}} = (0, 0, -C(t))$. With this transformation, the momentum equation written in the $\tilde{\mathbf{x}}$ coordinates (and dropping the tilde) becomes

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} - (0, 0, C(t))) \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{\text{Re}} \nabla^2 \mathbf{u}. \tag{9.3}$$

Two different numerical simulations are carried out - one with the ball initially located at radial position $d/R = 0.2$, and one with the initial position $d/R = 0.75$. Because the proper initial condition for the fluid is unknown, the ball's position is fixed for a short time at the beginning of the simulation so that the parabolic flow profile has time to adjust to the new geometry that includes the ball. Figure 9.2 shows the time evolution of the ball's position d/R. Table 9.2 shows the detailed results of our simulations using the "direct scheme with boundary fitting", and Table 9.3 shows the results of other authors using a variety of methods. Even at coarse resolutions, our results agree with the results of other authors, all of which predict an equilibrium position of about $d/R = 0.6$. Our higher resolution results agree very well with the Arbitrary Lagrangian-Eulerian method of [73] which the authors of [52] use as their benchmark result.

9.3 Migration Of A Disc In A Channel, Re $\approx$ 26.7

In this section, we perform the 2D analog of the test in Section 9.2, i.e., we find the equilibrium position of a migrating disc in a channel rather than a ball in a pipe. All parameters are identical to those in Section 9.2 other than the dimension of the problem and the initial release positions of the disc, which are now $d_0/R = 0.4$ and $d_0/R = 0.5$. From numerical testing, the disc should migrate towards an equilibrium position of about $d/R = 0.45$. The purpose of this test is to compare the convergence properties of some variants of the methods in this thesis: The "direct scheme with boundary fitting" using the interpolation surface integral of Section 6.3, the "direct scheme with boundary fitting" using the interpolation-free surface integral of Section 6.4, the FDM of Chapter 8, and the FDM with the discrete momentum conserving property in Section 8.2. Figure 9.3 shows the time evolution of the disc as computed by all four methods using a high grid resolution. Figure 9.4 shows the same simulations but using a lower grid resolution. For determining the equilibrium position of the disc, we can conclude from the figures that the interpolation surface

h	Δt	CPU	d_0/R	d/R	speed	ang. vel.	$\|\nabla \cdot \mathbf{u}\|_{L^\infty(\Omega)}$
1/10	1/500	0.3	0.2	0.58897	12.640	4.5008	1.8e-04
			0.75	0.59269	12.562	4.5330	2.9e+00
1/15	1/750	0.9	0.2	0.59594	12.491	4.5912	2.1e-05
			0.75	0.59611	12.486	4.5925	1.9e-05
1/20	1/1000	2.0	0.2	0.59517	12.501	4.5870	4.6e-05
			0.75	0.59638	12.472	4.5905	3.5e-05
1/25	1/1250	4.0	0.2	0.59696	12.462	4.6213	6.8e-07
			0.75	0.59905	12.412	4.6367	1.2e-04
1/30	1/1500	7.1	0.2	0.59955	12.403	4.6338	4.1e-06
			0.75	0.60115	12.364	4.6458	1.2e-03
1/35	1/1750	11.4	0.2	0.60135	12.361	4.6594	2.8e-05
			0.75	0.60137	12.360	4.6596	4.9e-05
1/40	1/2000	17.3	0.2	0.60135	12.362	4.6531	3.7e-06
			0.75	0.60146	12.359	4.6539	4.3e-05

Table 9.2: Results for the ball in 3D Poiseuille flow. The initial position of the ball is d_0/R. The position and velocities of the ball and the divergence of the fluid are presented again at $t = 50.0$. CPU time is measured in seconds per time step using one Intel Xeon processor at 2.4 GHz.

Distributed Lagrange Multiplier FDM of [52]						
Num. Velocity Nodes	Δt	CPU	d_0/R	d/R	speed	ang. vel.
2.16×10^6 (uniform)	1/1000	74	0.75	0.60582	12.237	4.6286
			0.2	0.60579	12.235	4.6359
Arbitrary Lagrangian-Eulerian method of [73]						
Num. Velocity Nodes	Δt	CPU	d_0/R	d/R	speed	ang. vel.
1.46×10^5	N/A	N/A	0.75	0.60108	12.364	4.6513
			0.2	0.60108	12.364	4.6513
Non Lagrange Multiplier FDM of [68]						
Num. Velocity Nodes	Δt	CPU	d_0/R	d/R	speed	ang. vel.
1.07×10^6 (non uniform)	1/200	190	0.75	0.6139	12.1806	4.7790
			0.2	0.6106	12.2585	4.7574

Table 9.3: Results of other authors for the ball in 3D Poiseuille flow. CPU times are the author's published values scaled down to approximate a 2.4GHz CPU.

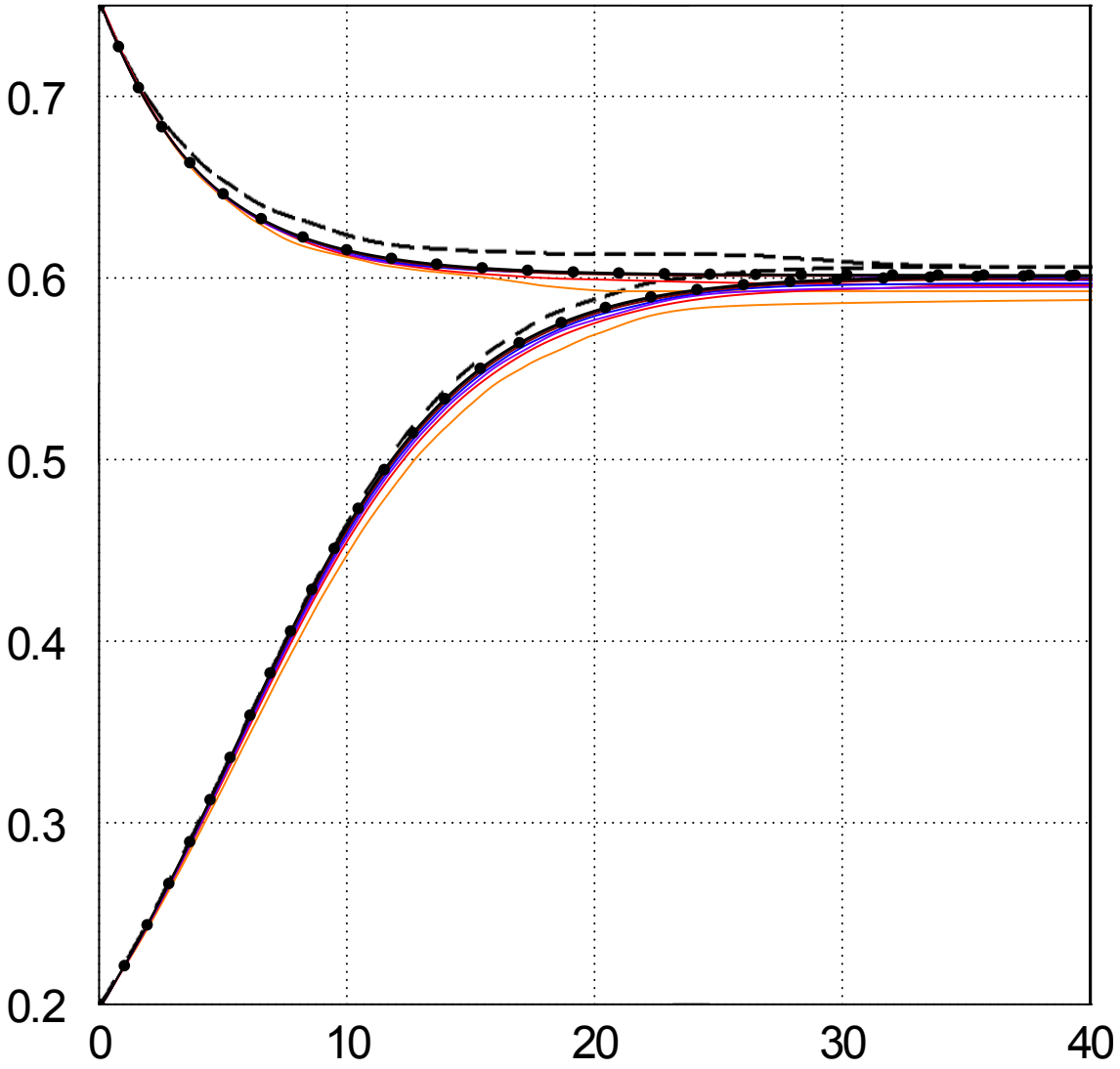

Figure 9.2: Time evolution of the ball's position d/R in the 3D Poiseuille flow. The vertical axis is d/R and the horizontal axis is time. The solid lines are our numerical simulations corresponding to Table 9.2, and the black dots indicate our $h = 1/40$ simulation. The two dashed lines are the Distributed Lagrange Multiplier results of [52] which are in approximate agreement with our results.

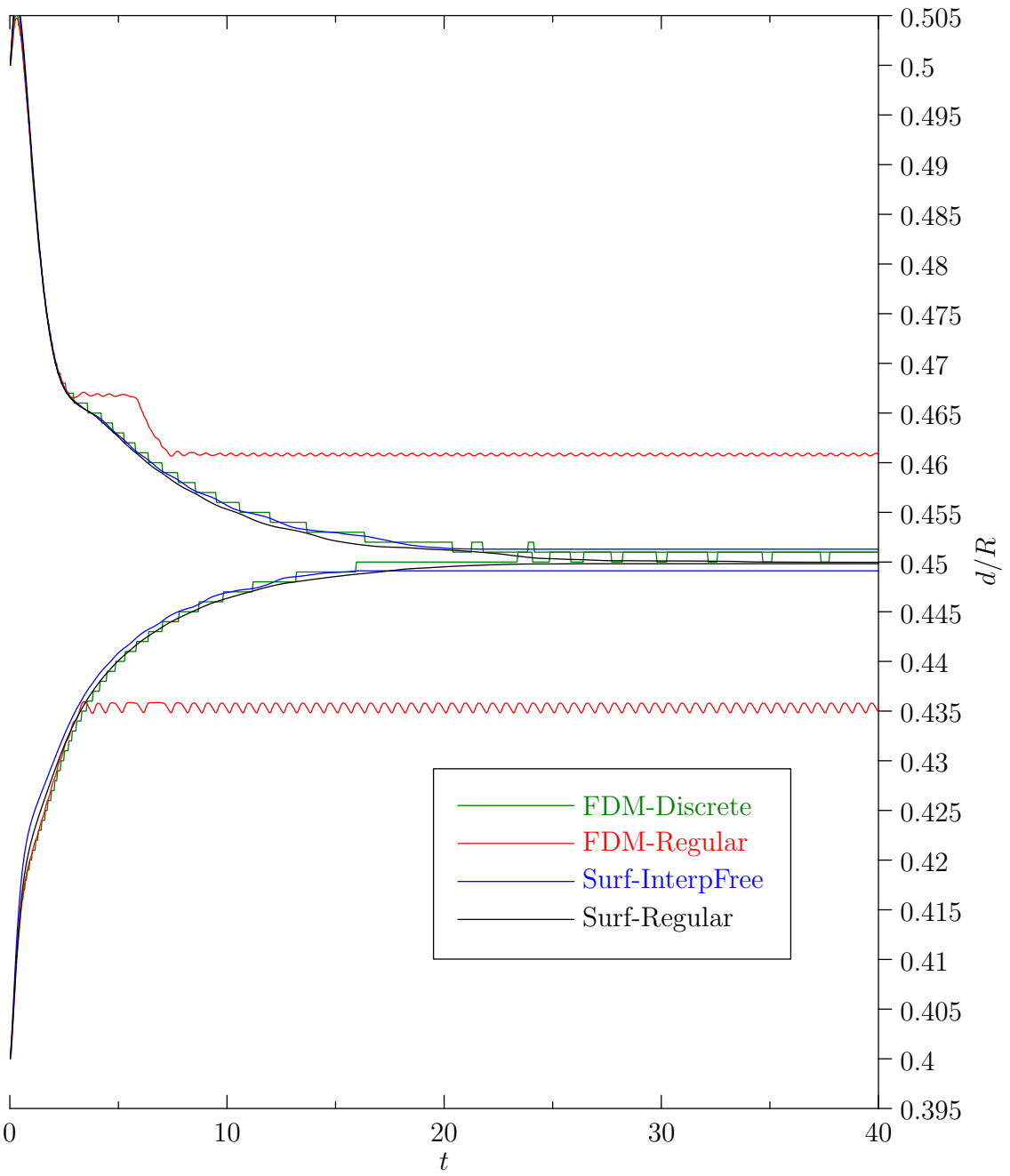

Figure 9.3: Time evolution of a disc in 2D Poiseuille flow using high resolution $h = 1/200$ and $\Delta t = 1/5000$. The regular surface integral method bounds the equilibrium position by $0.44985 \leq d/R \leq 0.44995$. The interpolation-free surface integral method only bounds the equilibrium position accurate to two decimal places. The regular FDM has significant error in comparison to both surface integral methods. The coordinate-discretized FDM shows a significant improvement over the regular FDM and seems to follow the solution of the surface integral methods. Both FDM methods have an oscillatory behaviour that comes from not fitting the boundary of the disc.

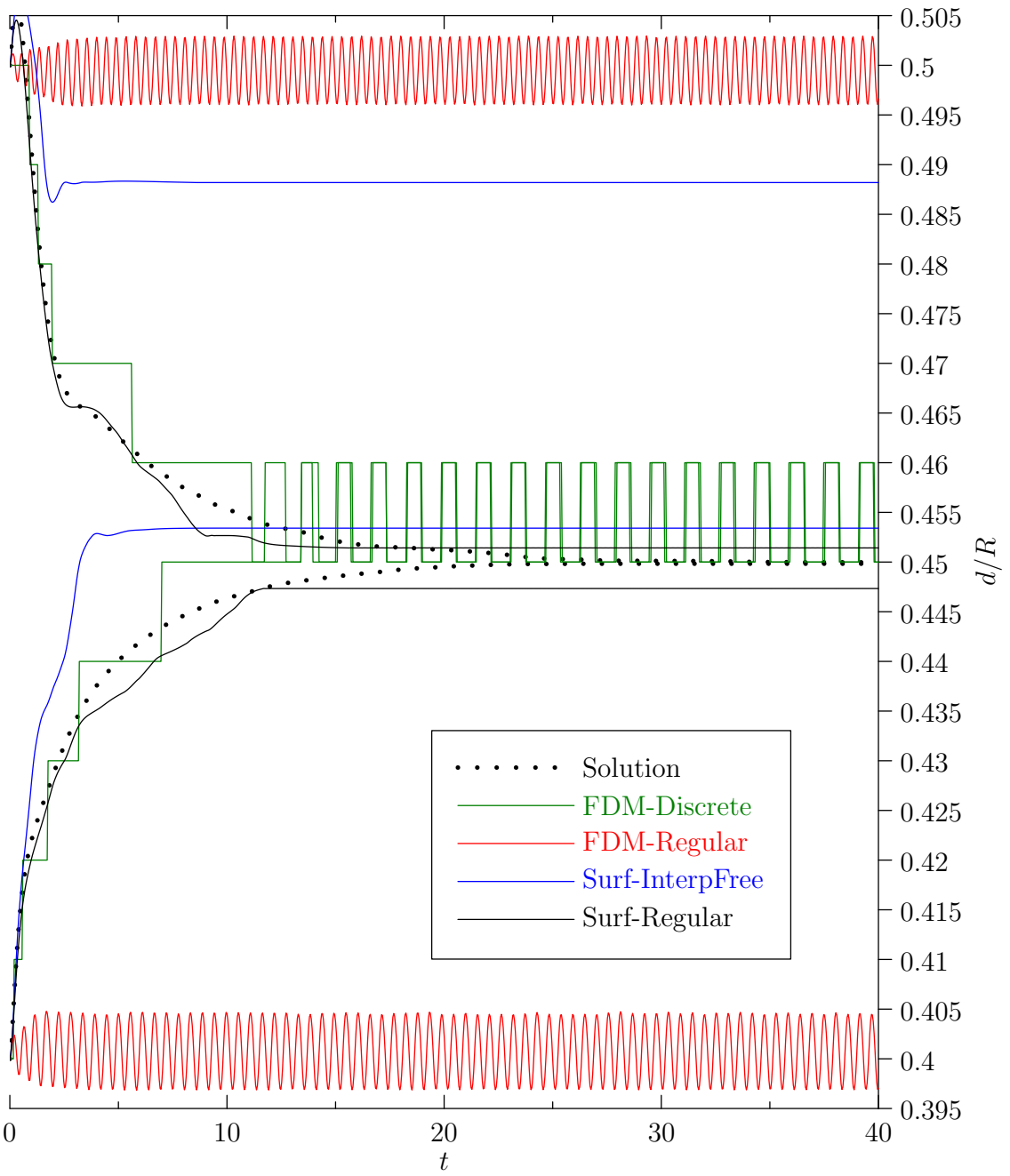

Figure 9.4: Time evolution of a disc in 2D Poiseuille flow using a lower resolution $h = 1/20$ and $\Delta t = 1/1000$. The high resolution surface integral method is used as a reference solution. The low resolution surface integral method still looks reasonably accurate. However, the interpolation-free surface integral method now has significant error especially when the disc is released from $d/R = 0.5$. The regular FDM is terrible at this resolution. The coordinate-discretized FDM is again a significant improvement over the regular FDM. Both FDM methods have oscillations of significant magnitude at this lower resolution.

integral method is clearly the most accurate, followed by the interpolation-free surface integral method and the FDM with discrete momentum conservation, and the regular FDM performs significantly worse than the other three methods.

9.4 Vortex Street Behind 2D Disc, Re=100

It is well known that the wake behind a circular cylinder produces a "von Karman vortex street". This flow has been studied extensively in the literature - see for example [21]. In this section, we validate the "direct scheme with boundary fitting" by performing the same simulation as in [21]. The domain is $[0, 32] \times [0, 16]$, and a disc of diameter 1 is fixed at the position $(8, 8.01)$, where the y position is perturbed from the center of the domain in order to quickly destabilize the flow. Dirichlet boundary conditions with velocity $(u, v) = (1, 0)$ are prescribed on the bottom, top, and left boundaries. The approximately stress free condition $p = 0$ and $\frac{\partial \mathbf{u}}{\partial x} = 0$ is prescribed at the right boundary. Based on the diameter of the disc and the velocity $u_c = 1$, the Reynolds number is 100. We simulate this flow using uniform grids of several different resolutions all of which have a maximum CFL number of 0.34:

$$\begin{array}{ll} h = 1/12.5 & \Delta t = 1/50 \\ h = 1/25 & \Delta t = 1/100 \\ h = 1/50 & \Delta t = 1/200 \\ h = 1/100 & \Delta t = 1/400 \\ h = 1/200 & \Delta t = 1/800 \end{array} \tag{9.4}$$

Figure 9.5 shows a picture of the disc and the vortex street appearing in the wake. Figure 9.6 shows the drag coefficient C_D and lift coefficient C_L of the disc as a function of time, where these coefficients are defined by

$$C_D = \frac{F_x}{\frac{1}{2}\rho_f u_c^2 A}, \qquad C_L = \frac{F_y}{\frac{1}{2}\rho_f u_c^2 A}, \tag{9.5}$$

where (F_x, F_y) is the force on the disc, A is a reference "area" which in 2D is the diameter of the disc, and $\rho_f = 1$ in our case. Figure 9.7 shows the drag and lift coefficients compared with the "fine mesh" Q2-Q1 (third-order) finite element benchmark results of [21, Fig. 6,7]. Our results for the drag and lift on the disc match the benchmark very well.

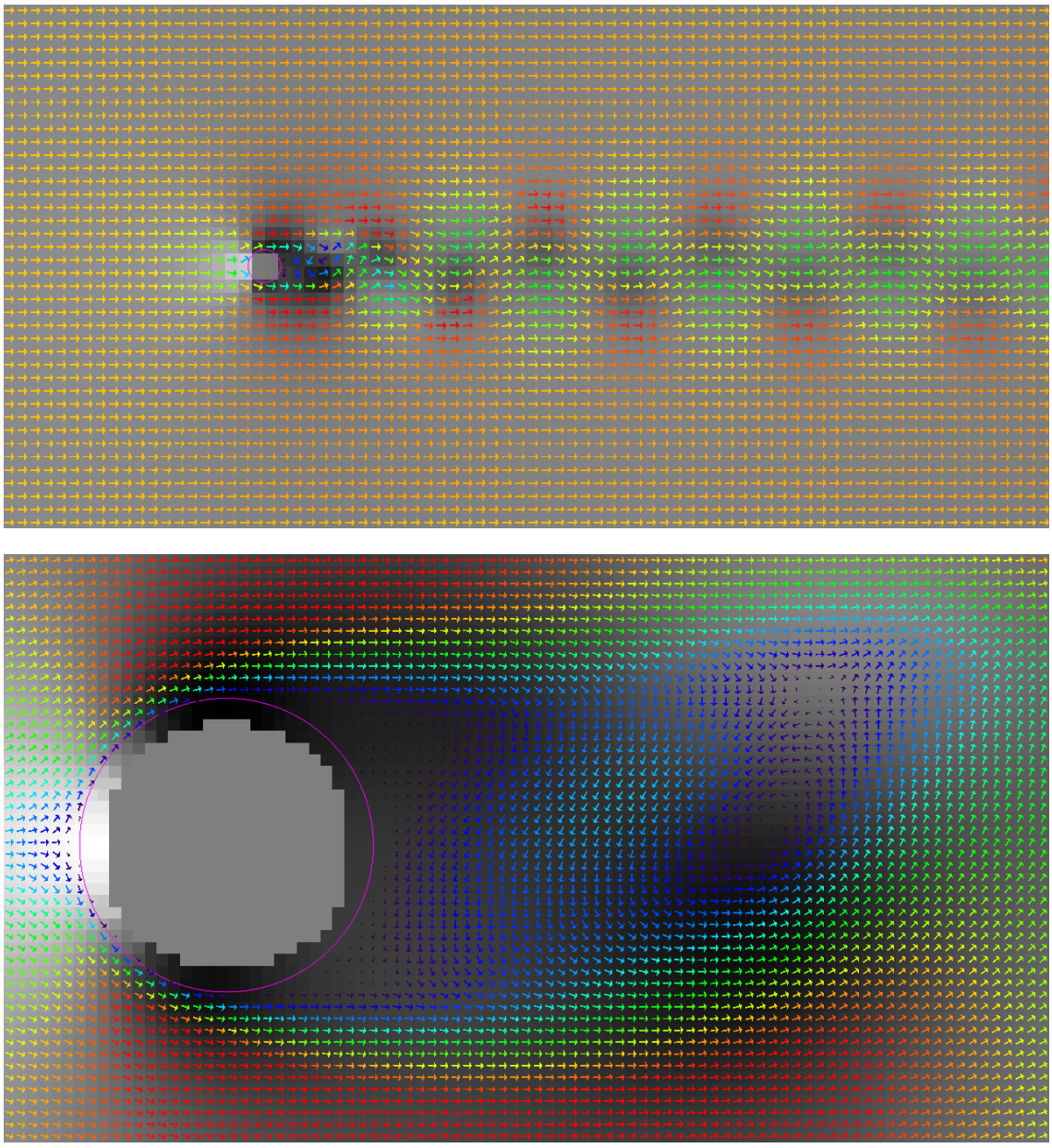

Figure 9.5: Top figure shows the entire computational domain for the "von Karman vortex street" test. The bottom figure shows a close-up of the disc shedding a vortex. In the bottom figure, each colored arrow represents one MAC cell using the grid resolution $h = 1/25$. Red colored arrows indicate fast moving fluid and blue colored arrows indicate slow moving fluid. White background indicates high pressure and black indicates low pressure. The disc is defined by the thin pink circle, not the set of gray cells inside the disc. The pressure has been explicitly extended inside the solid to a depth of 2 MAC cells using the procedure in Section 3.3.1.

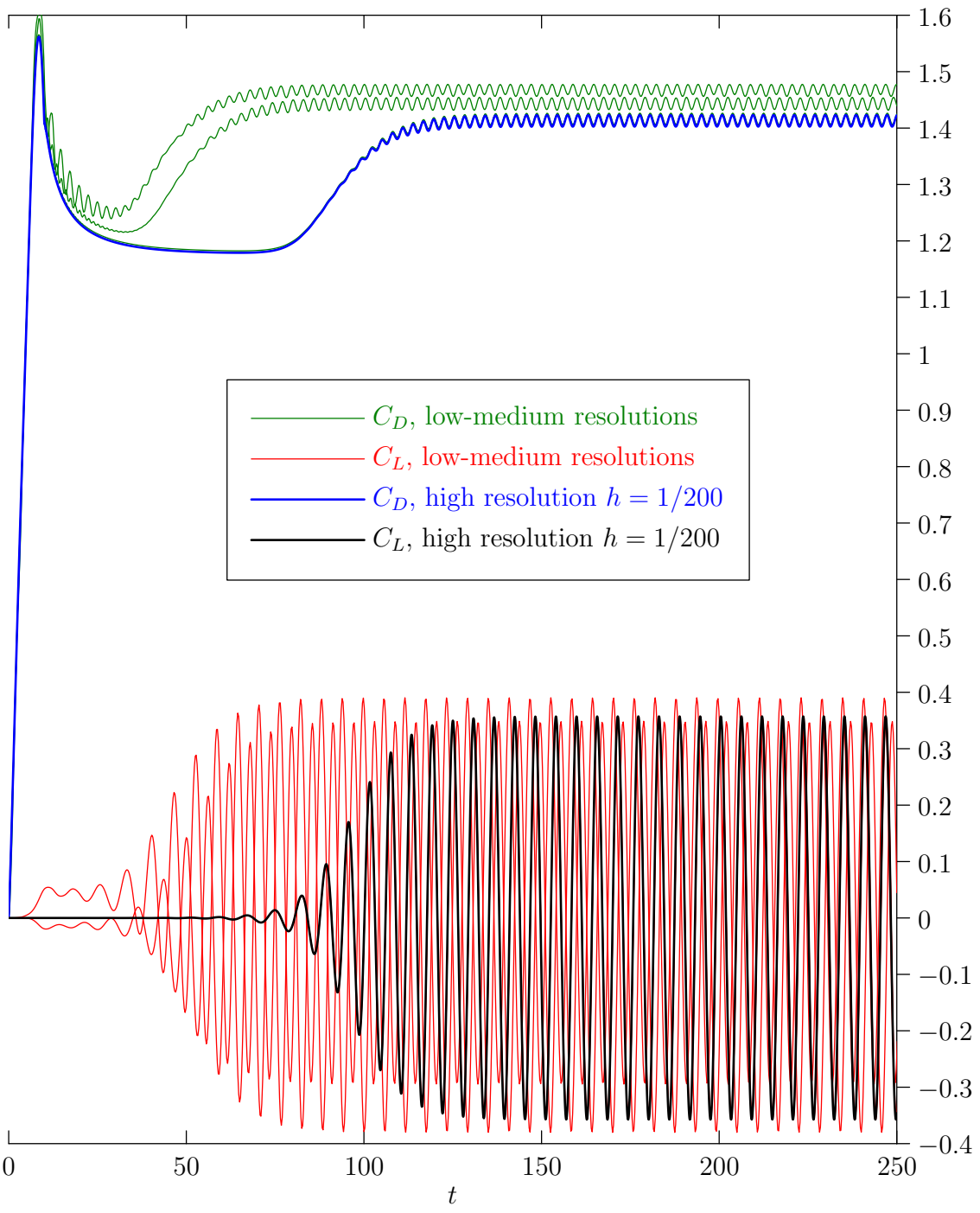

Figure 9.6: Drag and Lift coefficients as a function of time for the disc in the "von Karman vortex street" test. This graph shows the results of five simulations using the grid resolutions (9.4). The graphs for the three higher resolution simulations are very close and on this graph they appear to be drawn on top of each other. A close-up view as in Figure 9.7 shows that they differ by a small amount.

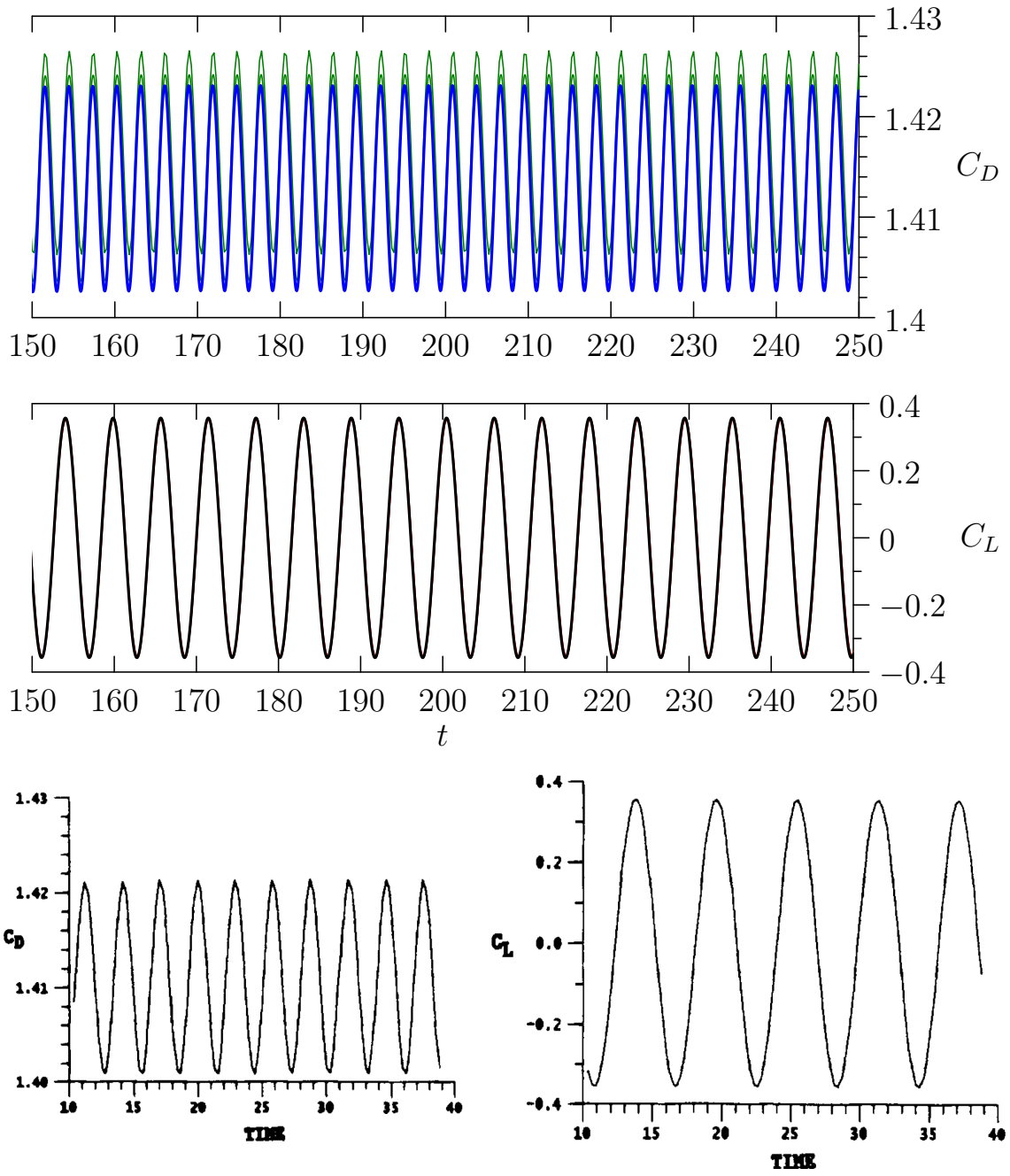

Figure 9.7: The top two figures show close-ups of the data in Figure 9.6 using the $h = 1/50$, $h = 1/100$, and $h = 1/200$ simulations. The bottom figures are the benchmark results of [21]. Our results agree that two vortices are shed every 5.8 time units. Our C_L is almost identical to the benchmark, and our C_D is very close. The top figure indicates that grid refinements beyond $h = 1/200$ (blue graph) will reduce C_d slightly and match the benchmark exactly.

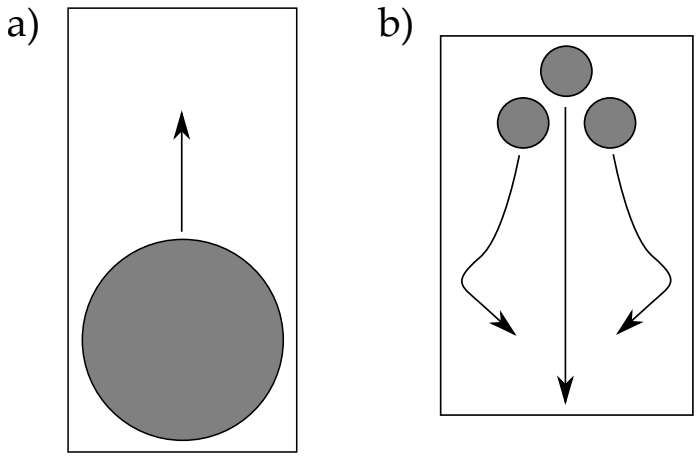

Figure 9.8: Two examples of 2D fluid flows containing rigid objects.

9.5 Large Buoyant Disc In 2D, Re $\approx$ 747

In this section, we consider Figure 9.8a, where a disc of relative density $\rho_{\text{disc}}/\rho_f = 1/4$ and diameter 1 rises vertically due to buoyancy (Fr=4321/5000) from an initial position of $(2500/4321, 2500/4321)$ in a rectangular domain $[0, 5000/4321] \times [0, 10000/4321]$. The Reynolds number is $(4321)^2/25000 \approx 747$. The purpose of this test is to demonstrate stability and accuracy of the numerical scheme in a situation where the solid object is significantly less dense than the fluid. In particular, the stabilizing factor $\eta_{\min}$ given by equation (6.7) is required for stability of this test. Table 9.4 shows numerical results of this simulation using the "direct scheme with boundary fitting" and the FDM in this thesis. As a comparison of accuracy, we can see from Table 9.4 that the "direct scheme with boundary fitting" converges with far fewer time step and spatial grid refinements than the FDM.

9.6 Three Falling Discs In 2D, Re=100

In this section, we consider Figure 9.8b, where three sedimenting discs interact with each other (without collisions) as they fall to the floor. The purpose of this test is to demonstrate that the "direct scheme with boundary fitting" also converges much faster than the FDM in more complicated situations involving multiple objects traveling in curved paths. The three discs each have unit diameter and the domain is

a 2D box of size $[0, 5] \times [0, 7.5]$. The relative density of all three discs is $\rho_{\text{disc}}/\rho_f = 1.1$, the Froude number is $\text{Fr} = 0.1$, and the initial positions of the disc centroids are $(5/2, 6.75)$, $(5/3, 5.625)$, and $(10/3, 5.625)$. The middle disc falls straight down due to symmetry, and it pushes the other two discs out of the way without actually touching them. Table 9.5 shows results of this simulation using the "direct scheme with boundary fitting" and the FDM in this thesis. The results again show that the "direct scheme with boundary fitting" is accurate even with coarse spatial resolutions and large time steps. The FDM scheme requires many refinements in both space and time in order to produce a similar result.

Direct Scheme with Boundary Fitting

h	Δt	v	$\|\nabla \cdot \mathbf{u}\|_{L^\infty(\Omega)}$
800/34568	1/50	0.096284	4.40
400/34568	1/100	0.095116	3.73
200/34568	1/200	0.094849	1.72
100/34568	1/400	0.094714	0.75
50/34568	1/800	0.094695	0.37
25/34568	1/1600	0.094692	0.19
25/34568	1/6400	0.094694	0.17

Fictitious Domain Method

h	Δt	v	$\|\nabla \cdot \mathbf{u}\|_{L^\infty(\Omega)}$
800/34568	1/50	0.106755	42.2
400/34568	1/100	0.108302	63.1
200/34568	1/200	0.103837	28.7
100/34568	1/400	0.099415	39.4
50/34568	1/800	0.097917	57.3
25/34568	1/1600	0.097004	83.2
800/34568	1/800	0.102700	7.6
400/34568	1/1600	0.104604	8.0
200/34568	1/3200	0.100745	10.0
100/34568	1/6400	0.096921	11.8
50/34568	1/12800	0.095954	16.5
25/34568	1/25600	0.095505	20.4

Table 9.4: Numerical results for the 2D buoyant disc test. This table presents the vertical velocity v of the disc and the maximum norm of the divergence, both of which are averaged over the time interval $[5.0, 6.0]$. We can see that the "direct scheme with boundary fitting" has converged in the fifth decimal place. This is because spatial error usually dominates the scheme, and for this problem the disc is spatially well resolved. Also, reducing the time step of the "direct scheme with boundary fitting" does not change the result, so this solution can be taken as correct to at least four or five decimal places. Even after using several more grid and time step refinements, the FDM has converged only to the second decimal place.

Direct Scheme with Boundary Fitting

h	Δt	ω_z	y	v
1/8	1/20	-0.0831	0.9317	-0.7194
1/16	1/40	-0.1029	0.7199	-0.7038
1/32	1/80	-0.0999	0.7212	-0.7104
1/64	1/160	-0.0961	0.7045	-0.7107
1/128	1/320	-0.0939	0.7021	-0.7125
1/256	1/640	-0.0927	0.7008	-0.7129
1/512	1/1280	-0.0921	0.7007	-0.7132
1/1024	1/2560	-0.0918	0.7015	-0.7137

Fictitious Domain Method

h	Δt	ω_z	y	v
1/8	1/20	-0.1471	0.9763	-0.6920
1/16	1/40	-0.0862	0.6088	-0.6637
1/32	1/80	-0.0725	0.5762	-0.6303
1/64	1/160	-0.0795	0.6069	-0.6575
1/128	1/320	-0.0840	0.6367	-0.6805
1/256	1/640	-0.0868	0.6568	-0.6928
1/512	1/1280	-0.0885	0.6705	-0.7000
1/1024	1/2560	-0.0895	0.6803	-0.7047
1/8	1/320	-0.1892	0.6049	-0.6818
1/16	1/640	-0.1047	0.5625	-0.5740
1/32	1/1280	-0.0854	0.6220	-0.6708
1/64	1/2560	-0.0884	0.6294	-0.6779
1/128	1/5120	-0.0893	0.6708	-0.7006
1/256	1/10240	-0.0903	0.6825	-0.7062
1/512	1/20480	-0.0908	0.6914	-0.7099
1/1024	1/10240	-0.0905	0.6916	-0.7097

Table 9.5: Numerical results for the three falling discs test. This table includes the angular velocity ω_z of the rightmost disc, the vertical position y of the middle disc, and the vertical velocity v of the middle disc. All quantities are averaged over the time interval $[9.75, 10.0]$ which is a few time units before the middle disc hits the floor. Both the FDM and the "direct scheme with boundary fitting" suggest that the exact solution accurate to two decimal places is $v = -0.71$, and $\omega_z = -0.09$. However, the "direct scheme with boundary fitting" achieves this result at a very coarse resolution compared to the FDM.

Chapter 10

DNS Of A 3D Fluidized Bed

In this chapter, we demonstrate the ability of the "direct scheme with boundary fitting" to solve very large problems. In particular, we simulate a fluidized bed, which is a container full of a large number of solid particles suspended in an upward flowing fluid as in Figure 10.1. Fluidized beds have many industrial applications for mixing, washing, or combustion of particles. We use "no-flux, no-slip" boundary conditions on the walls of the container and an upward flowing velocity of magnitude 1 as the Dirichlet boundary condition on the floor. The roof of the domain is the approximate stress-free condition $p = 0$ and $\frac{\partial \mathbf{u}}{\partial z} = 0$. We use the modified Douglas splitting and boundary-fitted spatial operators for diffusion and divergence. We choose $\chi = 1$, $h = 1/5$, $\Delta t = 1/80$, and collision model parameters $k = 10^5$, $\gamma = 100$. The Reynolds number based on the particle diameter is 10, the Froude number is 1, and the relative particle density is $\rho_p/\rho_f = 7.78$ (steel in water). We run two simulations, one containing 14000 spheres in the domain $[0, 44] \times [0, 44] \times [0, 44]$ solved using 1 CPU, and another simulation containing 2.2 million spheres in the domain $[0, 259.2] \times [0, 259.2] \times [0, 172.8]$ solved using 144 Xeon X5675 CPU cores and 205GB of distributed memory, which takes 18 seconds per time step. Both simulations are identical resolution, just different problem sizes. The initial condition of the spheres is a slightly perturbed Cartesian configuration with substantial spacing between each sphere so that the fluid can flow through the particles at the beginning of the simulation rather than forcing the entire group of spheres to the ceiling and causing them to "plug" the only outflow from the domain. Figure 10.2 shows the initial condition of the spherical particles along with a reference plane of the MAC grid to show spatial resolution. The spheres (diameter 1) have 5 MAC cells across their

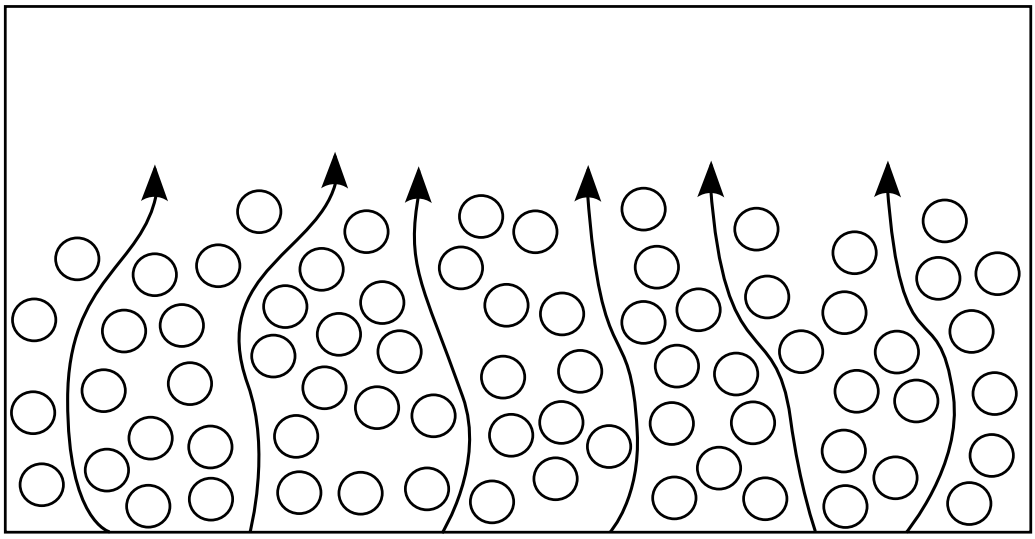

Figure 10.1: A fluidized bed in 2D containing a small number of particles.

diameter and contain 65 MAC cells by volume. The results of the 3D sedimenting ball simulation in Table 9.1 (column BF3) demonstrate that 5 MAC cells across the diameter is sufficient to resolve each object and the surrounding flow with less than 5% error. Figure 10.3 shows a picture of the 14000 sphere simulation, showing several plumes of fluid that bubbles up from below, which is a known feature of fluidized beds. Figures 10.4, 10.5 and 10.6 show pictures of the 2.2 million sphere simulation. Violent plumes can again be seen on the surface of the bed of suspended particles.

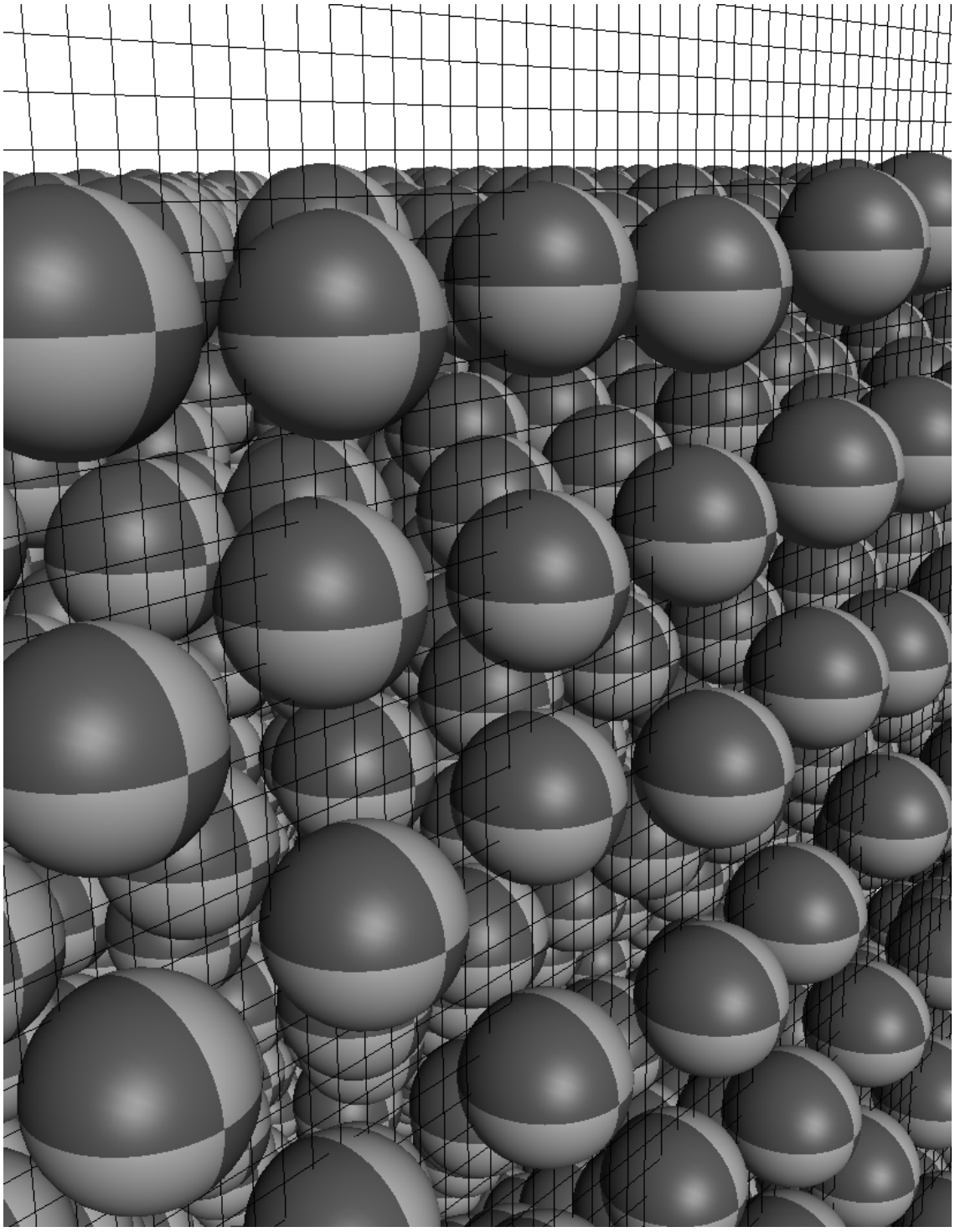

Figure 10.2: Initial position of spheres and a reference plane of the MAC grid.

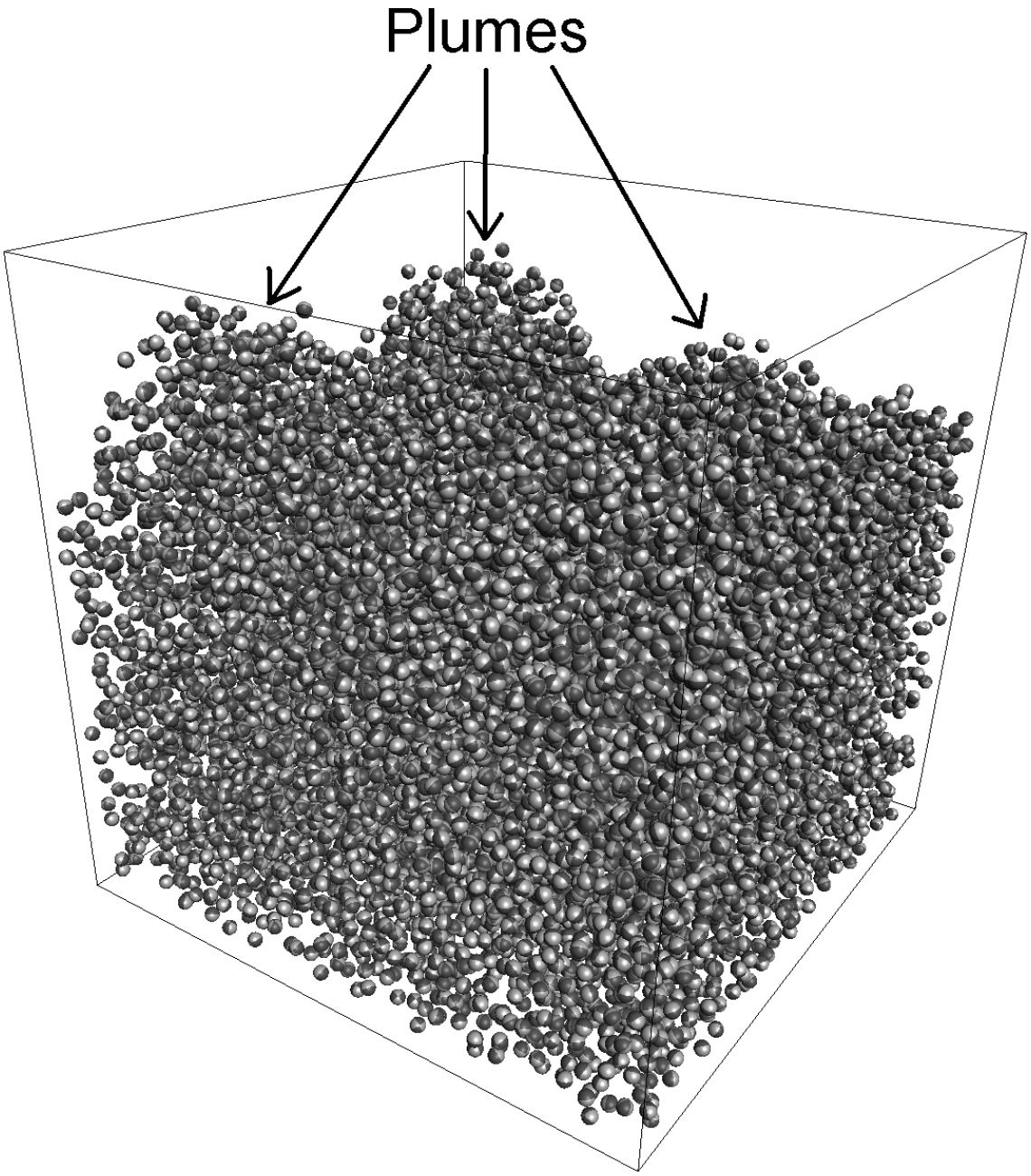

Figure 10.3: 3D fluidized bed simulation of 14000 spheres showing a typical configuration of the system. In this picture there are three plumes of fluid bubbling up from below and locally raising the surface height of the bed.

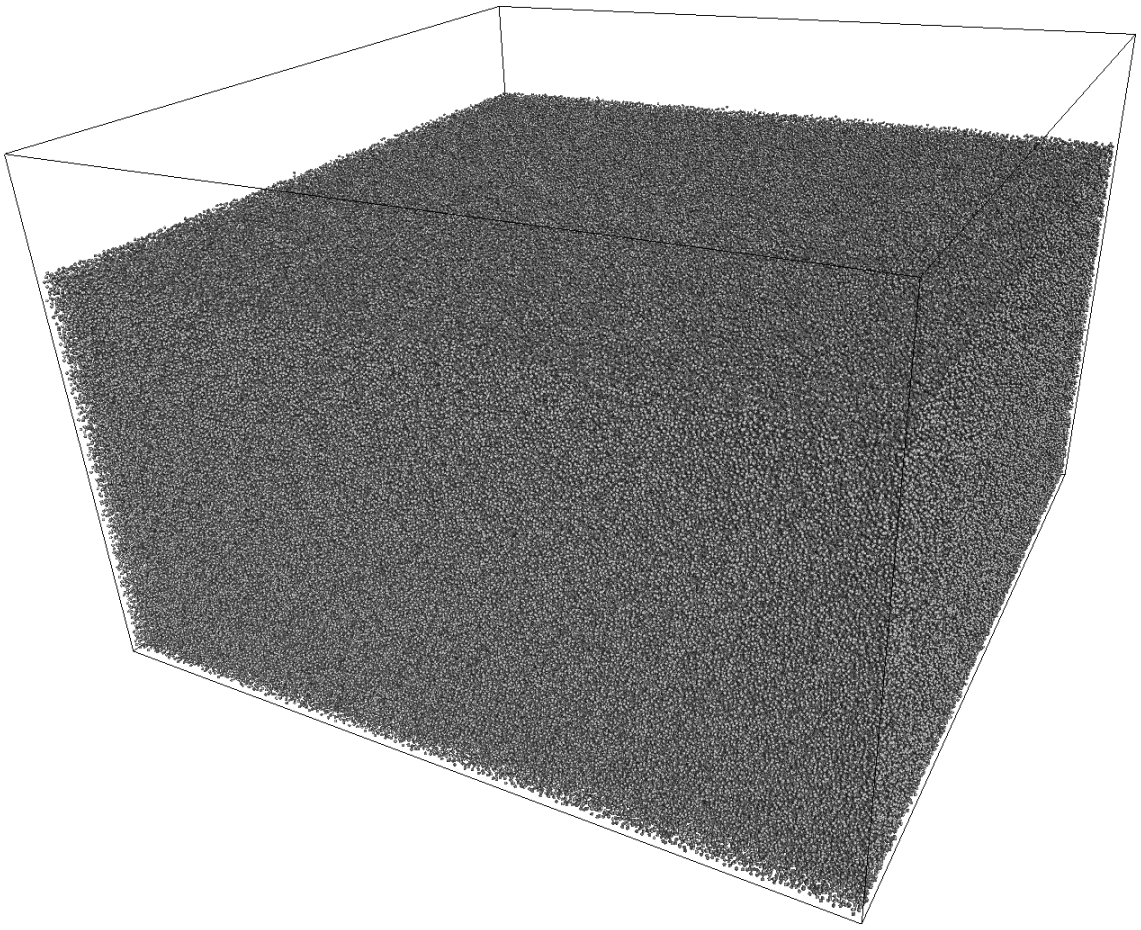

Figure 10.4: Initial condition of the 3D fluidized bed simulation of 2.2 million spheres.

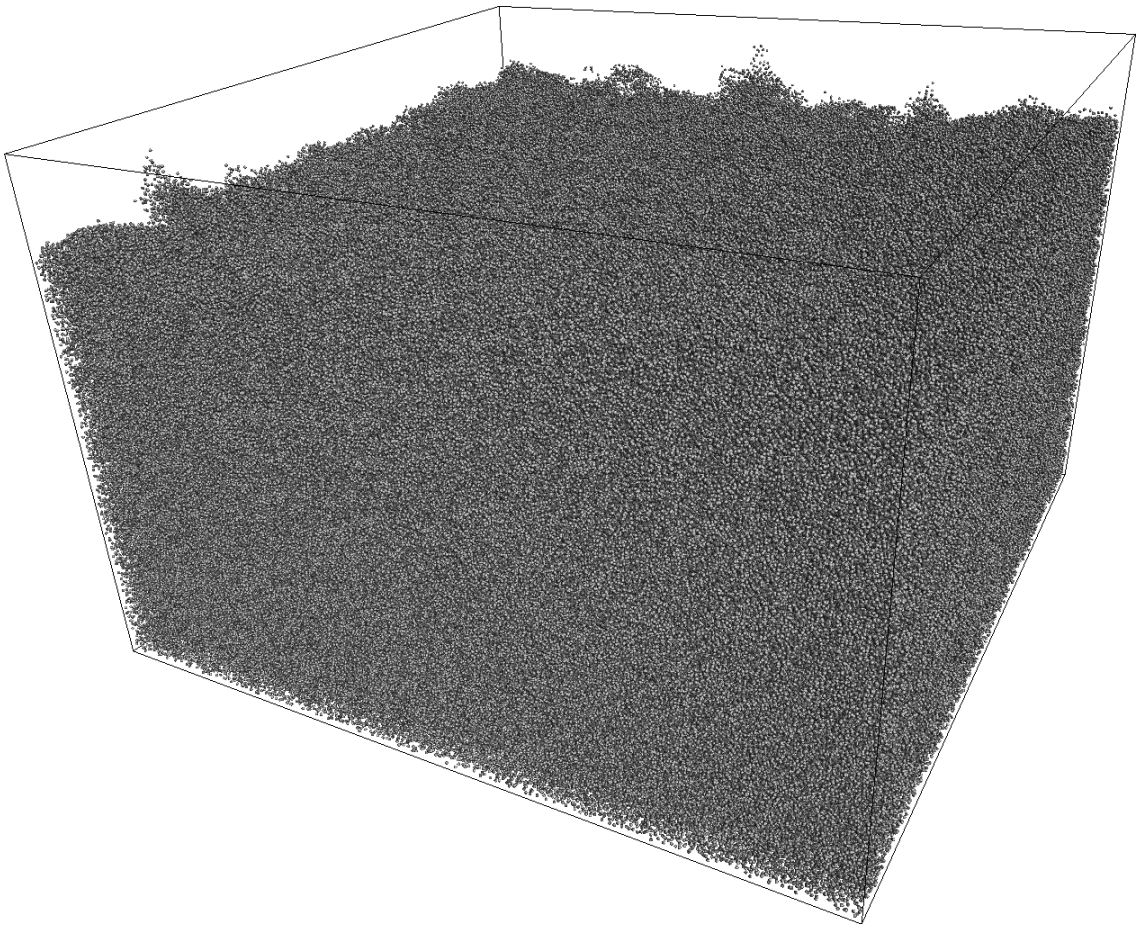

Figure 10.5: Mid-simulation of the 3D fluidized bed containing 2.2 million spheres.

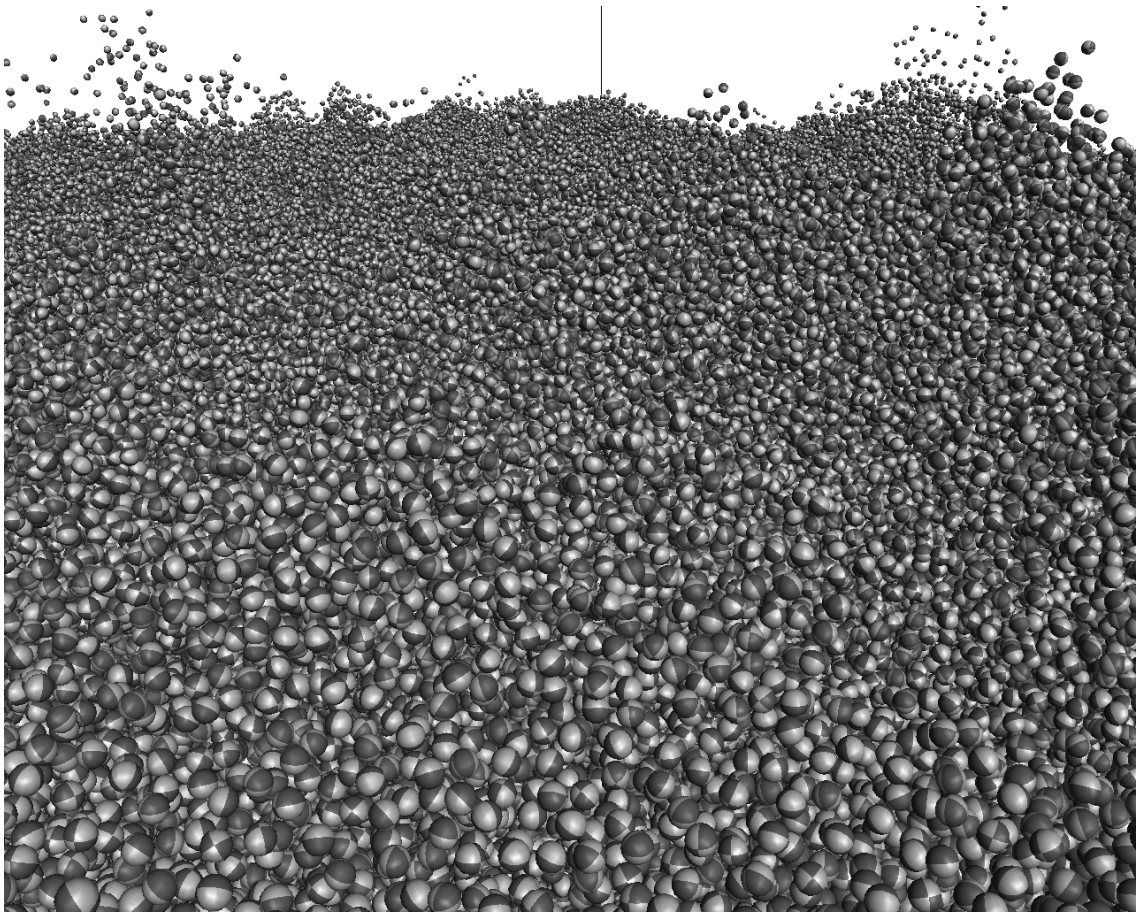

Figure 10.6: Surface of the 3D fluidized bed containing 2.2 million spheres.

10.1 Parallel Scalability

When measuring the parallel scalability of a given computer code, we are usually interested to know if the code is CPU bound or memory bound. CPU bound code typically does extensive work on small or medium amounts of data, and memory bound code typically does small amounts of work on large amounts of data. Strong scaling and weak scaling are two frequently used measures of how well a computer code utilizes different numbers of CPU cores. Strong scaling measures the efficiency of a CPU bound code to divide a fixed problem size into smaller parts using many CPU cores, where each CPU has a reduced workload. The strong parallel efficiency using N CPU cores is defined as $T_1/(NT_N)$, where T_1 is the time taken to solve the problem on 1 CPU core, and T_N is the time taken to solve the exact same problem using N CPU cores. Weak scaling measures the efficiency of a memory bound code to solve larger problems by keeping the workload per CPU fixed, and increasing the problem size when additional CPUs are used. The weak parallel efficiency using N CPU cores is defined as T_1/T_N, where T_1 is the time taken to solve problem size 1 using 1 CPU core, and T_N is the time taken to solve problem size N using N CPU cores.

Figure 10.7 shows the strong and weak parallel efficiency of the "direct scheme with boundary fitting" tested on 1, 8, 64, 216, and 512 CPU cores. All scaling results were performed on a Compute Canada WestGrid [71] cluster of 2.5GHz Intel Xeon L5420 CPU cores, 2GB DDR2 RAM per core, 8 cores per node, and each node connected via a 20GB/s network. The CPU scaling results were calculated using the real time taken to simulate 50 time steps.

For the strong scaling test, the problem size was 432x432x432 grid cells and 108000 solid objects, all of which takes approximately 12GB of RAM. The reduction in strong scaling efficiency at more than 200 CPU cores occurs because the 108000 solid objects in the simulation are distributed across all processors, and the objects must be communicated at each of the 100 soft-sphere collision sub-steps as in equation (6.12) that were used for this test. For the 512 CPU test, about 50% of the real time is spent communicating object positions and velocities between processors at these sub time steps. Thus, the strong scaling efficiency decline at more than 200 CPU cores is somewhat artificial for two reasons. First, if the sub time step communication is consuming all of the simulation time, then one can simply reduce the number of sub time steps per time step. Second, for a fixed problem size of only 108000

solid objects, there is sufficient memory to simply store all the solid objects on each processor, and then each processor can compute its own collision forces which requires no communication at all during the sub time steps.

Since we are primarily interested in solving huge problems, our code is memory bound and weak scaling is the relevant measure. For the weak scaling test, we use the same parameters as the strong scaling test except that the problem size is equal to 204x204x204 grid cells and 13000 solid objects per CPU, which takes about 1.4GB of RAM per CPU. This results in a problem size of 1632x1632x1632 grid cells and 6,656,000 solid objects when scaled up to 512 CPU cores. Figure 10.7 shows a significant drop in efficiency going from 1 to 8 CPUs, but this is largely due to the fact that 1 CPU core using only 1.4GB of RAM does not need to share the memory bandwidth or cache utilization on its cluster node (which contains another 7 idle CPUs). We can see from Figure 10.7 that there is very little reduction in parallel efficiency after all 8 CPU cores on a node are used. Eventually the parallel efficiency of the code must decline, of course, but Figure 10.7 suggests that this limit is far away. In particular, since we use direction splitting, all CPU communication is either "neighbour-to-neighbour" or in one dimension only. This means that communication costs should increase roughly as the cubed root of the problem size, and Figure 10.7 suggests that thousands of CPU cores could be utilized with high enough efficiency to solve problems containing one billion solid objects.

Our numerical method compares very well to other large parallel simulations in the literature. For example, the simulations of [26] utilized 294912 processor cores and 50TB of memory to simulate 264 million objects. While we do not have access to such a large computer, we can compare resource usage. Since our method utilizes 144 CPU cores and 205GB of memory to simulate 2.2 million objects, simple scaling suggests that with 294912 cores and 410 TB of memory we would be able to simulate 4.5 billion objects. One can also speculate that the computer code of [26] is optimized to a higher level than the computer code used for this thesis, judging simply by the fact that [26] is published in a computer science journal and the fact that any code which is allowed to run on such a large computer should be heavily optimized. Also keep in mind that the method of [26] is a fully explicit lattice-Boltzmann scheme, and our method solves Navier–Stokes with second-order accuracy using implicit methods. While it is relatively easy to make any fully explicit scheme parallelize well, the ability to perform large parallel direct numerical simulations using an implicit method is non-existent (or very rare) in the literature.

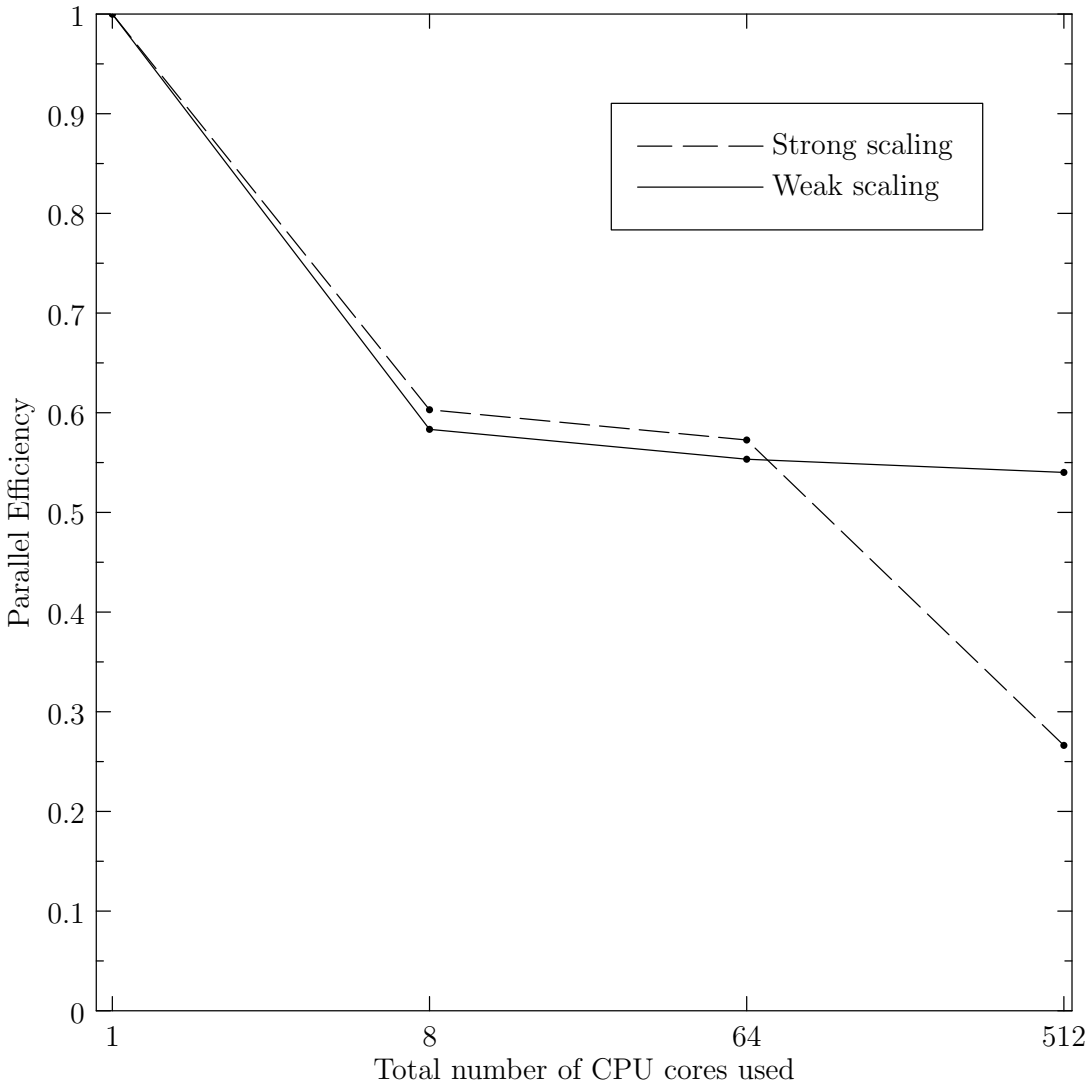

Figure 10.7: Parallel scalability for 3D problem using up to 512 CPU cores.

It is also worthwhile mentioning that a quick literature search can uncover a large number of works where the authors have been able to simulate large numbers of particles *without* using direct numerical simulation. For example, the recent work of [40] utilizes graphics processing units (GPUs) as well as CPUs in order to simulate 25 million particles using an ordinary desktop computer. However, in this case the particles are treated as points and the spatial resolution of the fluid is only one grid point for every 10 particles. Numerical models such as [40] rely on experimentally measured parameters in order to match similar experiments, and this is not direct numerical simulation. [11] is another recent example of a "volume average" type numerical model which is capable of simulating a large number of particles, but again the fluid is resolved using a very coarse grid that is much larger than the particle scale, and again this is not direct numerical simulation. If no fluid is included at all (particles only) then up to 100 billion particle systems have been computed by [1].

It is interesting to know what kinds of problems could actually be solved using direct numerical simulation. According to [67], an industrial-sized fluidized bed typically has a volume of between 2 and 200 cubic meters, and it usually contains millimeter-sized particles. Therefore, such a fluidized bed would contain 1 to 200 billion particles, depending on the shape of the particles and the void fraction. It seems plausible that all of the particles in a small industrial fluidized bed could be resolved using a scheme similar to the one in this thesis with today's supercomputers, however, the larger industrial fluidized beds would be a stretch for any fully resolved direct numerical simulation.

Chapter 11

Conclusions

In this thesis, we discussed several different direction-splitting schemes in the context of using boundary-fitted spatial operators for complex-shaped domains. In order to address stability issues associated with some splitting schemes in these situations, we introduced a new "modified Douglas" direction-splitting scheme. Numerical evidence suggests that this new splitting scheme is unconditionally stable for the diffusion equation. Using the "modified Douglas" scheme combined with a direction-splitting scheme for the incompressibility constraint, we constructed a fully second-order accurate scheme for solving the Navier–Stokes equations in 3D complex-shaped domains.

Based on the direction-splitting scheme for solving Navier–Stokes in complex-shaped domains, we created a numerical method for direct numerical simulation of particulate flows. The fluid forces on the particles are computed using boundary-fitted surface integrals of the fluid stress. The method as a whole was demonstrated to be second-order accurate in both time and space. The method was also validated on a number of real fluid flow problems and shown to agree well with experimental results and numerical results of other authors.

A second method for direct numerical simulation of particulate flows was also constructed using the direction-splitting Navier–Stokes scheme combined with a fictitious domain approach. Although the fictitious domain method is not as accurate as the second-order method, we discussed how it could be improved by using iterations at each time-step and by improving the accuracy of the volume integrals, particularly for conserving momentum.

For dealing with particle collisions, we introduced two different models. The first model was a simple dry soft-sphere viscoelastic model and the second model was an

elastic-plastic model based on lubrication theory. We showed that the lubrication theory model agrees with experiments of submerged collisions, and we explained how such a model could be incorporated into the larger scale numerical scheme.

Finally, we demonstrated the efficiency and scalability of the numerical methods of this thesis by performing DNS of a 3D fluidized bed containing 2.2 million spherical particles. Parallel scaling results indicated that the numerical scheme could be used to simulate fluid flows containing one billion particles.

11.1 Future Work

There are several areas of future work for this thesis. The most natural and immediate extension is to complete the work of submerged collision modeling by performing quantitative tests of the viscoelastic collision model in Section 6.5. If the model is determined to be insufficiently accurate, then the subgrid collision model described in Chapter 7 could be incorporated. Furthermore, angular momentum transfer between particles during a collision may also require subgrid modeling.

Another useful task for future work is to extend the numerical scheme to include solid objects (particles) of non-spherical shape. This task is straightforward in theory, however, the primary difficulty involves writing more parallel-CPU computer code to perform surface integrals. Also, the numerical scheme could be extended to work for "particles" of time-dependent shape, such as gas bubbles. This is a more complicated task but should still be possible under the current framework.

Although this thesis contains a lot of work related to boundary-fitting the discrete divergence operator, the options presented in Sections 3.4.1 or 3.4.2 can likely be improved. In particular, some local accuracy could be lost near object boundaries, and more numerical testing could be done in this area. Numerical results on coarse grids are affected by the way in which the discrete divergence is boundary fitted.

Since this thesis ends with a demonstration of ability to solve large problems, a natural place for future work is to simply use the method to solve large problems. For example, some statistics could be generated for fluidized beds or sedimentation problems. However, before performing very large simulations, the scheme should be tested on smaller problems and verified to be quantitatively correct, for example, does the fluidized bed have the correct height? Also, for such simulations it could be useful to first modify the scheme to use an adaptive time step and/or use the implicit

advection (4.101) in order to avoid violating the CFL condition halfway through a simulation involving thousands of CPU cores.

Finally, since the numerical scheme in this thesis involves many features that are relatively new, it would be useful to perform more quantitative comparisons against benchmark numerical or experimental results. With a sufficiently accurate benchmark, more grid convergence studies could be performed to verify convergence rates on complicated flow problems.

Bibliography

[1] N. Allsopp, G. Ruocco, and A. Fratalocchi. Molecular dynamics beyonds the limits: Massive scaling on 72 racks of a bluegene/p and supercooled glass dynamics of a 1 billion particles system. *Journal of Computational Physics*, 231:3432–3445, 2012.

[2] P. Angot. Analysis of singular perturbations on the Brinkman problem for fictitious domain models of viscous flows. *Mathematical Methods in the Applied Sciences*, 22(16):1395–1412, 1999.

[3] P. Angot, J.-P. Caltagirone, and P. Fabrie. A fast vector penalty-projection method for incompressible non-homogeneous or multiphase navier-stokes problems. *Applied Mathematics Letters*, 25:1681–1688, 2012.

[4] P. Angot, J. Keating, and P. Minev. A direction splitting algorithm for incompressible flow in complex geometries. *Computer Methods in Applied Mechanics and Engineering*, 117:111–120, 2012.

[5] S. V. Apte and J. R. Finn. A variable-density fictitious domain method for particulate flows with broad range of particle-fluid density ratios. *Journal of Computational Physics*, 243:109–129, 2013.

[6] S. V. Apte, M. Martin, and N. A. Patankar. A numerical method for fully resolved simulation (FRS) of rigid particle-flow interactions in complex flows. *Journal of Computational Physics*, 228:2712–2738, 2009.

[7] N. J. Balmforth, C. Cawthorn, and R. V. Craster. Contact in a viscous fluid. Part 2. A compressible fluid and an elastic solid. *Journal of Fluid Mechanics*, 646:339–361, 2010.

[8] G. Barnocky and R. H. Davis. Elastohydrodynamic collision and rebound of spheres: Experimental verification. *Physics of Fluids*, 31:1324–1329, 1988.

[9] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, pages 1–137, 2005.

[10] S. Bertoluzza, M. Ismail, and B. Maury. Analysis of the fully discrete fat boundary method. *Numerische Mathematik*, 118(1):49–77, 2011.

[11] J. Capecelatro and O. Desjardins. An euler-lagrange strategy for simulating particle-laden flows. *Journal of Computational Physics*, 238:1–31, 2013.

[12] A. Cate, C. Nieuwstad, J. Derksen, and H. V. den Akker. Particle imaging velocimetry experiments and lattice-Boltzmann simulations on a single sphere settling under gravity. *Physics of Fluids*, 14:4012, 2002.

[13] Y. A. Cengel, R. H. Turner, and J. M. Cimbala. *Fundamentals of Thermal-Fluid Sciences*. McGraw-Hill Higher Education, 3rd edition, 2008.

[14] S. Chen and G. D. Doolen. Lattice Boltzmann method for fluid flows. *Annual Review Fluid Mechanics*, 30:329–364, 1998.

[15] A. Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, 2:12–26, 1967.

[16] A. Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, 22:745–762, 1968.

[17] M. Coquerelle and G. H. Cottet. A vortex level set method for the two-way coupling of an incompressible fluid with colliding rigid bodies. *Journal of Computational Physics*, 227:9121–9137, 2008.

[18] R. H. Davis, J.-M. Serayssol, and E. J. Hinch. The elastohydrodynamic collision of two spheres. *Journal of Fluid Mechanics*, 163:479–497, 1986.

[19] N. Deen, M. V. S. Annaland, M. V. der Hoef, and J. Kuipers. Review of discrete particle modeling of fluidized beds. *Chemical Engineering Science*, 62(1-2):28–44, 2007.

[20] J. Douglas, Jr. Alternating direction methods for three space variables. *Numerische Mathematik*, 4:41–63, 1962.

[21] M. S. Engelman and M.-A. Jamnia. Transient flow past a circular cylinder: a benchmark solution. *International Journal for Numerical Methods in Fluids*, 11:985–1000, 1990.

[22] Z.-G. Feng and E. E. Michaelides. Proteus: a direct forcing method in the simulations of particulate flows. *Journal of Computational Physics*, 202:20–51, 2005.

[23] S. Ghosh and J. M. Stockie. Numerical simulations of particle sedimentation using the immersed boundary method. *Submitted to Journal of Computational Physics*, April 2013.

[24] R. Glowinski, T. Pan, T. Hesla, and D. Joseph. A distributed Lagrange multiplier/fictitious domain method for particulate flows. *International Journal of Multiphase Flow*, 25:755, 1999.

[25] R. Glowinski, T. Pan, T. Hesla, D. Joseph, and J. Periaux. A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: Application to particulate flow. *Journal of Computational Physics*, 169:363, 2001.

[26] J. Götz, K. Iglberger, M. Stürmer, and U. Rüde. Direct numerical simulation of particulate flows on 294912 processor cores. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '10, pages 1–11, Washington, DC, USA, 2010. IEEE Computer Society.

[27] J. Guermond and P. Minev. A new class of fractional step techniques for the incompressible Navier–Stokes equations using direction splitting. *Comptes Rendus Mathematique*, 348:581–585, 2010.

[28] J. Guermond and P. Minev. Start-up flow in a three-dimensional lid-driven cavity by means of a massively parallel direction splitting algorithm. *International Journal for Numerical Methods in Fluids*, 68:856–871, 2012.

[29] J.-L. Guermond and P. Minev. A new class of splitting methods for the incompressible Navier–Stokes equations using direction splitting. *Computer Methods in Applied Mechanics and Engineering*, 200:2083–2093, 2011.

[30] J.-L. Guermond, P. Minev, and A. Salgado. Convergence analysis of a class of massively parallel direction splitting algorithms for the Navier-Stokes equations in simple domains. *Mathematics of Computation*, 81:1951–1977, 2012.

[31] J.-L. Guermond, P. Minev, and J. Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195:6011–6054, 2006.

[32] J.-L. Guermond and A. Salgado. A splitting method for incompressible flows with variable density based on a pressure poisson equation. *Journal of Computational Physics*, 228(8):2834 – 2846, 2009.

[33] J.-L. Guermond and A. Salgado. Error analysis of a fractional time-stepping technique for incompressible flows with variable density. *SIAM Journal on Numerical Analysis*, 49(3):917 – 944, 2011.

[34] J.-L. Guermond and J. Shen. On the error estimates for the rotational pressure-correction projection methods. *Mathematics of Computation*, 73(248):1719–1737, 2003.

[35] B. J. Hamrock, S. R. Schmid, and B. O. Jacobson. *Fundamentals of Fluid Film Lubrication*. CRC Press 2004, 2004.

[36] X. He and L.-S. Luo. Lattice Boltzmann model for the incompressible navier-stokes equation. *Journal of Statistical Physics*, 88:927–944, 1997.

[37] H. H. Hu, N. Patankar, and M. Zhu. Direct numerical simulations of fluid-solid systems using the arbitrary Lagrangian-Eulerian technique. *Journal of Computational Physics*, 169:427–462, 2001.

[38] I.A.P.W.S. Revised release on the IAPWS industrial formulation 1997 for the thermodynamic properties of water and steam. *International Association for the Properties of Water and Steam*, 2007.

[39] I.A.P.W.S. Release on the IAPWS formulation 2008 for the viscosity of ordinary water substance. *International Association for the Properties of Water and Steam*, 2008.

[40] D. Jajcevic, E. Siegmann, C. Radeke, and J. G. Khinast. Large-scale cfd-dem simulations of fluidized granular systems. *Chemical Engineering Science*, 98:298–310, 2013.

[41] G. Joseph, R. Zenit, M. Hunt, and A. Rosenwinkel. Particle-wall collisions in a viscous fluid. *Journal of Fluid mechanics*, 433:329–346, 2001.

[42] A. A. Kantak and R. H. Davis. Oblique collisions and rebound of spheres from a wetted surface. *Journal of Fluid mechanics*, 509:63–81, 2004.

[43] J. Kim, D. Kim, and H. Choi. An immersed-boundary finite volume method for simulations of flow in complex geometries. *Journal of Computational Physics*, 171:132–150, 2001.

[44] H. Kleine, S. Tepper, K. Takehara, T. Etoh, and K. Hiraki. Cavitation induced by low-speed underwater impact. *26th International Symposium on Shock Waves*, pages 895–900, 2009.

[45] T. A. Laursen, E. J. Kim, and B. Yang. Recent extensions of mortar-based contact formulations: Lubrication modeling and parallel implementations. *IUTAM Symposium on Multiscale Problems in Multibody System Contacts*, pages 123–146, 2006.

[46] S. Luding. Contact models for very loose granular materials. *IUTAM Symposium on Multiscale Problems in Multibody System Contacts*, pages 135–150, 2006.

[47] T. Manteuffel and A. White, Jr. The numerical solution of second-order boundary value problems on nonuniform meshes. *Mathematics of Computation*, 47(176):511–535, 1986.

[48] S. Marella, S. Krishnan, H. Liu, and H. Udaykumar. Sharp interface cartesian grid method I: An easily implemented technique for 3D moving boundary computations. *Journal of Computational Physics*, 210:1–31, 2005.

[49] B. Maury. A many-body lubrication model. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 325(9):1053–1058, 1997.

[50] A. Mitchell. Splitting methods in partial differential equations. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 36(1):45–56, 1971.

[51] R. Mittal, H. Dong, M. Bozkurttas, F. Najjar, A. Vargas, and A. von Loebbecke. A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. *Journal of Computational Physics*, 227:4825–4852, 2008.

[52] T.-W. Pan and R. Glowinski. Direct simulation of the motion of neutrally buoyant balls in a three-dimensional Poiseuille flow. *Comptes Rendus Mécanique*, 333:884, 2005.

[53] N. Patankar. A formulation for fast computations of rigid particulate flows. *Center for Turbulence Research, Annual Research Briefs*, pages 185–196, 2001.

[54] D. Peaceman and H. Rachford, Jr. The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for Industrial and Applied Mathematics*, 3(1):28–41, 1955.

[55] C. Peskin. Numerical analysis of blood flow in the heart. *Journal of Computational Physics*, 25:220, 1977.

[56] C. S. Peskin. The immersed boundary method. *Acta Numerica*, 11:479–517, 2002.

[57] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3rd edition, 2007.

[58] R. Rannacher. *On Chorin's projection method for the incompressible Navier-Stokes equations*. Lecture Notes in Mathematics, vol. 1530, 1992.

[59] A. Samarskii, P. Matus, and Vabishchevich. Difference schemes with operator factors. In M. Hazewinkel, editor, *Mathemativs and Its Applications, v. 546*. Kluwer Academic Publishers, Dordrecht/Boston/London, 2002.

[60] A. Samarskii and A. Vabishchevich. *Additive schemes for problems of Mathematical Physics (in Russian)*. Nauka, Moskva, 1999.

[61] G. Segré and A. Silberberg. Behaviour of macroscopic rigid spheres in Poiseuille flow Part 2. Experimental results and interpretation. *Journal of Fluid Mechanics*, 14:136–157, 1962.

[62] N. Sharma and N. Patankar. A fast computation technique for the direct numerical simulation of rigid particulate flows. *Journal of Computational Physics*, 205:439–457, 2005.

[63] A. B. Stevens and C. M. Hrenya. Comparison of soft-sphere models to measurements of collision properties during normal impacts. *Powder Technology*, 154:99–109, 2005.

[64] J. R. Taylor. *Classical mechanics*. University Science Books, 2005.

[65] R. Temam. Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires II. *Archive for Rational Mechanics and Analysis*, 33:377–385, 1969.

[66] L. Timmermans, P. Minev, and F. V. D. Vosse. An approximate projection scheme for incompressible flow using spectral elements. *International Journal for Numerical Methods in Fluids*, 22:673–688, 1996.

[67] M. van der Hoef, M. van Sint Annaland, N. Deen, and J. Kuipers. Numerical simulation of dense gas-solid fluidized beds: A multiscale modeling strategy. *Annual Review of Fluid Mechanics*, 40:47–70, 2008.

[68] C. Veeramani, P. Minev, and K. Nandakumar. A fictitious domain formulation for flows with rigid particles: A non-Lagrange multiplier version. *Journal of Computational Physics*, 224:867–879, 2007.

[69] C. Veeramani, P. Minev, and K. Nandakumar. Collision modeling between tro non-Brownian particles in multiphase flow. *International Journal of Thermal Sciences*, 48:226–233, 2009.

[70] D. Wan and S. Turek. An efficient multigrid-FEM method for the simulation of solid-liquid two phase flows. *Journal of Computational and Applied Mathematics*, 203:561–580, 2007.

[71] Westgrid. `http://www.westgrid.ca`, 2013.

[72] N. N. Yanenko. On a method for solving the multi-dimensional heat equation (in Russian). *Doklady Akademii Nauk SSSR*, 125:1207–1210, 1959.

[73] B. Yang, J. Wang, D. Joseph, H. Hu, T. Pan, and R. Glowinski. Migration of a sphere in a tube flow. *Journal of Fluid mechanics*, 540:109, 2005.

[74] C. You, X. Wang, H. Qi, R. Yang, and D. Xu. Direct numerical simulation of particle collisions in two-phase flows with a meshless method. *Chemical Engineering Science*, 63:3474–3484, 2008.

[75] Z. Zhang and A. Prosperetti. A second-order method for three-dimensional particle simulation. *Journal of Computational Physics*, 210:292–324, 2005.

Appendix A

Taylor–Couette Flow

The 2D Taylor–Couette laminar flow between two concentric rotating cylinders (depicted in Figure A.1) has a well known analytical solution to the incompressible Navier–Stokes equations (2.22). When written in 2D polar coordinates, the steady-state solution in Ω_f between the two cylinders of radius R_1 and R_2 rotating at angular velocities ω_1 and ω_2 is

$$v_r = 0, \quad v_\theta = ar + \frac{b}{r}, \quad a = \frac{\omega_2 R_2^2 - \omega_1 R_1^2}{R_2^2 - R_1^2}, \quad b = \frac{(\omega_1 - \omega_2) R_1^2 R_2^2}{R_2^2 - R_1^2}, \tag{A.1}$$

$$p = \frac{a^2 r^2}{2} + 2ab \ln r - \frac{b^2}{2r^2}, \tag{A.2}$$

where v_r is the component of the velocity in the radial direction and v_θ is the component of the velocity in the θ direction. The velocity (u, v) in Cartesian coordinates is

$$u = -y v_\theta / r, \qquad v = x v_\theta / r. \tag{A.3}$$

We can embed the fluid domain Ω_f into a larger box domain Ω. Outside the fluid domain, i.e., in $\Omega \backslash \Omega_f$, the velocity field represents rigid rotation of the solid cylinders,

$$v_r = 0, \quad v_\theta = \omega r. \tag{A.4}$$

The regularity of this analytical solution is C^∞ in Ω_f, C^∞ in $\Omega \setminus \Omega_f$, but only C^0 in Ω. In other words, the first derivative of the velocity has a jump discontinuity across the fluid domain boundary $\partial\Omega_f$.

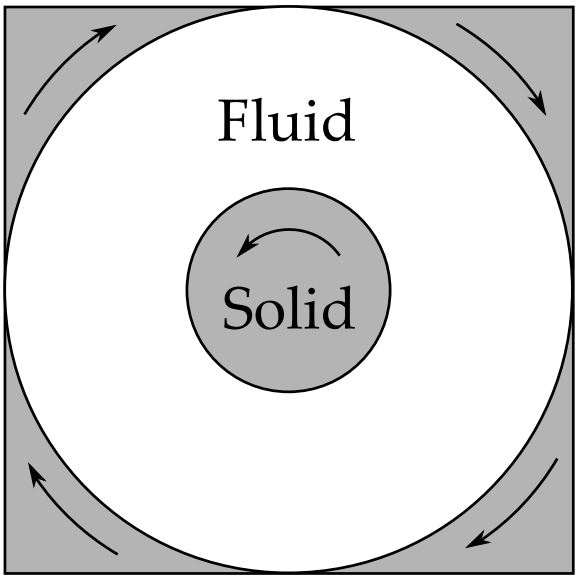

Figure A.1: Taylor–Couette flow between two solid cylinders.

The Taylor–Couette velocity solution (A.1) can also be used as a solution for the Stokes equations or the diffusion equation. In particular, (A.1) satisfies the diffusion equation (5.5) exactly. Therefore, (A.1) together with the zero pressure field $p = 0$ satisfies the Stokes equations (2.26).

Appendix B

Pressure Projection With Variable Density

If a fluid is incompressible but the density is *not* constant, then the variable-density incompressible Navier–Stokes equations are:

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

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

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) = 0, \tag{B.3}$$

where μ is the viscosity and ρ is the density and the new equation (B.3) is the density transport equation. Similar to the Chorin and Temam pressure projection in Section 4.1.1, one can split (B.1) into

$$\frac{\tilde{\mathbf{u}}^{n+1} - \mathbf{u}^n}{\Delta t} + \tilde{\mathbf{u}}^n\cdot\nabla\tilde{\mathbf{u}}^n = \frac{\mu}{\rho^{n+1}}\nabla^2\tilde{\mathbf{u}}^{n+1}, \tag{B.4}$$

$$\frac{\mathbf{u}^{n+1} - \tilde{\mathbf{u}}^{n+1}}{\Delta t} = -\frac{1}{\rho^{n+1}}\nabla p^{n+1}. \tag{B.5}$$

Taking the divergence of (B.5) and applying (B.2) gives the following variable coefficient Poisson problem for p^{n+1},

$$\frac{\nabla \cdot \tilde{\mathbf{u}}^{n+1}}{\Delta t} = \nabla \cdot \left(\frac{1}{\rho^{n+1}} \nabla p^{n+1} \right), \tag{B.6}$$

which is more difficult than the equivalent problem when the density is constant,

$$\frac{\nabla \cdot \tilde{\mathbf{u}}^{n+1}}{\Delta t} = \frac{1}{\rho} \nabla \cdot \left(\nabla p^{n+1} \right) = \frac{1}{\rho} \nabla^2 p^{n+1}. \tag{B.7}$$

Appendix C

Implementation Details

In this chapter, we briefly review some of the features of the computer code that was written along with this thesis. Designing, writing, testing, and debugging code took about 2/3 of my time during the last three years. However, at least half of this time was spent writing code for variants of the numerical scheme that have since been abandoned and we will not discuss any of these variants.

The code is written in C++, although almost all of the code is essentially C code. None of the C++ standard library is used, no "exception throwing", no "virtual functions", etc., since most of these constructs reduce the execution speed of the code. No third-party libraries are used, and all of the code was written from scratch (by me). This makes the code very portable, and it compiles and runs "problem free" on Linux, Microsoft windows, Mac OS, supercomputing clusters, etc. The code can be stopped and restarted by saving and loading the entire state from disk, and there are several compile-time parameters that can be adjusted to balance execution speed, memory usage, floating-point accuracy, and error checking. Detailed CPU timing and memory usage statistics are recorded, as well as flow statistics such as CFL number and the magnitude of the divergence.

The code can solve the diffusion equation, Stokes equations, or Navier–Stokes equations with arbitrary right-hand-side forcing function, and most combinations of Dirichlet, Neumann, or periodic boundary conditions are supported. Almost all of the numerical scheme variants listed in this thesis are supported, for example, BDF1 or Douglas or modified Douglas splitting, non-incremental or incremental pressure correction, and standard ($\chi = 0$) or rotational ($\chi > 0$) forms. Non-uniform grids are supported in most of the code.

C.1 Solving Tridiagonal Linear Systems

Since both the velocity and pressure require solving tridiagonal linear systems, it is important to efficiently utilize all CPUs to solve these systems quickly. For concreteness, suppose we have three CPUs and we want to solve the following 13x13 tridiagonal linear system (with possible corner entries for periodic domains):

$$\begin{bmatrix}
b_0 & d_0 & & & & & & & & & & & q_1 \\
a_0 & b & c & & & & & & & & & & \\
 & a & b & c & & & & & & & & & \\
 & & a & b & c_0 & & & & & & & & \\
 & & & e_0 & b_1 & d_1 & & & & & & & \\
 & & & & a_1 & b & c & & & & & & \\
 & & & & & a & b & c & & & & & \\
 & & & & & & a & b & c_1 & & & & \\
 & & & & & & & e_1 & b_2 & d_2 & & & \\
 & & & & & & & & a_2 & b & c & & \\
 & & & & & & & & & a & b & c & \\
 & & & & & & & & & & a & b & c_2 \\
q_2 & & & & & & & & & & & e_2 & b_3
\end{bmatrix}
\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix}
=
\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \\ f_{10} \\ f_{11} \\ f_{12} \end{bmatrix} \tag{C.1}$$

By simply re-ordering matrix rows and re-ordering the unknowns $x_0 - x_{12}$ as they appear in vector form, one can write the matrix equation (C.1) as:

$$\left[\begin{array}{cccccccccc|ccc}
b & c & & & & & & & & & a_0 & & \\
a & b & c & & & & & & & & & & \\
 & a & b & & & & & & & & & c_0 & \\
 & & & b & c & & & & & & & a_1 & \\
 & & & a & b & c & & & & & & & \\
 & & & & a & b & & & & & & & c_1 \\
 & & & & & & b & c & & & & & a_2 \\
 & & & & & & a & b & c & & & & \\
 & & & & & & & a & b & & & & \\
\hline
d_0 & & & & & & & & & & b_0 & & q_1 \\
 & & e_0 & d_1 & & & & & & & & b_1 & \\
 & & & & & e_1 & d_2 & & & & & & b_2 \\
 & & & & & & & & e_2 & & q_2 & & b_3
\end{array}\right]
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_5 \\ x_6 \\ x_7 \\ x_9 \\ x_{10} \\ x_{11} \\ \hline x_0 \\ x_4 \\ x_8 \\ x_{12} \end{bmatrix}
=
\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_5 \\ f_6 \\ f_7 \\ f_9 \\ f_{10} \\ f_{11} \\ \hline f_0 \\ f_4 \\ f_8 \\ f_{12} \end{bmatrix} \tag{C.2}$$

The matrix equation (C.2) is in block-tridiagonal form, with possible corner entries in the lower right block. We denote the blocks of (C.2) by

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \left[\begin{array}{c} X \\ \hline Y \end{array}\right] = \left[\begin{array}{c} F \\ \hline G \end{array}\right], \tag{C.3}$$

which is the same as

$$AX + BY = F, \tag{C.4}$$

$$CX + DY = G. \tag{C.5}$$

This linear system can be solved for the unknown vectors X and Y by using the Schur complement method. First, $X = A^{-1}(F - BY)$ can be obtained from (C.4) and then substituted into (C.5) to get $CA^{-1}(F - BY) + DY = G$. Then terms involving Y are isolated on the left hand side to obtain $(D - CA^{-1}B)Y = G - CA^{-1}F$. Denote the Schur complement matrix by $S = (D - CA^{-1}B)$ and denote the right-hand-side vector by $R = G - CA^{-1}F$. The vectors G and F contain finite-difference operators for the advection and explicit Laplacian (if the tridiagonal problem is for the momentum equation of Navier–Stokes) or finite-difference operators for the divergence (if the tridiagonal problem is for the pressure projection). In either case, these operators have stencil widths of several grid points, so on CPU boundaries the operators must be decomposed into parts. Since the advection operator is nonlinear, it cannot be decomposed as easily and some grid point values must be communicated in a neighbour-to-neighbour fashion beforehand. This is done using non-blocking MPI "ready" sends, and much of this communication is hidden while performing other calculations. Using the decomposed operators, each CPU computes a part of the Schur complement right-hand-side vector R and also a part of the Schur complement matrix S. Four floating point values for S and two floating point values for R are constructed by each CPU and sent to a single "master" CPU in order to assemble the linear system $SY = R$. This communication is performed using the MPI "gather" routine, which blocks until all CPUs have called it. The master CPU inverts $SY = R$ to obtain Y (where S is always tridiagonal in our case) and two floating point values of Y are sent back to each CPU using the MPI "scatter" routine, which blocks again

until all CPUs have called it. After each CPU receives the necessary parts of Y, the solution $X = A^{-1}(F - BY)$ can be obtained for the remaining unknowns by solving only local tridiagonal linear systems.

In order to efficiently solve a local tridiagonal linear system, the code contains several different routines that are all optimized for different situations. For example, solving $Ax = F$, $Ay = G$, $Az = H$ for the three unknown vectors x, y, z can be done about 30% faster by solving the single matrix problem $A(x, y, z) = (F, G, H)$. In situations where this can't be done but the same matrix still needs to be inverted more than once, the code optimizes the matrix on the first pass so that on the second problem, most of the work has already been done. This is similar to an LU factorization with some additional floating-point specific optimizations.

All of the communication for the parallel tridiagonal solver is done in one dimension only, but it is done for the entire perpendicular "area". In other words, on a computer with 1000 CPU cores that solves a 3D problem using $10 \times 10 \times 10$ CPUs, each communication would involve only 10 CPUs but 100 CPUs would be performing this communication at the same time. In this case, the Schur complement problem $SY = R$ associated with each tridiagonal problem contains only 10+1=11 unknowns. Even on machines with millions of CPU cores, $SY = R$ is a small problem that can be solved serially without any significant penalty. Also, in cases where the shape of the domain Ω_f is time-independent, the code supports the option to precompute and store the Schur complement matrix, which means that the communication required for solving tridiagonal systems in parallel is reduced from six floating-point values per grid line to two floating-point values per grid line.

C.2 Moving Rigid Objects

There are two ways to store moving rigid objects (particles) in computer memory on a computer system with many CPU cores. The first option is to redundantly store all particles on the shared memory associated with each CPU, and the second option is to distribute storage of the particles between the CPUs.

In the case where all particles are stored redundantly, each CPU can compute all collision sub-steps (see Section 6.5) without requiring any communication with other CPUs. However, all CPUs must still synchronize their objects every regular

time step otherwise floating-point inconsistencies can eventually cause each CPU to believe the objects exist at different positions, and this causes numerical instability.

In the case where the particle storage is distributed between CPUs, a given CPU is only aware of a small subset of particles at any given time. Each particle is always "owned" by exactly one CPU, but it may be visible by several CPUs if the particle is near a CPU boundary. Each CPU maintains a dynamic list of owned particles as well as a list of nearby "visible" particles that reside on adjacent CPUs. These lists need to be updated occasionally via neighbour-to-neighbour CPU communication, and some attributes of the particles (for example the velocity and position) need to be communicated during every collision sub-step in order for collisions to be properly identified and resolved. This communication is accomplished using non-blocking MPI "ready" sends and some of the communication is hidden while performing other calculations.

In order to efficiently identify pair-wise particle collisions, each CPU sorts all visible particles into buckets so that checking all particle pairs is done in $O(N)$ time rather than $O(N^2)$ time, where N is the number of particles.

In order to compute the fluid forces on each object, a surface integral must be computed as in Section 6.3. This is nontrivial to do in parallel because each velocity derivative in the fluid stress is a three-point stencil involving two fluid points that must be interpolated in a boundary-fitted way, but there is no guarantee that any of the grid points required by this complicated operation even exist on the CPU that needs to compute it. In order to handle this issue, the code abstracts the concept of a grid point such that grid points which do not exist on the processor can still be treated in the usual way. By exploiting linearity of operators, we can set the value at these "abstracted" grid points to zero, and then the complete surface integral works out to be the direct sum of what each CPU computes. However, care must still be taken to make sure that only one CPU contributes certain terms, otherwise they end up being counted more than once. Also, care must be taken to ensure that important floating-point conditional statements follow the same branch on every CPU. If not, the direct sum of the result computed by each CPU may be nonsense. All communication for the surface integrals is performed using non-blocking MPI "ready" sends.

C.3 Other Miscellaneous Optimizations

The code for the numerical scheme also contains many optimizations to reduce memory requirements and execution speed. For example, if two large-memory variables are never required at the same time, then these variables are stored at overlapping memory locations in order to reduce overall memory consumption. This trick reduces the overall memory usage by about 25%.

Cache efficiency is a big part of code speed on modern computers, and all loops in the code traverse multidimensional arrays in the cache-optimal order, with a few exceptions. Since tridiagonal linear systems need to be solved in each direction, only one of these directions can be cache-optimal, and traversing multidimensional arrays in the other two directions uses a large stride which hurts cache efficiency. In most of these cases, the code uses loop blocking / loop tiling to enhance cache performance.

In order to efficiently build the tridiagonal matrices for one-dimensional boundary-fitted operators, one must be able to efficiently determine intersection points of grid lines with a large number of particles. The code performs this task by assigning a number to each grid point which indicates whether or not the point is inside a particle, and if so, which particle. By looping over all particles and using bounding boxes to limit the number of grid points that need to be checked per particle, the process of marking each grid point is performed optimally in $O(N + M)$ time, where N is the total number of grid points and M is the total number of particles. When the matrix for a given grid line is constructed, grid points are are traversed along the line and the precomputed mapping of grid points to particles allows boundary-fitting to be performed in $O(1)$ time for each point, which means the matrix is built in $O(N)$ time, and the entire process takes $O(N + M)$ time, which is optimal.

The code also contains a substantial quantity of micro-optimizations. For example, inner-most loops use lookup tables for sin() and cos(), floating point divisions are precomputed, etc.

C.4 3D Viewer

When dealing with vector and scalar fields in many dimensions, clear and accurate data visualization is important. Therefore, I developed two visualization methods. The first is a simple "ascii art" drawing that the numerical code can output in order to precisely identify which grid nodes are considered fluid or solid. The second is

a 3D OpenGL-based visualization tool (3D Viewer) for viewing 2D or 3D velocity fields, pressure fields, and solid objects specifically in the context of the MAC grid. The 3D Viewer is interactive and allows the user to navigate the numerical data and view it from any angle or position. Zooming out allows at-a-glance visualization of macroscopic features of the flow fields, and zooming in shows numerical detail right down to individual grid points. The grid point accurate "zoom in" capability is extremely useful for identifying problems with the numerical scheme or bugs in the code. In particular, this tool allowed me to identify subtle implementation specific issues that would have otherwise gone unnoticed and ended up decreasing the accuracy of the scheme.

Using the 3D Viewer, each time step can be viewed as one of many "frames" that can be played forwards, backwards, or paused. The 3D Viewer can also output image sequences which can then be compiled by a third-party tool (such as ffmpeg) into videos. Several pictures in this thesis are screen-captures using the 3D Viewer tool, for example, Figures 9.5, 10.2, 10.3, 10.6.