An Immersed Interface Method for Incompressible Flow with Moving Boundaries and High Order Time Integration

by

James Gabbard

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2020

Author .....
Department of Mechanical Engineering
January 15, 2020

Certified by .....
Wim van Rees
Assistant Professor
Thesis Supervisor

Accepted by .....
Nicolas Hadjiconstantinou
Chairman, Department Committee on Graduate Theses

An Immersed Interface Method for Incompressible Flow with Moving Boundaries and High Order Time Integration

by

James Gabbard

Submitted to the Department of Mechanical Engineering
on January 15, 2020, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

In this work we present a novel Immersed Interface Method (IIM) for simulating two dimensional incompressible flows involving moving rigid bodies immersed in an unbounded fluid domain. To do so, we solve the Navier-Stokes equations in vorticity-stream function form, using a second order IIM spatial discretization that allows for the use of high order explicit Runge-Kutta time integration.

We begin by reviewing existing work on the immersed interface method, and developing novel algorithms for stencil calculation, geometry processing, and integration over irregular domains. We then introduce a stable IIM discretization of the advection-diffusion equation, and describe an improved version of the IIM Poisson solver developed by Gillis [9]. We review vorticity-based formulas for calculating the local tractions and global forces acting on an immersed body, and present a novel extension of the control-volume force calculation methods developed by Noca [16]. This first section culminates in the presentation of an IIM Navier Stokes solver for problems on stationary domains, which is shown to have second-order spatial accuracy and third-order temporal accuracy.

The second portion of this work develops a general IIM framework for discretizing PDEs on moving domains. We focus on schemes that are compatible with explicit high-order Runge-Kutta methods, and demonstrate that our method introduces a mixed spatial-temporal error term not seen in stationary IIM discretizations. We also consider CFL-like restrictions that limit the maximum time step used in problems with moving domains, and develop geometric criteria to ensure that these restrictions are met. Using these new methods, we extend our existing IIM Navier Stokes solver to allow for moving boundaries, and verify that the method retains its second-order spatial and third order temporal accuracy. Finally, we demonstrate the applicability of the algorithm to complex two-dimensional flow problems by calculating the time-dependent lift, thrust, and moment coefficients of a flapping airfoil.

Thesis Supervisor: Wim van Rees
Title: Assistant Professor

Contents

Chapter 1

Introduction

The natural world contains abundant examples of living creatures that propel themselves through an incompressible fluid, be it birds in the sky or fish that traverse the open ocean. There are strong economic incentives for human beings to do the same, and we have been successful so far in developing commercial aircraft, ships and submarines. However, there remains a fundamental gap between animal propulsion strategies and human propulsion strategies. While the creatures around us move with great agility via complex deformations of their bodies, man-made vehicles remain largely static, and are propelled by the rotary motion of propellers or the acceleration of fluid in a jet engine.

The study of biological propulsion strategies is an attempt to bridge this fundamental gap. By studying the locomotion patterns of natural creatures, we too can learn the lessons taught to them by millions of years of evolutionary time, and use this knowledge to improve the efficiency of our own propulsion systems. Computational fluid dynamics is an essential part of this pursuit, since experimental work with living creatures is difficult, expensive, and at times ethically dubious. Given an accurate measurement of an animal, computational methods allow us to reconstruct the flow field around that animal with great accuracy.

Unfortunately, there are a variety of factors that make biological propulsion difficult to simulate. While many problems in aerodynamics can be adequately described in a steady or time-averaged way, traditional swimming and flapping motions are inherently unsteady problems. Compounding this, many of the most effective methods in CFD require the use of a mesh that conforms to the fluid domain, leading to an unacceptably high mesh-generation cost for problems involving deforming geometries. Many creatures inhabit a fluid domain that is effectively unbounded, a fact that must be handled correctly to accurately capture the fluid dynamics at play.

Vortex methods provide an effective answer to many of these issues, and there are a wealth of recent papers that successfully simulate biological flows by tracking the vorticity field. The vorticity-velocity form of the Navier-Stokes equations naturally handles unbounded domains, and for many biologically-inspired flows the vorticity field is effectively localized on the object's surface and in key vortex structures in its wake. The use of vortex-particle methods, which reformulate these equations in a Lagrangian way, allows for efficient advection strategies with relaxed stability criteria compared to traditional Eulerian formulations.

Vortex methods alone do not provide a clear path forward for the simulation of deforming geometries. To avoid the use of body-fitted grids, we focus on immersed methods, which operate on a regular grid that need not be coincident with solid boundaries. There are a whole zoo of these methods, including penalization methods, immersed boundary methods, the ghost fluid method, cut-cell methods, and the immersed interface method (IIM). Among these, there is no single method which is clearly superior; all represent a trade off between simplicity, accuracy, and computational efficiency.

The starting point for the material presented here is the work of Yves Marichal [15] and Thomas Gillis [10], which merges a re-meshed vortex-particle method with an IIM boundary treatment to simulate flow past stationary bluff bodies in two and three dimensions. In this thesis we extend these

immersed interface methods to flow problems with moving boundaries, a significant step towards fluid-structure interaction and other biological propulsion problems. To achieve this, we replace the vortex-particle method with a finite difference based transport scheme, and develop improved algorithms for IIM stencil calculations, IIM-based geometry processing, integration over irregular domains, and force calculation in incompressible flow problems. The presentation of this material proceeds as follows:

- **Chapter 2** begins with a traditional presentation of the Explicit Jump Immersed Interface Method (EJIIM), and offers some conceptual simplifications that lead to more efficient stencil computation. We then present a novel algorithm for efficient geometry processing, and discuss techniques for accurate integration over irregular domains.
- **Chapter 3** presents a Von Neumann stability analysis of several finite difference discretizations of the advection diffusion equation. We then present a novel immersed interface boundary treatment, and demonstrate that the resulting discretization is stable over a broad range of Péclet numbers.
- **Chapter 4** improves an existing IIM Poisson solver developed by Gillis in [9]. We then motivate and describe the local vorticity boundary condition used successfully in [10] and in the remainder of this thesis.
- **Chapter 5** is intended as convenient, self-contained reference for calculating global and local forces from the vorticity-velocity formulation of the Navier-Stokes equations. We also present a novel control volume formulation for calculating aerodynamic moments.
- **Chapter 6** presents an IIM Navier Stokes solver built from material presented in the previous four chapters. The convergence of this solver is measured, and the resulting numerical solutions are compared with computational reference data from other authors.
- **Chapter 7** develops a novel method for handling moving bodies with the immersed interface method, allowing for explicit high-order time integration.
- **Chapter 8** demonstrates the validity of an IIM Navier Stokes solver for moving bodies. We verify the spatial and temporal convergence properties of the solver, and compare the resulting numerical results to existing reference data. Finally, we conclude by using this solver to simulate a flapping airfoil, a flow problem that is relevant to biological propulsion.

In this thesis, no attempt is made to achieve complete mathematical rigour. We will frequently motivate our discretizations with heuristics, and arrive at results via quick arguments instead of watertight proofs. It is exceedingly difficult to find a stability proof for a fully-discrete Navier stokes solver on a multidimensional irregular domain, and we will not attempt proofs of this sort here. Instead, we demonstrate the stability and convergence of our solvers through extensive numerical experimentation, in hopes that the efficiency and accuracy of the algorithms presented here will make up for the lack of absolute rigour in their presentation.

Chapter 2

The Immersed Interface Method

The Explicit Jump Immersed Interface Method (EJIIM), introduced by Wiegmann and Bube in [25], is a method of correcting finite difference schemes to account for jump discontinuities in the solution to a PDE. The method has been successful in treating a wide range of problems, especially those in which the magnitude of the jump discontinuity can be derived from the governing PDE. As an example, the Young-Laplace equation states that the pressure jump across a curved fluid interface is given by $[p] = 2\gamma H$, where γ is a surface tension parameter and H is the mean curvature of the interface. In an immersed interface method, this knowledge can be used to more accurately discretize fluid-fluid interfaces that do not align with the computational grid.

The EJIIM can also been extended problems involving irregular geometries immersed in a Cartesian grid, by treating the domain boundary as a jump discontinuity. This is the scenario considered in this thesis, where the Navier-Stokes equations in vorticity-velocity form are solved in the vicinity of an irregularly shaped object immersed in a Cartesian grid. This particular application of the EJIIM has been explored several times over the last 15 years [14, 15, 10], and produces solutions that maintain second-order or higher convergence properties at the fluid-solid boundary.

In this chapter, we revisit the traditional presentation of the one-dimensional EJIIM, and demonstrate that it can be simplified considerably when applied to the problem of irregular domains. We then develop some notation for multidimensional immersed interface problems, and use it to discretize a model two-dimensional PDE. Finally, we describe an efficient algorithm for processing implicitly defined geometries, and conclude with a novel treatment of integration using immersed-interface methods.

2.1 Explicit Jump IIM

2.1.1 Generalized Taylor Series

Traditional finite difference methods fail on discontinuous functions because the Taylor series approximations they are built on are only valid for functions with suitable smoothness properties. To extend finite difference methods to discontinuous problems, the authors of [25] propose a modified Taylor series which correctly approximates functions with jump discontinuities, provided that the jumps in the function and its derivatives are of known magnitude. Consider a function $f : \mathbb{R} \to \mathbb{R}$ that is smooth except at a point x_α, where there is a jump singularity in f and its derivatives $f^{(k)}$. Let $[f^{(k)}]$ denote the magnitude of the jump in the k^{th} derivative of f, so that

$$[f^{(k)}] = \lim_{x \to x_\alpha^+} f^{(k)} - \lim_{x \to x_\alpha^-} f^{(k)} = (f^{(k)})^+ - (f^{(k)})^-. \tag{2.1}$$

Teh function f can be expanded about $x < x_\alpha$ using a generalized Taylor series that accounts for the discontinuity:

$$\begin{aligned} f(x+h) &= \sum_{k=0}^{n} \frac{h^k}{k!} f^{(k)}(x) + J_\alpha + O(h^{n+1}), \\ J_\alpha &= \sum_{k=0}^{n} \frac{(h^+)^k}{k!} [f^{(k)}]. \end{aligned} \tag{2.2}$$

Here $h^+ = (x+h) - x_\alpha$ is the distance from the singularity to the evaluation point ($h^+ > 0$). The first half of (2.2) is a standard Taylor expansion of f about x; the second is a "jump correction" that must be added whenever the Taylor expansion crosses the discontinuity. For points $x > x_\alpha$, we use a similar expansion:

$$\begin{aligned} f(x-h) &= \sum_{k=0}^{n} (-1)^k \frac{h^k}{k!} f^{(k)}(x) - J_\alpha + O(h^{n+1}), \\ J_\alpha &= \sum_{k=0}^{n} (-1)^k \frac{(h^-)^k}{k!} [f^{(k)}]. \end{aligned} \tag{2.3}$$

Here $h^- = x_\alpha - (x-h)$ is the distance from the evaluation point to the singularity ($h^- > 0$). Both of these expansions are illustrated in Figure 2-1

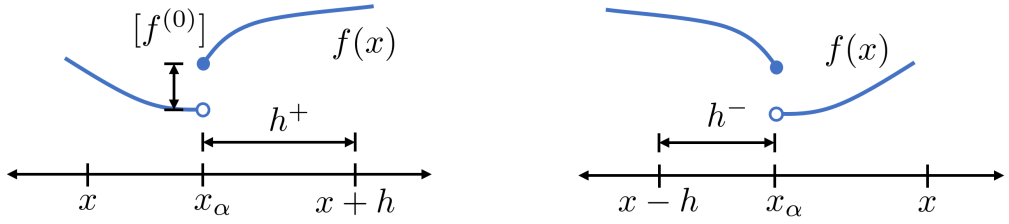


Figure 2-1: Setup for Equation 2.2 (left) and for Equation 2.3 (right).

A formal derivation of (2.2) and (2.3) is given in [25], but they can be motivated quite simply. Consider a function which is equal to $f(x)$ for $x < x_\alpha$ and equal to $g(x)$ for $x > x_\alpha$. The jump correction J is simply a Taylor expansion of $(g-f)(x)$ about the point x_α, which is added to a Taylor approximation of $f(x)$ whenever we would like to approximate $g(x)$ instead.

2.1.2 Jump-Corrected Finite Differences

In the EJIIM, the generalized Taylor series developed above are used to construct jump-corrected finite difference schemes. To begin this process, define a uniform grid spacing h and a origin $x_0 \in \mathbb{R}$, and for $i \in \mathbb{Z}$ let $x_i = x_0 + ih$. Consider a smooth function $f(x)$, which is regularly sampled to give values $f_i = f(x_i)$. We can approximate the second derivative of f using the standard second-order centered finite difference stencil $f_0^{(2)} = (f_{-1} - 2f_0 + f_1)/h^2 + \mathcal{O}(h^2)$, which can be derived by writing f_1 and f_{-1} as a Taylor expansion about f_0:

$$\begin{aligned} f_{-1} &= f_0 - h f_0^{(1)} + h^2 f_0^{(2)}/2 - h^3 f_0^{(3)}/6 + \mathcal{O}(h^4); \\ f_1 &= f_0 + h f_0^{(1)} + h^2 f_0^{(2)}/2 + h^3 f_0^{(3)}/6 + \mathcal{O}(h^4). \end{aligned}$$

Summing the two expansions and subtracting $2f_0$ gives the expected result, which is valid so long as $f \in C^4[x_{-1}, x_1]$. If f has a jump discontinuity at $x_\alpha \in (x_0, x_1)$, then we lose the second-order accuracy of the approximation. To regain this accuracy, we must re-expand f_1 using the generalized Taylor series

$$f_1 = f_0 + h f_0^{(1)} + h^2 f_0^{(2)}/2 + h^3 f_0^{(3)}/6 + J_\alpha + \mathcal{O}(h^4),$$

with J_α defined as in (2.2) and $h^+ = x_1 - x_\alpha$. Adding in an expansion of f_{-1} and subtracting $2f_0$ gives the modified finite difference stencil

$$\begin{aligned} f^{(2)}(x_0) &= \frac{1}{h^2}[f_{-1} - 2f_0 + (f_1 - J_\alpha)] + \mathcal{O}(h^2), \\ J_\alpha &= \sum_{k=0}^{3} \frac{(h^+)^k}{k!}[f^{(k)}]. \end{aligned} \tag{2.4}$$

For a discontinuity at $x_\alpha \in (x_{-1}, x_0)$, we can carry out the same procedure to obtain

$$\begin{aligned} f^{(2)}(x_0) &= \frac{1}{h^2}[(f_{-1} + J_\alpha) - 2f_0 + f_1] + \mathcal{O}(h^2), \\ J_\alpha &= \sum_{k=0}^{3} (-1)^k \frac{(h^-)^k}{k!}[f^{(k)}], \end{aligned} \tag{2.5}$$

with $h^- = x_\alpha - x_{-1}$.

2.1.3 A One-dimensional Example: The Diffusion Equation

To illustrate the concepts presented above, consider an initial-boundary value problem (IBVP) for the diffusion equation on the interval $\Omega = [x_\alpha, x_\beta]$, with Dirichlet boundary conditions:

$$\begin{aligned} \frac{\partial f}{\partial t} &= \nu \frac{\partial^2 f}{\partial x^2}, \\ f(x, 0) &= f_0(x), \\ f(x_\alpha, t) &= g(t), \\ f(x_\beta, t) &= h(t). \end{aligned} \tag{2.6}$$

To approximate the solution to this IBVP, we use the method of lines, in which the system is discretized spatially and the resulting system of ODEs is integrated with a standard numerical integrator. We choose $N+1$ points $x_i \in [0, 1]$ separated by a constant grid spacing $h = 1/N$, so that $x_i = ih$. For simplicity, we can assume that $x_{-1} < x_\alpha < x_0$ and $x_N < x_\beta < x_{N+1}$, so that the entire problem domain is contained within $[x_{-1}, x_{N+1}]$. Let $f_i(t)$ be the value of the numerical solution at point x_i, with initial value $f_i(0) = f_0(x_i)$. Similarly, let f_α and f_β be the numerical solution at x_α and x_β. The given boundary conditions require that $f_\alpha(t) = g(t)$ and that $f_\beta(t) = h(t)$. Since $f_{-1}(t)$ and $f_{N+1}(t)$ are not in the problem domain, we assume that $f_{-1}(t) = f_{N+1}(t) = 0$. This setup is shown in Figure 2-2.

For $1 < i < N - 1$, we approximate (2.6) with the second-order finite difference stencil

$$\frac{\mathrm{d}f_i}{\mathrm{d}t} = \frac{\nu}{h^2}(f_{i-1} - 2f_i + f_{i+1}). \tag{2.7}$$

This approximation breaks down at the boundary points x_0 and x_N, because it requires values of f that lie outside of the interval $[x_\alpha, x_\beta]$. To remedy this, we consider approximating the second derivative of a function

$$\tilde{f}(x) = \begin{cases} 0, & x \notin [x_\alpha, x_\beta] \\ f(x), & x \in [x_\alpha, x_\beta] \end{cases}. \tag{2.8}$$

This function has jump discontinuities at x_α and x_β, and can be expanded using the EJIIM. At x_α, the value of any jump $[\tilde{f}^{(k)}]$ is equal to the function value $\tilde{f}^{(k)}(x_\alpha)$. The Dirichlet boundary condition specifies a value for $\tilde{f}^{(0)}(x_\alpha)$, but for $k > 0$ the jumps must be interpolated using existing values of $\tilde{f}$ that lie on the problem domain. Using methods described in [15], we can construct a set

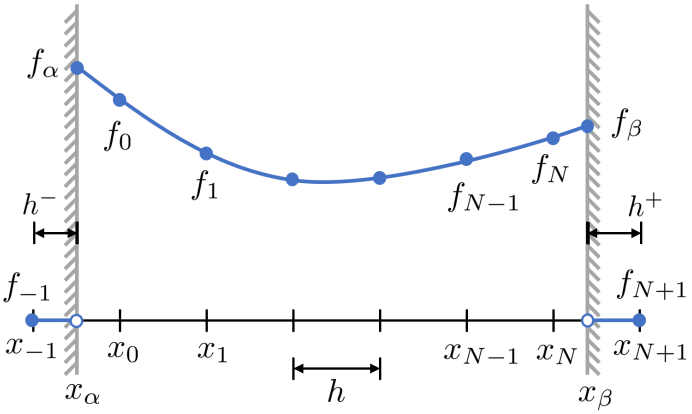


Figure 2-2: The model diffusion problem, discretized on a regular grid with irregular boundaries.

of interpolation coefficients $S_{k,i}$ so that

$$f^{(k)}(x_\alpha) = S_{k,\alpha} f(x_\alpha) + \sum_{i=1}^{3} S_{k,i} f_i + \mathcal{O}\big(h^{4-k}\big) \quad \text{for all} \quad f \in C^4[x_\alpha, x_\beta]. \tag{2.9}$$

These coefficients vary with the irregular spacing $h^- = x_\alpha - x_{-1}$, and the point x_0 is omitted from the calculation to prevent the interpolation from becoming ill conditioned when $x_0 - x_\alpha$ is small. Applying (2.5) with $f_{-1} = 0$, we obtain

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} f_0(t) &= \frac{\nu}{h^2}(J_\alpha - 2f_0 + f_1), \\ J_\alpha &= \sum_{k=0}^{3} (-1)^k \frac{(h^-)^k}{k!} f^{(k)}(x_\alpha) \\ &= f_\alpha + \sum_{k=1}^{3} \frac{(h^-)^k}{k!} \left(S_{k,\alpha} f_\alpha + \sum_{i=1}^{3} S_{k,i} f_i \right) \end{aligned} \tag{2.10}$$

A similar procedure can be conducted at x_β, using a new set of coefficients $S_{k,i}$ which depend on the spacing $h^+ = x_{N+1} - x_\beta$. Applying (2.4) with $f_{N+1} = 0$, we obtain

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} f_N(t) &= \frac{\nu}{h^2}(f_{N-1} - 2f_N + J_\beta), \\ J_\beta &= \sum_{k=0}^{3} \frac{(h^+)^k}{k!} f^{(k)}(x_\beta) \\ &= f_\beta + \sum_{k=1}^{3} \frac{(h^+)^k}{k!} \left(S_{k,\beta} f_\beta + \sum_{i=1}^{3} S_{k,i} f_{N-i} \right) \end{aligned} \tag{2.11}$$

This completes the second order spatial discretization of the original IBVP, and provides a linear system of ODEs for the unknown functions $f_i(t)$ which can be integrated to find an approximate solution to the continuous problem.

2.1.4 Implementation

The immersed interface method outlined above uses modified stencils only at the boundary-adjacent points x_0 and x_N. Consequently, we can implement this immersed interface method as a small modification to an existing finite-difference scheme. There are two main strategies we could use to make this modification. Both begin by pre-computing and storing the stencil coefficients $S_{k,i}$, and in both the time derivatives f_0' through f_N' are computed with the regular stencil (2.7) at each time step. To account for the irregular boundaries, we choose one of the following:

- Before computing f_0' and f_N', we set $f_{-1} = f_{N+1} = 0$. Afterwards, we compute the jump corrections $\nu J_\alpha/h^2$ and $\nu J_\beta/h^2$, and add these corrections to f_0' and f_N' respectively.
- Alternatively, we begin each time step by computing the jump corrections, then set $f_{-1} = J_\alpha$ and $f_{N+1} = J_\beta$. When we compute f_0' and f_N' with regular finite difference stencils, the jump corrections are automatically included.

We will refer to these as the post-correction and pre-correction schemes, respectively. In one dimension both procedures are equivalent. In higher dimensions the two approaches produce different discretizations, and we will choose to use one or the other depending on the type of problem at hand. The key takeaway from this example is that immersed interface methods operate in tandem with an existing finite difference scheme, and consume only a small amount of additional computational resources.

2.2 Simplifying Jump Corrections with Interpolating Polynomials

Before moving to higher-dimensional problems, we will take some time to streamline the computation of jump corrections. In a series of papers beginning in 2011 [4, 6, 3, 5], Brehm and Fasel simplify (2.10) and (2.11) by noting that both the finite difference stencil and jump corrections are calculated using a linear combination of the the solution values f_i. This allows the two to be combined into a single irregular finite difference stencil with coefficients C_i, so that (2.10) becomes

$$\frac{\mathrm{d}}{\mathrm{d}t} f(x_0, t) = \frac{\nu}{h^2}\left(C_\alpha f_\alpha + \sum_{i=0}^{3} C_i f_i\right) + \mathcal{O}\big(h^2\big). \tag{2.12}$$

This stencil contains five distinct evaluations of f, even though only four are required to obtain a second order approximation. Consequently, the authors are able to derive a system of four linear equations which the five C_i must satisfy to maintain $\mathcal{O}\big(h^2\big)$ accuracy. Solving these equations yields a one-parameter family of stencils $C_i(\gamma)$ that all provide the same formal order of accuracy. The authors then attempt to optimize over γ to obtain a stencil with improved stability properties, and provide some heuristics for choosing optimal γ in more complex settings. This approach eliminates the unnecessary complexity of $S_{k,i}$ coefficients, and dispenses with the need to invoke the EJIIM when deriving a spatial discretization. However, it also loses the convenient idea of small additive corrections to a standard discretization, opting instead to fundamentally alter the calculations performed at the boundary points.

In this thesis we will take an intermediate stance, and continue to use the language of jump corrections while moving away from the EJIIM and the calculation of the $f^{(k)}$ through $S_{k,i}$ stencils. To do so, we revisit the definition of the jump corrections J_α, reproduced here for convenience:

$$\begin{aligned} f(x+h) &= \sum_{k=0}^{n} \frac{h^k}{k!} f^{(k)}(x) + J_\alpha \\ J_\alpha &= \sum_{k=0}^{n} \frac{(h^+)^k}{k!} [f^{(k)}] + \mathcal{O}\big(h^{n+1}\big). \end{aligned}$$

Assume now that a single Dirichlet boundary condition f_α is given at x_α, and that the function f is identically zero for $x > x_\alpha$. If we know the solution values f_i at several points x_i near the boundary, then we can construct a polynomial of degree n which interpolates f at x_α and n of its neighbors x_i (Figure 2-3). If f is at least $n+1$ times differentiable, then this polynomial is guaranteed to approximate f and its derivatives, in that

$$f^{(k)}(x) - p_n^{(k)}(x) \sim \mathcal{O}\big(h^{n+1-k}\big) \quad \text{for all} \quad 0 \le k \le n. \tag{2.13}$$

If we replace f with p_n when calculating J_α, we find that

$$\begin{aligned} J_\alpha &= \sum_{k=0}^{n} \frac{(h^+)^k}{k!} \Big(p_n^{(k)}(x_\alpha) + \mathcal{O}\big(h^{n+1-k}\big) \Big) \\ &= \sum_{k=0}^{n} \frac{(h^+)^k}{k!} p_n^{(k)}(x_\alpha) + \mathcal{O}\big(h^{n+1}\big). \end{aligned} \tag{2.14}$$

The polynomial p_n is defined on all of $\mathbb{R}$, and is exactly equal to its $(n+1)$ term Taylor expansion about any point x_0:

$$p_n(x) = \sum_{k=0}^{n} \frac{(x-x_0)^k}{k!} p_n^{(k)}(x_0). \tag{2.15}$$

Combining (2.14) and (2.15), the jump correction can be written succinctly as

$$J_\alpha = p_n(x_{-1}) + \mathcal{O}\big(h^{n+1}\big),$$

demonstrating that J_α is approximated to order h^{n+1} by the value of a degree n interpolating polynomial. This offers a very simple interpretation of the immersed interface method: whenever a finite difference stencil requires a value outside of the domain, we reconstruct this value using a polynomial extrapolation from inside the domain.

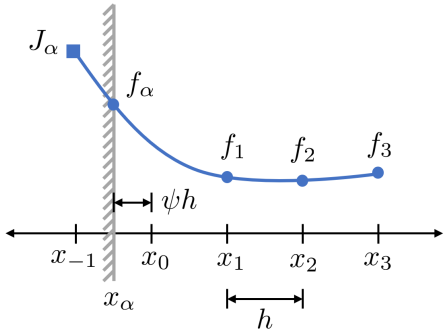


Figure 2-3: Fourth order jump correction calculated using polynomial extrapolation.

The polynomial $p_n(x)$ does not need to be constructed explicitly. Because we are only interested in the single value $p_n(x_{-1})$, we can construct an interpolation stencil to compute only this value. For convenience, define an index set $\mathcal{I} = \{\alpha, 1, 2, ...n\}$ so that $\{x_i\}_{i\in\mathcal{I}}$ includes the intersection point, excludes its immediate neighbor in the domain, and includes the next n points on that grid line (Figure 2-3). We seek an $n+1$ point stencil s_i satisfying

$$J_\alpha = \sum_{i\in\mathcal{I}}^{n} s_i f_i. \tag{2.16}$$

One easy way to construct s_i is with a basis of Lagrange interpolating polynomials ℓ_i, which are a set of n-th degree polynomials satisfying $\ell_i(x_j) = \delta_{ij}$. Then $p_n(x) = \sum_i f_i \ell_i(x)$ and $p_n(x_{-1}) = \sum_{i\in\mathcal{I}} f_i \ell_i(x_{-1})$. Using the Lagrange interpolation formula, we can write

$$s_j = \ell_j(x_{-1}) = \prod_{i\neq j} \frac{(x_{-1} - x_i)}{(x_j - x_i)}. \tag{2.17}$$

Alternatively, we can formulate a small linear system that determines s_i. Let $p_k(x)$ be a polynomial of degree less than or equal to n. Since any such polynomial is exactly equal to the degree n polynomial taking the same values on $\{x_i\}_{i\in\mathcal{I}}$, our coefficients must satisfy $p_k(x_{-1}) = \sum_{i\in\mathcal{I}} s_i p_k(x_i)$. Choosing $n+1$ such polynomials P_i, all linearly independent, we can form at a system of $(n+1)$ linear equations in $(n+1)$ unknowns:

$$\sum_j P_i(x_j) s_j = P_i(x_{-1}). \tag{2.18}$$

Because the only free parameter in the set $\{x_i\}_{i\in\mathcal{I}}$ is the non-dimensional intersection distance $\psi = (x_0 - x_\alpha)/h$, our stencil s_i can generally be expressed as a function of ψ. As an example, consider the fourth-order jump correction shown in Figure 2-3; by choosing the set of monomials $P_k = (x - x_\alpha)^k/h^k$, the linear system (2.18) can be reduced to

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & (1+\psi) & (2+\psi) & (3+\psi) \\ 0 & (1+\psi)^2 & (2+\psi)^2 & (3+\psi)^2 \\ 0 & (1+\psi)^3 & (2+\psi)^3 & (3+\psi)^3 \end{bmatrix} \begin{bmatrix} s_\alpha \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 1 \\ \psi - 1 \\ (\psi-1)^2 \\ (\psi-1)^3 \end{bmatrix}. \tag{2.19}$$

This method generalizes well to Neumann boundary conditions, and is the method of choice for all of the stencil calculations in this thesis. In practice, the cost of stencil computation is vanishingly small compared to other finite difference operations, so any convenient interpolation method will perform acceptably well.

The shift from $S_{k,i}$ stencils to direct polynomial interpolation reduces computational overhead, emphasizes the equivalence between jump corrections and extrapolated function values, and more clearly exposes the continuous polynomial approximation that underlies the discrete finite-difference stencil. With this tool at our disposal, we are fully prepared to move from one-dimensional problems to irregular domains in two dimensions.

2.3 Extending the IIM to Two Dimensions

The immersed interface method extends well to multidimensional operators that do not involve mixed derivatives, like the gradient and the Laplacian. Let Ω be an irregular two-dimensional domain, and let the $\mathcal{G}$ be the set containing all the nodes of a uniform Cartesian grid. To discuss the discretization of multidimensional immersed interface schemes, we define some convenient terminology and notation:

- **Control points** (denoted by $\mathcal{C}$) are intersections between the Cartesian grid and the immersed boundary $\partial\Omega$. In two dimensions, these can be subdivided into two disjoint sets $\mathcal{C}_x$ and $\mathcal{C}_y$, which consist of intersections involving grid lines parallel to the x and y axes, respectively.

- **Affected points** (denoted by $\mathcal{A}$) are regular grid points that are adjacent to a control point. These can be subdivided into two disjoint sets A^+ and $\mathcal{A}^-$, which represent points that are inside and outside Ω, respectively.

- **Interpolation points** (denoted by $\mathcal{I}_n$) are points that can be used to interpolate the value of a function at the control points $\mathcal{C}$. This set varies with the order of the immersed interface operation.

The spaces $\mathcal{C}$ and $\mathcal{A}$ are shown in Figure 2-4. There are a natural maps $A^+ : \mathcal{C} \to \mathcal{A}^+$ and $A^- : \mathcal{C} \to \mathcal{A}^-$ which link each control point to the affected points that lie adjacent to it. These mappings are not injective: any affected point can be adjacent to multiple control points, and it is quite common for an affected point to have two or even three such neighbors.

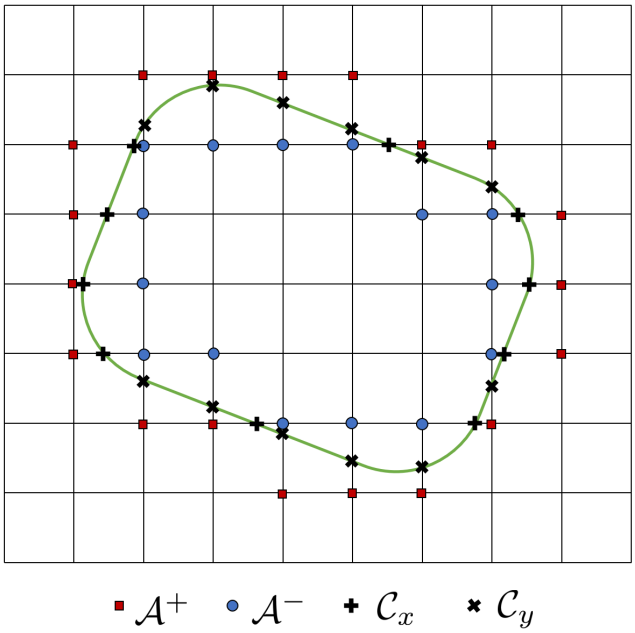


Figure 2-4: A typical 2D IIM discretization, including the control points $\mathcal{C}_x$ (+) and $\mathcal{C}_y$ (x) and the affected points $\mathcal{A}^+$ (□) and $\mathcal{A}^-$ (○).

In general, an intersection wiht $\partial\Omega$ can occur in any orientation and along any coordinate line. To avoid a confusing set of direction and sign conventions, all immersed interface operations associated with the control point $\mathbf{x}_c \in \mathcal{C}$ are considered in a local coordinate system (ξ, η), where ξ is the intersection direction and η is the transverse direction. This system is characterized by unit vectors $\hat{\boldsymbol{\xi}}$ and $\hat{\boldsymbol{\eta}}$, which are chosen so that the inward facing normal vector $\hat{\mathbf{n}} = [n_\xi, n_\eta]$ has non-negative components. Note that this system may not be right-handed. We will define a local origin $\mathbf{x}_{0,0} = A^+(\mathbf{x}_c)$, and relabel the nearby grid points so that $\mathbf{x}_{a,b} = \mathbf{x}_{0,0} + ah\hat{\boldsymbol{\xi}} + bh\hat{\boldsymbol{\eta}}$ whenever local stencils are discussed. Analogously, we relabel the local function values $f_{a,b} = f(\mathbf{x}_{a,b})$. These conventions are illustrated in Figure 2-5.

2.3.1 Jump Corrections in Two Dimensions

As an example of higher-dimensional IIM, we consider discretizing the Laplacian operator at the point $\mathbf{x}_{0,0} \in \mathcal{A}^+$ shown in Figure 2-5. Working in the local (ξ, η) coordinate system, we can write

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = \frac{\partial^2 f}{\partial \xi^2} + \frac{\partial^2 f}{\partial \eta^2}.$$

This simple conversion comes from the fact that the Laplacian is invariant under sign changes $x \to -x$ and $y \to -y$, as well as the excange $x \leftrightarrow y$. Using the standard second-order centered stencil, we are free to write

$$\frac{\partial^2 f}{\partial \eta^2} = \frac{1}{h^2}(f_{0,-1} - 2f_{0,0} + f_{0,1}) + \mathcal{O}(h^2).$$

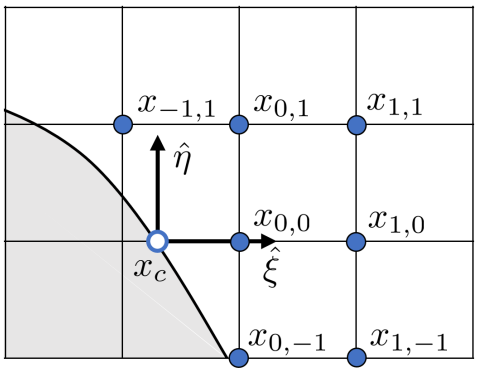


Figure 2-5: Standardized reference frame for multidimensional immersed interface problems.

In the ξ direction, we lack a function value $f_{-1,0}$, and cannot make the same standard approximation. Instead we treat the computation of $\partial^2 f/\partial\xi^2$ as its own one-dimensional immersed interface problem on the grid line $\mathbf{x}_{i,0}$ (Figure 2-5). After calculating the non-dimensional intersection distance $\psi = |\mathbf{x}_{0,0} - \mathbf{x}_c|/h$, we solve the linear system (2.19) to obtain stencil coefficients $s_{\xi,i}$. We can then write

$$\frac{\partial^2 f}{\partial \xi^2} = \frac{1}{h^2}(J_\alpha - 2f_{0,0} + f_{0,1}) + \mathcal{O}(h^2), \quad \text{with} \quad J_\alpha = \sum_{i\in\mathcal{I}} s_{\xi,i} f_{i,0}.$$

This completes the IIM discretization of the Laplacian for any affected point that is adjacent to only one control point. In two or more dimensions, we must also consider affected points that are adjacent to multiple control points. Luckily, a little bit of geometric reasoning shows that the one-dimensional finite difference stencils which require jump corrections are in one-to-one correspondence with the control points. Instead of looping over $\mathcal{A}^+$ and seeking corrections from neighboring control points, it is sufficient to loop over every $\mathbf{x}_c \in \mathcal{C}$ and correct only the local ξ-direction finite difference at $A^+(x_c)$. This is extremely convenient for implementation: a list of intersection points is equivalent to a list of corrections which must be made to the standard finite difference scheme.

2.3.2 Revisiting the Diffusion Equation in Two Dimensions

Here we consider the discretization of a model PDE with the immersed interface method, now in two dimensions. Consider an irregular region $\Omega \in \mathbb{R}^2$ with boundary $\partial\Omega$. A well-posed IBVP for the diffusion equation on this domain is

$$\begin{aligned} \frac{\partial f}{\partial t} &= \nu\left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial x^2}\right), \\ f(\mathbf{x}, 0) &= f_0(\mathbf{x}) \quad \text{for} \quad \mathbf{x} \in \Omega, \\ f(\mathbf{x}, t) &= g(\mathbf{x}, t) \quad \text{for} \quad \mathbf{x} \in \partial\Omega. \end{aligned} \tag{2.20}$$

We once again use the method of lines to approximate this continuous problem with a system of ODEs. Consider a regular grid of points $\mathbf{x}_{i,j}$ with uniform grid spacing h, and let $f_{i,j}(t)$ be the value of the numerical solution at point $\mathbf{x}_{i,j}$. We use the initial conditions $f_{i,j}(0) = f_0(\mathbf{x}_{i,j})$ for points in Ω, and $f_{i,j}(0) = 0$ otherwise.

The calculation begins by determining the set of control points $\mathcal{C}$. An efficient way to do this is described in section 2.4.1, but for now this step will be taken for granted. At each control point $\mathbf{x}_c \in \mathcal{C}$, we record the associated affected points $A^+(\mathbf{x}_c)$ and $A^-(\mathbf{x}_c)$, the local coordinate vectors $\hat{\xi}$, $\hat{\eta}$, and the non-dimensional intersection distance ψ, which can be used to construct the the jump stencil $s_{\xi,i}(\psi)$.

Once this is done, we begin the time-marching procedure. At each time step, we set $f_{i,j}(t) = 0$

for every $\mathbf{x}_{i,j} \notin \Omega$, and update the Dirichlet boundary condition $g(\mathbf{x}_c, t)$ at each $\mathbf{x}_c \in \mathcal{C}$. Time derivatives $f'_{i,j}$ for all $\mathbf{x}_{i,j} \in \Omega$ are calculated using the standard five-point stencil

$$\frac{\mathrm{d}}{\mathrm{d}t} f_{i,j} = \frac{\nu}{h^2}(f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1} - 4f_{i,j}).$$

We then revisit each $\mathbf{x}_c \in \mathcal{C}$, taking advantage of the one-to-one correspondence between intersection points and necessary jump corrections. Each jump correction is calculated using

$$J_\alpha(\mathbf{x}_c) = s_{\xi,\alpha} g(\mathbf{x}, t) + \sum_{i=1}^{3} s_{\xi,i} f_{\xi,i}, \tag{2.21}$$

and at the associated point $\mathbf{x}_{i,j} = A^+(\mathbf{x}_c)$ we make the additive correction

$$\frac{\mathrm{d}}{\mathrm{d}t} f_{i,j} \to \frac{\mathrm{d}}{\mathrm{d}t} f_{i,j} + \frac{\nu}{h^2} J_\alpha(\mathbf{x}_c).$$

This completes the spatial discretization of the problem, giving a set of linear ODEs that can integrated to approximate the solution of (2.20).

The above is a straightforward generalization of the one-dimensional post-correction scheme to two-dimensions. The analogous pre-correction scheme introduces one additional complexity. One can imagine modifying the procedure presented above so that at each point $x_c \in \mathcal{C}$, the associated jump correction J_α is computed at the beginning of each time step and immediately stored with the affected point $A^-(x_c)$. If this is possible, then the jump corrections will be automatically included when the regular five-point finite difference stencil is applied. However, the map $A^- : \mathcal{C} \to \mathcal{A}^-$ is not injective, so that each point in $\mathcal{A}^-$ may receive corrections from multiple control points.

Since the jump corrections are all n-th order polynomial extrapolations of the same function, we expect that they differ only by an $\mathcal{O}\big(h^{n+1}\big)$ error term. Thus we will preserve the formal accuracy of the finite difference scheme if we assign to each affected point the average value of all the corrections it receives. In theory, we could attempt to choose a value which minimizes truncation error, or maximizes a local stability criterion. However, for all of the problems in this thesis a simple average is sufficient. This method of extending a function by computing and storing immersed interface jump corrections is simple and convenient, and we will later use it to correct third order advection stencils and develop high-order time integration for problems on moving domains.

2.4 Geometry Processing for IIM

One immediate obstacle to the implementation of a multidimensional immersed interface method is the representation of complex geometry. In this thesis, all objects are represented by a signed-distance function (SDF), which returns the minimum distance between a point and the boundary of the object. The sign of the function is positive outside of the object and negative inside, allowing us to quickly determine whether a given point lies inside or outside the object. In general there is no obstacle to using a more general level set representation, but the SDF does reduce the need for some normalization operations.

The use of an implicit representation does not limit the applicability of the immersed interface methods. In [23], Upretti provides methods for efficiently generating implicit, algebraic representations of rational curves and surfaces, as well determining algebraically the parameter value of any query point on these surfaces. This opens the IIM to efficient geometry processing of a very wide class of shapes, including those typically produced by CAD packages. Although we do not use these tools here, this section will explore algorithms for identifying control points and calculating integrals which rely on an implicit representation of the immersed geometry.

2.4.1 Efficiently Identifying Control Points

In the immersed interface method, geometry is represented entirely by the intersections between a boundary curve and the lines of a Cartesian grid, as well as normal vector to the boundary at these intersections. Below we present a local algorithm which determines these intersections with $\mathcal{O}(h^4)$ accuracy and normal vectors with $\mathcal{O}(h^3)$ accuracy, using only signed distance function values available at the grid points.

We begin our search for intersections by evaluating the signed distance function ϕ at each point $\mathbf{x}_{i,j} \in \mathcal{G}$. We then enter a loop over all points in the domain, checking for points $\mathbf{x}_{i,j}$ that satisfy $0 \leq \phi(\mathbf{x}_{i,j}) \leq h$. If this is true, we loop over the neighbors of $\mathbf{x}_{i,j}$, denoted by $\mathcal{N}(\mathbf{x}_{i,j})$, in search of a point $\mathbf{x}_{k,l}$ satisfying $\phi(\mathbf{x}_{k,l}) < 0$. Every such point corresponds to an intersection; by continuity of ϕ, there is a point $\mathbf{x}_c$ on the grid-line connecting $\mathbf{x}_{i,j}$ and $\mathbf{x}_{k,l}$ for which $\phi(\mathbf{x}_c) = 0$. This point is a member of $\mathcal{C}$ satisfying $A^+(\mathbf{x}_c) = \mathbf{x}_{i,j}$ and $A^-(\mathbf{x}_c) = \mathbf{x}_{k,l}$.

Having determined the existence of an intersection, we must locate it precisely. To do so, define a local coordinate vector $\hat{\xi}_c = \mathbf{x}_{i,j} - \mathbf{x}_{k,l}$, which lies along the grid-line in question. The one-dimensional function $\phi_\xi(z) = \phi(\mathbf{x}_{i,j} + z\hat{\xi}_c)$ will have a zero at the intersection point, allowing us to locate $\mathbf{x}_c$ by determining the roots of ϕ_ξ. To avoid any additional evaluations of ϕ, we will instead find the roots of a polynomial which interpolates ϕ_ξ. We begin by collecting the values $\phi_m = \phi(\mathbf{x}_{i,j} + m\hat{\xi}_c)$ for $-2 \leq m \leq 1$, as shown in Figure 2-6. As a first approximation, we find the root of a linear interpolating polynomial passing through $(-h, \psi_{-1})$ and $(0, \psi_0)$, giving $z_0 = -\phi_0 h/(\phi_0 - \phi_{-1})$. This approach is only second order accurate, but serves as a useful starting point for a higher order method.

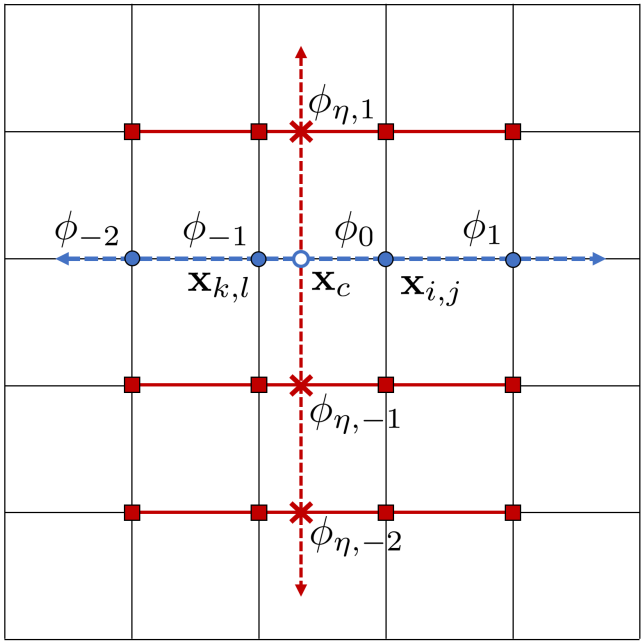


Figure 2-6: Notation for the local intersection algorithm.

To locate $\mathbf{x}_c$ with fourth order accuracy we construct a cubic polynomial $p_3(z)$ which interpolates ϕ_ξ at the four points (mh, ϕ_m), $-2 \leq m \leq 1$. One root (z_1) can be found using Newton's method with z_0 as an initial guess, and it is likely to lie in $[-h, 0]$. If it does not, we extract a quadratic

factor from p_3 using this root:

$$\begin{aligned} p_3(z) &= a_3 z^3 + a_2 z^2 + a_1 z + a_0 \quad \text{and} \quad p_3(r) = 0 \quad \text{implies that} \\ p(x) &= (z-r)(b_2 z^2 + b_1 z + b_0), \quad \text{with} \\ &\begin{cases} b_2 &= a_3, \\ b_1 &= a_2 + r b_2, \\ b_0 &= a_1 + r b_1. \end{cases} \end{aligned}$$

The remaining two roots of p_3 can then be found with the quadratic formula,

$$z_2,\ z_3 = \frac{-b_1 \pm \sqrt{b_1^2 - 4 b_0 b_2}}{2 b_0}.$$

One of these roots will lie in $[-h, 0]$; call it $z_{\mathcal{I}}$. We now record the following information:

$$\begin{aligned} \psi &= -z_{\mathcal{I}}/h + \mathcal{O}\big(h^3\big), & \mathbf{x}_c &= \mathbf{x}_{i,j} - h\psi\hat{\xi}_c + \mathcal{O}\big(h^4\big), \\ \frac{\partial \phi}{\partial \xi}(\mathbf{x}_c) &= p'(z_{\mathcal{I}}) + \mathcal{O}\big(h^3\big), & \frac{\partial^2 \phi}{\partial \xi^2}(\mathbf{x}_c) &= p''(z_{\mathcal{I}}) + \mathcal{O}\big(h^2\big). \end{aligned}$$

Our only remaining task is to compute the normal vector $\hat{\mathbf{n}} = \nabla\phi(\mathbf{x}_c)$. To do so, we must take a derivative of ϕ along the η direction.Since the sign of η is not yet known, we select an arbitrary unit vector $\hat{\eta}$ satisfying $\hat{\eta}\cdot\hat{\xi} = 0$. As shown in Figure 2-6, there are no points $\mathbf{x} \in \mathcal{G}$ with $\mathbf{x} = \mathbf{x}_c + z\hat{\eta}$, so we cannot directly apply a finite difference stencil. We must first interpolate the values $\phi_{\eta,k} = \phi(\mathbf{x}_c + k\hat{\eta})$ for $-2 \le k \le 1$, using a fourth-order interpolation stencil tailored to the intersection distance ψ. With these values available, we calculate

$$\begin{aligned} \frac{\partial \phi}{\partial \eta}(\mathbf{x}_c) &= \frac{1}{6h}(2\phi_{\eta,1} + 3\phi_{\eta,0} - 6\phi_{\eta,-1} + \phi_{\eta,-2}) + \mathcal{O}\big(h^3\big), \quad \text{and} \\ \frac{\partial^2 \phi}{\partial \eta^2}(\mathbf{x}_c) &= \frac{1}{h^2}(\phi_{\eta,1} - 2\phi_{\eta,0} + \phi_{\eta,-1}) + \mathcal{O}\big(h^2\big). \end{aligned}$$

The normal vector is now known to third order, since,

$$\hat{\mathbf{n}} = \left(\frac{\partial \phi}{\partial \xi}\hat{\boldsymbol{\xi}} + \frac{\partial \phi}{\partial \eta}\hat{\boldsymbol{\eta}}\right),$$

and the sign of the η coordinate can be reversed if it was initially selected incorrectly. We also have a second order estimate of the boundary curvature κ, since

$$\kappa = \nabla^2\phi = \frac{\partial^2 \phi}{\partial \xi^2} + \frac{\partial^2 \phi}{\partial \eta^2}$$

for a signed distance function ϕ. Although we have no direct use for kappa at the moment, the quantity dimensionless quantity κh will be useful for problems involving moving domains. This completes our characterization of the boundary at $\mathbf{x}_c$, and returns us to our original loop over the neighbors of $\mathbf{x}_{i,j}$. Taken all together, this procedure allows for the location of all grid line intersections relevant to the immerse interface method, in a way that requires only a single global evaluation of the signed distance function ϕ.

2.4.2 Integration over an Irregular Boundary

In incompressible flow problems, the total force and moment acting on an immersed solid body can be determined by integrating the local traction vector of the body's surface. With an immersed interface method, this surface may not coincide with the regular grid, and special approximations are necessary to produce an integral accurate to better than first order. This problem is considered

by Gillis' doctoral thesis [10], which builds on the work of Smereka [21]. Although we don't provide a complete treatment here, an abridged version is useful to motivate the treatment of integrals over irregular domains given in the next section.

Let S be the boundary of a two-dimensional region defined by the signed distance function ϕ. Given the set of intersection $\mathcal{C}$ between S and a regular Cartesian grid $\mathbf{x}_{i,j} \in \mathcal{G}$, the objective is to find a set of lengths $\ell(\mathbf{x}_c)$ so that

$$\oint_S f \, \mathrm{d}s = \sum_{\mathbf{x}_c \in \mathcal{C}} \ell(\mathbf{x}_c) f(\mathbf{x}_c) + \mathcal{O}(h^n),$$

where n is the order of the method. Smereka solves a very similar problem associated with level set methods, where the support of a smooth function f is the entire Cartesian grid, and we wish to make the approximation

$$\oint_S f \, \mathrm{d}s = \int_{\mathbb{R}^2} \delta(\phi(\mathbf{x})) f(\mathbf{x}) \, \mathrm{d}A = \sum_{\mathbf{x}_{i,j} \in \mathcal{A}} \delta(\mathbf{x}_{i,j}) f(\mathbf{x}_{i,j}) + \mathcal{O}(h^n).$$

Here $\delta(\mathbf{x})$ is the Dirac delta function, and $\delta_{i,j}$ is its discrete counterpart. Smereka provides second- and fourth-order methods for computing the weights $\delta_{i,j}$ using the values of a level set $\phi_{i,j}$. Much like a multidimensional immersed interface operation, the second-order method uses weights that contain additive contributions from each control point adjacent to a given affected point. Consequently, we can construct the set of weights $\delta(\mathbf{x}_{i,j})$ by considering every $\mathbf{x}_c \in \mathcal{C}$, each of which corresponds to an additive contribution of δ_c^+ to the point $A^+(\mathbf{x}_c)$ and δ_c^- to the point $A^-(\mathbf{x}_c)$. Gillis' insight was to pull these contributions back to the control points, so that $\ell(\mathbf{x}_c) = \delta_c^+ + \delta_c^-$ and

$$\oint_S f \, \mathrm{d}s = \sum_{\mathbf{x}_c \in \mathcal{C}} \left[\delta_c^+ + \delta_c^-\right] f(\mathbf{x}_c) + \mathcal{O}(h^n). \tag{2.22}$$

This is shown experimentally to be second order accurate. The computation of δ^+ and δ^- is nontrivial, and we leave the details to [10].

2.4.3 Integration over an Irregular Domain

To build on the work described in the previous section, we describe here a method for evaluating integrals over an irregular region $\Omega \in \mathbb{R}^2$ that is immersed in a Cartesian grid with uniform grid spacing h. Let $f(x)$ be a scalar function on Ω, and $f_{i,j}$ be its value at the grid point $\mathbf{x}_{i,j}$. Naively, one can integrate f approximately using

$$\int_\Omega f(x) \, \mathrm{d}x \approx h^2 \sum_{\mathbf{x}_{i,j} \in \Omega} f(\mathbf{x}_{i,j}).$$

This approach is only first order accurate. To see this, note that an n-dimensional domain has an $(n-1)$-dimensional boundary, meaning that the set of affected points $\mathcal{A}$ contains $\mathcal{O}\big(h^{-(n-1)}\big)$ points. The error committed at of each of these points point is an arbitrary fraction of the local volume element, which has magnitude $\mathcal{O}(h^n)$; summing up $\mathcal{O}\big(h^{-(n-1)}\big)$ of these errors leads to an overall $\mathcal{O}(h)$ error in the integral.

Just as an approximation of the delta function led to second-order IIM surface integrals, an approximation of the Heaviside function allows for second-order IIM domain integrals. Consider a function $f : \mathbb{R}^2 \to \mathbb{R}$, and a domain Ω defined implicitly by the signed distance function ϕ, so that $x \in \Omega$ iff $\phi(x) \geq 0$. We can write

$$\int_\Omega f(x) \, \mathrm{d}x = \int_{\mathbb{R}^2} f(x) H(\phi(x)) \, \mathrm{d}x \,, \tag{2.23}$$

where $H(x)$ is the one-dimensional Heaviside function

$$H(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0. \end{cases}$$

To approximate (2.23), we immerse the domain Ω in a regular Cartesian grid $\mathcal{G}$ with grid spacing h, and seek a discrete Heaviside function $\tilde{H}_{i,j}$ constructed to satisfy

$$\int_{\mathbb{R}^2} f(x) H(\phi(x)) \, \mathrm{d}x = h^2 \sum_{\mathbf{x}_{i,j} \in \mathcal{G}} f(\mathbf{x}_{i,j}) \tilde{H}_{i,j} + \mathcal{O}\big(h^2\big).$$

One such discrete Heaviside function is derived by Tower in [22]. To calculate the weights $\tilde{H}_{i,j}$ used in Tower's Heaviside function, we consider the unit ramp function

$$I(x) = \int_{-\infty}^{x} H(u) \, \mathrm{d}u = \begin{cases} 0, & x < 0 \\ x, & 0 \leq x. \end{cases}$$

Applying the ramp function to $\phi(x)$ and differentiating, we find that

$$\nabla I(\phi) = H(\phi) \nabla \phi, \quad \text{so that} \quad \nabla I(\phi) \cdot \nabla \phi = H(\phi).$$

To discretize this relationship, define the discrete gradient operator ∇_h to be the gradient operator discretized using a centered finite difference stencils, so that

$$\nabla_h f_{i,j} = \frac{1}{2h}(f_{i+1,j} - f_{i-1,j}) \hat{\mathbf{e}}_x + \frac{1}{2h}(f_{i,j+1} - f_{i,j-1}) \hat{\mathbf{e}}_y.$$

Tower's second-order Heaviside function is given by

$$\tilde{H}_{i,j} = \nabla_h I(\phi_{i,j}) \cdot \nabla_h \phi_{i,j}.$$

Let $\mathbf{x}_{i,j}$ be a point with a set of neighbors $\mathcal{N}(\mathbf{x}_{i,j})$ that is entirely inside or entirely outside of Ω. Then then to second order the evaluation of $\tilde{H}_{i,j}$ can be simplified to

$$\tilde{H}_{i,j} = \begin{cases} 0, & \mathcal{N}(\mathbf{x}_{i,j}) \subset \Omega \\ 1, & \mathcal{N}(\mathbf{x}_{i,j}) \subset \mathbb{R}^2 \setminus \Omega. \end{cases}$$

Thus to second order, $\tilde{H} = H(\phi)$ for all points $\mathbf{x}_{i,j} \notin \mathcal{A}$. It's convenient to define the function $\delta H_{i,j} = \tilde{H}_{i,j} - H(\phi_{i,j})$, which is nonzero only on $\mathcal{A}$; then,

$$\int_{\Omega} f(x) \, \mathrm{d}x = h^2 \sum_{\mathbf{x}_{i,j} \in \mathcal{G}} f_{i,j} H(\phi_{i,j}) + h^2 \sum_{\mathbf{x}_{i,j} \in \mathcal{A}} f_{i,j} \delta H_{i,j} + \mathcal{O}\big(h^2\big) \tag{2.24}$$

The first sum is the naive, first order discretization of the integral, while the second sum is restricted to the boundary and acts as a higher-order correction. The evaluation of δH can be cast neatly into our local immersed interface language: we see that the inner product $\nabla_h I(\phi) \cdot \nabla_h \phi$ breaks into an additive contribution from each spatial direction, and that every contribution to δH is tied to a point $\mathbf{x}_c \in \mathcal{C}$.

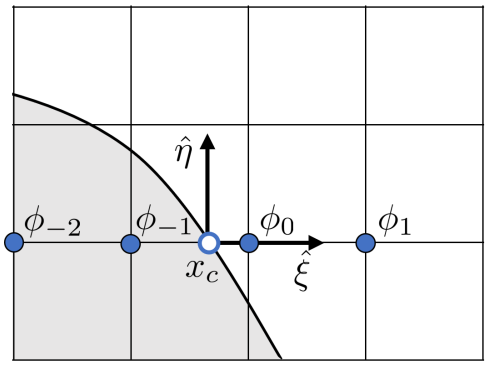


Figure 2-7: Nomenclature for constructing $\tilde{H}_{i,j}$.

Figure 2-7 illustrates the notation needed to describe this calculation. Consider a control point $\mathbf{x}_c \in \mathcal{C}$ and the associated points $x_{-1} = A^-(\mathbf{x}_c)$ and $x_0 = A^+(\mathbf{x}_c)$ which lie adjacent to $\mathbf{x}_c$ on the local ξ axis. Conveniently the inner product $\nabla_h I(\phi) \cdot \nabla_h \phi$ is invariant under the exchange $x \leftrightarrow y$ and the transformations $x \to -x$ or $y \to -y$, so that we can confidently neglect changes in the sign or orientation of the local coordinate system. The contribution from $\mathbf{x}_c$ to $\delta H(x_{-1})$ is

$$\delta H_c^- = \left(\frac{I(\phi_0) - I(\phi_{-2})}{2h}\right)\left(\frac{\phi_0 - \phi_{-2}}{2h}\right) = \frac{\phi_0(\phi_0 - \phi_{-2})}{4h^2}.$$

To calculate the contribution to $\delta H(x_0)$, we use the approximation $H(x_0) = 1 \approx |\nabla_h \phi|^2$ and the identity $I(x) - x = I(-x)$ to write

$$\begin{aligned}\tilde{H} - H &= \nabla_h I(\phi) \cdot \nabla_h \phi - \nabla_h \phi \cdot \nabla_h \phi \\ &= \nabla_h [I(\phi) - \phi] \cdot \nabla_h \phi \\ &= \nabla_h I(-\phi) \cdot \nabla_h \phi.\end{aligned}$$

From the last identity, we see that

$$\delta H_c^+ = \left(\frac{I(-\phi_1) - I(-\phi_{-1})}{2h}\right)\left(\frac{\phi_1 - \phi_{-1}}{2h}\right) = \frac{\phi_{-1}(\phi_1 - \phi_{-1})}{4h^2}.$$

This leaves only one missing piece in (2.24). While $f(x_0)$ is well-defined, the value $f(x_{-1})$ is outside of the domain, so we cannot truly form the sum $\sum_{\mathbf{x}_{i,j} \in \mathcal{A}} f_{i,j} \delta H_{i,j}$. Instead we replace $f(x_{-1})$ with a jump correction, and write

$$\int_\Omega f(x)\,\mathrm{d}x = h^2 \sum_{\mathbf{x}_{i,j} \in \Omega} f_{i,j} + h^2 \sum_{\mathbf{x}_c \in \mathcal{C}} \left[\delta H_c^+ f(A^+(\mathbf{x}_c)) + \delta H_c^- J_\alpha(\mathbf{x}_c)\right] + \mathcal{O}\big(h^2\big). \tag{2.25}$$

A quick analysis shows that we need only first order accuracy in evaluating $f(A^+(\mathbf{x}_c))$ and J_α to maintain global second order accuracy. We are then free to pull δH back to the control points, and write

$$\int_\Omega f(x)\,\mathrm{d}x = h^2 \sum_{\mathbf{x}_{i,j} \in \Omega} f_{i,j} + h^2 \sum_{\mathbf{x}_c \in \mathcal{C}} \big(\delta H_c^+ + \delta H_c^-\big) f(\mathbf{x}_c) + \mathcal{O}\big(h^2\big). \tag{2.26}$$

The discrete integral (2.25) can be used without boundary data, or when only a Neumann boundary condition is available. The form given (2.26) is restricted to problems with Dirichlet conditions, but is more conveniently implemented. In this thesis we choose to evaluate all of our domain integrals with (2.25), since we generally have an accurate jump correction left over from other immersed

interface operations. Although this method has been developed in two dimensions, its extension to N dimensions is clear, and the signed distance function ϕ can easily be replaced with an arbitrary level set by inserting the normalizing factors of $|\nabla\phi|$ which have been neglected here.

Chapter 3

IIM for Transport Problems

3.1 The Advection Diffusion Equation

Consider an incompressible fluid confined to a domain $\Omega \in \mathbb{R}^2$, moving with velocity $\mathbf{u}(\mathbf{x}, t)$. Let $f(\mathbf{x}, t)$ be the concentration of a scalar quantity that is advected by the flow, and also diffuses through the fluid with a constant diffusivity ν. These two processes give rise to a flux of the scalar quantity that can be described by the vector

$$\mathbf{q} = \mathbf{u}f + \nu\nabla f,$$

so that a conservation law for f of the form $\partial_t f + \nabla \cdot \mathbf{q} = 0$ yields the advection diffusion equation

$$\frac{\partial f}{\partial t} + (\mathbf{u} \cdot \nabla)f = \nu\nabla^2 f. \tag{3.1}$$

To non-dimensionalize this relation, we introduce a reference length L, velocity U, and scalar concentration F. Define the re-scaled variables

$$f = f^* F, \qquad \mathbf{u} = \mathbf{u}^* U, \qquad \mathbf{x} = \mathbf{x}^* L, \qquad t = t^* \frac{L}{U};$$

after making substitutions and dropping the stars, (3.1) becomes

$$\frac{\partial f}{\partial t} + (\mathbf{u} \cdot \nabla)f = \frac{1}{\mathrm{Pe}}\nabla^2 f, \tag{3.2}$$

where Pe is the Péclet number,

$$\mathrm{Pe} = \frac{LU}{\nu}. \tag{3.3}$$

This quantity characterizes the relative magnitude of the convective and diffusive fluxes, and is analogous to the Reynolds number in fluid flow. The advection equation and diffusion equation can be used to simplify processes with extreme Péclet numbers:

$$\frac{\partial f}{\partial t} + (\mathbf{u} \cdot \nabla)f = \nu\nabla^2 f \rightarrow \begin{cases} \dfrac{\partial f}{\partial t} + (\mathbf{u} \cdot \nabla)f = 0, & \mathrm{Pe} \gg 1; \\ \dfrac{\partial f}{\partial t} = \nu\nabla^2 f, & \mathrm{Pe} \ll 1. \end{cases}$$

The diffusive limit is relatively well behaved. However, the advective limit changes the order of the PDE, leading to a singular perturbation. Physically, this indicates that solutions to (3.1) with $\mathrm{Pe} \gg 1$ form thin boundary layers, which can trigger instabilities in some numerical methods. This chapter first examines the stability of free-space advection diffusion discretizations, and then develops a novel immersed interface method for imposing boundary conditions in a stable and accurate way.

3.2 Free-Space Discretization

3.2.1 Stability of Finite Difference Advection Schemes

Any immersed interface discretization must begin with a stable finite difference scheme. This section reviews results on the stability of explicit Runge-Kutta schemes and Von-Neumann stability analysis for one-dimensional finite difference schemes on infinite domains. These tools are then applied to a method-of-lines discretizations of the advection equation, using common first-, second-, and third-order finite differences.

Consider a simple initial-value problem for an unknown function $u(t) : \mathbb{R} \to \mathbb{R}^n$,

$$\frac{\mathrm{d}u}{\mathrm{d}t} = Au \quad \text{with} \quad u(0) = u_0, \tag{3.4}$$

where $A : \mathbb{R}^n \to \mathbb{R}^n$ is a linear map. The solution of this equation is $u(t) = \exp(tA)u_0$, where $\exp(M)$ is defined by the power series

$$\exp(At) = I + M + \frac{1}{2}M^2 + ... = \sum_{n=0}^{\infty} \frac{1}{n!}M^n. \tag{3.5}$$

A typical Runge-Kutta integration scheme approximates the evolution of u by truncating the above power series. For an N-th order Runge-Kutta scheme,

$$u(t+\tau) = \exp(\tau A)u(t) = \left(\sum_{n=0}^{N} \frac{1}{n!}\tau^n A^n\right)u(t) + \mathcal{O}\big(\tau^{N+1}\big). \tag{3.6}$$

In general, we would like to know that any error introduced during numerical integration does not grow as time progresses. If A admits a full basis of eigenvectors, then any small perturbation can be broken into components that are also eigenvectors of A, and it is sufficient to study the behavior of these components. Let u be an eigenvector of A with eigenvalue $\lambda \in \mathbb{C}$, so that

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \lambda u \quad \text{with} \quad u(0) = u_0. \tag{3.7}$$

If λ has negative real part, then the exact solution will decay with time. This is not necessarily true for the approximate solution, which satisfies

$$u(t+k\tau) = \left(1 + \tau\lambda + ... + \frac{\tau^N\lambda^N}{N!}\right)^k u_0. \tag{3.8}$$

If the approximate solution is to decay in time, then the stability polynomial

$$P_N(\tau\lambda) = \big(1 + \tau\lambda + ... + \tau^N\lambda^N/N!\big)$$

must have a magnitude less than unity. This defines the stability region for an N-th order Runge-Kutta scheme,

$$\left|\sum_{n=0}^{N} \frac{(\tau\lambda)^n}{n!}\right| \leq 1. \tag{3.9}$$

To apply this analysis to a finite difference scheme, we must be able to find eigenvalues and eigenvectors of finite difference operators. Consider a infinite one-dimensional domain $\Omega = \mathbb{R}$, and a uniformly separated set of points $x_n = hn$ for $n \in \mathbb{Z}$. Any real-valued function $F : \Omega \to \mathbb{R}$ can be approximated on this grid the values $f_n = F(x_n)$. A finite difference scheme uses a linear combination of values f_i to approximate derivatives of the original function F. The finite differences

considered in this thesis are

$$\begin{aligned}
\left.\frac{\mathrm{d}F}{\mathrm{d}x}\right|_{x_0} &= \frac{1}{h}(f_1 - f_0) + \mathcal{O}(h) && \text{[1D]: First-order downwind} \\
&= \frac{1}{h}(f_0 - f_{-1}) + \mathcal{O}(h) && \text{[1U]: First-order upwind} \\
&= \frac{1}{2h}(f_1 - f_{-1}) + \mathcal{O}(h) && \text{[2C]: Second-order centered} \\
&= \frac{1}{2h}(-3f_0 + 4f_1 - f_2) + \mathcal{O}\big(h^2\big) && \text{[2D]: Second-order downwind} \\
&= \frac{1}{2h}(f_{-2} - 4f_{-1} + 3f_0) + \mathcal{O}\big(h^2\big) && \text{[2U]: Second-order upwind} \\
&= \frac{1}{6h}(-2f_{-1} - 3f_0 + 6f_1 - f_2) + \mathcal{O}\big(h^3\big) && \text{[3D]: Third-order downwind} \\
&= \frac{1}{6h}(f_{-2} - 6f_{-1} + 3f_0 + 2f_1) + \mathcal{O}\big(h^3\big) && \text{[3U]: Third-order upwind}
\end{aligned}$$

Although notated for x_0, these approximations are valid if the indices are shifted by an arbitrary integer. Each finite difference scheme can be seen as a linear operator acting on the vector of values f_i, which returns another vector f'_i that approximates the exact derivative $F'(x)$. In this framework, it is convenient to decompose each finite difference into a linear combination of translation operators T_m, which satisfy

$$(T_m f)_n = f_{m+n}. \tag{3.10}$$

The eigenfunctions of each translation operator are the complex exponential functions $f(n) = e^{inkh}$, which satisfy

$$T_m e^{inkh} = e^{imkh} e^{inkh}. \tag{3.11}$$

The exponential $f(n) = e^{inkh}$ is traditional for a Von-Neumann stability analysis, and represents the oscillatory function $F(x) = e^{ikx}$ restricted to the grid. Because of aliasing effects, only functions for which $kh \in [0, \pi)$ are linearly independent. The eigenfunctions of finite difference approximations are also exponential functions, and can be extracted from the discretization in a straightforward way. As an example, consider the second-order upwind operator $[2U] = (T_{-2} - 4T_{-1} + 3T_0)/2h$, which has eigenvalues of the form

$$[2\mathrm{U}]e^{inkh} = \frac{1}{2h}(e^{-2ikh} - 4e^{-ikh} + 3)e^{inkh}.$$

For convenience, let $z = e^{ikh}$, so that z ranges over the uppers half of the unit circle. Then the spectra of the finite difference operators defined above are

$$\begin{aligned}
[1\mathrm{D}] &: \lambda(z) = \frac{1}{h}(z - 1) \\
[1\mathrm{U}] &: \lambda(z) = \frac{1}{h}(1 - z^{-1}) \\
[2\mathrm{C}] &: \lambda(z) = \frac{1}{2h}(z - z^{-1}) \\
[2\mathrm{D}] &: \lambda(z) = \frac{1}{2h}(-3 + z - z^2) \\
[2\mathrm{U}] &: \lambda(z) = \frac{1}{2h}(z^{-2} - 4z^{-1} + 3) \\
[3\mathrm{D}] &: \lambda(z) = \frac{1}{6h}(-2z^{-1} - 3 + 6z - z^2) \\
[3\mathrm{U}] &: \lambda(z) = \frac{1}{6h}(z^{-2} - 6z^{-1} + 3 + 2z).
\end{aligned}$$

To evaluate the stability of a particular discretization of the advection problem, we discretize the

operator $-u\frac{\mathrm{d}}{\mathrm{d}x}$ using a finite difference scheme, and check that for all $z \in [0, \pi)$, the stability polynomial $P_N(\tau\lambda(z))$ has magnitude less than one. For advection discretizations using any of the finite differences listed above, $\tau\lambda(z)$ is proportional to the Courant number

$$C = \frac{u\tau}{h}, \tag{3.12}$$

which is the only non-dimensional combination of numerical parameters affecting the stability of the advection schemes discussed here. In general, any advection scheme that uses explicit time integration has a critical Courant number above which it is unstable. We can determine this critical Courant number numerically, by performing a bisection search to find the lowest value positive of C for which

$$\max_z |P_N(-hC\lambda(z))| > 1.$$

The results of this analysis for all of the finite difference operators considered so far are given in the table below; an "X" indicates unconditional instability.

	RK1	RK2	RK3
1U	1	1	1.25
2C	X	X	1.75
2U	X	0.50	0.62
3U	X	0.88	1.62

This table does not include any of the downwind differences; these are all unconditionally unstable for any Runge-Kutta scheme. The higher order upwind differences require at least RK2 to maintain stability, while the second-order centered scheme performs well only for RK3.

In this thesis, we focus on the [3U] scheme. First order advection schemes provide insufficient accuracy for fluid simulations, and second order schemes have a leading truncation error that produces dispersive phenomena. The third order upwind scheme, however, has a leading truncation error proportional to $f^{(4)}(x)$, which introduces a hyper-diffusive effect that tends to smooth out high wavenumber error modes. This scheme also enjoys the least restrictive CFL constraint for RK2 integration, and is competitive with [2C] when integrated with RK3.

3.2.2 Stability of Diffusion Schemes

For diffusion processes, we use the standard centered stencil

$$\left.\frac{\mathrm{d}^2 F}{\mathrm{d}x^2}\right|_{x_0} = \frac{1}{h^2}(f_{-1} - 2f_0 + f_1) + \mathcal{O}\big(h^2\big)$$

which we will call [DF]. This finite difference operator has eigenvalues

$$\lambda(z) = \frac{1}{h^2}(z^{-1} - 2 + z).$$

Since z lies on the unit circle, z and z^{-1} are conjugate, so that $\lambda(z)$ is real and satisfies $\lambda(z) \in [-4, 0]$. Applying this to the full diffusion operator $\nu\frac{\mathrm{d}^2}{\mathrm{d}x^2}$, we find that $\tau\lambda(z)$ is proportional to the Fourier number

$$r = \frac{\nu\tau}{h^2}.$$

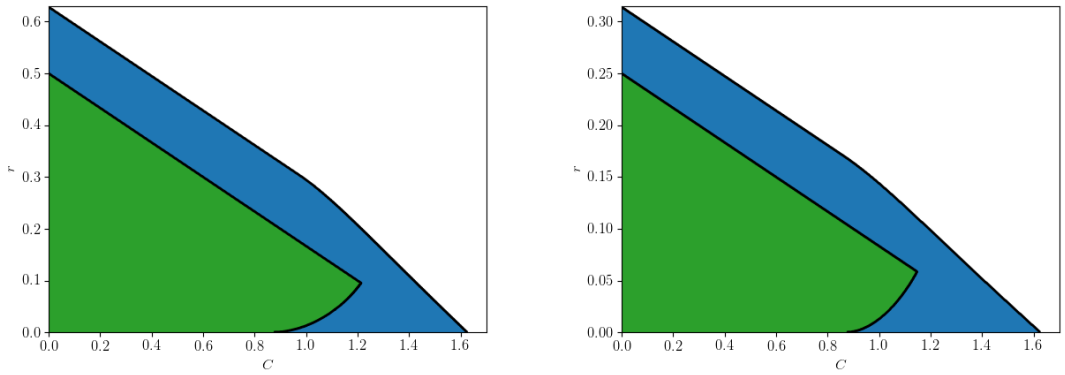
In one dimension the maximum allowable Fourier number is 0.5 for RK1 or RK2 time stepping, and approximately 0.62 for RK3. This is a significantly stricter stability requirement than the CFL condition, since τ must scale with h^2 as the spatial discretization is refined.

3.2.3 Stability of Advection Diffusion Schemes

Stability behavior becomes more complex for the full advection-diffusion equation. We limit our discussion to the [3U] advection scheme in combination with the standard diffusion stencil discussed above. Because both the advection and diffusion terms are linear, the eigenvalues of our discretization are simply the sum of the eigenvalues for each individual operator:

$$\lambda(z) = -u\lambda_{[3U]}(z) + \nu\lambda_{[DF]}(z). \tag{3.13}$$

These spectra now depend on both the Courant number C and Fourier number r, which can be varied independently by changing the parameters ν. Using the same brute force method as in section 3.2.1, we can determine a region in the (C, r) for which our discretization is stable; this region is shown in Figure 3-1a.



(a) One-dimensional stability Region.

(b) Two-dimensional Stability Region

Figure 3-1: Stability region in the (C, r) plane for the one and two-dimensional advection-diffusion discretization considered here.

To analyze this finite difference discretization in two dimensions, we must consider a grid indexed by two integers n and m, and consider complex exponentials of the form $e^{ih(nk_x+mk_y)}$. If our velocity field $\mathbf{u} = [u_x, u_y]$ does not vary in space, then we still perform a Von-Neumann stability analysis. However, we must now consider two wave numbers k_x and k_y, as well as two Courant numbers $C_x = u_x\tau/h$ and $C_y = u_y\tau/h$. Let $z_x = e^{ik_xh}$ and $z_y = e^{ik_yh}$, so that the eigenvalues of of our two-dimensional advection diffusion discretization take the form

$$\lambda(z_x, z_y) = -u_x\lambda_{[3U]}(z_x) - u_y\lambda_{[3U]}(z_y) + \nu(\lambda_{[DF]}(z_x) + \lambda_{[DF]}(z_y)) \tag{3.14}$$

Given the values for C_x, C_y, and r, our finite difference scheme will be stable with an N-th order Runge Kutta scheme if

$$\max_{z_x,z_y} \left|P_N(-hC_x\lambda_{[3U]}(z_x) - hC_y\lambda_{[3U]}(z_y) + h^2r(\lambda_{[DF]}(z_x) + \lambda_{[DF]}(z_y)))\right| \leq 1.$$

Some numerical experimentation indicates that this stability criteria depends on C_x and C_y through their sum only, motivating the definition of a two-dimensional Courant number $C_{2D} = C_x + C_y$. Figure 3-1b shows the stability region of our two-dimensional advection diffusion equation in the (C_{2D}, r) plane, for $u_x, u_y > 0$. The behavior is very similar (though not identical) to the one-dimensional case with a re-scaled Fourier number.

If the velocity field $\mathbf{u}$ is not constant, then the Von-Neumann stability analysis performed above is no longer strictly applicable. However, it is well established that we can still achieve a stable discretization if at every point the local Courant number $C_{2D}(\mathbf{x})$ lies in the stability region associated with the constant coefficient problem. We must also ensure that our upwind stencil remains locally

upwind at each point in the flow. To do this, we calculate $\partial_x f$ using [3U] when the local velocity component u_x is positive, and using [3D] when $u_x \leq 0$. The procedure for $\partial_y f$ is identical.

3.3 Boundary Conditions

3.3.1 Diffusion Problems

A successful immersed interface discretization of the diffusion equation with Dirichlet or Neumann boundary conditions was developed by Marichal in [15]. The Dirichlet case was already described in Chapter 2, and a stability analysis performed by Marichal indicates that this scheme has a critical Fourier number that is only slightly below the free-space scheme. No subtlety is needed to enforce a Dirichlet boundary conditions for this problem; a BC is required on every surface for both the continuous problem and the numerical problem, and all jump corrections can safely be calculated from the given BC. Neumann problems are also treated in [15], but they won't be discussed here.

To confirm the convergence behavior reported by Marichal, we set up a model IBVP for the advection diffusion equation with known analytical solution. For simplicity, the velocity field $\mathbf{u}$ and diffusivity ν are both assumed to be constant in space. Following Marichal [15], we consider a Gaussian distribution with height g_0, standard deviation σ, and moving center $\mathbf{x}_g(t) = \mathbf{x}_g(0) + \mathbf{u}t$, which has an analytical expression

$$g(\mathbf{x}, t) = g_0 \frac{\sigma^2}{(\sigma^2 + 4\nu t)} \exp\left[\frac{-|\mathbf{x} - \mathbf{x}_g(t)|^2}{(\sigma^2 + 4\nu t)}\right] \tag{3.15}$$

and satisfies the advection-diffusion equation on an unbounded domain. For our numerical problem, let the computational domain Ω_c be the unit square $[0, 1] \times [0, 1]$, and let the problem domain Ω be equal to Ω_c a minus circular region with center $\mathbf{x}_c$ and radius R. We can preserve the above solution by constructing an IBVP with (3.15) as an initial condition and Dirichlet boundary condition,

$$\begin{aligned} \frac{\partial f}{\partial t} &= -(\mathbf{u} \cdot \nabla) f + \nu \nabla^2 f, \\ f(\mathbf{x}, t_0) &= g(\mathbf{x}, t_0) \quad \text{for} \quad \mathbf{x} \in \Omega, \\ f(\mathbf{s}, t) &= g(\mathbf{s}, t) \quad \text{for} \quad \mathbf{s} \in \partial\Omega. \end{aligned} \tag{3.16}$$

By construction, this IBVP is well-behaved at large Péclet numbers, since the analytical solution (3.15) contains no boundary layers. The continuous system (3.16) is discretized using the immersed interface method described above, and the resulting system of ODEs is integrated over the interval $t \in [0, T]$ using an explicit Runge-Kutta method. To measure the spatial convergence of the solver, an extremely small time-step is chosen, so that the dominant error in the numerical solution comes from the spatial discretization. We approximate the continuous L_2 and L_∞ error norms

$$\epsilon_2(T) = \frac{1}{g_0} \sqrt{\frac{1}{A} \int_\Omega [f(\mathbf{x}, T) - g(\mathbf{x}, t)]^2 \, \mathrm{d}A}$$

$$\epsilon_\infty(T) = \frac{1}{g_0} \max_{\mathbf{x} \in \Omega} |f(\mathbf{x}, T) - g(\mathbf{x}, T)|$$

by their discrete analogs

$$\epsilon_2(T) = \frac{1}{g_0} \sqrt{\frac{1}{A} \sum_{i,j} h^2 [f_{i,j}(T) - g(\mathbf{x}_{i,j}, T)]^2}, \tag{3.17}$$

$$\epsilon_\infty(T) = \frac{1}{g_0} \max_{i,j} |f_{i,j}(T) - g(\mathbf{x}_{i,j}, T)|, \tag{3.18}$$

where A is the unit area of Ω_c and the sum and max are over all points $\mathbf{x}_{i,j} \in \Omega$.

Letting $\mathbf{u} = 0$ in the IBVP described above, we recover the diffusion equation considered by Marichal. After selecting parameters

$$\begin{array}{llll} x_c = 0.513, & x_g = 0.4, & r = 0.123, & \nu = 0.01, \\ y_c = 0.537, & y_g = 0.4, & g_0 = 1.0, & \sigma = 0.6, \end{array}$$

and spatial resolutions $h = 1/32, /1/64, /1/128, 1/256$, and $1/512$, the resulting immersed interface discretizations are integrated from time $t = 0$ to $T = 0.1$ with time step $\tau = 10^{-5}$. Figure 3-2 shows the variation of the resulting L_2 and L_∞ error with the grid spacing h; as expected, the method is second order in both.

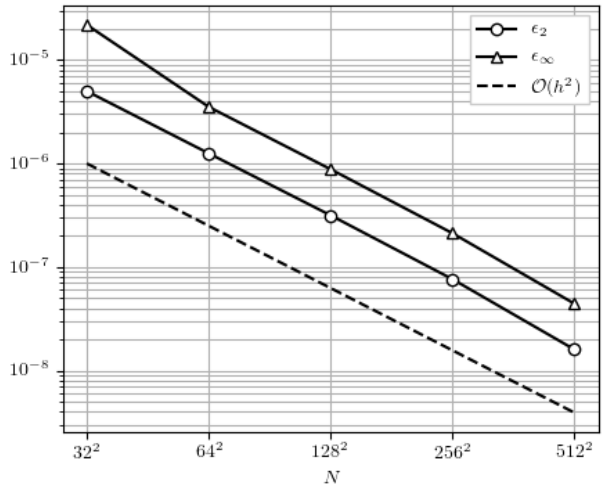


Figure 3-2: Second Order Convergence for purely diffusive process.

Having achieved a satisfactory method of handling boundary conditions for the diffusion equation, we will continue to use this method whenever we calculate the diffusive term in the advection-diffusion equation.

3.3.2 Advection Diffusion Part 1

We now turn to boundary conditions for the advection term in the advection-diffusion equation. The third-order advection scheme discussed in section 3.2.3 produces a stencil that extends to two neighboring points in the upwind direction. Consequently, we must consider immersed interface corrections for points in $\mathcal{A}^+$ and for points with neighbors in $\mathcal{A}^+$. This issue exists for any standard finite difference discretization of order higher than two, and has been addressed in various ways by other authors. For example, Linnick and Fasel [14] avoid this issue in their fourth-order diffusion scheme by using compact finite differences. Here we adopt the following strategy:

- For points adjacent to the boundary, the simplest way to handle this issue is to revert to second-order central differencing. The primary reason for the use of a third-order advection scheme was free-space stability; if a second order boundary treatment leads to a stable advection diffusion scheme, than we have accomplished our original goal.

- To account for points adjacent to $\mathcal{A}^+$, the immersed interface corrections are stored in $\mathcal{A}^-$ before calculating the advection term. This ensures that any point which is not in $\mathcal{A}^+$ but requires corrections will receive them.

To implement this discretization in an advection-diffusion solver, we must take the following steps:

- Loop over all $\mathbf{x}_c \in \mathcal{C}$, and calculate a third order jump correction using the given Dirichlet boundary condition. This correction is stored in $A^+(\mathbf{x}_c)$; if there multiple candidates for the value of $A^+(\mathbf{x}_c)$, the average value is used.
- Calculate the advection term using a third-order upwind finite difference stencil.
- Loop over all $\mathbf{x}_c \in \mathcal{C}$ again. At each point, recalculate $u_\xi \partial_\xi f$ using a third-order upwind difference, remove this contribution from $(\mathbf{u} \cdot \nabla) f$ at $A^+(\mathbf{x}_c)$, and replace it with $u_\xi \partial_\xi f$ calculated using a second-order centered stencil.
- Zero all of the points in $\mathcal{A}^-$, and calculate the diffusion term as described in section 3.3.1.

To test this strategy, we return to the numerical setup established in the previous section. This time the chosen geometric parameters are

$$x_c = 0.417, \quad y_c = 0.409, \quad r = 0.123,$$

along with solution parameters

$$\begin{array}{lll} u_x = 0.8, & x_g = 0.3, & g_0 = 1.0, \\ u_y = 0.4, & y_g = 0.3, & \sigma = 0.5, \end{array}$$

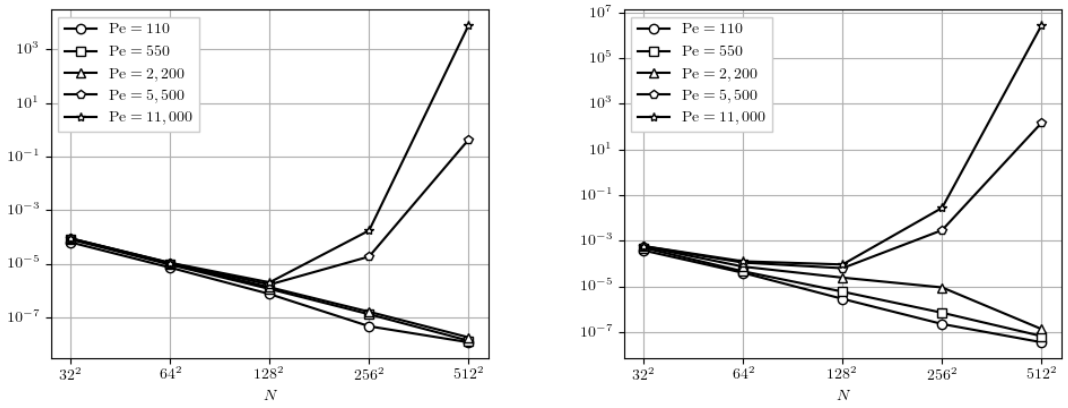
so that the free-stream velocity $\mathbf{u}$ is nonzero. To explore a range of Péclet numbers, the diffusivity is chosen from the set

$$\nu \in \{2 \times 10^{-3},\ 4 \times 10^{-4},\ 1 \times 10^{-4},\ 4 \times 10^{-5},\ 1 \times 10^{-5}\},$$

which correspond to Péclet numbers

$$\mathrm{Pe} \in \{110,\ 550,\ 2200,\ 5500,\ 11000\}.$$

Figure 3-3 shows the resulting convergence behavior. At lower Pe, the numerical solution appears to converge to the analytical solution with normal power-law behavior. However, at higher Pe the numerical solution fails to convergence at all, due to a numerical instability. This lack of stability



(a) L_2 Error. (b) L_∞ Error.

Figure 3-3: Instability leads to a lack of convergence at high Pe for a naive IIM advection-diffusion discretization.

makes sense for the high Pe limit, which allows for the formation of increasingly thin boundary layers. At low resolutions, the hyper-diffusive error term in the [3U] stencil helps to damp out high-frequency modes associated with instability. As the grid is refined, the magnitude of this numerical

diffusion decreases, and we reach a resolution that is neither coarse enough to damp the instability nor fine enough to resolve the boundary layer. At this point, instability creeps in.

Although this failure is logical, it certainly isn't desirable. We would like a robust transport scheme that fails gracefully during under-resolved simulations, providing an answer that is at least qualitatively correct instead of a numerical blow-up. We would also like to see smooth convergence behavior as we sweep from a heavily under-resolved test case to a well-resolved test case, instead of encountering a middle region where there is no convergence at all. With this in mind, we continue our search for an IIM discretization that remains stable at arbitrarily high Pe.

3.3.3 Advection Problems

To understand the advection-diffusion equation at high Péclet numbers, we take inspiration from the advection equation

$$\frac{\partial f}{\partial t} + (\mathbf{u} \cdot \nabla) f = 0. \tag{3.19}$$

For (3.19), as well as for any hyperbolic system, imposing boundary conditions everywhere on the domain boundary can lead to an ill-posed IBVP. An easy way to see this is to move to a Lagrangian formulation of the same problem, using the method of characteristics. Let $\sigma(t)$ be the path of a fluid particle, which satisfies

$$\frac{\mathrm{d}\sigma}{\mathrm{d}t} = \mathbf{u}(\sigma, t). \tag{3.20}$$

Then the scalar concentration $f(\sigma(t))$ in this fluid particle satisfies

$$\frac{\mathrm{d}}{\mathrm{d}t} f(\sigma(t)) = \frac{\partial f}{\partial t} + \frac{\mathrm{d}\sigma}{\mathrm{d}t} \cdot \nabla f = 0, \tag{3.21}$$

so that the value of particle's scalar concentration is constant in time. Each particle entering the domain must receive a value, which is determined at the time of entry by the imposed boundary condition. Each particle leaving the domain brings its predetermined value back to the boundary, where it may or may not agree with the boundary condition that is imposed there. To avoid conflicts, we can define the inlet region $\mathcal{I}(t)$ to be the set of boundary points for which $\mathbf{u}(\mathbf{x}, t) \cdot \hat{\mathbf{n}} > 0$, where $\hat{\mathbf{n}}$ is a normal vector pointing into the fluid domain. All particles must enter the domain through the inlet, and cannot return to the inlet on their way out of the domain. We can now construct the well-posed problem

$$\begin{aligned} \frac{\partial f}{\partial t} + (\mathbf{u} \cdot \nabla) f &= 0 \\ f(\mathbf{x}, 0) &= f_0(\mathbf{x}) \\ f(\mathbf{x}, t) &= \hat{f}(\mathbf{x}, t) \quad \text{for} \quad \mathbf{x} \in \mathcal{I}(t). \end{aligned}$$

This offers some inspiration for the advection diffusion equation: if the advection term is calculated using a boundary condition only on the inlet, improved convergence properties could follow.

To explore this idea further, we consider imposing numerical boundary conditions on a one-dimensional problem. Consider a one-dimensional discretization of the advection equation on $\Omega = [0, 1]$, with constant wave speed $u > 0$. Let the domain be represented by $N + 1$ equally spaced points $x_i = i/N$, and let f_i be the value of the approximate solution at x_i. Fluid enters the domain at x_0, and leaves the domain at x_N, so that a suitable boundary condition is to fix the value of f_0. The value of f_N must remain as an unknown, since no boundary condition is given; placing an artificial Dirichlet or Neumann condition here would over-determine the continuous problem.

Having established what we can and cannot do in one-dimension, we move on to a two-dimensional advection problem that behaves like a collection of one-dimensional problems. Consider the two-dimensional advection equation with constant velocity $\mathbf{u} = [u_x, u_y] = [1,\ 1]$. Any function of the form

$$\tilde{f}(\mathbf{x}, t) = g(x - u_x t) \tag{3.22}$$

will satisfy this advection equation on an unbounded domain. Although the velocity field does not align with the grid lines, making the physical problem inherently two-dimensional, the solution (3.22) purposely has no y-dependence.

For a finite domain Ω, we can impose (3.22) as an initial condition and as a boundary condition, so that the IVBP

$$\begin{aligned} \frac{\partial f}{\partial t} + (\mathbf{u} \cdot \nabla) f = 0 \quad &\text{for} \quad \mathbf{x} \in \Omega \\ f(\mathbf{x}, 0) = \tilde{f}(\mathbf{x}, 0) \quad &\text{for} \quad \mathbf{x} \in \Omega \\ f(\mathbf{x}, t) = \tilde{f}(\mathbf{s}, t) \quad &\text{for} \quad \mathbf{s} \in \mathcal{I} \end{aligned}$$

has $\tilde{f}(\mathbf{x}, t)$ as its unique solution. For concreteness, let the problem domain be the square $(x, y) \in [0, 1] \times [0, 1]$ with a circular region removed. We introduce the angular coordinate θ to talk about regions on the boundary of the circle, with $\theta = 0$ representing the point on the circle with the maximal x coordinate. With this convention, the inlet for the continuous problem is $\mathcal{I} = [-\pi/4,\ 3\pi/4]$.

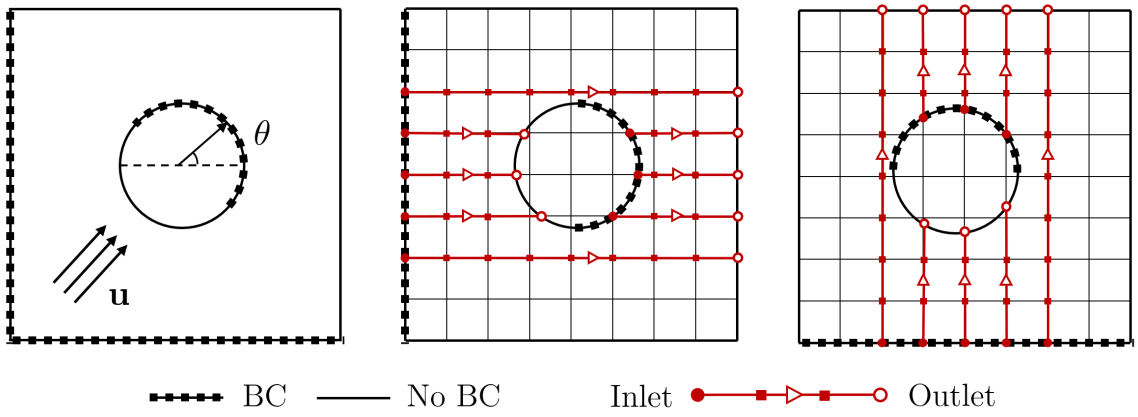


Figure 3-4: An illustration of the continuous problem presented here, along with the two quasi-one-dimensional discretizations it produces. The red overlay illustrates the decoupled one-dimensional advection problems, and the heavy dashed lines indicates locations where a boundary condition must be enforced to ensure stability or well-posedness.

If this IBVP is discretized spatially with a regular grid, centered second-order finite differences, and a standard IIM boundary treatment, then the two-dimensional discretization is reduced to a series of one-dimensional advection problems coupled only by truncation error. To show this, we apply the spatial operators to the exact solution $\tilde{f}_{i,j}$:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} f_{i,j} &= \frac{u_x}{2h}(f_{i+1,j} - f_{i-1,j}) + \frac{u_y}{2h}(f_{i,j+1} - f_{i,j-1}) \\ &= \frac{u_x}{2h}[g(x_{i+1} - u_x t) - g(x_{i-1} - u_x t)] + \frac{u_y}{2h}[g(x_i - u_x t) - g(x_i - u_x t)] + \mathcal{O}\big(h^2\big) \\ &= \frac{u_x}{2h}[g(x_{i+1} - u_x t) - g(x_{i-1} - u_x t)] + \mathcal{O}\big(h^2\big). \end{aligned}$$

Thus the x-direction contribution to each time derivative is $\mathcal{O}(1)$, while the y-direction contribution is at most $\mathcal{O}\big(h^2\big)$. If we neglect the contribution from the y-direction, we see that the dynamics at each point depend only on that point's left and right neighbors, so that each x-gridline is completely decoupled from the others.

We know that, in order to be well-posed, each one-dimensional advection problem will need a boundary condition on its inflow, and cannot have a boundary condition on its outflow. This suggests

that our discretization will be well-behaved if we impose a boundary condition on the numerical inlet $\mathcal{I}_x = [-\pi/2,\ \pi/2]$. We can also repeat the above exercise with a new quasi-one-dimensional solution of the form

$$\tilde{f}(\mathbf{x}, t) = h(y - u_y t), \tag{3.23}$$

leading to a system of decoupled y-gridlines that require a numerical inlet $\mathcal{I}_y = [0,\ \pi]$. Neither $\mathcal{I}_x$ or $\mathcal{I}_y$ coincides with $\mathcal{I}$ for the continuous problem, so it seems highly unlikely that enforcing boundary conditions only on $\mathcal{I}$ will lead to a stable discretization. For a pure advection problem, we are stuck: there is no boundary condition provided on $\mathcal{I}_x \setminus \mathcal{I}$ or $\mathcal{I}_y \setminus \mathcal{I}$, even though we require a BC on these regions for stability.

3.3.4 Advection Diffusion Part 2

If we return to the advection diffusion equation, where a BC is prescribed on all boundaries, then we can make headway on the conundrum presented in the previous section. One simple way to discretize both the x-gridline and y-gridline problems simultaneously is to impose a boundary condition only at x-intersections that lie in $\mathcal{I}_x$ and at y-intersections that lie in $\mathcal{I}_y$. In the local frame of each control point, we impose a boundary condition only when $u_\xi > 0$. Thus we use a boundary condition only when $\mathbf{x}_c$ acts as an inlet for the local ξ-direction advection term, and choose to use a one-sided stencil when $\mathbf{x}_c$ acts as an outlet. Incorporating this idea into the previously proposed advection-diffusion scheme, we obtain the following procedure:

- Loop over all $\mathbf{x}_c \in \mathcal{C}$, and calculate a third order jump correction using the given Dirichlet boundary condition. This correction is stored in $A^+(\mathbf{x}_c)$; if there multiple candidates for the value of $A^+(\mathbf{x}_c)$, the average value is used.
- Calculate the advection term using a third-order upwind finite difference stencil.
- Loop over all $\mathbf{x}_c \in \mathcal{C}$ again. At each point, recalculate $u_\xi \partial_\xi f$ using a third-order upwind difference, remove this contribution from $(\mathbf{u} \cdot \nabla) f$ at $A^+(\mathbf{x}_c)$. Then:
  - If $u_\xi > 0$, recalculate $u_\xi \partial_\xi$ with a centered second-order stencil.
  - If $u_\xi < 0$, recalculate $u_\xi \partial_\xi$ with a one-sided second-order stencil.

  This new value of $u_\xi \partial_\xi$ is added back to $(\mathbf{u} \cdot \nabla) f$ at $A^+(\mathbf{x}_c)$.
- Zero all of the points in $\mathcal{A}^-$, and calculate the diffusion term as described in section 3.3.1.

To test the stability of this discretization, we return to the test problem from section 3.3.1. As a challenge to our scheme, consider the advection diffusion equation with a diffusivity that is nonzero, but smaller than machine epsilon. The resulting IBVP is still parabolic, and requires a boundary condition on all surfaces; however, the numerical value of diffusion term is so small that it can effectively be neglected allowing us to set $\nu = 0$ in our discretization. Rerunning the established convergence test, we find that the numerical solution converges to the exact solution without any stability issues (Figure 3-5), even though the Péclet number in our numerical solver is effectively infinite. Although the boundary treatment is second order, the resulting solutions are $\mathcal{O}(h^3)$ accurate in both the L_2 and L_∞ error norms.

To confirm that this discretization is still valid at moderate Pe, the same test case is repeated using diffusivities

$$\nu \in \{2 \times 10^{-3},\ 4 \times 10^{-4},\ 1 \times 10^{-4},\ 4 \times 10^{-5},\ 1 \times 10^{-5}\},$$

which correspond to Péclet numbers

$$\mathrm{Pe} \in \{110,\ 550,\ 2200,\ 5500,\ 11000\}.$$

Figure 3-6 shows the resulting convergence behavior, which varies with both Péclet number and resolution. For each choice of Pe, spatial convergence is initially dominated by the third order

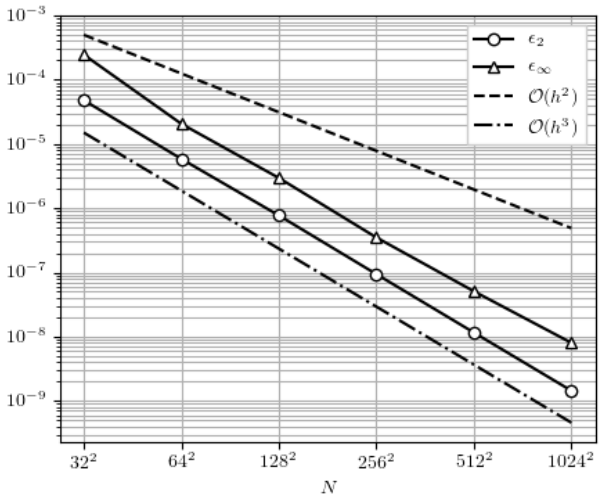
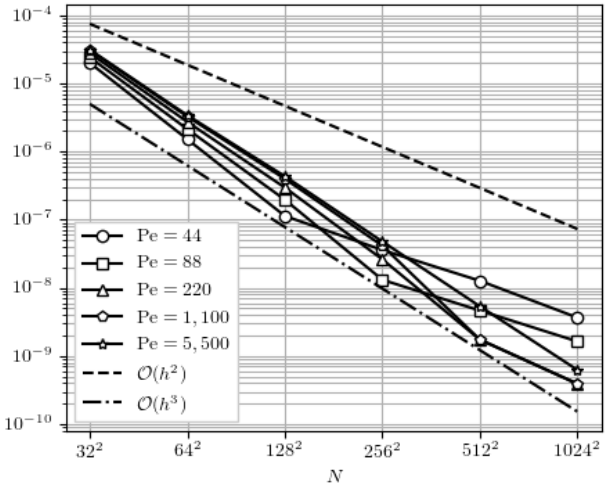
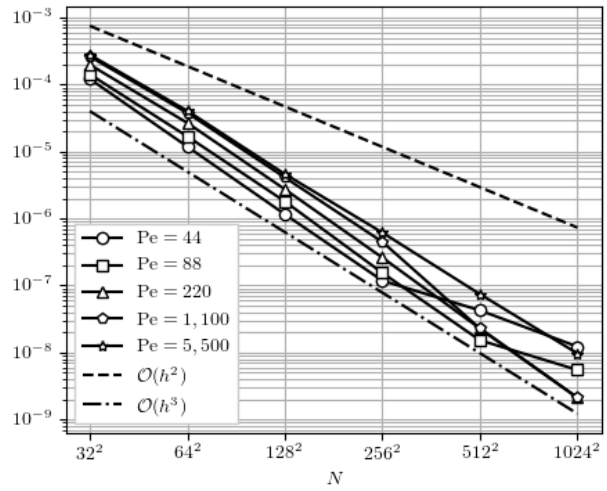


Figure 3-5: Third Order Convergence of an advection-diffusion process with vanishing diffusivity.

advection term, transitioning to second order past a critical resolution. This critical resolution is finer at higher Pe. Having verified that this advection-diffusion discretization is stable and between second and third order accurate over a broad range of Péclet numbers, we will continue to use it as the default transport scheme throughout the rest of this thesis.



(a) L_2 Error.



(b) L_∞ Error.

Figure 3-6: Mixed second and third order spatial convergence for an advection-diffusion discretization over a wide range of Péclet numbers.

Chapter 4

IIM for the Navier Stokes Equations

In vorticity-velocity form, the Navier-Stokes equations for an unbounded two-dimensional domain are

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \nabla^2 \omega, \tag{4.1}$$

$$\nabla \wedge \mathbf{u} = \omega, \tag{4.2}$$

along with the incompressibility constraint $\nabla \cdot \mathbf{u} = 0$. Having developed immersed interface tools to solve (4.1) in the previous section, we turn our eye to the reconstruction of a divergence-free velocity field $\mathbf{u}$ satisfying (4.2). With the introduction of an immersed body, we must also consider boundary conditions on (4.1) and (4.2) which ensure that the no-slip and no-through-flow boundary conditions are satisfied at the body surface. This section reviews the immersed interface tools that have been developed for both of these problems, completing the theory necessary to successfully discretize the Navier stokes equations.

4.1 Kinematics of Vorticity

4.1.1 Kinematics on an Simply-Connected Domain

Vortex methods track the evolution of a velocity field through the vorticity field $\boldsymbol{\omega} = \nabla \wedge \mathbf{u}$. Consequently, an essential part of vortex methods is the reconstruction of a velocity field given only its associated vorticity field. This can be done in a variety of ways. One option is to take the curl of (4.2), which gives a vector Poisson equation relating the two fields:

$$\nabla^2 \mathbf{u} = -\nabla \wedge \boldsymbol{\omega}. \tag{4.3}$$

We can also describe the the solenoidal field $\mathbf{u}$ through a stream function $\boldsymbol{\psi}$, so that the velocity field $\mathbf{u} = \nabla \wedge \boldsymbol{\psi}$ is automatically divergence free. If the stream function satisfies the vector Poisson equation

$$\nabla^2 \boldsymbol{\psi} = -\boldsymbol{\omega}, \tag{4.4}$$

then the resulting velocity field will also satisfy (4.3).

This approach is particularly efficient in two dimensions, where both the vorticity and the stream function become scalar fields. We define a two-dimensional stream function ψ that determines the velocity field through

$$u_x = \partial_y \psi \quad \text{and} \quad u_y = -\partial_x \psi. \tag{4.5}$$

Note that ψ is not unique; for any constant c, ψ and $\psi + c$ determine the same velocity field. In two dimensions, this stream function must obey the scalar Poisson equation $\nabla^2 \psi = -\omega$. To determine

boundary conditions which ψ must satisfy, we transfer (4.5) to the domain boundaries, giving

$$\mathbf{u}_b \cdot \hat{\mathbf{n}} = \partial_s \psi \quad \text{and} \quad \mathbf{u}_b \cdot \hat{\mathbf{s}} = -\partial_n \psi.$$

If we wish to enforce the no-through-flow condition on the domain boundary, we must enforce $\partial_s \psi = \mathbf{u}_b \cdot \hat{\mathbf{n}}$. This is traditionally integrated over the boundary to give a Dirichlet boundary condition $\psi = \psi_b + \bar{\psi}$, where $\bar{\psi}$ is an arbitrary constant. In a simply connected domain Ω, the non-uniqueness of ψ implies that the choice of $\bar{\psi}$ will not affect the resulting velocity field, so we are free to set $\bar{\psi} = 0$. Collecting the previous results gives an elliptic boundary value problem for the stream function,

$$\begin{aligned} -\nabla^2 \psi &= \omega \\ \psi &= \psi_b \quad \text{on} \quad \partial\Omega. \end{aligned} \tag{4.6}$$

4.1.2 Kinematics on a Multiply-Connected Domain

In a multiply-connected domain, the ambiguity in the Dirichlet boundary condition is no longer negligible. Consider a collection of N objects immersed in an unbounded fluid domain, each occupying a connected region $\Omega_i \in \mathbb{R}^2$. We can integrate the condition $\partial_s \psi = \mathbf{u}_b \cdot \hat{\mathbf{n}}$ around each object boundary $\partial\Omega_i$, giving one arbitrary constant $\bar{\psi}_i$ for each object. Similarly, the far-field boundary condition

$$\lim_{|x| \to \infty} \nabla \wedge \psi = \mathbf{u}_\infty$$

provides an additional scalar degree of freedom, since it is unaffected by the addition of a global constant $\psi \to \psi + \bar{\psi}_\infty$. The non-uniqueness of ψ can absorb this global constant, but we are left with N undetermined constants $\bar{\psi}_i$. To choose one particular flow from this N-dimensional space of possible solutions, we must provide N additional constraints on ψ by prescribing the circulation around the boundary of each object:

$$\oint_{\partial\Omega_i} \mathbf{u} \cdot \mathrm{d}\mathbf{s} = \Gamma_i.$$

With these circulation constraints, we form the complete reconstruction problem for the stream function in an two-dimensional, unbounded, multiply-connected domain:

$$\begin{aligned} \nabla^2 \psi &= -\omega \\ \lim_{|x| \to \infty} \nabla \wedge \psi &= \mathbf{u}_\infty \\ \psi &= \psi_{b,i} + \bar{\psi}_i \quad \text{on} \quad \partial\Omega_i \\ \oint_{\partial\Omega_i} \mathbf{u} \cdot \mathrm{d}\mathbf{s} &= \Gamma_i. \end{aligned} \tag{4.7}$$

One effective strategy for solving the Poisson equation on an unbounded domain is to use Green's functions, which typically guarantee that $|\nabla \wedge \psi| \to 0$ as $|\mathbf{x}| \to \infty$. If the non-homogeneous boundary condition $\mathbf{u} = \mathbf{u}_\infty$ is prescribed, then it is convenient to make the decomposition $\psi = \tilde{\psi} + \psi_\infty$, where ψ_∞ is a free-stream stream function satisfying $\nabla \wedge \psi_\infty = \mathbf{u}_\infty$ and $\tilde{\psi}$ is a perturbation satisfying

$$\begin{aligned} \nabla^2 \tilde{\psi} &= -\omega \\ \lim_{|x| \to \infty} \nabla \wedge \tilde{\psi} &= \mathbf{0} \\ \tilde{\psi} &= \tilde{\psi}_{b,i} + \bar{\psi}_i \quad \text{on} \quad \partial\Omega \\ \oint_{\partial\Omega} \mathbf{u} \cdot \mathrm{d}\mathbf{s} &= \Gamma_b. \end{aligned} \tag{4.8}$$

Here $\tilde{\psi}_{b,i} = \psi_{b,i} - \psi_\infty$. For the rest of this thesis, we consider only a single obstacle, so that we can drop the subscript and simply write $\tilde{\psi} = \tilde{\psi}_b + \bar{\psi}$ on $\partial\Omega$.

4.1.3 Discretized Kinematics

In this thesis, (4.8) is solved using a small modification of the iterative immersed interface algorithm proposed by Gillis in [9] and in [10]. The rest of this section describes this modification, and assumes some familiarity with this prior work. To discuss the spatial discretization of (4.8), we adopt the notation for multidimensional IIM problems developed in Chapter 2, and define the following vector spaces:

- V_a - the space of functions defined on $\mathcal{A}$ (the affected points);
- V_i - the space of functions defined on $\mathcal{I}_n$ (the interpolation points);
- V_c - the space of functions defined on $\mathcal{C}$ (the control points);
- V_G - the space of functions defined on $\mathcal{G}$ (the entire Cartesian grid).

We then introduce linear operators to connect the spaces $\mathcal{A}$ and $\mathcal{I}_n$ to the larger Cartesian grid $\mathcal{G}$:

- $E_a : V_a \to V_G$ places a vector of values defined on $\mathcal{A}$ into their global position within $\mathcal{G}$;
- $E_i : V_i \to V_G$ places a vector of values defined on $\mathcal{I}_n$ into their global position within $\mathcal{G}$.

It's helpful to note that E_a and E_i are orthonormal, so that $E_a^T E_a = I \in V_G$ and $E_i^T E_i = I \in V_G$. Thus the transpose of each takes a field defined on the Cartesian grid and restricts it to the interpolation points or the affected points. To finish off the set, we define operators representing a discretization of the immersed interface method and the free-space Poisson solver.

- $C : V_i \times V_c \to V_a$ calculates IIM jump corrections based on function values at the interpolation points and a boundary condition at the control points. When convenient, we'll split C into two linear operators $A : V_i \to V_a$ and $B : V_c \to V_a$, so that $C(x_i, x_c) = Ax_i + Bx_c$ for $x_i \in V_i$ and $x_c \in V_c$.
- $\nabla_h^2 : V_G \to V_G$ is a finite difference discretization of the Laplacian operator. Its inverse $\nabla_h^{-2} : V_G \to V_G$ represents convolution with a free-space Lattice Green's Function, as detailed in [10].
- $\Sigma_a : V_a \to \mathbb{R}$ and $\Sigma_G : V_G \to \mathbb{R}$ sum the components of a vector in V_a or V_G respectively, and are useful in discretizing the circulation constraint in (4.8).

For this problem, our immersed interface corrections must be two-sided, so that any point $\mathbf{x}_c \in \mathcal{C}$ will contribute a jump correction J_α^+/h^2 to the point $A^+(\mathbf{x}_c)$ and a jump correction J_α^-/h^2 to the point $A^-(\mathbf{x}_c)$. Using the local stencil notation from section 2.2,

$$J_\alpha^+ = s_{\xi,\alpha} f_\alpha + \sum_{i=1}^{3} s_{\xi,i} f_i \quad \text{and} \quad J_\alpha^- = -f_0,$$

so that the corrected operator $\nabla_h^2 + E_a C$ evaluates correctly at $A^+(\mathbf{x}_c)$ and evaluates to zero at $A^-(\mathbf{x}_c)$.

Using the operators defined above, we can write an immersed interface discretization of 4.8 as

$$\nabla_h^2 \tilde{\psi} + E_a C(E_i^T \tilde{\psi}, \tilde{\psi}_b + \bar{\psi}) = -\omega, \tag{4.9}$$

with $\tilde{\psi}$ and ω now restricted to $\mathcal{G}$, $\tilde{\psi}_b$ restricted to $\mathcal{C}$, and $\bar{\psi}$ representing a constant function defined on $\mathcal{C}$. Splitting C into its linear components A and B and rearranging, we see that

$$\begin{aligned} \nabla_h^2 \tilde{\psi} + E_a A E_i^T \tilde{\psi} + E_a B(\tilde{\psi}_b + \bar{\psi}) = -\omega, \quad \text{or} \\ (\nabla_h^2 + E_a A E_i^T)\tilde{\psi} = -E_a B(\tilde{\psi}_b + \bar{\psi}) - \omega. \end{aligned}$$

The operator $\nabla_h^2 + E_a A E_i^T$ is a low-rank perturbation of the free-space Laplacian, and can be inverted efficiently with the Sherman-Morrison-Woodbury formula:

$$\tilde{\psi} = \nabla_h^{-2}(-E_a A x_b - E_a B(\tilde{\psi}_b + \bar{\psi}) - \omega), \tag{4.10}$$

where $x_b \in V_i$ is a vector satisfying

$$(I_i + E_i^T \nabla_h^{-2} E_a A) x_b = E_i^T \nabla_h^{-2}(-E_a B(\tilde{\psi}_b + \bar{\psi}) - \omega). \tag{4.11}$$

We can determine $\bar{\psi}$ by enforcing the circulation constraint. Instead of setting the circulation around the immersed obstacle, we can equivalently set the total discrete vorticity in the domain, so that

$$\Sigma_G \omega + \Sigma_a C(E_i^T \tilde{\psi}, \tilde{\psi}_b - \bar{\psi}) = 0. \tag{4.12}$$

Equations 4.10, 4.11, and 4.12 are the linear system for the unknowns $(\tilde{\psi}, \bar{\psi})$ developed by Gillis in [9]. In this thesis, we improve Gillis' discretization by using a slightly different application of the SMW formula, replacing (4.10) with

$$\tilde{\psi} = \nabla_h^{-2}(-E_a y_b - E_a B(\tilde{\psi}_b + \bar{\psi}) - \omega), \tag{4.13}$$

where $y_b \in V_a$ is a vector satisfying

$$(I_a + A E_i^T \nabla_h^{-2} E_a) y_b = A E_i^T \nabla_h^{-2}(-E_a B(\tilde{\psi}_b + \bar{\psi}) - \omega). \tag{4.14}$$

Because the unknown y_b is defined on the affected points rather than the interpolation points, this system is smaller than (4.11), which involves as many equations as there are interpolation points. To simplify further, define a new vector $z_b = y_b + B(\tilde{\psi}_b + \bar{\psi})$, so that (4.13) and (4.14) reduce to

$$\tilde{\psi} = -\nabla_h^{-2}(\omega + E_a z_b) \tag{4.15}$$

$$(I_a + A E_i^T \nabla_h^{-2} E_a) z_b = -B(\tilde{\psi}_b + \bar{\psi}) - A E_i^T \nabla_h^{-2} \omega. \tag{4.16}$$

By inspecting (4.15), we see that the circulation constraint is expressed succinctly as

$$\Sigma_G \omega + \Sigma_a z_b = 0. \tag{4.17}$$

Equations 4.15, 4.16, and 4.17 are the full discretization of the stream function reconstruction problem used in this thesis.

4.1.4 Solving the Discrete Stream Function System

The discretized IIM Poisson equations described in the previous section are designed to be solved efficiently using an iterative method. The most expensive operation in the discrete linear system is the application of ∇_h^{-2}, which represents the convolution of a source term with a free-space Lattice Green's function. This convolution can be carried out in $\mathcal{O}(N \log N)$ time using the FFT-base Hockney-Eastwood algorithm described in [9], where N is the number of points in the computational domain. If we wish to apply the modified operator $E_i^T \nabla_h^{-2} E_a$, then we are assured that the source term is defined only on the affected points, and the only relevant output resides on the interpolation points. Consequently, we can carry out the Hockney-Eastwood convolution on any rectangular sub-domain domain containing all of the affected points and interpolation points in $\mathcal{O}(N_b \log N_b)$ time, where N_b is the number of points in this sub-domain. For a small obstacle in a large domain, we may have $N_b \ll N$, so that the application of $E_i^T \nabla_h^{-2} E_a$ is significantly faster than the full ∇_h^{-2} operator.

To solve (4.15) through (4.17), we begin with the discrete source term ω, and integrate over the domain to find the total circulation $\Sigma_G \omega$ appearing in (4.17). Next we perform the convolution $\tilde{\psi}_0 = -\nabla_h^{-2} \omega$, which must be carried out on the entire domain. With this done, we compute the vectors $A E_i^T \tilde{\psi}_0$ and $B \tilde{\psi}_b$ which appear on the right hand side of (4.16). To compute the unknowns

$(z_b, \bar{\psi})$, we rephrase (4.16) and (4.17) as a single linear system:

$$\begin{bmatrix} (I_a + AE_i^T \nabla_h^{-2} E_a) & B \\ \Sigma_a & 0 \end{bmatrix} \begin{bmatrix} z_b \\ \bar{\psi} \end{bmatrix} = \begin{bmatrix} -B\tilde{\psi}_b - AE_i^T \nabla_h^{-2} \omega \\ -\Sigma_G \omega \end{bmatrix}. \tag{4.18}$$

To avoid forming the dense matrix on the left-hand side of (4.18), we solve this system iteratively using a matrix-free GMRES algorithm. Once a solution is found within an acceptable tolerance, we perform one final convolution to reconstruct $\tilde{\psi} = \tilde{\psi}_0 - \nabla_h^{-2} E_a z_b$. The total cost of the solution is then two full Hockney-Eastwood convolutions and one smaller convolution for each GMRES iteration.

If this system must be solved repeatedly for many inputs $(\omega, \tilde{\psi}_b)$, as it would be during a flow simulation, we can further accelerate the GMRES algorithm using a recycling scheme. This procedure uses past solutions to construct an optimal initial guess, which can significantly reduce the number of iterations required to achieve convergence. As an example, the recycling procedure described in [10] reduces the number of iterations required to roughly one iteration per time step during the solution of three-dimensional flow problems.

4.1.5 Immersed Interface Curl Operator

The final (and so far neglected) step in velocity reconstruction is to derive a divergence-free velocity field from ψ using

$$u_x = \partial_y \psi \quad \text{and} \quad u_y = -\partial_x \psi. \tag{4.19}$$

This is done using the standard second-order centered finite difference and an immersed interface boundary treatment. To calculate the necessary third-order jump corrections, we must retain the Dirichlet boundary condition $\psi_b + \bar{\psi}$ after solving the stream function system. Unlike the jump corrections considered so far, we must develop separate procedures for the x-intersections $\mathcal{C}_x$ and y-intersections $\mathcal{C}_y$, since the x and y derivatives in (4.19) have different signs and operate on different components of the velocity field. Thus the computation of u_y requires jump corrections only from the points in $\mathcal{C}_x$, while the computation of u_x requires jump corrections only from the points in $\mathcal{C}_y$.

4.2 Vorticity Boundary Conditions

The vorticity-velocity form of the Navier-Stokes equations places a no-slip velocity boundary condition on immersed surfaces. However, because the velocity field is reconstructed from the vorticity field through an elliptic equation, it is difficult to translate the no-slip boundary condition into a traditional boundary condition for the vorticity transport equation. In the words of Rempfer [19], "there is no other formulation of the incompressible Navier Stokes equations that has seen such an extensive use of improper boundary conditions as can be found in the literature on numerical methods for the solution of the vorticity-stream function equations." In this thesis we adopt the boundary condition used by Gillis in his IIM vortex-particle scheme [10], which is quite similar to the approach of Linnick and Fasel in their compact-differencing-based IIM scheme [14] and reminiscent of the local boundary conditions described by E and Liu in [8]. This section reviews a very narrow selection of the literature on vorticity boundary conditions, and then described the specific numerical boundary condition used in this thesis.

4.2.1 Candidate Boundary Conditions

A comprehensive examination of boundary conditions for incompressible flow problems is provided by Quartapelle [18], who considers multiple formulations of the Navier Stokes equations in both two and three dimensions. In the two-dimensional vorticity-stream function formulation, Quartapelle demonstrates that the no-slip boundary condition is equivalent to the requirement that the vorticity field has a pre-determined projection onto to the space of harmonic functions (via the standard L_2 inner product). Thus if $\mathcal{H}(\Omega)$ is the set of functions f satisfying $\nabla^2 f = 0$ on Ω, Quartapelle's

integral condition is

$$\int_\Omega \omega f \, \mathrm{d}V = \oint_{\partial\Omega} a\frac{\partial f}{\partial n} - bf \, \mathrm{d}s \quad \text{for all} \quad f \in \mathcal{H}(\Omega),$$

where a and b are functions derived from the velocity boundary condition. To enforce this condition, Quartapelle develops a numerical method which relies on an implicit discretization of the diffusion term, and produces a symmetric linear system with dimension equal to the number of boundary nodes that must be solved at each time step. Although this solution is theoretically satisfying, the boundary problem is not sparse, meaning that a matrix representation of the problem requires $\mathcal{O}(N)$ memory in two dimensions and $\mathcal{O}\big(N^{4/3}\big)$ memory in three dimensions. Some preliminary research suggests that the condition could be implemented more efficiently via a matrix-free, FFT accelerated algorithm, but this is still considerably more expensive than a local boundary condition. The special implicit treatment of the diffusion term also obstructs any method-of-lines discretization of the incompressible flow problem.

Another widely-used boundary condition is the Lighthill model, which offers a physically motivated boundary treatment popular for vortex-particle methods [15]. In the Lighthill model, the vorticity field is allowed to evolve with a no-flux condition for a small time step τ. This does not enforce the no-slip condition, and at the end of the time step there is generally some spurious slip velocity $\boldsymbol{\gamma} = \mathbf{u} - \mathbf{u}_b$ present on the domain boundary. This spurious slip velocity is interpreted as a singular vortex sheet confined to the body surface, which should have been diffused into the flow over the course of the time step. Thus a pure diffusion step is carried out with the flux boundary condition $-\nu\partial\boldsymbol{\omega}/\partial n = \boldsymbol{\gamma}/\tau$. Variations on this procedure with particle-based advection are examined extensively by Marichal in his doctoral thesis [15]. The weakness of Lighthill schemes is their temporal accuracy, which is generally limited to first-order by the splitting of the diffusion step.

Alternative vorticity boundary conditions are the subject of a publication by E and Liu [8], who argue that global conditions like Quartapelle's do not perform significantly better than approximate local boundary conditions. By treating the diffusion term of the vorticity transport equation implicitly, the authors show that Quartapelle's global condition is well approximated by local conditions such as Thom's formula and Fromm's formula. They provide similar results for a family of global boundary conditions considered by Andersen [1]. Although it is well argued by Wu [29], Rempfer [19], Quartapelle [18], and others that there can be no correct local boundary condition on the vorticity field, we adopt here the view of E and Liu [8] that a local boundary condition and explicit integration, even if not rigorously justified, can still provide satisfactory results with a minimum of computational resources.

4.2.2 IIM Boundary Condition

The boundary condition used here in IIM Navier-Stokes solver developed here is taken directly from [10]. We begin each time step with a vorticity field defined on a grid $\mathcal{G}$, and proceed to reconstruct the velocity field $\mathbf{u}$ at these same points subject to a no-through flow boundary condition. The velocity field at control points $x_c \in \mathcal{C}$ is set equal to the boundary velocity, to satisfy the no-slip condition. Finally, with the velocity field fully defined, we compute the boundary vorticity by taking the curl of the velocity field, so that $\omega(x_c) = \nabla \wedge \mathbf{u}(x_c)$.

Anticipating large vorticity gradients near any solid boundary, we choose to use a second-order finite difference, to avoid oscillations associated with higher-order polynomial interpolation. The resulting finite difference stencils are shown in Figure 4-1. Working in a local frame,

$$\begin{aligned} \omega &= \partial_x u_y - \partial_y u_x \\ &= (-1)^k\big[(-1)^\ell \partial_\xi u_\eta - (-1)^m \partial_\eta u_\xi\big], \end{aligned} \tag{4.20}$$

where the integer k determines the left- or right-handedness of the local coordinate system and the integers $l,\ m$ determine the positive or negative orientation of the ξ and η axes respectively. The derivative $\partial_\xi u_\eta$ is relatively easy to calculate, as all values on the ξ-axis are readily available. To compute $\partial_\eta u_\xi$, we must interpolate values on the η-axis to third order before applying a standard one-sided finite difference stencil.

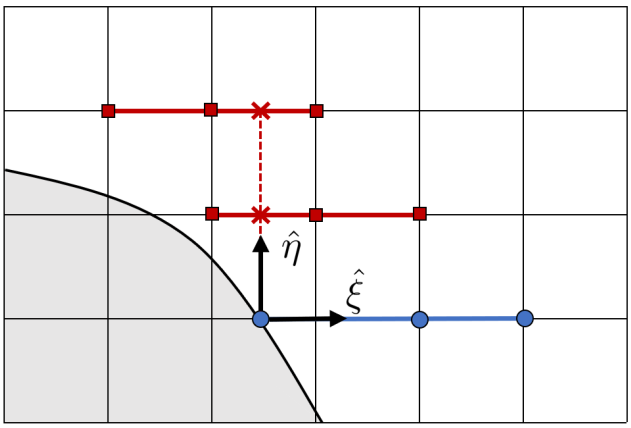


Figure 4-1: Finite Difference Stencils used in the computation of boundary vorticity.

The numerical results presented in Chapter 6 and in [10] demonstrate that this boundary condition leads to second-order spatial accuracy in lift and drag computations for immersed bodies. Though its correctness has not been rigorously proven, this condition provides a local, inexpensive method of enforcing the no-slip boundary condition.

Chapter 5

Force Calculation

This chapter is dedicated to determining the global and local forces acting on a fluid body, using information available in the velocity-vorticity formulation. In an incompressible Newtonian fluid, the local stress tensor $\boldsymbol{\sigma}$ can be written

$$\boldsymbol{\sigma} = \mathbf{T} - p\mathbf{I},$$

where $\mathbf{T}$ is the viscous stress tensor and p is the scalar pressure field. Consequently, all of the loading information we are interested in can be also additively decomposed into two contributions: one from the action of viscosity, and one from the surface pressure distribution. Viscous forces arise from shearing processes, and are easily recovered from the vorticity field. Pressure forces present a greater challenge, since the pressure field is excluded from the vorticity-velocity formulation.

Throughout this chapter, we assume a constant-density incompressible fluid, and for convenience we choose units so that $\rho = 1$. Consequently we'll work with the kinematic viscosity $\nu = \mu/\rho$ in place of the dynamic viscosity μ, and simply write p in place of p/ρ. For two-dimensional problems, all forces and moments are expressed per unit depth.

5.1 Traction an a Material Surface

Let S be a closed surface immersed in a fluid flow, and $\mathrm{d}\mathbf{S}$ be an infinitesimal surface element with area $\mathrm{d}S$ and normal vector $\hat{\mathbf{n}}$. From the definition of the stress tensor, the local traction on the element $\mathrm{d}\mathbf{S}$ is given by $\mathbf{t} = -p\hat{\mathbf{n}} + \mathbf{T}\hat{\mathbf{n}}$. If we assume the fluid is Newtonian, so that $\mathbf{T} = \nu(\nabla\mathbf{u} + \nabla\mathbf{u}^T)$, and that a no-slip boundary condition holds on S, then we can follow an argument presented by Wu et al. in [29] to re-express the local traction vector as

$$\mathbf{t} = -(p + 2\nu r_s)\hat{\mathbf{n}} + \nu(\boldsymbol{\omega} - 2\mathbf{W}) \wedge \hat{\mathbf{n}}, \tag{5.1}$$

where r_s is the local rate of surface stretching and $\mathbf{W}$ is the local angular velocity of the surface. In terms of the surface element,

$$r_s = \frac{1}{\mathrm{d}S}\frac{\mathrm{D}}{\mathrm{D}t}\,\mathrm{d}S \quad \text{and} \quad \mathbf{W} = \frac{\mathrm{D}}{\mathrm{D}t}\hat{\mathbf{n}}.$$

For a general deforming body, r_s and $\mathbf{W}$ can be determined entirely from the surface deformation, without reference to any external velocity field. Thus knowledge of the local pressure and vorticity fields, along with the surface kinematics, can completely characterize the local traction forces acting on an immersed body.

In this thesis we consider only rigid bodies, for which $r_s = 0$ and $\mathbf{W} = \boldsymbol{\Omega}$, the angular velocity of the body. The surface traction then simplifies to

$$\mathbf{t} = -p\hat{\mathbf{n}} + \nu(\boldsymbol{\omega} - 2\boldsymbol{\Omega}) \wedge \hat{\mathbf{n}}, \tag{5.2}$$

which we will split into the local pressure force $\mathbf{f}_p = -p\hat{\mathbf{n}}$ and the local viscous force $\mathbf{f}_v = \nu(\boldsymbol{\omega} - 2\boldsymbol{\Omega}) \wedge \hat{\mathbf{n}}$. We also define the integrated pressure force and moment

$$\mathbf{F}_p = \oint_S \mathbf{f}_p \, \mathrm{d}S \quad \text{and} \quad \mathbf{M}_{p,\bar{\mathbf{x}}} = \oint_S (\mathbf{x} - \bar{\mathbf{x}}) \wedge \mathbf{f}_p \, \mathrm{d}S \,,$$

and similarly define an integrated viscous force $\mathbf{F}_v$ and moment $\mathbf{M}_{v,\bar{\mathbf{x}}}$. The local viscous force $\mathbf{f}_v$ is determined by the vorticity field on the surface, which is readily available in a vorticity-velocity formulation of the Navier-Stokes equations. The integrated viscous loads also take simple forms; it can be shown using the divergence theorem that the identity $\nabla \cdot \mathbf{T} = -\nu \nabla \wedge \boldsymbol{\omega}$ implies

$$\mathbf{F}_v = \nu \oint_S \boldsymbol{\omega} \wedge \hat{\mathbf{n}} \, \mathrm{d}S \,,$$

while a direct integration gives the viscous moment

$$\mathbf{M}_{v,\bar{\mathbf{x}}} = \nu \oint_S (\mathbf{x} - \bar{\mathbf{x}}) \wedge (\boldsymbol{\omega} \wedge \hat{\mathbf{n}}) \, \mathrm{d}S - 2(N-1)\nu V \boldsymbol{\Omega}$$

for a body of volume V immersed in an N-dimensional flow. The integrated pressure forces are more subtle, and will be treated in the next section.

5.2 Integrated Pressure Forces

Although the pressure field p is not directly available in the vorticity-velocity formulation, the pressure gradient ∇p can be extracted directly from the the incompressible Navier Stokes equations

$$\frac{D\mathbf{u}}{Dt} = -\nabla p - \nu \nabla \wedge \boldsymbol{\omega}.$$

If the no-slip boundary condition holds on the immersed surface S, then the acceleration of the fluid is identical to the acceleration of the boundary $\mathbf{a}_b$, and we can write

$$\nabla p = -\nu \nabla \wedge \boldsymbol{\omega} - \mathbf{a}_b \quad \text{on} \quad S. \tag{5.3}$$

Using the the N-dimensional integral identity

$$(N-1) \oint_S \phi \hat{\mathbf{n}} \, \mathrm{d}S = -\oint_S \mathbf{x} \wedge (\hat{\mathbf{n}} \wedge \nabla \phi) \, \mathrm{d}S$$

presented in [16], we can write

$$-\oint_S p\hat{\mathbf{n}} \, \mathrm{d}S = \frac{1}{(N-1)} \oint_S \mathbf{x} \wedge (\hat{\mathbf{n}} \wedge \nabla p) \, \mathrm{d}S \,,$$

and after substituting (5.3) and simplifying considerably we obtain

$$\mathbf{F}_p = \frac{1}{N-1} \oint_S \mathbf{x} \wedge \left(\nu \frac{\partial \boldsymbol{\omega}}{\partial n} - \hat{\mathbf{n}} \wedge \mathbf{a}_b \right) \mathrm{d}S - \frac{\nu}{N-1} \oint_S \mathbf{x} \wedge (\nabla \boldsymbol{\omega} \cdot \hat{\mathbf{n}}) \, \mathrm{d}S \,.$$

In two dimensions, the second term is zero, and we can write

$$\mathbf{F}_{p,2D} = \oint_S \mathbf{x} \wedge \left(\nu \frac{\partial \boldsymbol{\omega}}{\partial n} - \hat{\mathbf{n}} \wedge \mathbf{a}_b \right) \mathrm{d}s \,. \tag{5.4}$$

The integrated pressure moment can be handled using the integral identity

$$\oint_S \frac{x^2}{2}\hat{\mathbf{n}} \wedge \nabla\phi \, \mathrm{d}S = \oint_S \mathbf{x} \wedge \phi\hat{\mathbf{n}} \, \mathrm{d}S \,,$$

which is proven in Appendix B. Letting $\phi = p$ and substituting (5.3), we eventually obtain

$$\mathbf{M}_{p,\bar{\mathbf{x}}} = \frac{1}{2}\oint_S (\mathbf{x}-\bar{\mathbf{x}})^2 \left(\nu\frac{\partial \boldsymbol{\omega}}{\partial n} - \hat{\mathbf{n}} \wedge \mathbf{a}_b\right) \mathrm{d}S - \frac{\nu}{2}\oint_S (\mathbf{x}-\bar{\mathbf{x}})^2 (\nabla\boldsymbol{\omega} \cdot \hat{\mathbf{n}}) \, \mathrm{d}S \,.$$

In two dimensions we can drop the last term to obtain

$$\mathbf{M}_{p,2D,\bar{\mathbf{x}}} = \frac{1}{2}\oint_S (\mathbf{x}-\bar{\mathbf{x}})^2 \left(\nu\frac{\partial \boldsymbol{\omega}}{\partial n} - \hat{\mathbf{n}} \wedge \mathbf{a}_b\right) \mathrm{d}s \,. \tag{5.5}$$

This shows that in both two and three dimensions, we can determine the global loading using only the surface vorticity field, the surface vorticity flux, and the surface kinematics.

5.3 Local Pressure Forces

Obtaining a local pressure value from the vorticity velocity formulation is considerably more difficult than obtaining the integrated pressure force. We'll start with a slightly simpler task, which is to obtain the surface pressure distribution up to an additive constant. In two dimensions we take the tangential component of (5.3),

$$\frac{\partial p}{\partial s} = \nu\frac{\partial \boldsymbol{\omega}}{\partial n} - \mathbf{a}_b \cdot \hat{\mathbf{s}}, \tag{5.6}$$

where s is an arc length coordinate on the boundary curve and $\hat{\mathbf{s}}$ is the tangential unit vector. Choosing an arbitrary point s_0 on the boundary, we find that

$$\tilde{p}(s) = p(s) - p(s_0) = \int_{s_0}^{s} \left(\nu\frac{\partial \boldsymbol{\omega}}{\partial n} - \mathbf{a}_b \cdot \hat{\mathbf{s}}\right) \mathrm{d}s \,. \tag{5.7}$$

If we integrate over the entire length of the boundary C, then we should find that $\tilde{p}(s_0) = \tilde{p}(s_0 + C)$; this guarantee is provided by Kelvin's theorem.

In three dimensions, we can introduce a projection operator $\mathbf{P}_n = \mathbf{I} - \hat{\mathbf{n}}\hat{\mathbf{n}}$, which projects a vector field onto the surface. We also introduce a surface gradient operator, a surface divergence operator, and a surface Laplacian operator, so that we can rewrite (5.3) as

$$\nabla_s p = \mathbf{g}, \quad \text{with} \quad \mathbf{g} = -\mathbf{P}_n(\nu\nabla \wedge \boldsymbol{\omega} + \mathbf{a}_b).$$

Taking the surface divergence of (5.3) yields the surface Poisson equation $\nabla_s^2 p = \nabla_s \mathbf{g}$. This scalar PDE can be solved on the two-dimensional surface, up to an additive constant, to give the local pressure distribution. Since this thesis is focused on two-dimensional flow, we won't pursue this surface PDE any further.

To remove the unknown additive constant from our pressure calculations, we must leave the immersed surface. Taking the divergence of the Navier-Stoke equations gives the usual pressure Poisson equation,

$$\nabla^2 p = -\nabla \cdot (\mathbf{u} \cdot \nabla)\mathbf{u} = \nabla\mathbf{u} : \nabla\mathbf{u}^T. \tag{5.8}$$

Although this is perfectly valid continuously, it is an inconvenient to implement numerically for unbounded problems. The issue lies in the decay rate of the right hand side: in three-dimensions, the far field velocity obeys $|\mathbf{u} - \mathbf{u}_\infty| \sim \mathcal{O}(x^{-3})$, and in a two-dimensions it worsens to $\mathcal{O}(x^{-2})$ for non-lifting flows and $\mathcal{O}(x^{-1})$ for lifting flows. The square of the velocity gradient then scales as $\mathcal{O}(x^{-8})$, $\mathcal{O}(x^{-6})$, and $\mathcal{O}(x^{-4})$, respectively. If the problem is solved on a finite computational domain, then the very small (but non-zero) portion of the source distribution lying outside the

domain will be neglected. To remedy this, we follow [13] and define the total pressure

$$H = (p - p_\infty) + \frac{1}{2}(u^2 - u_\infty^2),$$

which obeys the Poisson equation

$$\nabla^2 H = -(\mathbf{u} \wedge \boldsymbol{\omega}). \tag{5.9}$$

The new right hand side has the same support as the vorticity field, which is usually confined near immersed bodies and decays exponentially as $x \to \infty$. This PDE requires a boundary condition at infinity, as well as a surface boundary condition. By definition, H decays to zero as $x \to \infty$, and a Neumann boundary condition for H can be constructed from (5.3):

$$\begin{aligned}\left.\frac{\partial H}{\partial n}\right|_S &= \frac{\partial p}{\partial n} - \frac{\partial}{\partial n}\left(\frac{u^2}{2}\right) \\ &= -\mathbf{n} \cdot (\nu \nabla \wedge \omega + \mathbf{a}_b) - \mathbf{u}_b \cdot \frac{\partial \mathbf{u}}{\partial n}.\end{aligned}$$

Anticipating that our numerical surface pressure distribution may be much less noisy than the vorticity gradient, we can also use the surface pressure distributions discussed above as a Dirichlet boundary condition for H,

$$\left.H\right|_S = \tilde{p} + \bar{p} + \frac{u_b^2}{2}, \tag{5.10}$$

where $\bar{p}$ is still an unknown additive constant. To determine $\bar{p}$, we introduce the additional integral constraint

$$\begin{aligned}\oint_S \frac{\partial H}{\partial n}\,\mathrm{d}S &= \oint_S -\mathbf{n} \cdot (\nu \nabla \wedge \omega + \mathbf{a}_b) - \mathbf{u}_b \cdot \frac{\partial \mathbf{u}}{\partial n}\,\mathrm{d}S \\ &= -\oint_S \mathbf{n} \cdot \mathbf{a}_b + \mathbf{u}_b \cdot \frac{\partial \mathbf{u}}{\partial n}\,\mathrm{d}S\,,\end{aligned}$$

which does not involve the surface vorticity gradient.

5.4 Global Forces: CV Analysis

In an immersed interface method, calculations preformed on the immersed surface can be significantly noisier than calculations performed on the regular grid. This is compounded by the fact that fluid flow is characterized by thin boundary layers, leading to large and possibly under-resolved gradients in both velocity and vorticity near no-slip boundaries. Consequently, it's advantageous to calculate global loads in a way that avoids the use of surface quantities. To do so, we use the control volume approach developed by Flavio Noca [16], which makes no assumptions on the form of the viscous stress tensor $\mathbf{T}$, the location or position of the control volume, or the boundary conditions on any surface it contains. In this we consider Noca's "impulse 2", "momentum 4", and "flux 4" formulations, specialized to an immersed obstacle with no-slip and no-through-flow conditions on its boundary:

$$F_{f4} = -\frac{\mathrm{d}}{\mathrm{d}t}\oint_S \hat{\mathbf{n}} \cdot [(\mathbf{x} \cdot \mathbf{u})\mathbf{I} - \mathbf{x}\mathbf{u} + \mathbf{u}\mathbf{x}]\,\mathrm{d}S - \frac{\mathrm{d}}{\mathrm{d}t}\oint_{S_b} \hat{\mathbf{n}} \cdot (\mathbf{u}\mathbf{x})\,\mathrm{d}S + \oint_S \hat{\mathbf{n}} \cdot \boldsymbol{\gamma}\,\mathrm{d}S\,, \tag{5.11}$$

$$F_{i2} = -\frac{\mathrm{d}}{\mathrm{d}t}\int_V \mathbf{x} \wedge \boldsymbol{\omega}\,\mathrm{d}V + \frac{\mathrm{d}}{\mathrm{d}t}\oint_{S_b} \hat{\mathbf{n}} \wedge (\mathbf{x} \wedge \mathbf{u})\,\mathrm{d}S + \oint_S \hat{\mathbf{n}} \cdot \boldsymbol{\gamma}\,\mathrm{d}S\,, \tag{5.12}$$

$$F_{m4} = -\frac{\mathrm{d}}{\mathrm{d}t}\int_V \mathbf{u}\,\mathrm{d}V - \frac{\mathrm{d}}{\mathrm{d}t}\oint_S \hat{\mathbf{n}} \wedge (\mathbf{x} \wedge \mathbf{u})\,\mathrm{d}S + \oint_S \hat{\mathbf{n}} \cdot \boldsymbol{\gamma}\,\mathrm{d}S\,. \tag{5.13}$$

The quantity $\boldsymbol{\gamma}$ in all of the above is a tensor collecting miscellaneous terms evaluated on the control volume:

$$\boldsymbol{\gamma} = \frac{1}{2}u^2\mathbf{I} - \mathbf{u}\mathbf{u} - (\mathbf{u} - \mathbf{u}_S)(\mathbf{x} \wedge \boldsymbol{\omega}) + \boldsymbol{\omega}(\mathbf{x} \wedge \mathbf{u}) + [\mathbf{x} \cdot (\nabla \cdot \mathbf{T})\mathbf{I} - \mathbf{x}(\nabla \cdot \mathbf{T})] + \mathbf{T}. \tag{5.14}$$

The surface integral of $\boldsymbol{\gamma}$ is common to all three formulations, and can be simplified considerably by introducing the definition of $\mathbf{T}$ for a Newtonian fluid and noting that $\boldsymbol{\omega} \cdot \hat{\mathbf{n}} = 0$ in two dimensions. This gives a set of expressions involving only the fields $\boldsymbol{\omega}$ and $\mathbf{u}$:

$$\begin{aligned}\oint_S \hat{\mathbf{n}} \cdot \boldsymbol{\gamma}\,\mathrm{d}s &= \oint_S \mathbf{J}\,\mathrm{d}s + \nu \oint_S \mathbf{K}\,\mathrm{d}s\,, \quad \text{with} \\ \mathbf{J} &= \frac{1}{2}u^2\hat{\mathbf{n}} - (\hat{\mathbf{n}} \cdot \mathbf{u})\mathbf{u} - (\hat{\mathbf{n}} \cdot \mathbf{u})(\mathbf{x} \wedge \boldsymbol{\omega}) \\ \mathbf{K} &= (\mathbf{x} \cdot \hat{\mathbf{n}})(\nabla \wedge \boldsymbol{\omega}) - \mathbf{x} \cdot (\nabla \wedge \boldsymbol{\omega})\hat{\mathbf{n}} + \boldsymbol{\omega} \wedge \hat{\mathbf{n}}.\end{aligned}$$

For stationary obstacles, the flux formulation can be implemented using only quantities defined on a bounding box around the object, completely avoiding the use of IIM integration. For moving or deforming objects, the momentum formulation can be implemented without any knowledge of the surface kinematics.

Noca's published work does not include an analogous calculation of the moment acting on an immersed body. However, it can be derived using similar methods, and this calculation is presented in Appendix B. Specialized to an obstacle with no-slip and no through-flow boundary conditions, the resulting "impulse" and "momentum" forms are

$$\mathbf{M}_i = \frac{\mathrm{d}}{\mathrm{d}t}\int_{V(t)} \frac{x^2}{2}\omega\,\mathrm{d}V - \frac{\mathrm{d}}{\mathrm{d}t}\oint_{S_b(t)} \frac{x^2}{2}\hat{\mathbf{n}} \wedge \mathbf{u}\,\mathrm{d}S + \oint_{S(t)} \boldsymbol{\lambda}(\hat{\mathbf{n}})\,\mathrm{d}S\,, \tag{5.15}$$

$$\mathbf{M}_m = -\frac{\mathrm{d}}{\mathrm{d}t}\int_{V(t)} \mathbf{x} \wedge \mathbf{u}\,\mathrm{d}V + \frac{\mathrm{d}}{\mathrm{d}t}\oint_{S(t)} \frac{x^2}{2}\hat{\mathbf{n}} \wedge \mathbf{u}\,\mathrm{d}S + \oint_{S(t)} \boldsymbol{\lambda}(\hat{\mathbf{n}})\,\mathrm{d}S\,. \tag{5.16}$$

The quantity $\lambda(\hat{\mathbf{n}})$ is again introduced to collect miscellaneous surface terms, and can be written as the action of a two-tensor $\lambda = \boldsymbol{\Lambda}\hat{\mathbf{n}}$ if necessary (though it is not convenient to do so here):

$$\begin{aligned}\boldsymbol{\lambda}(\hat{\mathbf{n}}) = \mathbf{x} \wedge \frac{1}{2}u^2\hat{\mathbf{n}} - \frac{x^2}{2}\hat{\mathbf{n}} \wedge (\mathbf{u} \wedge \boldsymbol{\omega}) - \frac{x^2}{2}\hat{\mathbf{n}} \wedge (\nabla \cdot \mathbf{T}) - \left(\frac{1}{2}x^2\boldsymbol{\omega}\right)(\mathbf{u}_s \cdot \hat{\mathbf{n}}) \\ + \mathbf{x} \wedge (\mathbf{T} \cdot \hat{\mathbf{n}}) - (\mathbf{x} \wedge \mathbf{u})(\mathbf{u} \cdot \hat{\mathbf{n}}).\end{aligned} \tag{5.17}$$

It would be convenient to develop a third expression that did not involve any integrals over the computational domain. For force calculations, the identity $\mathbf{u} = \nabla \cdot (\mathbf{u}\mathbf{x})$ allows an easy transition between the flux and momentum forms, but a similar identity for $\mathbf{x} \wedge \mathbf{u}$ is difficult to find. Nevertheless, the momentum formulation given above does not depend on the kinematics of the immersed body, making it a convenient choice for moving bodies.

All of the global force formulas considered here involve integration over the computational domain, its outer boundary, or the boundary of an immersed obstacle. Integrals over a regular outer boundary are performed using a simple composite trapezoidal rule, while the other two are performed using the methods discussed in Chapter 2. The domain integrals can be performed with or without using a boundary condition on the immersed object; in this thesis, we always choose to use a boundary condition, which is constructed either from $\mathbf{u}_b$ or ω_b.

5.5 Calculating Vorticity Flux

The boundary vorticity flux plays a vital role in several of the formulas given here, but is not directly available from the Dirichlet boundary condition ω_b. It turns out that calculating this flux smoothly and accurately is exceptionally difficult, due to the formation of boundary layers near

no-slip boundaries. If the boundary layer thickness is a small distance δ, we expect that the wall vorticity scales as U_∞/δ, so that the wall vorticity flux scales as $\nu U_\infty/\delta^2$. Resolving this sharp gradient requires the use of a grid spacing h that is several times smaller than δ, which may be a prohibitively fine discretization for high Reynolds number flow. This difficulty is a key motivation for the use of control volume methods, which can give accurate results without directly probing the boundary layer.

The computation of vorticity flux in three dimensions is considered by Gillis in [10]. In two dimensions, the smoothest and most robust second-order method known to the author is somewhat roundabout. Using the signed distance function ϕ, we compute the value of $\nabla\phi \cdot \nabla\omega$ in the vicinity of the boundary, which represents a smooth extension of $(\hat{\mathbf{n}} \cdot \nabla)\omega$ into the domain. This is done with second-order centered finite difference stencils. For affected points, we must correct this operation using third-order jump corrections that incorporate the Dirichlet boundary condition ω_b. This flux field is then extrapolated to the immersed boundary, using a two-dimensional linear least squares fit at every control point (Figure 5-1). This procedure combines noisy wall information with values computed using regular stencils away from the boundary, and the two-dimensional nature of the final extrapolation improves somewhat the smoothness of the resulting boundary flux distribution. A simple alternative is to compute the wall gradient $\nabla\omega(x_c)$ directly, using the same finite difference

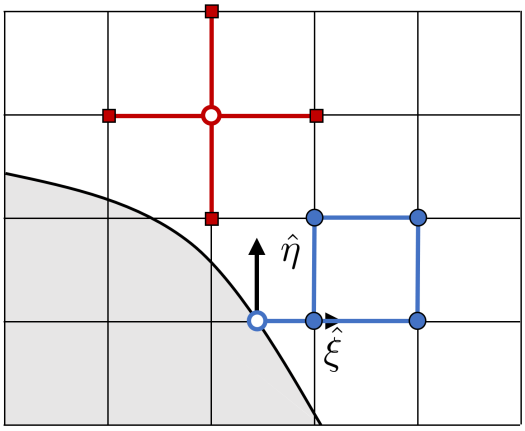
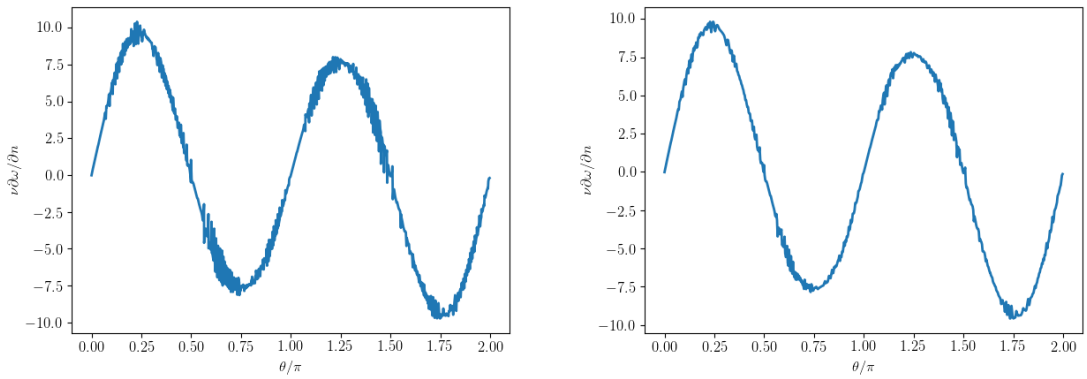


Figure 5-1: Stencils for computing wall vorticity flux. The field $\partial_n\omega$ is computed with a regular stencil (red) in the domain, and then interpolated to the wall using a four-point linear least-square fit (blue).

stencils as the Dirichlet vorticity boundary condition described in Chapter 4. This formulation relies entirely on second-order polynomials for the construction of interpolation and differencing stencils, which helps avoid the oscillatory behavior commonly observed in higher order polynomial interpolation. It also manages to incorporate a large two-dimensional patch of points near the boundary, even though the benefits of a true two-dimensional fitting operation are lost. Figure 5-2 shows side-by side the flux-distributions resulting from the each discretization discussed here, to illustrate the slight gains in smoothness form least-squares fitting. Though the vorticity flux itself may be quite noisy, the pressure force formulas given here involve only integrals of the flux. Thus we are able to construct a much smoother pressure distribution on the surface of an immersed obstacle, despite our inability to calculate a smooth flux. This capability will be demonstrated in the next chapter, where we consider an immersed interface discretization of the full Navier-Stokes equations.



(a) Wall Derivatives.

(b) Least Squares.

Figure 5-2: Sample vorticity flux distributions computed using the two methods proposed here. The flow problem considered is flow past a cylinder (Chapter 6), and the angular coordinate θ is defined to be zero at the forward stagnation point.

Chapter 6

Numerical Results: Navier Stokes with Stationary Boundaries

The material covered in the previous four chapters is enough to successfully discretize the Navier Stokes equations on a stationary domain using the Immersed Interface Method. Other successful IIM discretizations of the Navier-Stokes equations in vorticity-velocity formulation have already been presented by Linnick and Fasel [14], Marichal [15], and Gillis [10]. Our purpose here is not to re-invent the wheel; the method we present is only a slight modification of [10]. The goal of this chapter is simply to describe the full discretization, and demonstrate that it achieves the expected convergence properties and accurately predicts local and global loads. With this foundation in place, we will be well prepared to extend the discretization to flow problems with moving boundaries.

6.1 An IIM Navier Stokes Solver

In vorticity velocity formulation, the two-dimensional incompressible Navier Stokes equations are

$$\begin{aligned} \frac{\partial \omega}{\partial t} &= -\mathbf{u} \cdot \nabla \omega + \nu \nabla^2 \omega \\ \nabla^2 \psi &= -\omega \\ \mathbf{u} &= \nabla \wedge \psi. \end{aligned}$$

The first of these represents the transport of vorticity, while the last two are a convenient representation of the incompressibility constraint $\nabla \cdot \mathbf{u} = 0$ and the vorticity definition $\omega = \nabla \wedge \mathbf{u}$. For a single obstacle immersed in an unbounded fluid domain, we supplement these equations with the boundary conditions

$$\begin{aligned} \mathbf{u} &= \mathbf{u}_b \quad \text{on} \quad \partial\Omega, \\ \lim_{|x| \to \infty} \mathbf{u} &= \mathbf{u}_\infty. \end{aligned}$$

These are the no-slip condition imposed on the boundary of the object, and the condition that the free-stream flow is undisturbed infinitely far away from the obstacle. Given an initial vorticity distribution ω_0, these equations provide sufficient information to determine the vorticity field $\omega(\mathbf{x}, t)$ at any future time.

Numerically, we reformulate the continuous problem as a large nonlinear system of ODEs of the form

$$\frac{\mathrm{d}\omega}{\mathrm{d}t} = f(\omega, t),$$

where ω represents a vector containing the value of the vorticity field at every grid point in the computational domain. We begin by computing the signed distance function for our immersed

obstacle, and identifying the control points $\mathcal{C}$ using the algorithm described in section 2.4.1. Then, given a discrete vorticity field ω, we can calculate $\mathrm{d}\omega/\mathrm{d}t$ with the following steps:

1. **Poisson Equation.** Solve the reconstruction problem $\nabla^2\psi = -\omega$ using the immersed interface method described in Chapter 4. This explicitly enforces the no-through-flow boundary condition. For an unbounded domain, we use Kelvin's theorem to set the circulation constraint.

2. **Velocity Reconstruction.** Compute $\mathbf{u} = \nabla \wedge \psi$ at each grid point, using the immersed interface method described in Chapter 4. This explicitly enforces the incompressibility constraint.

3. **Boundary Condition.** Calculate the boundary vorticity $\omega(\mathbf{x}_c) = \nabla \wedge \mathbf{u}(x_c)$ for each control point $x_c \in \mathcal{C}$, using the local boundary condition described in Chapter 4. This dynamically enforces the no-slip boundary condition.

4. **Force Calculation.** With the velocity and vorticity available at all grid points and control points, we can calculate any of the integrals necessary to measure lift and drag or to calculate local traction forces. These integral formulations and their discretization are discussed in Chapter 5.

5. **Vorticity Transport.** Calculate the time derivative of ω, using the vorticity transport equation

$$\begin{aligned} \frac{\partial \omega}{\partial t} &= -\mathbf{u} \cdot \nabla\omega + \nu\nabla^2\omega \\ \omega &= \omega_b \quad \text{on} \quad \partial\Omega. \end{aligned}$$

   Here we use the discretization of the advection-diffusion equation presented in Chapter 3, and the boundary vorticity distribution calculated in step three.

Having developed a method to compute the field $\mathrm{d}\omega/\mathrm{d}t$ from a given ω, we provide our solver with an initial condition ω_0, and integrate the resulting initial value problem using an explicit n-th order Runge-Kutta scheme with time step τ. All of the immersed interface operations are performed with at least second-order spatial accuracy, so we expect that the error in our numerical solution scales as $\mathcal{O}\big(h^2\big) + \mathcal{O}(\tau^n)$. The computational cost of each step is asymptotically dominated by the reconstruction problem, which is nonlocal and requires $\mathcal{O}(N \log N)$ operations for N grid points. All other procedures are local and require $\mathcal{O}(N)$ or $\mathcal{O}(N_c)$ operations, where N_c is the total number of control points.

The main stability criteria for this method are the Courant number and Fourier number constraints associated with our explicit treatment of the transport problem. For high Reynolds number flows, the Fourier number constraint is usually negligible, and the time-step is entirely determined by the CFL constraint

$$\tau < \frac{hC_{\max}}{u_{\max}},$$

where $C_{\max}$ is a fixed, pre-determined critical Courant number and $u_{\max} = \max(|u_x| + |u_y|)$ is the maximum of the sum of the velocity components over the computational domain. For unsteady flows $u_{\max}$ changes with time, and we must recompute the value of τ at each time step.

6.2 Problem Setup: Impulsively Started Cylinder

A classic test case in two-dimensional incompressible flow is the impulsively started cylinder. Consider a cylinder of diameter D and center $\mathbf{x}_c = (x_c,\ y_c)$ immersed in a quiescent, unbounded fluid with kinematic viscosity ν. At time $t = 0$, the cylinder begins translating with constant velocity, which produces a free-stream velocity of $\mathbf{u}_\infty = (u_{\infty,x},\ u_{\infty,y})$ in a reference frame attached to the cylinder (Figure 6-1). We non-dimensionalize the problem by defining a non-dimensional position $\mathbf{x}^* = \mathbf{x}/D$, time $t^* = Ut/D$, and velocity $\mathbf{u}^* = \mathbf{u}/u_\infty$, leading to a non-dimensional vorticity

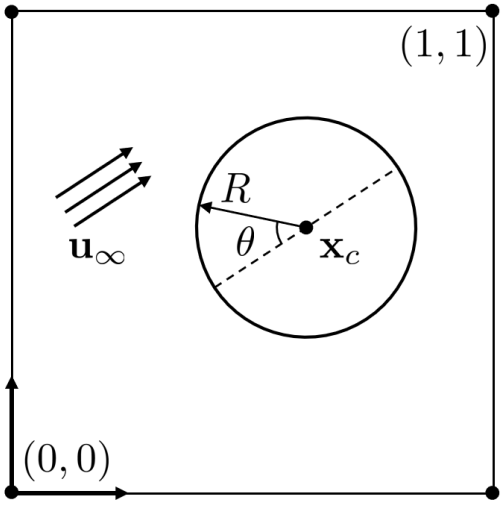


Figure 6-1: Parameters and computational domain for the impulsively started cylinder problem.

$\omega^* = \omega D/u_\infty$. Making a change of variables in the vorticity transport equation, we find that the dynamics of the problem depend only on the Reynolds number $\mathrm{Re} = u_\infty D/\nu$.

For the Reynolds numbers considered here, all of the solutions to this flow problem follow the same qualitative sequence. A thin boundary layer forms immediately after the impulsive start, and then separates from the cylinder, leaving a symmetrical re-circulation region downstream. As t^* increases, the symmetry of the flow is broken by small numerical perturbations, leading to the commonly observed vortex-shedding behavior associated with this problem. In this thesis we focus on the formation of the boundary layer and the initial symmetrical flow.

6.2.1 Handling Impulsive Starts

Impulsive starts lead to extremely large vorticity and vorticity flux values on the object boundary. To control the nearly-singular solution that develops at small but finite t, we begin the simulation with a ramped time stepping procedure, in which the time step begins several orders of magnitude below its maximal stable value and slowly increases to normal levels. Explicitly, given a number of "safe" steps n_s, the time step for step $n < n_s$ is

$$\tau = \frac{hC_{\max}}{u_{\max}} \times 10^{-5(1-n/n_s)}. \tag{6.1}$$

To further increase stability, we split the advection and diffusion steps of the transport equation during the first n_s steps, and break the diffusion step into n_{split} sub-steps. This is done improve the smoothness of the solution at early times, at the cost of reducing the method's temporal accuracy to $\mathcal{O}(\tau)$ near the initial singularity. For all of the impulsive starts shown in this thesis, $n_s \sim \mathcal{O}(100)$ and $n_{\text{split}} = 5$.

6.2.2 Re = 550: Temporal Convergence

Because this problem has been discretized using the method of lines, any explicit n-th order Runge-Kutta scheme should achieve $\mathcal{O}(\tau^n)$ temporal accuracy without any complications. To confirm this behavior, we select a fixed spatial resolution, and measure the variation of the numerical solution with respect to the time step τ. To do so, the spatially discretized system is integrated using a very small time step τ_{ref}, to approximate the exact trajectory of the spatially discrete system. As τ is increased, the resulting numerical solutions should deviate from this reference solution by $\mathcal{O}(\tau^n)$,

where $n = 2$ or $n = 3$ for the Runge-Kutta schemes considered here.

For a symmetrical flow, we expect no lift force and no aerodynamic moment acting on the cylinder, so that the drag force F_D is the only relevant load. Since we are working in a regime for which the vorticity field remains within the computational domain, we can measure the total vortical impulse of the flow

$$\mathbf{I}(t) = \int_V \mathbf{x} \wedge \omega(t)\, \mathrm{d}V\,,$$

and extract the drag force from its time derivative

$$\mathbf{F}_D = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{I}(t).$$

To avoid issues associated with the impulsive start, we begin our convergence test at a finite time t_0^*. To do so, we integrate the solution from time 0 to time t_0^* using the ramped time-stepping given in (6.1), and use this result as an initial condition for another integration beginning at t_0^*.

For this convergence test and the tests that follow, we choose a Reynolds number of $\mathrm{Re}_D = 550$, for which there is an abundance of data available from previous numerical publications. The physical parameters used here are

$$\begin{array}{lll} x_c = 0.291, & u_{\infty,x} = 0.9000, & D = 0.4, \\ y_c = 0.457, & u_{\infty,y} = 0.4359, & \nu = 7.273 \times 10^{-4}, \end{array}$$

and a grid spacing of $h = 1/256$. We choose to integrate from 0 to $t_0^* = 0.1$ with Courant number $C_{\max} = 0.4$ and initial ramping parameter $n_s = 150$. From $t_0^* = 0.1$ to final time $t_f^* = 0.5$, we choose time steps covering the range $C_{\max} \sim [0.03,\ 0.25]$. To quantify the error in the solution, we use the non-dimensional error norms

$$\begin{aligned} \epsilon_2 &= \frac{1}{D^2 U_\infty} \sqrt{\frac{1}{t_f - t_0} \sum_k \left[I(t_k) - I_{ref}(t_k)\right]^2 \tau} \\ \epsilon_\infty &= \frac{1}{D^2 U_\infty} \max_k |I(t_k) - I_{ref}(t_k)|, \end{aligned} \tag{6.2}$$

where $I(t_k)$ and is the magnitude of the vortical impulse at time step t_k. As shown in Figure 6-2, the discretization achieves the expected order of temporal accuracy for both RK2 and RK3 integration.

6.2.3 Re = 550: Spatial Convergence of Global Loads

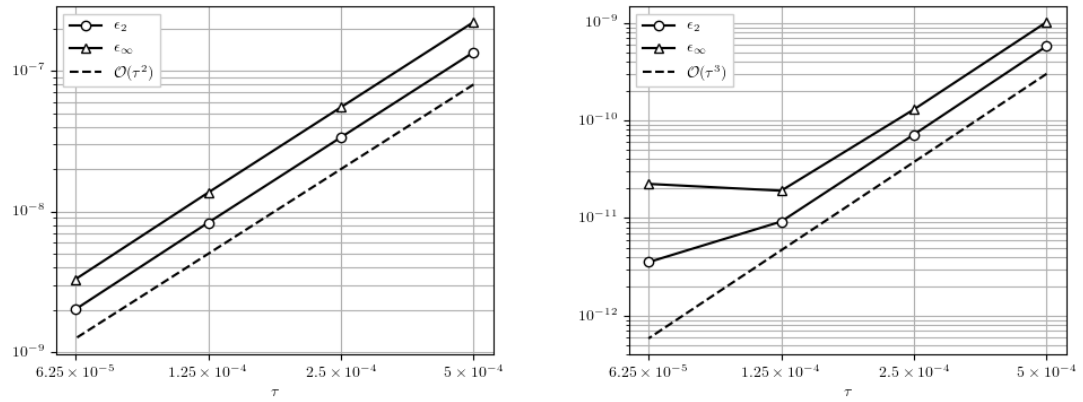
The leading truncation error in the Navier-Stokes discretization developed here is $\mathcal{O}(\tau^n) + \mathcal{O}\big(h^2\big)$, where $n = 2$ or $n = 3$ depending on the chosen Runge-Kutta integrator. For the impulsively started cylinder at the resolutions shown here, the dominant stability criteria is a CFL condition, which dictates that $\tau \sim \mathcal{O}(h)$ as h is decreased to maintain stability. Thus, if we fix a constant CFL and decrease the grid spacing h, we expect the solver to converge to an exact solution at $\mathcal{O}\big(h^2\big) + \mathcal{O}\big(C^3 h^3\big) \to \mathcal{O}\big(h^2\big)$.

Following Gillis [10], we measure the quality of our spatial discretization by an estimate of the number of grid points contained within the characteristic boundary layer thickness,

$$N_\delta = \frac{1}{\sqrt{\mathrm{Re}_D}} \left(\frac{D}{h}\right).$$

For low N_δ, the boundary layer is poorly resolved, which can trigger instability and adversely affect the accuracy of force computations. Based on the results in [10], $N_\delta \sim 8$ represents a well-resolved flow-field, and is usually sufficient for obtaining accurate global loading data.

To quantify the error in numerical solutions to the impulsive cylinder problem, we simulate the problem at the finest possible spatial resolution, and use this solution as a reference for computing the numerical error at coarser resolutions. As in the previous section, we simulate the impulsively started cylinder at $\mathrm{Re} = 550$, with parameters



(a) RK2 Integration

(b) RK3 Integration

Figure 6-2: Temporal convergence for the impulsively started cylinder (reference solution uses $\tau_{\text{ref}} = 1.25 \times 10^{-5}$). The L_2 and L_∞ error norms in the drag coefficient converge with $\mathcal{O}(\tau^2)$ for RK2 integration. For RK3 integration, the convergence is $\mathcal{O}(\tau^3)$ up until the temporal error and round-off error become comparable.

$$\begin{aligned} x_c &= 0.201, \quad u_{\infty,x} = 0.70711, \qquad D = 0.3, \\ y_c &= 0.203, \quad u_{\infty,y} = 0.70711, \quad \nu = 5.455 \times 10^{-4}, \end{aligned}$$

and the sequence of resolutions $h = 1/256$, $1/512$, $1/1024$, and $1/2048$, giving quality criteria of $N_\delta = 3.27$, 6.55, 13.1, and 26.2 respectively. The quantity of interest for these simulations is the drag coefficient

$$C_D(t^*) = \frac{\mathbf{F}(t^*) \cdot \hat{\mathbf{u}}_\infty}{\frac{1}{2} u_\infty^2}, \tag{6.3}$$

where $\hat{\mathbf{u}}_\infty$ is a unit vector in the free-stream direction. Using asymptotic methods, Bar-Lev and Yang [2] provide the analytic estimate

$$C_D = 4\sqrt{\frac{\pi}{t^* \text{Re}_D}} + 2\pi\left(9 - \frac{15}{\sqrt{\pi}}\right)\frac{1}{\text{Re}_D}$$

which is valid for small t^*. Numerically, the drag coefficient is computed using the Noca's "flux 4" control volume method described in Chapter 5, and the resulting drag forces are plotted in Figure 6-3a. The well-resolved ($N_\delta > 8$) results agree well with the short-time analytical solution, as well as with reference data provided by Gillis (G) [10], Marichal (M) [15], and Koumoutsakos and Leonard (K and L) [11]. Using the $h = 2048$ data as a reference solution, we also plot the error $C_D(t^*) - C_{D,\text{ref}}(t^*)$ in Figure 6-3b. These error plots show that the drag curve converges with increasing resolution, and that the time over which the impulsive start affects the accuracy of the solution decreases steadily as the grid is refined.

To quantify the spatial convergence rate of the error, we define the error norms

$$\begin{aligned} \epsilon_2 &= \sqrt{\frac{1}{t_f^* - t_0^*} \sum_k (C_D(t_k^*) - C_{D,\text{ref}}(t_k^*))^2 \tau}, \\ \epsilon_\infty &= \max_k |C_D(t_k^*) - C_{D,\text{ref}}(t_k^*)|, \end{aligned} \tag{6.4}$$

and choose the comparison interval $t^* \in [0.5,\ 2.5]$. Figure 6-3c plots the variation of the L_2 and L_∞ error norms with h, indicating second order convergence for $h = 256,\ 512$ and better than second order convergence for $h = 512,\ 1024$.

Because resolution of our reference solution is only twice the next resolution in the sequence,

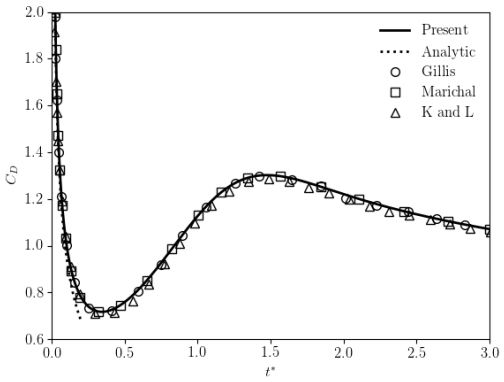
the convergence rates estimated in Figure 6-3c may be artificially high. To get another estimate of the spatial convergence rate, we can compare any three consecutive resolutions h_0, $h_1 = ah_0$, and $h_2 = a^2 h_0$, where $a < 1$ is a known refinement factor. Assuming that we have some numerical approximation of the form $f(h) = A + \epsilon h^n$,

$$\frac{f(h_0) - f(h_1)}{f(h_1) - f(h_2)} = \frac{h^n - a^n h^n}{a^n h^n - a^{2n} h^n} = a^{-n},$$

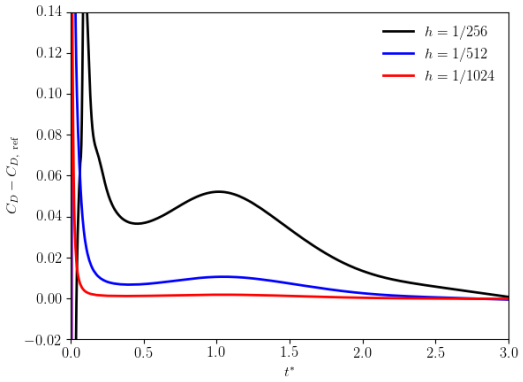
giving the estimate

$$n = -\log_a \left(\frac{f(h_0) - f(h_1)}{f(h_1) - f(h_2)} \right). \tag{6.5}$$

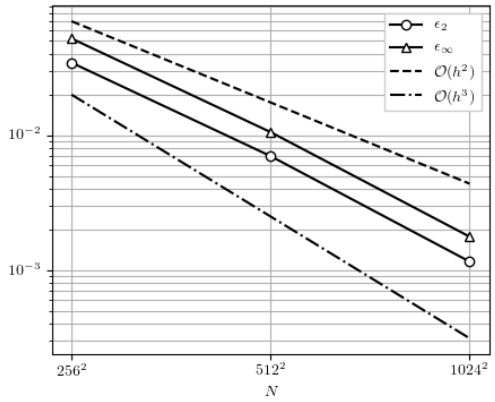
Replacing f with $C_D(t^*)$ in (6.5) provides a method for calculating the rate of convergence of C_d at a specific t^*. Figure 6-3d show these instantaneous convergence rates for all $t^* \in [0,\ 1.5]$. Away from the singularity associated with an impulsive start, the instantaneous convergence rate varies between second- and third-order spatial accuracy, which is consistent with the convergence results shown in Figure 6-3c.



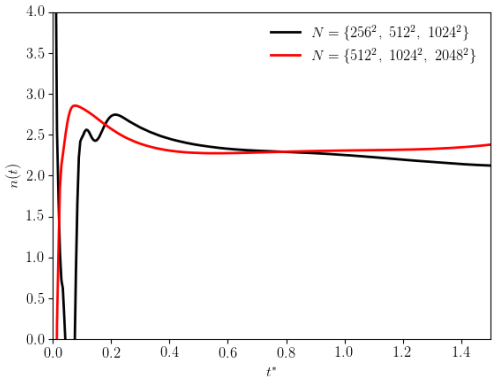
(a) Drag Forces.



(b) Error in Drag Forces.



(c) L_2 and L_∞ convergence.



(d) Instantaneous convergence.

Figure 6-3: Caption me.

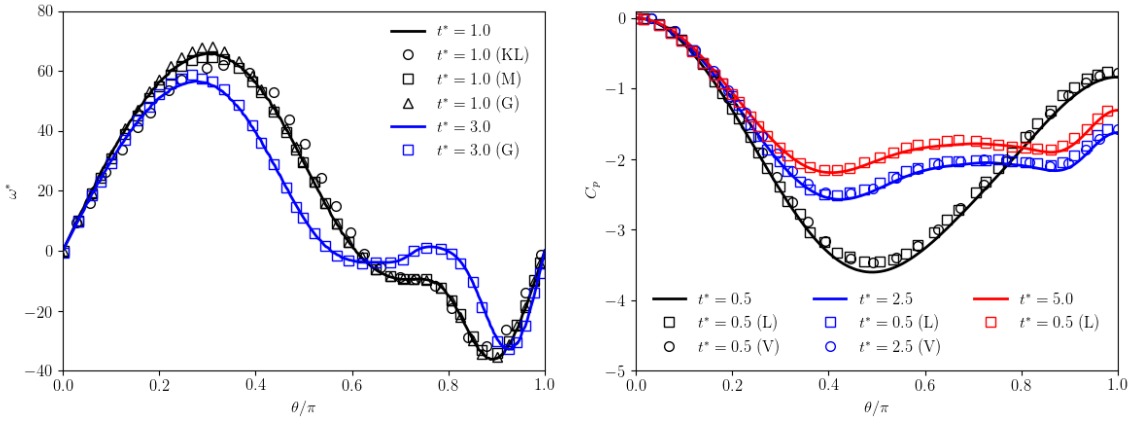
6.2.4 Re = 550: Local Loads

To test the computation of local traction forces, we examine a cylinder at Re = 550 with parameters

$$x_c = 0.201, \quad u_{\infty,x} = 0.7071, \quad D = 0.2,$$
$$y_c = 0.203, \quad u_{\infty,y} = 0.7071, \quad \nu = 3.636 \times 10^{-4},$$

on a grid with resolution $h = 1/1024$, giving a quality criterion of $N_\delta = 8.73$. The boundary of the cylinder is parametrized by an angle θ, so that $\theta = 0$ lies on the front stagnation point and $\theta = \pi$ lies on the rear stagnation point.

The local vorticity field is shown in Figure 6-4; the resulting shear stress distribution is directly proportional to this distribution. The results agree well with data from IIM solvers developed by Gillis (G) [10] and Marichal (M) [15] for $t^* = 1$ and $t^* = 3$, as well reference data from Koumoutsakos and Leonard (KL) [11] for $t^* = 1.0$. The relative pressure coefficient $C_p(\theta) = 2[p(\theta) - p(0)]/u_\infty^2$ is calculated by integrating the vorticity flux as described in Chapter 5, and the constant of integration is fixed so that $C_p(0) = 0$. The resulting pressure distribution is in good agreement with the results of Lee, Lee, and Suh (L) [13] and Verma et al.(V) [24], both of whom use a vorticity-based penalization method.



(a) Vorticity Distribution

(b) Pressure Distribution

Figure 6-4: Local loading for the impulsively started cylinder at Re = 550.

6.2.5 Higher Reynolds Number (Re = 3000)

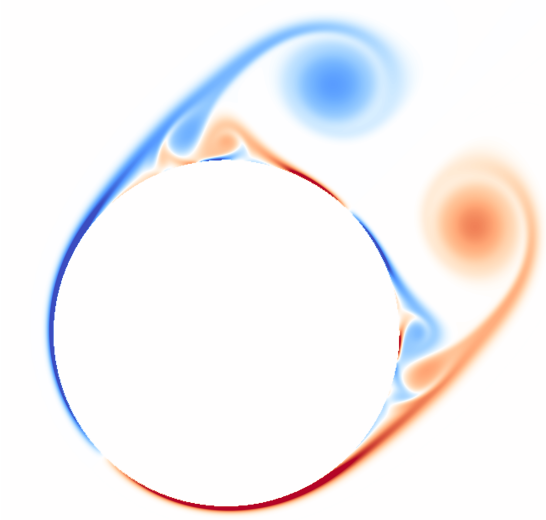
To demonstrate stability at moderate Re, we examine the impulsively started cylinder at Re = 3000 with parameters

$$x_c = 0.301, \quad u_{\infty,x} = 0.7071, \quad D = 0.4,$$
$$y_c = 0.303, \quad u_{\infty,y} = 0.7071, \quad \nu = 1.333 \times 10^{-4},\text{k}$$

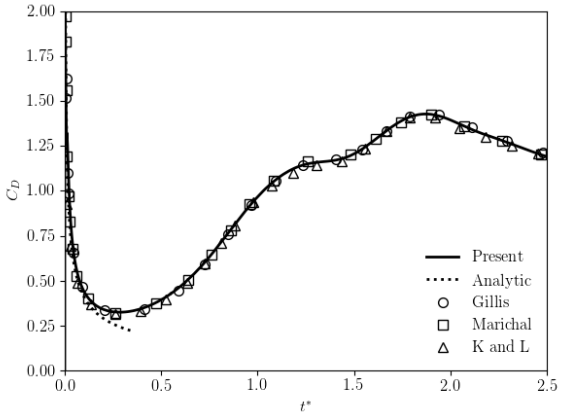
on a grid with resolution $h = 1/1024$, giving a quality criterion of $N_\delta = 7.48$. Figure 6-5 shows that the resulting drag forces for $t^* \in$ and local tractions for $t^* = 1.0$ are in good agreement with reference data provided by body-fitted grid-based methods from Anderson and Reider (AR) [1], Wu et al. (W) [27], and Qian and Vezza (Q) [17], as well as with vortex methods by Koumoutsakos and Leonard (K and L) and Gillis (G) [10].

6.3 Impulsively Rotated Cylinder

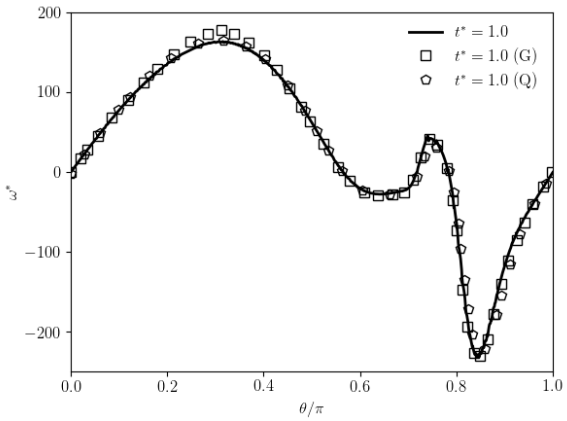
The short-time behavior of an impulsively started cylinder is an interesting test case for drag computation, but does not produce any non-trivial moment acting on the cylinder. To test our control-volume formulation for global moments, we examine a cylinder of radius R which begins rotating



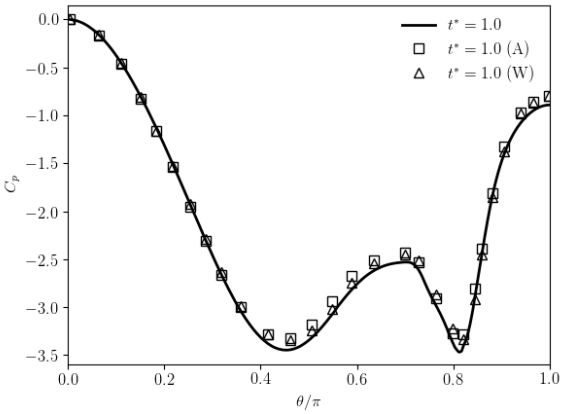
(a) Vorticity Field ($t^* = 2.05$)



(b) Drag Coefficient



(c) Vorticity Distribution ($t^* = 1.0$)



(d) Pressure Distribution ($t^* = 1.0$)

Figure 6-5: Results for the impulsively started cylinder at Re = 3000.

with angular velocity Ω at time $t = 0$. If we assume that the resulting flow is axisymmetric, then we can derive an analytical expression for wall vorticity (Appendix C),

$$\omega_w^*(t) = \frac{\omega_w(t)}{\Omega} = -\frac{2}{\pi}\int_0^\infty \Re\left\{\frac{K_0(ix)}{K_1(ix)}\right\} e^{-x^2 t}\,\mathrm{d}x\,. \tag{6.6}$$

where K_0 and K_1 represent modified Bessel functions of the second kind. Although this value cannot be written in closed form, the integrand in (6.6) decays exponentially, so that the improper integral can be evaluated numerically with good accuracy. The total moment acting on the cylinder can be calculated by integrating the resulting shear stress distribution, giving

$$M^* = \frac{M}{2\pi R^2 \nu \Omega} = \omega_w^* - 2. \tag{6.7}$$

We can define a Reynolds number

$$\mathrm{Re}_\Omega = \frac{R^2\Omega}{\nu}$$

based on the velocity of the cylinder's surface. Although the analytical solution is independent of Re_Ω, it's possible that small numerical perturbations could trigger an instability at high Re_Ω, leading to a chaotic, non-axisymmetric solution. To avoid this possibility, we simulate the impulsively rotating cylinder at $\mathrm{Re}_\Omega = 1$. Choosing parameters

$$\begin{aligned} x_c &= 0.501, \quad a = 0.2, \quad \nu = 0.04, \\ y_c &= 0.503, \quad \Omega = 1.0, \quad h = 1/256, \end{aligned}$$

and a Fourier number constraint $r = 0.2$, we integrate from $t^* = 0$ to $t* = 0.07$ using RK3 integration and the ramped time stepping scheme (6.1). The result moment $M^*(t^*)$ for is shown in Figure 6-6, and is hardly distinguishable from the analytical solution for $t^* > 0.003$.

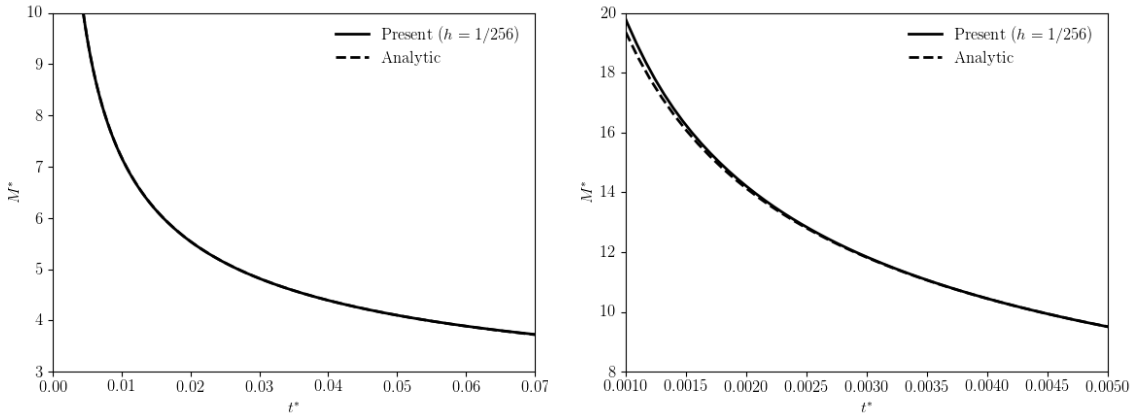


Figure 6-6: Numerical and analytic solutions for the moment acting on an impulsively rotated cylinder. In (a), the exact and numerical solutions are visibly indistinguishable. (b) focuses on the early-time behavior of the simulation, which shows excellent agreement with the analytical solution as early as $t^* = 0.003$.

Chapter 7

IIM with Moving Boundaries

The immersed interface method provides a convenient way to discretize PDEs on a stationary irregular domain without investing computational resources in mesh generation. However, for a stationary domain, the cost of mesh-generation may be negligible compared to the cost of time-marching. The real value of the IIM is its ability to handle problems with moving boundaries, which would otherwise require a significant investment in re-meshing the moving geometry. The use of an IIM for flow problems with moving boundaries is not original; Brehm and Fasel have demonstrated one such method in [6]. The novelty of our approach is that it is not tied any specific time integration scheme, or even a specific PDE. In fact, we will demonstrate that the techniques in this chapter are compatible with arbitrarily high-order explicit Runge-Kutta methods, and do not rely on any physical phenomena unique to flow problems.

In this chapter we apply the IIM spatial discretization tools developed in Chapters 2 and 3 to transport problems involving moving boundaries. The presentation is as follows: we begin first-order integration, focusing on the spatial discretization, geometry processing, and stability considerations. We then move to techniques that allow for higher order time integration, and present an error analysis that reveals a non-trivial coupling between spatial and temporal error. We close with numerical results that experimentally verify this error analysis, and demonstrate the utility of these schemes for transport problems. While the examples in this chapter involve moving rigid bodies, the methods described here apply equally well to arbitrarily deforming bodies.

7.1 First-Order Time Integration

Consider a single time step in the two-dimensional IIM diffusion solver described in Chapter 2, now taking place on a moving domain $\Omega(t)$ and integrated with explicit Euler time stepping. For clarity, let $\mathcal{W}(t)$ be the set of grid points that lie in the problem domain at time t, so that $x_{i,j} \in \mathcal{W}(t) \subset \mathcal{G}$ if and only if $x_{i,j} \in \Omega(t) \subset \mathbb{R}^2$. We begin at time t_0 with an initial value $f_{i,j}(t_0)$ for every point in $\mathcal{W}(t_0)$, and for each $\mathbf{x}_{k,l}$ not in $\mathcal{W}(t_0)$ we assume that $f_{k,l} = 0$. Using the immersed interface method, we approximate $f'_{i,j}(t_0)$ for each $x_{i,j} \in \mathcal{W}(t_0)$, and then advance each point in time via forward Euler time integration:

$$f_{i,j}(t_0 + \tau) = f_{i,j}(t_0) + \tau f'_{i,j}(t_0) \quad \text{for all} \quad \mathbf{x}_{i,j} \in \Omega(t_0).$$

Thus we arrive at time $t_1 = t_0 + \tau$, and immediately encounter an issue (Figure 7-1): because our domain has moved, $\mathcal{W}(t_0) \neq \mathcal{W}(t_1)$, and we have no nonzero value $f_{i,j}(t_1)$ for any point $x_{i,j}$ in $\mathcal{W}(t_1) \setminus \mathcal{W}(t_0)$. To have any hope of proceeding, we must provide a value for each point $\mathbf{x}_{i,j} \in \Omega(t_1) \setminus \Omega(t_0)$.

To do so, we begin with a key assumption: that any point which enters the domain at time t_1 has a neighbor lying inside of the domain at time t_0. More precisely, we require that

$$\mathcal{W}(t_1) \setminus \mathcal{W}(t_0) \subseteq \mathcal{A}^-(t_0). \tag{7.1}$$

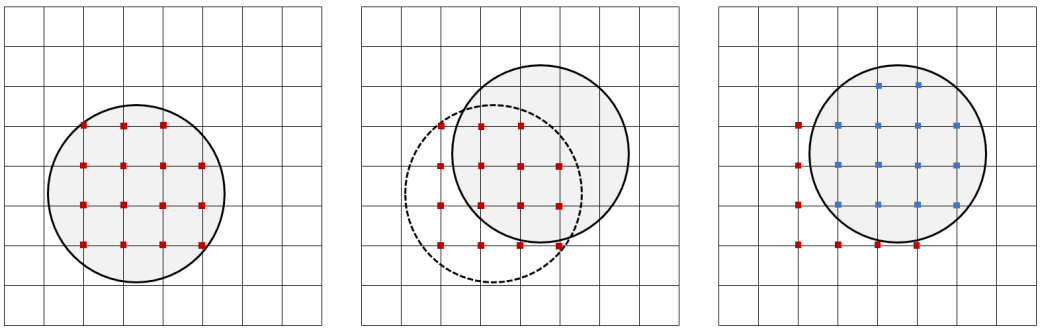


Figure 7-1: A moving obstacle uncovers points that were previously set to zero (shown in red).

This restriction places a CFL-like condition on the maximum time step used in the discretization, which is explored in section 7.1.1. Assuming that (7.1) holds, we can quickly come up with at least three ways of filling these new points using the immersed interface method:

1. One simple solution begins by updating every point in $\mathcal{W}(t_0)$ from t_0 to t_1 using forward Euler, without moving the domain. We then use an IIM field extension without boundary condition (section 2.3.2) to fill $\mathcal{A}^-(t_0)$ with jump corrections appropriate for time t_1. Finally, we update the domain, and discard any values not contained in $\mathcal{W}(t_1)$. Because no boundary condition is used in the extrapolation, there is no splitting error introduced when we delay moving the domain; the IIM machinery from time t_0 is simply a convenient tool used to identify $\mathcal{A}^-(t)$.

2. To improve the accuracy of our extrapolation and the stability of the resulting numerical scheme, we can use an extrapolation that incorporates a given boundary condition. To do so, we begin by updating our field from t_0 to t_1, and marking every point in $\mathcal{A}^-(t_0)$ with a flag value (such as NaN). We then move our domain and calculate the new control points $\mathcal{C}(t_1)$. Looping over every point $\mathbf{x}_c \in \mathcal{C}(t_1)$, we search for points $A^+(\mathbf{x}_c)$ that are marked with a flag value. Once found, these points are given new value using a polynomial interpolation in the local ξ direction that incorporates the boundary condition, as shown in Figure 7-2. Some points will have multiple candidate values; we choose to average these candidates, as we do for standard IIM field extensions.

3. If our boundary condition depends on the solution at time t_1 (like the vorticity BC described in Chapter 4), then a BC for time t_1 is not available until we have filled in the newly uncovered points, and procedure two breaks down. Knowing that we can have a BC at time t_0, we begin our time integration by filling $\mathcal{A}^-(t_0)$ using an IIM field extension constructed from this BC. We then calculate $f'_{i,j}(t_0)$ at every point in $\mathcal{W}(t_0)$, and extend the field $f'_{i,j}$ to $\mathcal{A}^-(t_0)$ using an IIM field extension without boundary condition. Finally, we update every point in $\mathcal{W}(t_0) \cup \mathcal{A}^-(t_0)$ using a forward Euler step, and zero any points that lie outside of $\mathcal{W}(t_1)$.

In this thesis we focus exclusively on procedure three, which can handle the boundary conditions used in our IIM Navier Stokes solver (Chapter 6). The method is entirely characterized by the order of the extrapolations used for the field and its derivative; for convenience, we'll refer to an N-th order field extrapolation with boundary condition and M-th order derivative extrapolation without boundary condition as an $(N,\ M)$ extrapolation procedure. Before tackling higher order time integration, we will take some time explore the consequences of assumption (7.1), and demonstrate that our extrapolation procedure does not degrade the spatial accuracy of the finite difference schemes that it is intended to integrate.

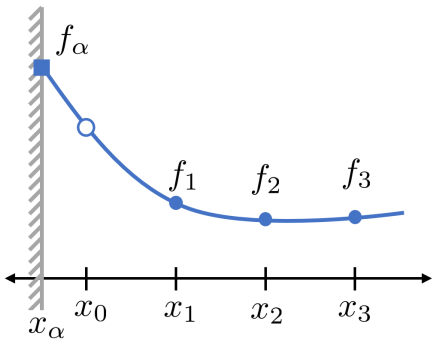


Figure 7-2: One-dimensional stencil proposed in procedure two, used to fill in the values of points that have just entered the computational domain.

7.1.1 Restrictions on Time Stepping

In section 7.1, it was mentioned that assumption (7.1) holds under a type of CFL restriction; here expand on this idea. Assumption (7.1) states that any point which enters the domain will have a neighbor in the domain during the step prior to its entry. Let $\mathbf{x}_{i,j}$ be a grid point lying inside of a *convex* domain Ω. If all of the neighbors of $\mathbf{x}_{i,j}$ are contained within Ω, then by convexity the diamond of side length $\sqrt{2}h$ surrounding $\mathbf{x}_{i,j}$ is fully contained within Ω. This implies that $\mathbf{x}_{i,j}$ can be no closer than $h/\sqrt{2}$ to the boundary of Ω, and if no point on the boundary travels a distance greater than $h/\sqrt{2}$ during a single time step, $\mathbf{x}_{i,j}$ cannot enter the domain during that time step. Thus if we can find a parametrization of the boundary $\mathbf{X}(s,t)$ that obeys

$$\max_s \int_{t_0}^{t_1} \left| \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{X}(s,t) \right| \mathrm{d}t < \frac{h}{\sqrt{2}}, \tag{7.2}$$

then (7.1) will be satisfied. For problems where the domain boundary is composed of well-defined material points, we can simplify this condition considerably. Let $u(s,t)$ be the velocity of a material point s on $\partial\Omega$, and define the velocity bound $u_{\max}(t_0,t_1)$ by

$$u_{\max}(t_0,t_1) = \max_{t\in[t_0,t_1]} \max_{s\in\partial\Omega} |\mathbf{u}(s,t)|. \tag{7.3}$$

Our general condition (7.2) is guaranteed to hold if we enforce the boundary CFL condition

$$\mathrm{CFL}_\Omega = \frac{u_{\max}\tau}{h} < \frac{1}{\sqrt{2}}. \tag{7.4}$$

If the region Ω is not convex, this line of reasoning quickly breaks down. However, if the boundary of Ω is "well-resolved", in a way made precise by the boundary curvature, then a similar guarantee can still be obtained. Let $\gamma(s)$ be an arc-length parametrization of $\partial\Omega$, and let $\hat{\mathbf{n}}$ be the outward facing normal to Ω on the boundary. Define the signed curvature $\kappa(s)$ to be

$$\kappa(s) = -\frac{\mathrm{d}\gamma}{\mathrm{d}s} \cdot \hat{\mathbf{n}}(s),$$

For a convex obstacle, this quantity is always positive; otherwise $\kappa(s)$ can be either positive or negative. For objects represented by a signed distance function, $\kappa = \nabla^2\phi$, and can be calculated inexpensively using the algorithm described in section 2.4.1.

If $|\kappa(s)h| \ll 1$, then the radius of curvature of the boundary is much larger than the grid spacing, and the boundary can be locally approximated by a circular arc. Figure 7-3 illustrates the basic

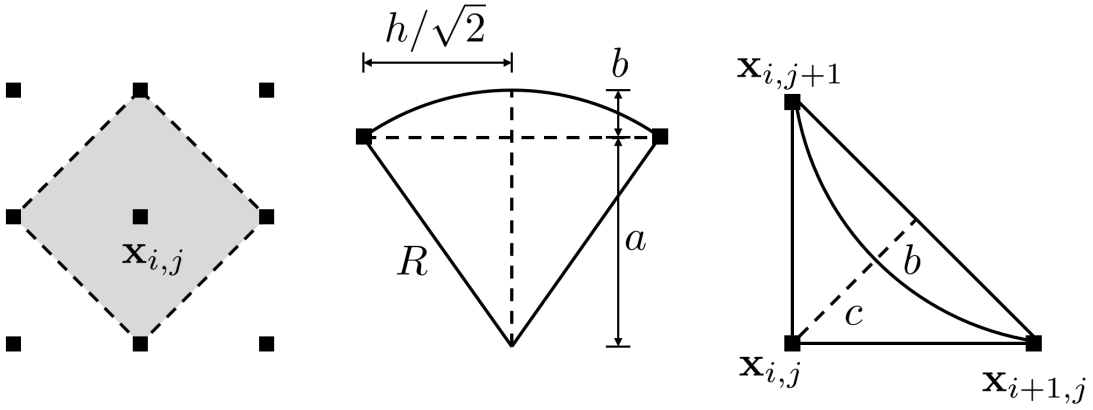


Figure 7-3: (Left) the convexity argument used to construct (7.4). (Center, Right) The geometry used to construct (7.5); the length of b can be calculated separately in both figures, and by equating the two we relate c to R and h.

geometry which determines how close the curve $\gamma(s)$ can approach $\mathbf{x}_{i,j}$. The final result is

$$\text{CFL}_\Omega < \frac{1}{\sqrt{2}} + \frac{1}{z} - \sqrt{\frac{1}{z^2} - \frac{1}{2}}, \quad \text{where} \quad z = h\kappa_{\min}. \tag{7.5}$$

For small z, this expression is more practically represented by the Taylor series

$$\text{CFL}_\Omega < \frac{1}{\sqrt{2}} + \frac{z}{4} + \frac{z^3}{32} + \mathcal{O}(z^5), \tag{7.6}$$

which indicates that for a well-resolved non-convex boundary, the maximum allowable CFL_Ω does not deviate far from $1/\sqrt{2}$.

7.1.2 Numerical Results

The procedures developed above for handling moving boundaries are only useful if they do not affect the stability and convergence properties of the transport schemes developed in Chapter 3. To verify this, we return to the model IBVP for the advection diffusion equation advection-diffusion equation used extensively in Chapter 3. The problem treated here is identical, except that the circular region removed from the domain now translates at at speed $\mathbf{u}_b = (u_{b,x},\ u_{b,y})$. The spatial location of this region can specified by its initial center point $\mathbf{x}_c = (x_{c,0},\ y_{c,0})$, and the error norms defined in Chapter 3 will be reused in this section without alteration. We select a pure diffusion problem, which is stable when integrated with forward Euler integration, and choose geometry parameters

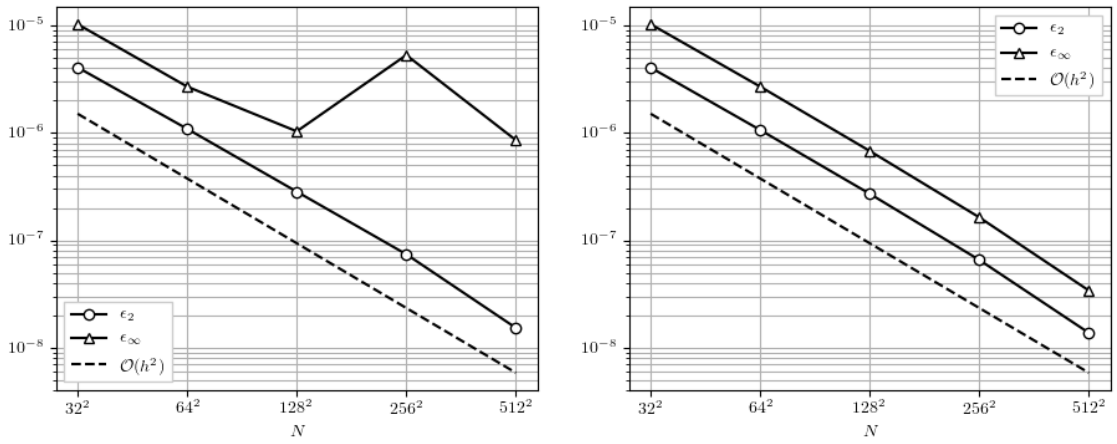
$$\begin{aligned} x_{c,0} &= 0.313, \quad u_{b,x} = 1.5, \quad r = 0.123, \\ y_{c,0} &= 0.337, \quad u_{b,y} = 0.75, \end{aligned}$$

and scalar distribution parameters

$$\begin{aligned} x_g &= 0.4, \quad \sigma = 0.6, \quad \nu = 0.01, \\ y_g &= 0.4, \quad \omega = 1.0. \end{aligned}$$

The resulting immersed interface discretization is integrated numerically to $t = 0.1$ with a small time step ($t = 10^{-5}$). Figures 7-4a and 7-4b show the resulting spatial convergence behavior for a (2, 2) extrapolation and a (3, 2) extrapolation, respectively. The (2, 2) extrapolation does not achieve second order convergence in the L_∞ norm, indicating that a (3, 2) extrapolation is the minimum

order necessary to avoiding disrupting the convergence of the second-order finite difference scheme. This result is somewhat surprising, since fourth order jump corrections are required in a second-order evaluation of the diffusion term. Although it would be interesting to explore this mismatch in accuracy requirements, we do not do so here, and will continue to rely on numerical experiments to determine the necessary order of extensions.



(a) (2, 2) Extension.

(b) (3, 2) Extension.

Figure 7-4: Spatial convergence of the diffusion equation with a moving domain and first-order time integration.

7.2 Higher Order Integration Methods

7.2.1 Explicit Higher Order Runge-Kutta Methods

If a Runge-Kutta type scheme is used, then additional extrapolation is necessary. In this thesis we primarily consider RK2 schemes and a low-storage three-stage RK3 scheme published by Williamson [26], which allow us to integrate the system $\partial_t u = f(u,t)$ while storing only the current u value and a single history field q. The n-th stage of Williamson's three-stage method with time step τ is

$$\begin{cases} q_{n+1} = \tau f(u_n, t_n) + a_n q_n, \\ u_{n+1} = u_n + b_n q_{n+1}, \\ t_{n+1} = t_n + c_n \tau, \end{cases} \tag{7.7}$$

where a_n, b_n, and c_n are taken from

$$a_k = \left\{0, -\frac{5}{9}, -\frac{153}{128}\right\}, \qquad b_k = \left\{\frac{1}{3}, \frac{15}{16}, \frac{8}{15}\right\}, \qquad c_k = \left\{\frac{1}{3}, \frac{5}{12}, \frac{1}{4}\right\}. \tag{7.8}$$

We can write any two-stage RK2 scheme in the same way, using constants

$$a_k = \left\{0,\ 2\alpha - 2\alpha^2 - 1\right\}, \quad b_k = \{\alpha,\ 1/2\alpha\}, \quad c_k = \{\alpha,\ 1-\alpha\}. \tag{7.9}$$

Here α is a free parameter, which does not affect the order of the method. We choose $\alpha = 2/3$, which corresponds to Ralston's method and minimizes truncation error; using $\alpha = 1/2$ gives the midpoint method, while $\alpha = 1$ gives Heun's method. If we allow for additional history fields, then

(7.7) can implement any Runge-Kutta method described by a Butcher tableau of the form

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \dots & \dots & \dots & \dots & & \\ c_n & a_{n1} & \dots & \dots & a_{n,n-1} & \\ \hline & b_1 & \dots & \dots & b_{n-1} & b_n. \end{array}$$

We demonstrate by systematically converting an arbitrary fourth order explicit scheme

$$\begin{aligned} k_1 &= \tau f(u_n,\ t_n) \\ k_2 &= \tau f(u_n + a_{21}k_1,\ t_n + c_2\tau) \\ k_3 &= \tau f(u_n + a_{31}k_1 + a_{32}k_2,\ t_n + c_3\tau) \\ k_4 &= \tau f(u_n + a_{41}k_1 + a_{42}k_2 + a_{43}k_3,\ t_n + c_4\tau) \\ u_{n+1} &= u_n + (b_1k_1 + b_2k_2 + b_3k_3 + b_4k_4), \end{aligned}$$

into a four step procedure

$$\begin{cases} k_1 = \tau f(t_0, y_0) \\ u_1 = u_0 + a_{21}k_1 \\ t_1 = t_0 + c_2\tau \end{cases} \qquad \begin{cases} k_3 = \tau f(t_2, y_2) \\ u_3 = u_1 + (a_{41} - a_{31})k_1 + (a_{42} - a_{32})k_2 + a_{43}k_3 \\ t_3 = t_1 + (c_4 - c_3)\tau \end{cases}$$

$$\begin{cases} k_2 = \tau f(t_1, y_1) \\ u_2 = u_1 + (a_{31} - a_{21})k_1 + a_{32}k_2 \\ t_2 = t_1 + (c_3 - c_2)\tau \end{cases} \qquad \begin{cases} k_4 = \tau f(t_3, y_3) \\ u_4 = u_1 + (b_1 - a_{41})k_1 + (b_2 - a_{42})k_2 + (b_3 - a_{43})k_3 + b_4k_4 \\ t_4 = t_1 + (1 - c_4)\tau. \end{cases}$$

Since k_4 is only used in the last stage, this method require the three history fields k_1, k_2, and k_3.

The purpose of rephrasing our Runge-Kutta schemes in this format is to ensure that they are automatically compatible with the extrapolation procedures we have already developed for forward Euler. When a new point $\mathbf{x}_{i,j}$ enters the domain during stage n of a high-order integration, it must be provided with both a new value $u^n_{i,j}$ and a new set of history fields. Define the boundary CFL constraint for an N step method beginning at t_0 and ending at t_N to be

$$\text{CFL}_\Omega = \frac{u_{\max}(t_0, t_N)\tau}{h} < \frac{1}{\sqrt{2}} + \frac{1}{z} - \sqrt{\frac{1}{z^2} - \frac{1}{2}}. \tag{7.10}$$

If our integrator takes the form of (7.7), then the extension procedure developed for forward Euler integration will automatically provide a complete history for every point that enters the domain.

To see that this is true, note that any point $\mathbf{x}_{i,j}$ will have a complete history as long it receives an evaluation $f_{i,j}(u,t)$ at every stage of the integration. During each stage, every point with a neighbor in the domain will receive an evaluation. For a point to enter the domain with an incomplete history, it must have no neighbors in the domain during one stage, and then enter the domain during another. Section 7.1.1 demonstrates that this cannot happen if the total time step τ is constrained by (7.10).

If we are willing to extend every history parameter at every time step, in addition to $f_{i,j}(u,t)$ and the field itself, then the CFL_Ω restriction can be loosened slightly. If the n-th stage of a high-order integrator advances a system from time t_n to time t_{n+1}, then the condition

$$\frac{u_{\max}(t_n,\ t_{n+1})(t_{n+1} - t_n)}{h} < \frac{1}{\sqrt{2}} + \frac{1}{z} - \sqrt{\frac{1}{z^2} - \frac{1}{2}}$$

will ensure that only points in $\mathcal{A}^-(t_n)$ can enter the domain at time t_{n+1}. However, since the CFL_Ω constraint (7.10) is not significantly stricter than the stability restrictions imposed by the transport

scheme developed in Chapter 3, we choose to extend only the derivative $f_{i,j}(u,t)$ and endure the stricter stability requirement.

7.3 Coupling of Spatial and Temporal Error

In the method-of-lines IIM discretizations we have used so far, each grid point in the problem domain becomes one unknown in a semi-discrete problem, represented by a large system of ODEs. Grid points which lie outside of the problem domain are essentially inactive, and aside from offering a location for storing jump corrections, they do not participate in the discretization.

For problems with moving domains, the picture is more complicated. If the motion is slight, so that no grid points cross the domain boundary, then the set of intersections $\mathcal{C}$ is constant, and we can account for the motion through the time-dependent intersection distances $\psi_c(t)$ and the resulting stencils $s_{\xi,i}(\psi_c(t))$. If the motion is large enough to move grid points across the boundary, then the problem changes fundamentally: every crossing alters the dimension of the system, so that the semi-discrete problem can no longer be represented by a single continuous system of ODEs. In this section we reconcile this issue by constructing a more complex semi-discrete problem, and then show that the application of standard numerical integrators to this semi-discrete system introduces a mixed error term of order τh^N for moving problems, where N is the order of the spatial discretization.

Let $\mathcal{L}$ be a spatial differential operator, and consider an IBVP $\partial_t u = \mathcal{L}u$ on a moving domain $\Omega(t)$ over the time interval $[0,T]$. Assuming that the motion of the domain and the regular Cartesian grid are known in advance, we can compute the exact time that any grid point crosses the domain boundary. If we order these times, we obtain a set of disjoint intervals $[t_i, t_{i+1}]$ during which no crossings occur, with $[0,T] = \cup_i [t_i, t_{i+1}]$. Using the immersed interface method, we can construct a different system of ODEs $\partial_t u_i = f_i(u_i,t)$ for each uninterrupted interval, where $f_i(u_i,t)$ is a continuous function of time. We define an initial value for each system via spatial interpolation operators $\mathcal{I}_i$, so that $u_i(t_i) = \mathcal{I}_i[u_{i-1}(t_i)]$. These operators represent the IIM extrapolations used in this chapter: each $\mathcal{I}_i$ removes points from u_i when they leave $\Omega(t)$, preserves points that remain inside $\Omega(t)$, and provides initial values for points as they enter $\Omega(t)$. We'll assume that each $\mathcal{I}_i$ has spatial accuracy greater than or equal to N, so that the error introduced by the $\mathcal{I}_i$ does not dominate the truncation error of the immersed interface method. If we solve the systems $\partial_t u_i = f_i(u_i,t)$ exactly

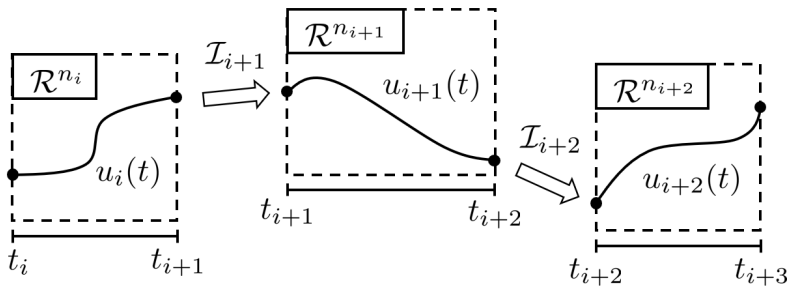


Figure 7-5: A conceptual sketch of the semi-discrete system constructed in section 7.3.

and in sequence, applying the $\mathcal{I}_i$ at each crossing time t_i, then we arrive at a spatially-discrete but time-continuous version of the moving immersed interface discretizations developed in this chapter (Figure 7-5). The error in this approximation is entirely spatial; for any time $t \in [t_i, t_{i+1}]$, the N-th

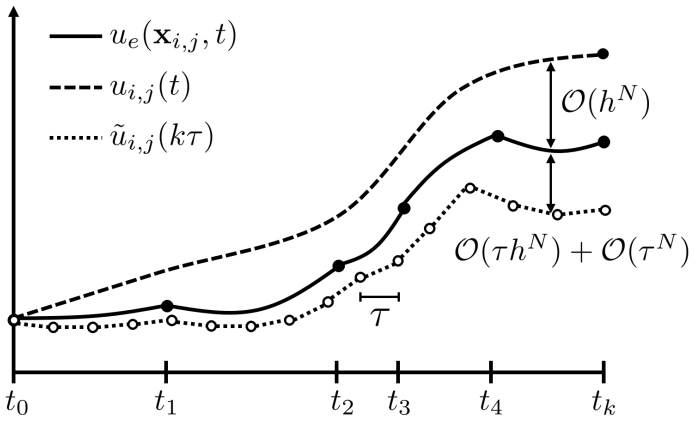


Figure 7-6: An illustration of the exact solution $u_e(\mathbf{x}, t)$ at the point $\mathbf{x}_{i,j}$, the semi-discrete approximation $u_{i,j}(t)$, and the fully discretized solution $\tilde{u}_{i,j}(k\tau)$. The semi-discrete system constructed here is continuous, but may not be differentiable at the crossing times t_i . This spatial discretization introduces an $\mathcal{O}\big(h^N\big)$ error, while the time stepping (which does not align with the crossing times) introduces the expected $\mathcal{O}\big(\tau^M\big)$ error along with an "overshoot" error of order τh^N.

order immersed interface discretization $f_i(u_i, t)$ guarantees that

$$\frac{\mathrm{d}}{\mathrm{d}t} u_i(t, \mathbf{x}_{i,j}) = \mathcal{L} u_e(\mathbf{x}_{i,j}, t) + \mathcal{O}\big(h^N\big), \quad \text{so that}$$
$$u_i(\mathbf{x}_{i,j}, t) = u_e(\mathbf{x}_{i,j}, t) + \mathcal{O}\big(h^N\big).$$

If we integrate each individual system numerically with an M-th order integrator and time step τ_i, still applying the operators $\mathcal{I}_i$ at each crossing time, then we introduce an additional error of order $\mathcal{O}\big(\tau_i^M\big)$, which is propagated from one system to the next via the interpolation operators. If $\tau_{\max}$ is the largest of these time steps, then by the end of the integration we expect that

$$u_i(\mathbf{x}_{i,j}, T) = u_e(\mathbf{x}_{i,j}, T) + \mathcal{O}(h^n) + \mathcal{O}\big(\tau_{\max}^M\big).$$

Keeping the spatial discretization fixed while allowing τ to approach 0, we recover the semi-discrete solution constructed above.

For two or three-dimensional problems, the time between boundary crossings can be prohibitively short. To get around this, we would like to be able to take time steps of arbitrary size, applying the operators $\mathcal{I}_i$ at the end of each time step to fill in any newly uncovered points. If we do so, then we inevitably take a numerical step that lies partly in one interval $[t_{i-1}, t_i]$ and partly in the next interval $[t_i, t_{i+1}]$. If the integration is explicit, then we have approximated the evolution of the system $\partial u_{i-1} = f_{i-1}(u_{i-1}, t)$ for a time τ, when in fact we should have done so for only a fraction of the step before switching to the system $\partial u_i = f_i(u_i, t)$. In general the fraction of a step that we misattributed to f_{i-1} is $\mathcal{O}(\tau)$, and the difference $|f_{i-1}(u, t) - f_i(u, t)|$ cannot exceed $\mathcal{O}\big(h^N\big)$ at any grid point common to both systems, since they are N-th order approximations of the same continuous operator $\mathcal{L}u$. So we estimate that the error due to "overshooting" a single crossing time is of order τh^N. If our overshoot covers more than two intervals, we obtain the same estimate, since each individual system in question obeys the same $\mathcal{O}\big(h^N\big)$ accuracy requirement.

Taking the overshoot error into account leads to two final error estimate for the moving IIM discretizations developed here: $\mathcal{O}\big(h^N\big) + \mathcal{O}\big(\tau^M\big)$ for integration which respects the crossing times, and $\mathcal{O}\big(h^N\big) + \mathcal{O}\big(\tau^M\big) + \mathcal{O}\big(\tau h^N\big)$ for integration which does not respect the crossing times. If τ is a small parameter (as it is in any useful numerical scheme), then the mixed error term will always be dominated by the spatial error term. However, we can still observe the existence of this error term by

fixing a spatial discretization and slowly decreasing the time step τ in an integrator for which $M > 1$. Eventually we will arrive at a point where the mixed error and temporal error are of comparable magnitude, and our integrator appears to switch from M-th order accuracy to first order accuracy. This is no cause for concern: the mixed error of order τh^N cannot be the asymptotically dominant contributor to the total error of a numerical discretization, even though we have manufactured a situation for which it dominates the temporal error.

7.3.1 Numerical Results

If the above error estimate holds, then we have developed a scheme which improves the total temporal error of our discrete system from $\mathcal{O}(\tau)$ for forward Euler to $\mathcal{O}\big(\tau h^N\big) + \mathcal{O}\big(\tau^M\big)$ for an M-th order Runge Kutta method. To confirm these gains, we begin with a simple one-dimensional example. Consider the one-dimensional advection diffusion equation,

$$\frac{\partial f}{\partial t} = -u\frac{\partial f}{\partial x} + \nu\frac{\partial^2 f}{\partial x^2} \tag{7.11}$$

with u, $\nu > 0$, which has free-space solutions of the form

$$g(x,t) = e^{-k^2\nu t}\sin(kx - kct). \tag{7.12}$$

For $k = 2\pi$, these solutions remain valid on the restricted domain $\Omega \in [0,1]$ with the periodic boundary condition $f(0) = f(1)$. To allow for an immersed interface discretization, we remove the region $[x_\ell(t), x_r(t)]$ from Ω, and add the Dirichlet boundary conditions $f(x_\ell(t)) = g(x_\ell(t))$ and $f(x_r(t)) = g(x_r(t))$. For simplicity, let the removed interval have a constant length $x_{r,0} - x_{\ell,0}$ and constant translation speed u_b, so that

$$\begin{cases} x_\ell(t) &= x_{\ell,0} + u_b t, \\ x_r(t) &= x_{r,0} + u_b t. \end{cases}$$

Numerically, this problem is discretized using the immersed-interface transport scheme developed in Chapter 3, so that advection term is calculated without the boundary condition at x_ℓ. To account for the moving boundaries, we use a (3, 2) extension procedure. Choosing parameters

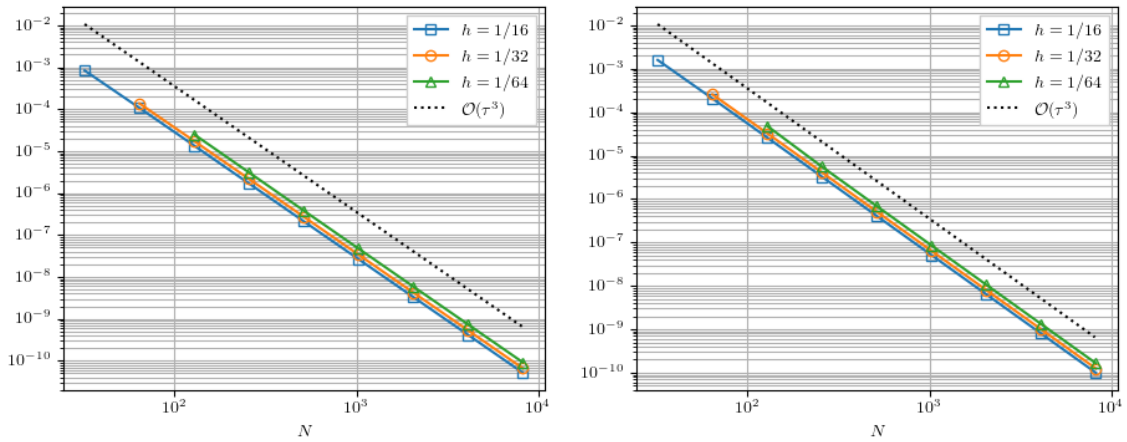
$$\begin{aligned} x_{\ell,0} &= 0.25 + 10^{-10}, \quad & u &= 1.0, \quad & \nu &= 3.125 \times 10^{-3}, \\ x_{r,0} &= 0.50 + 10^{-10}, \quad & u_b &= 0.25, \quad & t &\in [0,1], \end{aligned}$$

along with numerical parameters of the form $h = 2^{-n}$ and $\tau = 2^{-m}$ ensures that all crossing times align with the natural time stepping. The slight offset in $[x_\ell, x_r]$ ensures that the (3, 2) extrapolation is consistently applied just after each crossing. We define the error norms

$$\begin{aligned} \epsilon_2 &= \sqrt{\sum_i h(f_i(T) - f_{i,\mathrm{ref}}(T)^2)}, \\ \epsilon_\infty &= \max_i \big|f_i(T) - f_{i,\mathrm{ref}}(T)^2\big|, \end{aligned}$$

where $f_{i,\mathrm{ref}}(t)$ is a reference solution computed with the same spatial resolution and a much smaller time step. Figure 7-7 plots the L_2 and L_∞ error norms as a function of τ for several spatial resolutions, demonstrating that that the numerical error is $\mathcal{O}\big(\tau^3\big)$ with a prefactor that is largely independent of the spatial discretization.

If the interval used above is replaced by $[x_{\ell,0},\ x_{r,0}] = [0.261,\ 0.447]$, then the crossing times are no longer aligned with the temporal discretization. Figure 7-8 plots the new temporal errors; as predicted, the behavior is initially $\mathcal{O}\big(\tau^3\big)$, and eventually drops to $\mathcal{O}(\tau)$ with an $\mathcal{O}\big(h^3\big)$ prefactor. The third-order spatial convergence indicates that spatial error from the advection term still dominates the second-order diffusion term.



(a) L_2 Error.

(b) L_∞ Error

Figure 7-7: Temporal convergence of a one-dimensional advection-diffusion discretization integrated with RK3, for time integration that aligns with the boundary crossings. The convergence rate is $\mathcal{O}(\tau^3)$, with a prefactor that is independent of the spatial resolution.

To verify that our high-order time stepping does not affect the stability or spatial convergence of our two-dimensional transport discretization, we return to the example presented in section 7.1.2. The geometry and scalar distribution parameters are left unchanged, and the resulting immersed interface discretization is integrated numerically to $t = 0.1$ with a small time step ($\tau = 10^{-5}$) and RK3 time integration. Figures 7-9a and 7-9b show the resulting convergence behavior for a $(2,\ 2)$ extension and a $(3,\ 2)$ extension, respectively. As with forward Euler time integration, the $(2,\ 2)$ case fails to achieve second order convergence, indicating that $(3,\ 2)$ is the minimally accurate extension needed for the transport scheme developed in Chapter 3.

The rest of the numerical examples here will use this $(3, 2)$ extension procedure. The test described above is repeated for an advection diffusion problem with vanishing viscosity, using geometry parameters

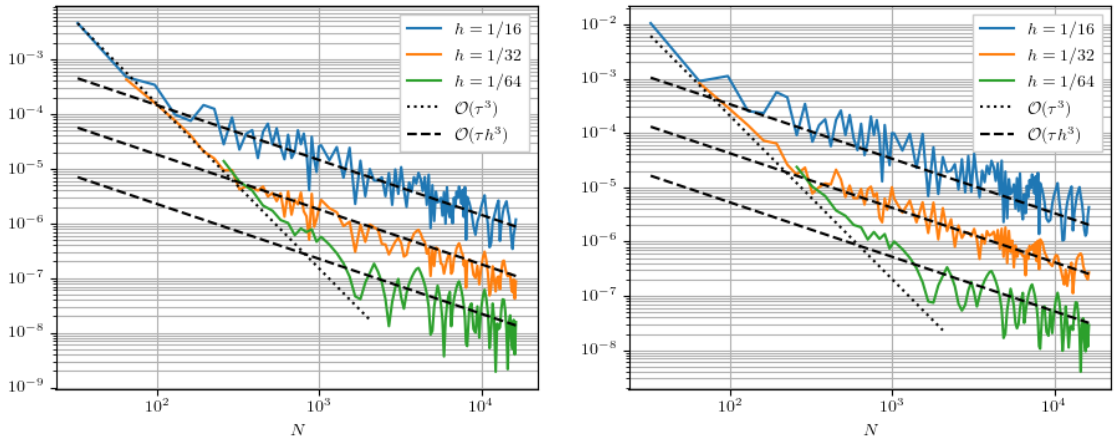
$$\begin{aligned} x_{c,0} &= 0.693, \quad u_{b,x} = -0.39, \quad r = 0.123, \\ y_{c,0} &= 0.697, \quad u_{b,y} = -0.19, \end{aligned}$$

along with scalar distribution parameters

$$\begin{aligned} x_g &= 0.3, \quad \sigma = 0.6, \quad u_x = 0.4, \\ y_g &= 0.3, \quad \omega = 1.0, \quad u_y = 0.2. \end{aligned}$$

The resulting system is integrated to $t = 0.5$ using the small time step $\tau = 1.25 \times 10^{-4}$ and RK3 time integration. Figure 7-10a shows the results, which indicate third order spatial convergence in both the L_2 and L_∞ error norms. We also repeat the same test with a no through-flow boundary, setting $\mathbf{u} = \mathbf{u}_b = [-0.2, -0.4]$; the results are shown in Figure 7-10b. In this case any error introduced at the boundary remains concentrated near the boundary as the simulation progresses, leading to an L_∞ convergence rate that is only second order.

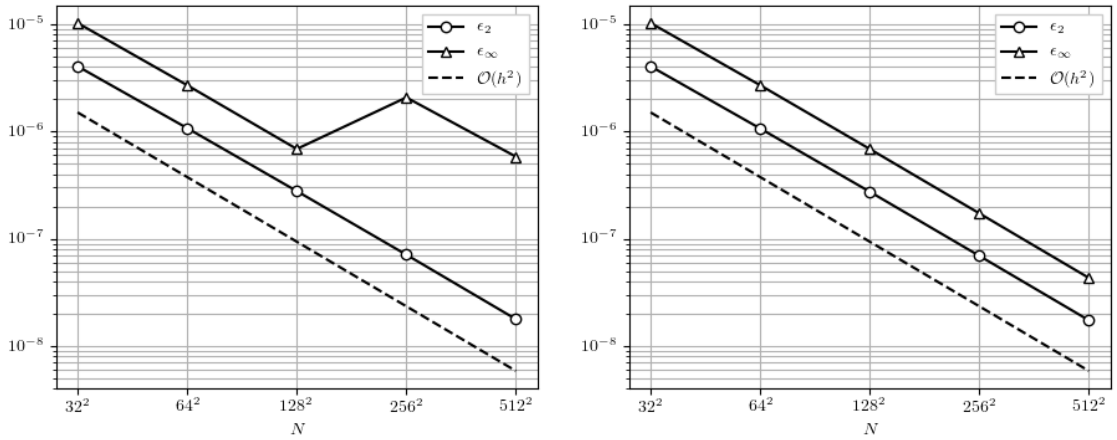
To demonstrate convergence for the full advection diffusion equation, we introduce viscosity $\nu = 0.001$, and reduce the initial width of the Gaussian distribution to $\sigma = 0.3$. The resulting system is integrated to $t = 1.0$ using a time step $t = 2 \times 10^{-4}$, and the results are plotted in Figure 7-11. The spatial error initially converges at $\mathcal{O}(h^3)$, while the advective error dominates, but drops to $\mathcal{O}(h^2)$ at the finest discretizations. This convergence behavior is identical to the stationary solver, indicating that the moving IIM techniques developed here are successful in discretizing transport problems on moving domains. With this knowledge, we are fully prepared to extend the stationary Navier Stokes solver constructed in Chapter 6 to problems with moving domains.



(a) L_2 Error.

(b) L_∞ Error

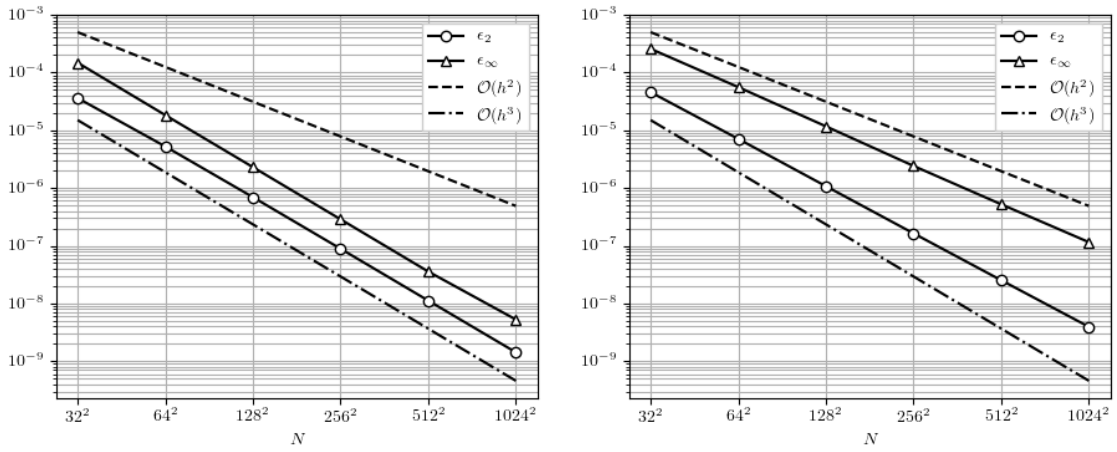
Figure 7-8: Temporal convergence of a one-dimensional advection-diffusion discretization integrated with RK3, for time integration that does not align with the boundary crossings. The initial convergence is $\mathcal{O}(\tau^3)$, which eventually weakens to $\mathcal{O}(\tau)$ with a prefactor that scales with h^3.



(a) (2, 2) Extension.

(b) (3, 2) Extension.

Figure 7-9: Spatial convergence of the diffusion equation with a moving domain and third-order time integration.



(a) With through-flow.

(b) No through-flow.

Figure 7-10: Convergence of the advection-diffusion equation with vanishing viscosity and a moving domain. Convergence of the L_2 error is nearly third-order for both test cases; the L_∞ error is between second and third order, and noticeably larger for the no through-flow case.

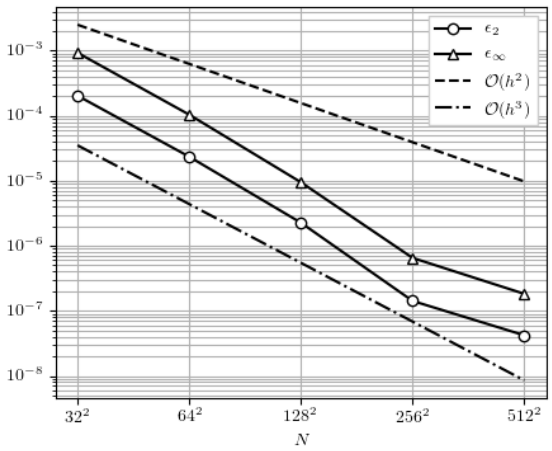


Figure 7-11: Convergence of the advection-diffusion equation with moving domain and $\mathrm{Pe}_D = 110$. The resulting spatial convergence is initially $\mathcal{O}\big(h^3\big)$, dropping to $\mathcal{O}\big(h^2\big)$ for the finest resolutions as the diffusive error term begins to dominate.

Chapter 8

Numerical Results: Navier Stokes with Moving Boundaries

Having developed a methodology for generalizing immersed-interface discretizations to moving domains, we turn our focus back to the Navier Stokes equations. We will demonstrate through several numerical examples that the immersed interface method described here achieves second order spatial accuracy at the moving domain boundary, and allows for second or third order time integration. The problems shown here involve a rigid body with a prescribed trajectory immersed in an unbounded fluid domain. However, the extension to deforming bodies and bodies moving under the action of a prescribed force is relatively simple, and we intend to demonstrate this capability in a future publication.

8.1 A Moving Boundary Navier Stokes Solver

The spatial discretization used in this chapter is nearly identical to the one presented in Chapter 6, now with an added spatial interpolation step to account for moving boundaries. As before, all time integration is done with RK2 (7.9) or a low-storage RK3 method (7.8) due to Williamson [26]. The complete algorithm for solving the Navier Stokes equations on a moving domain is described in below; although only steps one, seven, and eight are new, we have reproduced the others for convenience.

1. **Geometry Processing.** Evaluate the signed distance function $\phi(\mathbf{x}, t)$ for the current boundary configuration, and zero all points that do not lie in the domain. Using the techniques described in section 2.4.1, identify the control points $\mathcal{C}$ and the associated affected points $\mathcal{A}^+$ and $\mathcal{A}^-$. Then pre-compute all the necessary immersed interface stencils (Appendix A).

2. **Poisson Equation.** Solve the reconstruction problem $\nabla^2\psi = -\omega$ using the immersed interface method described in Chapter 4. This explicitly enforces the no-through-flow boundary condition. For an unbounded domain, we use Kelvin's theorem to set the circulation constraint.

3. **Velocity Reconstruction.** Compute $\mathbf{u} = \nabla \wedge \psi$ at each grid point, using the immersed interface method described in Chapter 4. This explicitly enforces the incompressibility constraint.

4. **Boundary Condition** Calculate the boundary vorticity $\omega(\mathbf{x}_c) = \nabla \wedge \mathbf{u}(x_c)$ for each control point $x_c \in \mathcal{C}$, using the local boundary condition described in Chapter 4. This dynamically enforces the no-slip boundary condition.

5. **Force Calculation.** With the velocity and vorticity available at all grid points and control points, we can calculate any of the integrals necessary to measure lift and drag or to calculate local traction forces. These integral formulations and their discretization are discussed in Chapter 5.

6. **Vorticity Transport.** Calculate the time derivative of ω, using the vorticity transport equation

$$\frac{\partial \omega}{\partial t} = -\mathbf{u} \cdot \nabla \omega + \nu \nabla^2 \omega$$
$$\omega = \omega_b \quad \text{on} \quad \partial\Omega.$$

   Here we use the discretization of the advection-diffusion equation presented in Chapter 3, and the existing boundary vorticity distribution.

7. **Field Extensions.** Using the vorticity boundary condition, extend the vorticity field to $\mathcal{A}^-$ at third order. The time derivative of vorticity is extended to $\mathcal{A}^-$ at second order, without boundary condition.

8. **Time Integration.** The vorticity field, history field, object position, and time variable are updated according to the chosen Runge-Kutta scheme.

The most expensive step associated with moving bodies is the evaluation of the signed distance function and creation of control points. Unless a clever scheme is employed to bound a moving obstacle, these processes must consider every point in the domain, and take $\mathcal{O}(N)$ resources. The extensions and re-computation of stencils require $\mathcal{O}(N_b)$ resources. All of the stability constraints mentioned in Chapter 6 apply to the moving case, along with the $\text{CFL}_{\partial\Omega} < 1/\sqrt{2}$ restriction discussed in Chapter 7.

8.2 Impulsively Started Cylinder

To verify the accuracy of the moving-boundary Navier Stokes solver, we return to the impulsively started cylinder problem. In Chapter 6, this problem was presented in a frame of reference that moved with the cylinder, leading to a nonzero free-stream velocity and a fixed problem domain. Here we move to a reference frame with no free-stream velocity, in which the cylinder begins translating with a fixed velocity. The parameters that define this problem are nearly identical to the stationary case, except that we now list the cylinder's velocity instead of a free-stream velocity, and we locate the cylinder in space by listing its initial position $(x_{c,0},\ y_{c,0})$. To form non-dimensional quantities, we must replace the free-stream velocity $\mathbf{u}_\infty$ with the body velocity $\mathbf{u}_b$, giving

$$\text{Re}_D = \frac{D u_b}{\nu}, \qquad t^* = \frac{u_b t}{D}, \quad \text{and} \quad \omega* = \frac{\omega D}{u_b}.$$

We will not redefine all of the error measures and quality criteria, since all of the results listed here have stationary counterparts described in Chapter 6.

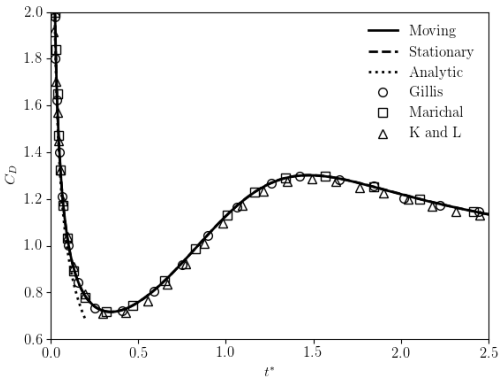
8.2.1 $\text{Re} = 550$: Spatial Convergence

With the given discretization, we expect that the error in any quantity of interest behaves as $\mathcal{O}\big(h^2\big) + \mathcal{O}\big(\tau^N\big) + \mathcal{O}\big(\tau h^2\big)$ when N-th order Runge-Kutta integration. Because the CFL criteria for our discretization requires that $\tau \sim \mathcal{O}(h)$ as h decreases, the $\mathcal{O}\big(h^2\big)$ spatial error term should still dominate as $h \to 0$ if the CFL constraint remains constant. To confirm this, we consider a moving cylinder with parameters

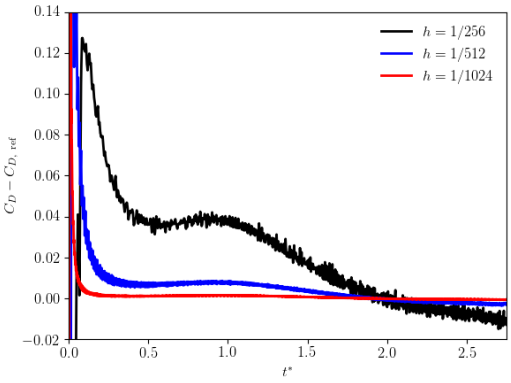
$$\begin{aligned} x_{c,0} &= 0.143, \quad u_{b,x} = 0.7071, \qquad D = 0.2, \\ y_{c,0} &= 0.147, \quad u_{b,y} = 0.7071, \quad \nu = 3.636 \times 10^{-4}, \end{aligned}$$

with the same sequence of resolutions h and quality criteria N_δ used in Chapter 6: $h = 1/256$, $1/512$, $1/1024$, $1/2048$, and $N_\delta = 3.27$, 6.55, 13.1, and 26.2. The resulting drag coefficients, errors, convergence rate, and instantaneous convergence rates are shown in Figure 8-1. The drag data agrees well with reference data from other authors, and is visually indistinguishable from the results

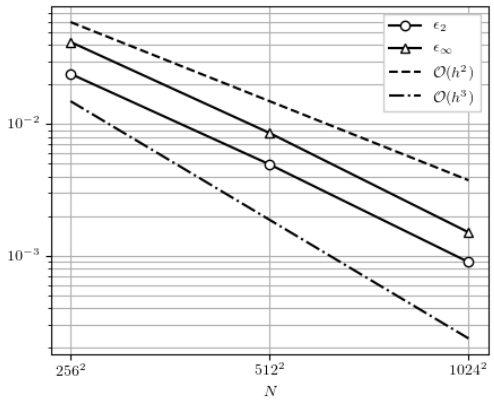
of the stationary simulations shown in Chapter 6. New to the moving cylinder is the introduction of a small amount of noise in the function $C_D(t^*)$, due to the noisy truncation error that comes from integrating over a constantly-changing numerical domain. This noise does not affect the second order convergence of the L_2 or L_∞ error norms, which are shown in Figure 8-1c. The instantaneous convergence rates also remain between second and third order away from the impulsive start, with the exception of a region surrounding $t^* \approx 1.9$. Here the errors shown in Figure 8-1b change sign, leading to extremely small error magnitudes that cause the convergence rate estimate (6.5) to become ill conditioned.



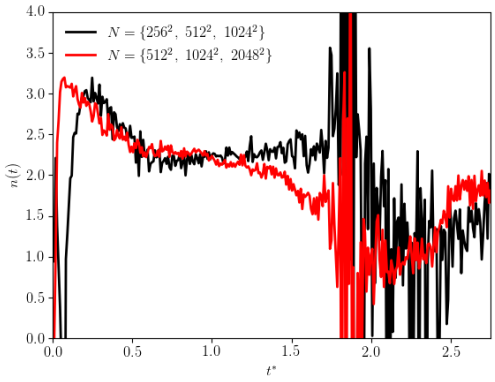
(a) Drag Forces.



(b) Error in Drag Forces.



(c) L_2 and L_∞ convergence.



(d) Instantaneous convergence.

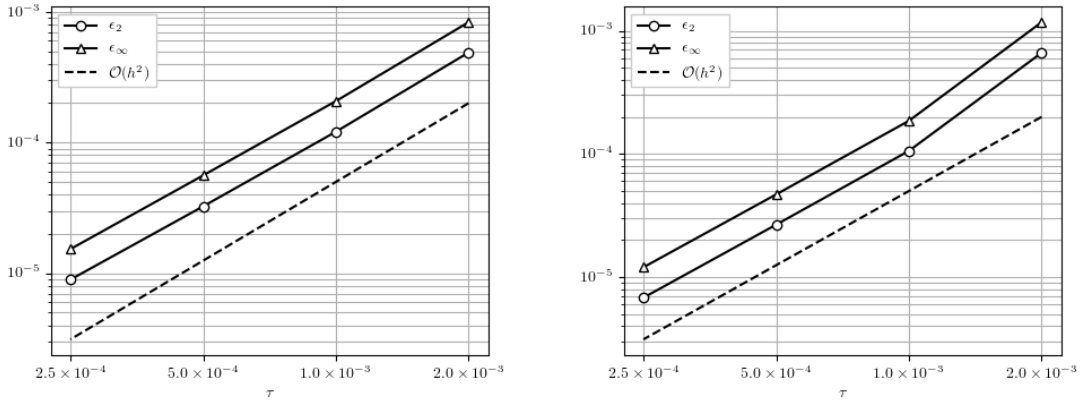
Figure 8-1: Spatial convergence for the moving impulsively started cylinder problem at $\mathrm{Re} = 550$.

8.2.2 $\mathrm{Re} = 550$: Temporal Convergence

If we wish to investigate the temporal error, we can fix h and allow τ to approach zero. As mentioned in Chapter 7, we expect see an error of $\mathcal{O}(\tau^N)$ for larger τ, which is eventually dominated by the $\mathcal{O}(h^2\tau)$ term. Practically, this means that to observe the $\mathcal{O}(\tau^N)$ convergence, we must choose a small N and reasonably fine spatial discretization. For this reason we choose to simulate the moving cylinder with RK2 time integration and spatial resolutions $h = 1/128$, $h = 1/256$. We use the physical parameters

$$\begin{aligned} x_{c,0} &= 0.291, \quad u_{b,x} = 0.995, \quad D = 0.4, \\ y_{c,0} &= 0.457, \quad u_{b,y} = 0.100, \quad \nu = 7.273 \times 10^{-4}, \end{aligned}$$

which correspond to Re = 550. As in the stationary case, we integrate to $t^* = 0.25$ with a single fixed time step, and then use this result as an initial condition for the convergence test. The total fluid impulse $\mathbf{I}(t)$ is calculated on the interval $t^* \in [0.25, 0.88]$ using a range of time steps, and compared to a well-resolved reference solution. The resulting L_2 and L_∞ errors for both resolutions are shown in Figure 8-2, and indicate second order temporal convergence.



(a) $h = 1/128$ (b) $h = 1/256$

Figure 8-2: Second order temporal convergence of the moving cylinder test case. The observed error does not vary significantly when the spatial discretization is changed, indicating that the observed error is purely temporal. Errors are with respect to a reference time step $\tau_{\text{ref}} = 5 \times 10^{-5}$.

8.2.3 Re = 550: Local Forces

To test the computation of local traction forces, we examine a moving cylinder at Re = 550 with parameters

$$\begin{aligned} x_{c,0} &= 0.143, \quad u_{b,x} = 0.7071, \qquad D = 0.2, \\ y_{c,0} &= 0.147, \quad u_{b,y} = 0.7071, \quad \nu = 3.636 \times 10^{-4}, \end{aligned}$$

on a grid with resolution $h = 1/1024$, giving a quality criterion of $N_\delta = 8.73$. The resulting vorticity and pressure distributions are shown in Figure 8-3, and agree well with stationary reference data from Gillis (G) [10], Marichal (M) [15], Lee et al. (L) [13], and Verma et al. (V) [24], as well as with the stationary results from Chapter 6 (S).

8.2.4 Higher Reynolds Number (Re = 3000)

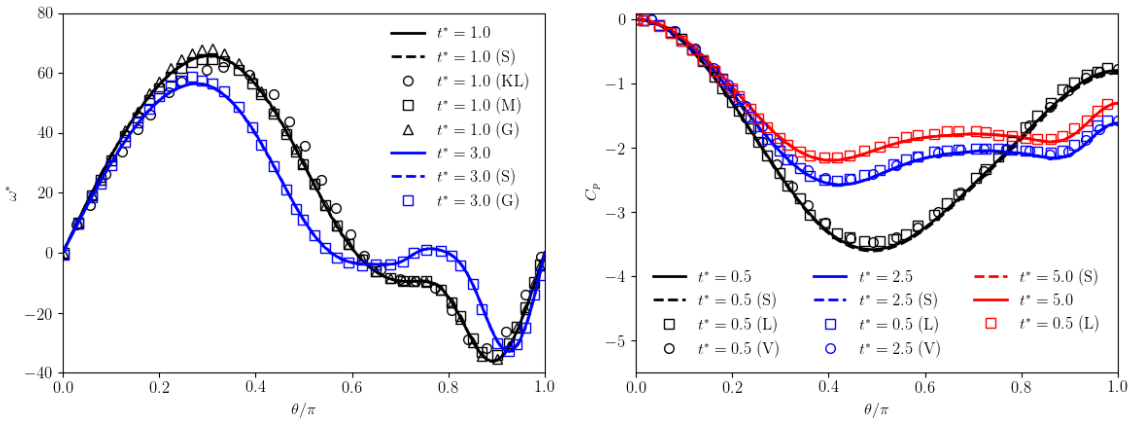
To demonstrate stability for moving domains at moderate Re, we examine the impulsively started cylinder at Re = 3000 with parameters

$$\begin{aligned} x_c &= 0.301, \quad u_{\infty,x}, = 0.7071, \qquad D = 0.4, \\ y_c &= 0.303, \quad u_{\infty,y}, = 0.7071, \quad \nu = 1.333 \times 10^{-4}, \end{aligned}$$

and a grid resolution $h = 1/1024$, giving a quality criterion of $N_\delta = 7.48$. Figure 8-4 shows that the resulting drag forces and tractions agree well with same the stationary reference data used in Chapter 6.

8.3 Flapping Ellipse

To demonstrate the versatility of our discretization, we turn to the flapping foil, a problem with a non-circular moving boundary and non-steady trajectory. We'll use a two-dimensional setup taken



(a) Vorticity Distribution

(b) Pressure Distribution

Figure 8-3: Local loading for the moving impulsively started cylinder at $\mathrm{Re} = 550$.

from by Dong [7], who provides extensive exploration of the foil's parameter space in both two and three dimensions. Consider an ellipse with center $(x_c,\ y_c)$ and major axes a_x and a_y immersed in a fluid with kinematic viscosity ν and free stream velocity U_∞ in the x direction. The ellipse is oriented so that the major axis forms an angle θ with the x-axis (Figure 8-5), and its flapping motion is described by

$$\begin{aligned} x_c(t) &= x_0 \\ y_c(t) &= y_0 + A_y \sin(2\pi f t) \\ \theta(t) &= \theta_0 + A_\theta \cos(2\pi f t). \end{aligned} \tag{8.1}$$

To non-dimensionalize this geometry, we use a_x as a length scale, and select parameters

$$\frac{a_y}{a_x} = 0.12, \qquad \frac{A_y}{a_x} = 0.5, \qquad A_\theta = 30^\circ, \qquad \theta_0 = 0.$$

The non-dimensional quantities which govern the flow and dynamics are the Reynolds number and Strouhal number, given by

$$\mathrm{Re} = \frac{U_\infty a_x}{\nu} = 200, \quad \text{and} \quad \mathrm{St} = \frac{2A_y f}{U_\infty} = 0.6.$$

We also define non-dimensional thrust, lift, and moment coefficients

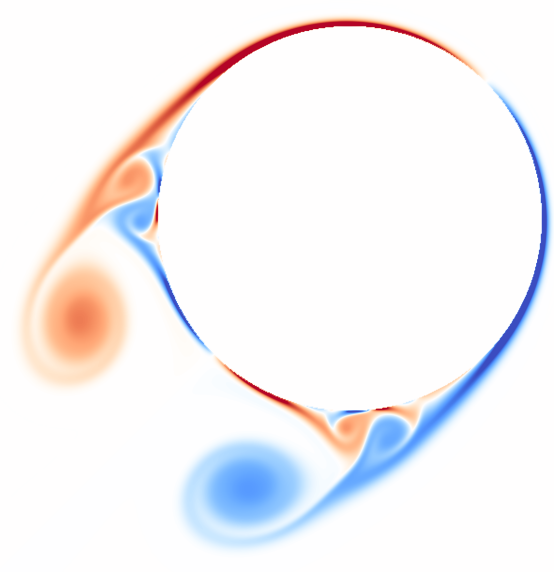
$$C_T = \frac{T}{\frac{1}{2}U_\infty^2 a_x}, \qquad C_L = \frac{L}{\frac{1}{2}U_\infty^2 a_x}, \qquad C_M = \frac{M}{\frac{1}{2}U_\infty^2 a_x^2},$$

which are the main quantities of interest for this test case. Lastly, we define a non-dimensional time $t^* = ft$.

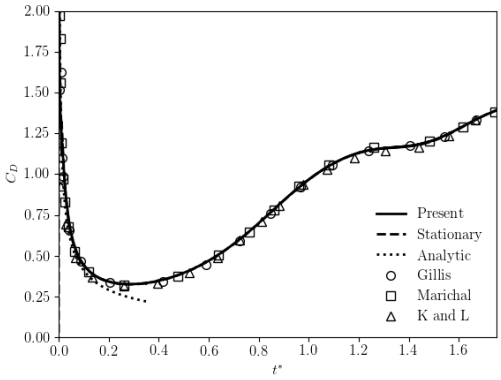
For the results shown here we select $a_x = 0.15$, $U_\infty = 1.0$, and $\mathbf{x}_0 = (0.175,\ 0.500)$; the rest of the dimensional quantities follow from the non-dimensional parameters given above. The simulation begins with zero vorticity in the domain, which corresponds to an impulsive start. Consequently we continue to use the time step ramping procedure (6.1) defined for the impulsively started cylinder.

8.3.1 Results

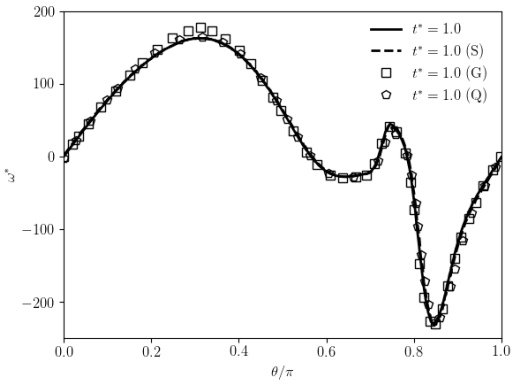
A snapshot of the vorticity field generated by the flapping foil is shown in Figure 8-6. Unlike the translating cylinder, the flapping foil is an accelerating body, which adds an extra term to the control



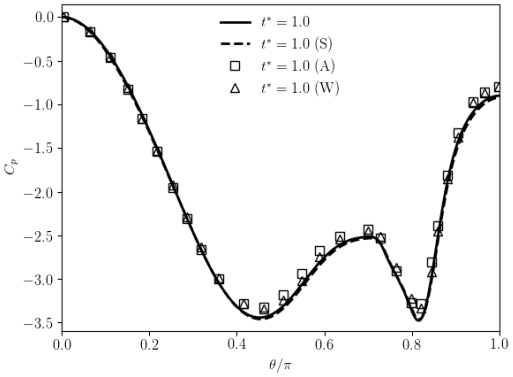
(a) Vorticity Field ($T = 1.0$)



(b) Drag Coefficient



(c) Vorticity Distribution ($T = 1.0$)



(d) Pressure Distribution ($T = 1.0$)

Figure 8-4: Results for the moving impulsively started cylinder at Re = 3000.

volume force formulas based on impulse (see Chapter 5). To avoid this complication, the global loads are calculated using the momentum formulas (5.13) and (5.16). To obtain reference data for these global loads, the same problem is simulated using MRAG-I2D [20], a penalization-based vorticity-velocity solver designed specifically for moving two-dimensional bodies in unbounded domains. The resulting global thrust and lift coefficients are shown in Figures 8-7 and 8-8 respectively, while the moment coefficient is shown in Figure 8-9. Overall the agreement between the two solvers is quite good, particularly for the lift coefficient. The current method shows a slightly increased peak thrust coefficient compared to the reference, and a significantly noisier aerodynamic moment.

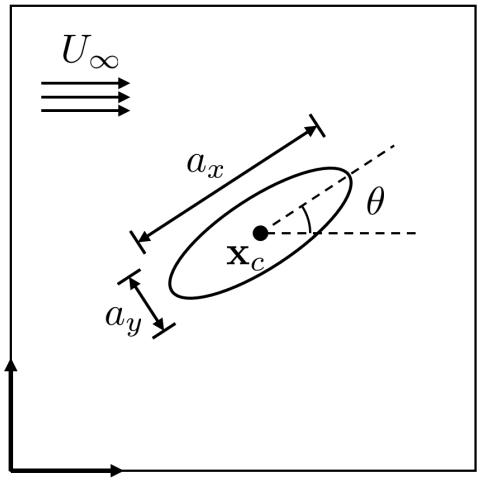


Figure 8-5: Geometric setup for the flapping foil problem.

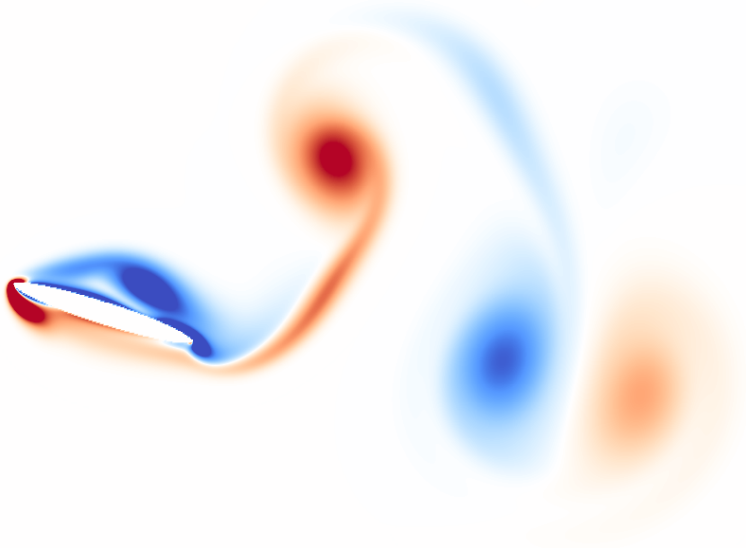


Figure 8-6: Snapshot of the vorticity field around a flapping ellipse at $t^* = 1.6$.

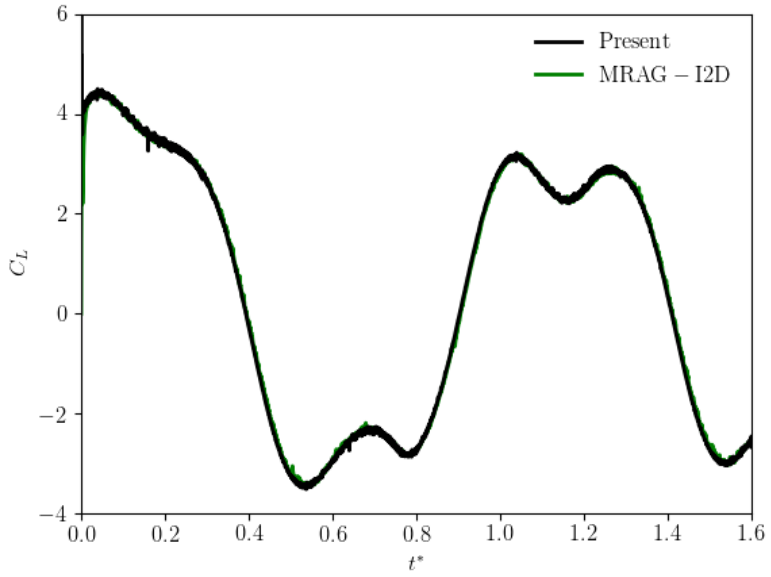


Figure 8-7: Time-dependent lift coefficient for the flapping ellipse at $Re = 200$, $St = 0.6$.

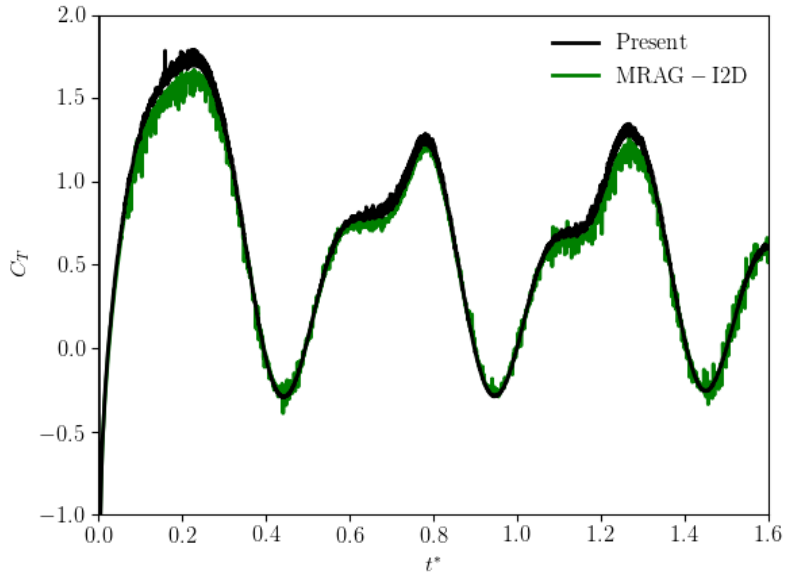


Figure 8-8: Time-dependent thrust coefficient for the flapping ellipse at $Re = 200$, $St = 0.6$.

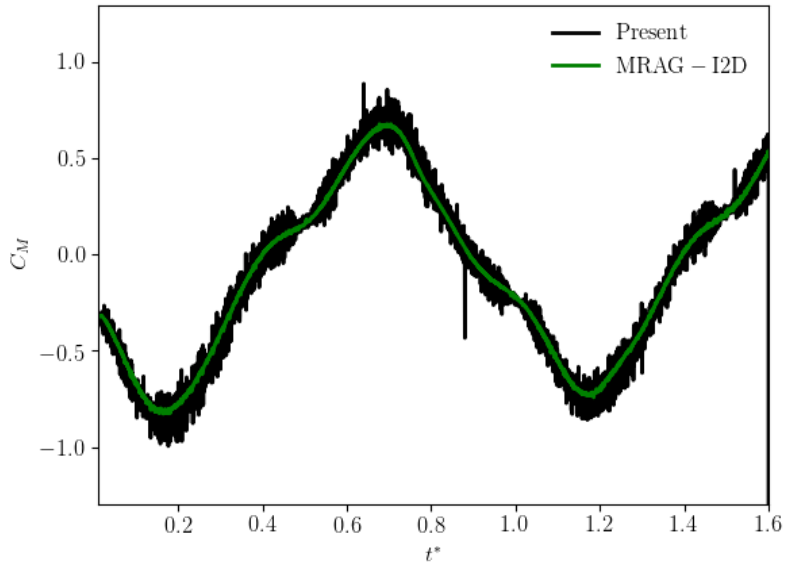


Figure 8-9: Time-dependent moment coefficient for the flapping ellipse at $Re = 200$, $St = 0.6$.

Chapter 9

Conclusions

The culmination of all the ideas developed in this thesis is the moving Navier Stokes algorithm presented in Chapter 8. The success of this method in predicting the local traction forces on a moving body demonstrates that the immersed interface method is a viable alternative to re-meshing techniques for simulations involving biological propulsion and fluid structure interaction. These results are the direct product of the novel immersed interface techniques developed in Chapter 7, which allow for the simulation of PDEs on irregular moving domains using explicit high-order Runge Kutta time integration. Taken together, Chapters 7 and 8 form the main body of original research presented in this thesis.

This research would not have been possible without the solid foundation provided by Gillis, Marichal, Fasel, and all of the other authors who have explored IIM discretizations of the Navier Stokes equations in vorticity-velocity form. Their collective contributions form the backbone of this thesis, and chapters Chapters 2 through 6 are primarily devoted to reviewing this pre-existing work. However, the development of this material has been interspersed with novel improvements to existing discretization techniques, as well as several entirely original contributions to the immersed interface literature:

- In Chapter 2, we clarify the underlying mechanics of the EJIIM by recasting jump corrections explicitly as a one-dimensional polynomial extrapolation, allowing for significant improvements in the methods traditionally used to calculate them. We also provide a novel algorithm for identifying control points from an arbitrary level set, greatly increasing the class of geometries that can be efficiently simulated with the immersed interface method. Finally, by rephrasing existing level set integration methods in the language of the immersed interface method, we allow for second-order evaluation of the integrals necessary to compute global global forces in incompressible flow problems.

- Although the transport scheme introduced in Chapter 3 may be somewhat ad hoc, the IIM boundary treatment developed there is entirely original, and allows for the stable simulation of advection diffusion problems that cannot be handled by the traditional IIM alone.

- In Chapter 4, we reformulate the boundary problem in Gillis' IIM Poisson solver in a way that greatly reduces the dimension of the associated linear system, significantly reducing the memory cost of the rGMRES algorithm and Gillis' chosen recycling scheme.

- Finally, Chapter 5 collects several novel presentations of results relevant to force calculations that are difficult to locate in existing literature, including what we believe to be a novel extension of the work of Noca [16] to the calculation of aerodynamic moments.

Now that this work is drawing toward a close, we turn our eyes towards future research directions. Because the test cases in this thesis were limited to single, simply-connected rigid bodies, one clear path forward is to extend the current formulation to multiple arbitrarily deforming bodies. Together with the force reconstruction techniques considered in Chapter 5, this capability should allow for the

simulation of fully-coupled fluid-structure interaction problems, provided that the associated solid mechanics solver can interface with a traction distribution defined on the control points. There are a plethora of other research projects that could feasibly grow out of material presented in this thesis, and we present the list below as a small sample.

- All of the geometry considered in this thesis was assumed to be convex. However, most interesting and realistic geometries are non-convex. Handling these geometries with the immersed interface method is not trivial, and there is an immediate need for efficient IIM geometry processing algorithms that account for non-convexity .
- The IIM Navier Stokes solver here suffers from stability issues at low resolutions caused by under-resolved boundary layers. These instabilities originate on the boundary, and should be greatly affected by the choice of vorticity boundary condition. Although a global vorticity boundary conditions require increased computational resources, this global coupling may hold the key to increased stability at low resolutions and high Reynolds numbers.
- All of the simulations presented in this thesis involve relatively short time scales. This is because the closure of the two-dimensional reconstruction problem used in Chapter 4 relies on Kelvin's theorem, in a way that requires the entire vorticity field to remain within the computational domain. To move towards simulations that allow outflow, a proper outflow boundary condition for the transport scheme presented in Chapter 3 is necessary, as well as a new strategy for closing the stream function reconstruction system.
- Although Chapter 5 considered several methods of reconstructing the surface pressure field from the boundary vorticity flux, only one was explore in Chapters 6 and 8. The most effective formulation of the pressure Poisson equation in an immersed interface vorticity-velocity framework remains an open question. The calculation of smooth and accurate vorticity flux data is another useful goal for force calculations, since the methods presented in Chapter 5 and by Gillis in [10] are noisy and require a well-resolved simulation to be at all effective.
- We have not touched on implementation strategies for the algorithms described in this thesis. However, there are exciting possibilities for parallel implementations, multiresolution solvers, and other high performance computing related work. There is also a lack of literature on efficient data structures for handling the control points of an immersed interface discretization, which becomes a significant issue for three-dimensional simulations and parallel implementations.

Finally, the simulation of fluid phenomena represents only small portion of the applications that the IIM has found in current computational literature. The use of moving immersed interface techniques for solid mechanics, or as coupling method in multi-physics simulations, is another potentially rewarding area of research.

Bibliography

[1] Christopher R. Anderson and Marc B. Reider. "A high order explicit method for the computation of flow about a circular cylinder". In: *Journal of Computational Physics* 125.1 (1996), pp. 207–224.

[2] M. Bar-Lev and H. T. Yang. "Initial flow field over an impulsively started circular cylinder". In: *Journal of Fluid Mechanics* 72.4 (Dec. 1975), pp. 625–647.

[3] C. Brehm and H. F. Fasel. "A novel concept for the design of immersed interface methods". In: *Journal of Computational Physics* 242 (2013), pp. 234–267.

[4] C. Brehm and H. F. Fasel. "Novel immersed interface method based on local stability conditions". In: *40th AIAA Fluid Dynamics Conference* July (2010), pp. 1–22.

[5] C. Brehm, C. Hader, and H. F. Fasel. "A locally stabilized immersed boundary method for the compressible Navier-Stokes equations". In: *Journal of Computational Physics* 295 (Aug. 2015), pp. 475–504.

[6] Christoph Brehm and Hermann Fasel. "Immersed Interface Method for Solving the Incompressible Navier-Stokes Equations with Moving Boundaries". In: January (2011), pp. 1–19.

[7] H. Dong, R. Mittal, and F. M. Najjar. "Wake topology and hydrodynamic performance of low-aspect-ratio flapping foils". In: *Journal of Fluid Mechanics* 566 (Nov. 2006), pp. 309–343.

[8] Weinan E and Jian Guo Liu. "Vorticity boundary condition and related issues for finite difference schemes". In: *Journal of Computational Physics* 124.2 (1996), pp. 368–382.

[9] T. Gillis, G. Winckelmans, and P. Chatelain. "Fast immersed interface Poisson solver for 3D unbounded problems around arbitrary geometries". In: *Journal of Computational Physics* 354 (Feb. 2018), pp. 403–416.

[10] Thomas Gillis. "Accurate and efficient treatment of solid boundaries for the vortex particle-mesh method". PhD thesis. UCLouvain, 2019.

[11] P. Koumoutsakos and A. Leonard. "High-Resolution simulations of the flow around an impulsively started cylinder using vortex methods". In: *Journal of Fluid Mechanics* 296 (1995), pp. 1–38.

[12] Paco Lagerstrom. *Laminar Flow Theory*. Princeton, N.J.: Princeton University press, 1996.

[13] Seung Jae Lee, Jun Hyeok Lee, and Jung Chun Suh. "Computation of pressure fields around a two-dimensional circular cylinder using the vortex-in-cell and penalization methods". In: *Modelling and Simulation in Engineering* 2014 (2014).

[14] Mark N. Linnick and Hermann F. Fasel. "A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains". In: *Journal of Computational Physics* 204.1 (Mar. 2005), pp. 157–192.

[15] Yves Marichal. "An immersed interface vortex particle-mesh method". PhD thesis. UCLouvain, 2014.

[16] Flavio Noca. "On the evaluation of time-dependent fluid-dynamic forces on bluff bodies". PhD thesis. California Institute of Technology, 1997.

[17] L. Qian and M. Vezza. "A vorticity-based method for incompressible unsteady viscous flows". In: *Journal of Computational Physics* 172.2 (Sept. 2001), pp. 515–542.

[18] L. Quartapelle. "The incompressible Navier—Stokes equations". In: *Numerical Solution of the Incompressible Navier-Stokes Equations*. Basel: Birkhäuser Basel, 1993, pp. 1–11.

[19] Dietmar Rempfer. "On boundary conditions for incompressible Navier-Stokes problems". In: *Applied Mechanics Reviews* 59.1-6 (2006), pp. 107–125.

[20] Diego Rossinelli et al. "MRAG-I2D: Multi-resolution adapted grids for remeshed vortex methods on multicore architectures". In: *Journal of Computational Physics* 288 (May 2015), pp. 1–18.

[21] Peter Smereka. "The numerical approximation of a delta function with application to level set methods". In: *Journal of Computational Physics* 211.1 (2006), pp. 77–90.

[22] John D. Towers. "Discretizing delta functions via finite differences and gradient normalization". In: *Journal of Computational Physics* 228.10 (2009), pp. 3816–3836.

[23] Kritika Upreti. "Algebraic level sets for CAD/CAE integration and moving boundary problems". PhD thesis. Purdue, 2014.

[24] Siddhartha Verma et al. "Computing the force distribution on the surface of complex, deforming geometries using vortex methods and Brinkman penalization". In: *International Journal for Numerical Methods in Fluids* 85.8 (Nov. 2017), pp. 484–501.

[25] Andreas Wiegmann and Kenneth P. Bube. *The Explicit-Jump Immersed Interface Method: Finite Difference Methods for PDES with Piecewise Smooth Solutions*. 2000.

[26] J. H. Williamson. "Low-storage Runge-Kutta schemes". In: *Journal of Computational Physics* 35.1 (Mar. 1980), pp. 48–56.

[27] Chunlin Wu et al. "A conservative viscous vorticity method for unsteady unidirectional and oscillatory flow past a circular cylinder". In: *Ocean Engineering* 191 (Nov. 2019).

[28] J. C. Wu. "Theory for Aerodynamic Force and Moment in Viscous Flows". In: *AIAA Journal* 19.4 (1981), pp. 432–441.

[29] Jie-Zhi Wu, Hui-Yang Ma, and Ming-De Zhou. *Vorticity and Vortex Dynamics*. Springer, 2005, p. 782.

Appendix A

IIM Stencil Calculations

The immersed interface method requires repeated calculations of stencil components for polynomial interpolation. As discussed in section 2.2, each of these stencils can be calculated by solving a single small linear system. Here we list these systems for all of the stencils used in this thesis, as a helpful reference for implementations.

A.1 Interpolation and Extrapolation on a Regular Grid

The intersection algorithm presented in section 2.4.1, the vorticity boundary condition presented in Chapter 4, and the vorticity flux computation discussed in Chapter 5 all require derivatives taken in transverse direction. To provide values for these calculations, we must interpolate values that lie along the local transverse coordinate axis (see Figure 2-6). For N-th order interpolation, consider N consecutive points x_0 through x_{N-1}, as shown in Figure A-1.

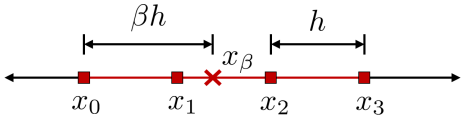


Figure A-1: Labeling scheme for interpolation stencils.

To interpolate the value of $x_\beta = x_0 + \beta h$, we write

$$f_\beta = \sum_{i=0}^{N-1} s_i f_i.$$

For $N = 4$, the stencil coefficients s_i must satisfy

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 4 & 9 \\ 0 & 1 & 8 & 27 \end{bmatrix} \begin{bmatrix} s_\alpha \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 1 \\ \beta \\ \beta^2 \\ \beta^3 \end{bmatrix}. \tag{A.1}$$

For $N = 3$, we drop the last row and last column of the system, so that

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} s_\alpha \\ s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} 1 \\ \beta \\ \beta^2 \end{bmatrix}. \tag{A.2}$$

The extension to higher and lower orders is straightforward.

A.2 Jump Corrections with Dirichlet Condition.

In section 2.2, the following system was derived to calculate the necessary stencil for a fourth order jump correction using a Dirichlet BC

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & (1+\psi) & (2+\psi) & (3+\psi) \\ 0 & (1+\psi)^2 & (2+\psi)^2 & (3+\psi)^2 \\ 0 & (1+\psi)^3 & (2+\psi)^3 & (3+\psi)^3 \end{bmatrix} \begin{bmatrix} s_\alpha \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 1 \\ \psi - 1 \\ (\psi-1)^2 \\ (\psi-1)^3 \end{bmatrix}. \tag{A.3}$$

This system can be truncated to give for third jump order corrections,

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & (1+\psi) & (2+\psi) \\ 0 & (1+\psi)^2 & (2+\psi)^2 \end{bmatrix} \begin{bmatrix} s_\alpha \\ s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} 1 \\ \psi - 1 \\ (\psi-1)^2 \end{bmatrix}, \tag{A.4}$$

and second order jump corrections,

$$\begin{bmatrix} 1 & 1 \\ 0 & (1+\psi) \end{bmatrix} \begin{bmatrix} s_\alpha \\ s_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \psi - 1 \end{bmatrix}. \tag{A.5}$$

A.3 Jump Corrections without a Boundary Condition.

Jump corrections that do not use a boundary condition are both a special case of the above ($\psi = 0$) and a special case of the interpolation stencils from section A.1 ($\beta = -1$). There are no free geometric parameters, so we can solve these systems in advance, giving

$$J_\alpha = 2f_0 - f_1 \qquad \text{(Second Order)} \tag{A.6}$$

$$J_\alpha = 3f_0 - 3f_1 + f_2 \qquad \text{(Third Order)} \tag{A.7}$$

$$J_\alpha = 4f_0 - 6f_1 + 4f_2 - f_3 \qquad \text{(Fourth Order)} \tag{A.8}$$

Jump corrections that do not use a boundary condition are needed to extend the time derivative field in IIM discretizations with moving boundaries.

A.4 Wall derivatives

For third order wall derivatives, we write

$$f'(x_\alpha) = s_\alpha f_\alpha + \sum_{i=1}^{3} s_i f_i,$$

using stencil coefficients s_i that satisfy

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & (1+\psi) & (2+\psi) & (3+\psi) \\ 0 & (1+\psi)^2 & (2+\psi)^2 & (3+\psi)^2 \\ 0 & (1+\psi)^3 & (2+\psi)^3 & (3+\psi)^3 \end{bmatrix} \begin{bmatrix} s_\alpha \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 0 \\ h^{-1} \\ 0 \\ 0 \end{bmatrix}. \tag{A.9}$$

For a second-order wall-derivative, we truncate the above system, so that

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & (1+\psi) & (2+\psi) \\ 0 & (1+\psi)^2 & (2+\psi)^2 \end{bmatrix} \begin{bmatrix} s_\alpha \\ s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} 0 \\ h^{-1} \\ 0 \end{bmatrix}. \tag{A.10}$$

These stencils are used in the calculation of vorticity flux (Chapter 5) and the vorticity boundary condition (Chapter 4).

Appendix B

Control Volume Formulation for Moments

There are a plethora of control volume formulas for calculating forces and moments in the vorticity-velocity formulation, most of which are simplified to a handful of terms by making assumptions on the size or position of the control volume. It is more difficult, however, to find a formulation that makes no assumptions on the size or position of the control volume. Flavio Noca presents a handful of these general formulations in his doctoral thesis [16] which is aimed at the computation of forces in Particle-Image Velocimetry applications. The derivation of a similar formulation for moments can be done in a completely analogous way; however, the author has been unable to find this calculation in the literature. It is presented here because it is needed in Chapters 6 and 8, and with the hope that others who need it will be able to find this appendix instead of re-deriving it for themselves.

The notation used here is taken directly from Noca. We begin with an identity from J. C. Wu's "Theory for Aerodynamic Force and Moment in Viscous flows" [28],

$$\mathbf{x} \wedge \mathbf{a} = -\frac{1}{2} x^2 \nabla \wedge \mathbf{a} + \frac{1}{2} \nabla \wedge (x^2 \mathbf{a}), \tag{B.1}$$

where $x = |\mathbf{x}|$. Applying this to the fluid velocity $\mathbf{u}$ and integrating over a volume yields

$$\int_V \mathbf{x} \wedge \mathbf{u} \, \mathrm{d}V = -\frac{1}{2} \int_V x^2 \boldsymbol{\omega} \, \mathrm{d}V + \frac{1}{2} \oint_S \hat{\mathbf{n}} \wedge (x^2 \mathbf{u}) \, \mathrm{d}S \,. \tag{B.2}$$

The left hand side represents the total angular momentum of the flow, while the right hand side represents the angular impulse and a boundary term which disappears for an infinite domain. This relation is useful when considering a control volume analysis based on angular momentum. Assuming unit density, we express the conservation of angular momentum by

$$\begin{aligned} \mathbf{M} = -\frac{\mathrm{d}}{\mathrm{d}t} \int_{V(t)} \mathbf{x} \wedge \mathbf{u} \, \mathrm{d}V + \oint_{S(t)} \mathbf{x} \wedge [(-p\mathbf{I} + \mathbf{T}) \cdot \hat{\mathbf{n}}] \, \mathrm{d}S \\ - \oint_{S(t)} (\mathbf{x} \wedge \mathbf{u})(\mathbf{u} - \mathbf{u}_s) \cdot \hat{\mathbf{n}} \, \mathrm{d}S - \oint_{S_b(t)} (\mathbf{x} \wedge \mathbf{u})(\mathbf{u} - \mathbf{u}_s) \cdot \hat{\mathbf{n}} \, \mathrm{d}S \,, \end{aligned}$$

where $\mathbf{M}$ is the total moment acting on the immersed body. Using (B.2) to replace the first volume integral,

$$\begin{aligned} \mathbf{M} = \frac{\mathrm{d}}{\mathrm{d}t} \int_{V(t)} \frac{x^2}{2} \boldsymbol{\omega} \, \mathrm{d}V - \frac{\mathrm{d}}{\mathrm{d}t} \oint_{S(t)} \frac{x^2}{2} \hat{\mathbf{n}} \wedge \mathbf{u} \, \mathrm{d}S - \frac{\mathrm{d}}{\mathrm{d}t} \oint_{S_b(t)} \frac{x^2}{2} \hat{\mathbf{n}} \wedge \mathbf{u} \, \mathrm{d}S \\ + \oint_{S(t)} \mathbf{x} \wedge [(-p\mathbf{I} + \mathbf{T}) \cdot \hat{\mathbf{n}}] \, \mathrm{d}S - \oint_{S(t)} (\mathbf{x} \wedge \mathbf{u})(\mathbf{u} - \mathbf{u}_s) \cdot \hat{\mathbf{n}} \, \mathrm{d}S - \oint_{S_b(t)} (\mathbf{x} \wedge \mathbf{u})(\mathbf{u} - \mathbf{u}_s) \cdot \hat{\mathbf{n}} \, \mathrm{d}S \end{aligned} \tag{B.3}$$

Taking a cue from Noca's derivation of the impulse equations, we will adopt the following program:

1. Bring the time derivatives inside the integrals over the moving outer surface.

2. Replace any time derivatives of velocity using the Navier Stokes equations.

3. Transform any resulting ∇p-related terms into p-related terms using an integral identity.

These new pressure terms should exactly cancel the existing pressure term, leaving a pressure-free control volume formulation for the moment acting on a body. To accomplish the first step, we need the tensor identity

$$\frac{\mathrm{d}}{\mathrm{d}t}\oint_{S(t)} \mathbf{A}\hat{\mathbf{n}}\,\mathrm{d}S = \oint_{S(t)} \frac{\partial \mathbf{A}}{\partial t}\hat{\mathbf{n}}\,\mathrm{d}S + \oint_{S(t)} \nabla\cdot\mathbf{A}(\mathbf{u}_s\cdot\hat{\mathbf{n}})\,\mathrm{d}S\,, \tag{B.4}$$

where $(\nabla\cdot\mathbf{A})_i = \sum_j \partial_j A_{ij}$. This can be derived by converting the left hand side to a volume integral, applying Reynold's transport theorem, and then converting back to a surface integral. Let $[\mathbf{a}]$ be the cross-product matrix for a, so that $[\mathbf{a}]\mathbf{x} = \mathbf{a}\wedge\mathbf{x}$. Note that $\nabla\cdot[\mathbf{a}] = -\nabla\wedge\mathbf{a}$. Applying (B.4) to (B.3) gives

$$\begin{aligned}
-\frac{\mathrm{d}}{\mathrm{d}t}\oint_{S(t)} \frac{x^2}{2}\hat{\mathbf{n}}\wedge\mathbf{u}\,\mathrm{d}S &= \frac{\mathrm{d}}{\mathrm{d}t}\oint_{S(t)}\left[\frac{x^2}{2}\mathbf{u}\right]\hat{\mathbf{n}}\,\mathrm{d}S \\
&= \oint_{S(t)}\frac{\partial}{\partial t}\left[\frac{x^2}{2}\mathbf{u}\right]\hat{\mathbf{n}}\,\mathrm{d}S + \oint_{S(t)}\nabla\cdot\left[\frac{x^2}{2}\mathbf{u}\right](\mathbf{u}_s\cdot\hat{\mathbf{n}})\,\mathrm{d}S \\
&= \oint_{S(t)}\frac{x^2}{2}\frac{\partial\mathbf{u}}{\partial t}\wedge\hat{\mathbf{n}}\,\mathrm{d}S - \oint_{S(t)}\nabla\wedge\left(\frac{x^2}{2}\mathbf{u}\right)(\mathbf{u}_s\cdot\hat{\mathbf{n}})\,\mathrm{d}S \\
&= -\oint_{S(t)}\frac{x^2}{2}\hat{\mathbf{n}}\wedge\frac{\partial\mathbf{u}}{\partial t}\,\mathrm{d}S - \oint_{S(t)}\left(\mathbf{x}\wedge\mathbf{u}+\frac{1}{2}x^2\boldsymbol{\omega}\right)(\mathbf{u}_s\cdot\hat{\mathbf{n}})\,\mathrm{d}S
\end{aligned} \tag{B.5}$$

In the last step the vector identity (B.1) has been used. This completes step one of the program.

To eliminate the time derivative of velocity, we'll use the Navier Stokes equations in the slightly nontraditional form

$$\frac{\partial\mathbf{u}}{\partial t} = -\nabla\left(p+\frac{1}{2}u^2\right) + \mathbf{u}\wedge\boldsymbol{\omega} + \nabla\cdot\mathbf{T}.$$

Substituting this into one the first term of (B.5),

$$\begin{aligned}
-\oint_{S(t)}\frac{x^2}{2}\hat{\mathbf{n}}\wedge\frac{\partial\mathbf{u}}{\partial t}\,\mathrm{d}S &= -\oint_{S(t)}\frac{x^2}{2}\hat{\mathbf{n}}\wedge\left(-\nabla\left(p+\frac{1}{2}u^2\right)+\mathbf{u}\wedge\boldsymbol{\omega}+\nabla\cdot\mathbf{T}\right)\mathrm{d}S \\
&= \oint_{S(t)}\frac{x^2}{2}\hat{\mathbf{n}}\wedge\nabla\left(p+\frac{1}{2}u^2\right)\mathrm{d}S \\
&\quad -\oint_{S(t)}\frac{x^2}{2}\hat{\mathbf{n}}\wedge(\mathbf{u}\wedge\boldsymbol{\omega})\,\mathrm{d}S - \oint_{S(t)}\frac{x^2}{2}\hat{\mathbf{n}}\wedge(\nabla\cdot\mathbf{T})\,\mathrm{d}S\,.
\end{aligned} \tag{B.6}$$

The last two integrals on the right hand side are reasonably computable, and will not be manipulated further. This completes step two of the program.

To finish off the derivation, an identity is needed to transform the pressure integral. It turns out that the one we need is

$$\oint_S \frac{x^2}{2}\hat{\mathbf{n}}\wedge\nabla\phi\,\mathrm{d}S = \oint_S \mathbf{x}\wedge\phi\hat{\mathbf{n}}\,\mathrm{d}S\,.$$

This can be proven by noting that

$$\nabla\left(\frac{x^2}{2}\phi\right) = \phi\mathbf{x} + \frac{x^2}{2}\nabla\phi,$$

and integrating over a surface to obtain

$$\oint_S \mathbf{n} \wedge \nabla\left(\frac{x^2}{2}\phi\right) \mathrm{d}S = \oint_S \hat{\mathbf{n}} \wedge \phi\mathbf{x}\, \mathrm{d}S + \oint_S \frac{x^2}{2}\hat{\mathbf{n}} \wedge \nabla\phi\, \mathrm{d}S$$

Using the divergence theorem,

$$\oint_S \mathbf{n} \wedge \nabla\left(\frac{x^2}{2}\phi\right) \mathrm{d}S = \int_V \nabla \wedge \nabla\left(\frac{x^2}{2}\phi\right) \mathrm{d}V = 0,$$

and the identity follows. Picking up the earlier thread,

$$\begin{aligned}\oint_{S(t)} \frac{x^2}{2}\hat{\mathbf{n}} \wedge \nabla\left(p + \frac{1}{2}u^2\right) \mathrm{d}S &= \oint_{S(t)} \mathbf{x} \wedge \left(p + \frac{1}{2}u^2\right)\hat{\mathbf{n}}\, \mathrm{d}S \\ &= \oint_{S(t)} \mathbf{x} \wedge p\hat{\mathbf{n}}\, \mathrm{d}S + \oint_{S(t)} \mathbf{x} \wedge \frac{1}{2}u^2\hat{\mathbf{n}}\, \mathrm{d}S \end{aligned} \tag{B.7}$$

This completes step three of the program. To collect these results, we substitute (B.7) into (B.6), then substitute (B.6) into (B.5), and finally substitute (B.5) into (B.3), giving the expression

$$\begin{aligned}\mathbf{M} = \frac{\mathrm{d}}{\mathrm{d}t}\int_{V(t)} \frac{x^2}{2}\boldsymbol{\omega}\, \mathrm{d}V - \frac{\mathrm{d}}{\mathrm{d}t}\oint_{S_b(t)} \frac{x^2}{2}\hat{\mathbf{n}} \wedge \mathbf{u}\, \mathrm{d}S - \oint_{S_b(t)} (\mathbf{x} \wedge \mathbf{u})(\mathbf{u} - \mathbf{u}_s) \cdot \hat{\mathbf{n}}\, \mathrm{d}S + \oint_{S(t)} \mathbf{x} \wedge p\hat{\mathbf{n}}\, \mathrm{d}S \\ + \oint_{S(t)} \mathbf{x} \wedge \frac{1}{2}u^2\hat{\mathbf{n}} - \frac{x^2}{2}\hat{\mathbf{n}} \wedge (\mathbf{u} \wedge \boldsymbol{\omega}) - \frac{x^2}{2}\hat{\mathbf{n}} \wedge (\nabla \cdot \mathbf{T}) - \left(\mathbf{x} \wedge \mathbf{u} + \frac{1}{2}x^2\boldsymbol{\omega}\right)(\mathbf{u}_s \cdot \hat{\mathbf{n}})\, \mathrm{d}S \\ - \oint_{S(t)} \mathbf{x} \wedge p\hat{\mathbf{n}}\, \mathrm{d}S + \oint_{S(t)} \mathbf{x} \wedge (\mathbf{T} \cdot \hat{\mathbf{n}}) - (\mathbf{x} \wedge \mathbf{u})(\mathbf{u} \cdot \hat{\mathbf{n}}) + (\mathbf{x} \wedge \mathbf{u})(\mathbf{u}_s \cdot \hat{\mathbf{n}})\, \mathrm{d}S\end{aligned}$$

Canceling the pressure terms and collecting the surface terms brings us to an angular-impulse based control volume formula for moments,

$$\mathbf{M} = \frac{\mathrm{d}}{\mathrm{d}t}\int_{V(t)} \frac{x^2}{2}\boldsymbol{\omega}\, \mathrm{d}V - \frac{\mathrm{d}}{\mathrm{d}t}\oint_{S_b(t)} \frac{x^2}{2}\hat{\mathbf{n}} \wedge \mathbf{u}\, \mathrm{d}S - \oint_{S_b(t)} (\mathbf{x} \wedge \mathbf{u})(\mathbf{u} - \mathbf{u}_s) \cdot \hat{\mathbf{n}}\, \mathrm{d}S + \oint_{S(t)} \boldsymbol{\lambda}(\hat{\mathbf{n}})\, \mathrm{d}S\,, \tag{B.8}$$

where the quantity $\lambda(\hat{\mathbf{n}})$ collects miscellaneous surface terms:

$$\begin{aligned}\boldsymbol{\lambda}(\hat{\mathbf{n}}) = \mathbf{x} \wedge \frac{1}{2}u^2\hat{\mathbf{n}} - \frac{x^2}{2}\hat{\mathbf{n}} \wedge (\mathbf{u} \wedge \boldsymbol{\omega}) - \frac{x^2}{2}\hat{\mathbf{n}} \wedge (\nabla \cdot \mathbf{T}) - \left(\frac{1}{2}x^2\boldsymbol{\omega}\right)(\mathbf{u}_s \cdot \hat{\mathbf{n}}) \\ + \mathbf{x} \wedge (\mathbf{T} \cdot \hat{\mathbf{n}}) - (\mathbf{x} \wedge \mathbf{u})(\mathbf{u} \cdot \hat{\mathbf{n}}).\end{aligned} \tag{B.9}$$

We are also free to use (B.1) to transform the above back into an angular momentum formulation. Doing so removes the integral over the immersed boundary, in exchange for an extra integration around the edge of the domain:

$$\mathbf{M} = -\frac{\mathrm{d}}{\mathrm{d}t}\int_{V(t)} \mathbf{x} \wedge \mathbf{u}\, \mathrm{d}V + \frac{\mathrm{d}}{\mathrm{d}t}\oint_{S(t)} \frac{x^2}{2}\hat{\mathbf{n}} \wedge \mathbf{u}\, \mathrm{d}S - \oint_{S_b(t)} (\mathbf{x} \wedge \mathbf{u})(\mathbf{u} - \mathbf{u}_s) \cdot \hat{\mathbf{n}}\, \mathrm{d}S + \oint_{S(t)} \boldsymbol{\lambda}(\hat{\mathbf{n}})\, \mathrm{d}s \tag{B.10}$$

Equations B.8 and B.10 are the control volume formulations introduced in Chapter 5, and used in Chapters 6 and 8.

Appendix C

Analytical Solution for an Impulsively Rotated Cylinder

The impulsively rotated cylinder with imposed axi-symmetry is a flow problem simple enough to have an analytical solution, and after a bit of searching one can find an expression for the velocity field in Lagerstrom [12]. However, the author does not provide the details of the solution, and incorrectly states the resulting vorticity field, leading to an erroneous expression for the shear stress on the cylinder. Here we fill in the details leading to Lagerstrom's velocity field, and provide the correct expression for the shear stress.

Consider an infinitely long cylinder of radius a at rest in an unbounded domain. The fluid in this problem is incompressible and Newtonian, with kinematic viscosity ν. At $t = 0$, the cylinder begins to rotate with angular velocity Ω. The problem is axisymmetric and two dimensional, so it is easiest to work in polar coordinates. The symmetry of the problem implies that

$$u_r = 0 \quad \text{and} \quad \frac{\partial}{\partial \theta} = 0,$$

so that $\mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{0}$ and the Navier-Stokes equations reduce to a one-dimensional PDE for $u_\theta(r, t)$:

$$\frac{\partial u_\theta}{\partial t} = \nu\left(\frac{\partial^2 u_\theta}{\partial r^2} + \frac{1}{r}\frac{\partial u_\theta}{\partial r} - \frac{1}{r^2}u_\theta\right),$$

$$u_\theta(a, t) = \Omega a, \quad \text{and} \quad \lim_{r\to\infty} u_\theta(r, t) = 0,$$

$$u_\theta(r, 0) = 0.$$

To non-dimensionalize the problem, we make the substitutions

$$t^* = \frac{\nu t}{a^2}, \qquad u^* = \frac{u_\theta}{\Omega a}, \qquad r^* = \frac{r}{a}.$$

Then, after collecting terms and dropping the asterisks, the IBVP becomes

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} - \frac{1}{r^2}u,$$

$$u(1, t) = 1, \quad \text{and} \quad \lim_{r\to\infty} u(r, t) = 0,$$

$$u(r, 0) = 0.$$

The problem can be solved using a Laplace transform for the time variable. Let $F(s) = \mathcal{L}[f(t)] =$

$\int_0^\infty f(t)e^{-st}\,\mathrm{d}t$ denote the Laplace transform. For a function $f(t)$ and constant c,

$$\mathcal{L}\left[\frac{\mathrm{d}f}{\mathrm{d}t}\right] = sF(s) - f(0^-), \quad \text{and} \quad \mathcal{L}[c] = cs^{-1}.$$

Applied to the PDE, the transformed function $U(r,s)$ obeys

$$sU - u(r,0^-) = \frac{\partial^2 U}{\partial r^2} + \frac{1}{r}\frac{\partial U}{\partial r} - \frac{1}{r^2}U.$$

Applying the initial condition and boundary conditions, along with some trivial simplification, we obtain

$$r^2\frac{\partial^2 U}{\partial r^2} + r\frac{\partial U}{\partial r} - (1+sr^2)U = 0,$$

$$U(1,s) = \frac{1}{s} \quad \text{and} \quad \lim_{r\to\infty} U(r,s) = 0.$$

This strongly resembles a modified Bessel equation. To get rid of the $r^2 s$ term, we let $\beta = \sqrt{s}r$, so that

$$\frac{\partial}{\partial r} = \sqrt{s}\frac{\partial}{\partial \beta}, \quad \text{and}$$

$$\beta^2\frac{\partial^2 U}{\partial \beta^2} + \beta\frac{\partial U}{\partial \beta} - (1+\beta^2)U = 0.$$

The general solution to this equation $U(r,s) = c_1 I_1(\beta) + c_2 K_1(\beta)$, where I_1 and K_1 are modified Bessel functions of the first and second kind, respectively. Since $I_1(\sqrt{s}r)$ tends to infinity as $r \to \infty$, our boundary conditions dictate that $c_2 = 0$ and

$$c_1 K_1(\sqrt{s}) = \frac{1}{s}.$$

Solving for c_1, we arrive at an expression for our transformed velocity:

$$U(r,s) = \frac{1}{s}\frac{K_1(\sqrt{s}r)}{K_1(\sqrt{s})}.$$

To recover $u(r,t)$, the inverse Laplace transform is required. This is accomplished using the contour integral

$$u(r,t) = \frac{1}{2\pi i}\int_C \frac{K_1(\sqrt{s}r)}{K_1(\sqrt{s})}\frac{e^{st}}{s}\,\mathrm{d}s\,,$$

where the contour C is the Bromwich contour, $s = \gamma + ix$ for $x \in [-\infty, \infty]$. Another quantity of interest is the vorticity field, which can be evaluated using the identity

$$\frac{\mathrm{d}}{\mathrm{d}z}K_n(z) = -K_{n-1}(z) - \frac{n}{z}K_n(z).$$

Specializing to the current case,

$$\frac{\mathrm{d}}{\mathrm{d}r}K_1(\sqrt{s}r) + \frac{1}{r}K_1(\sqrt{s}r) = -\sqrt{s}K_0(\sqrt{s}r).$$

Applying this to the contour integral, we obtain the vorticity field

$$\omega = \frac{\partial u}{\partial r} + \frac{u}{r} = -\frac{1}{2\pi i}\int_C \frac{K_0(\sqrt{s}r)}{K_1(\sqrt{s})}\frac{e^{st}}{\sqrt{s}}\,\mathrm{d}s\,.$$

Evaluating this function at the wall gives the shear stress and moment on the cylinder, since

$$\frac{\tau}{\nu} = -\omega + 2\frac{\partial u}{\partial r} = \frac{\partial u}{\partial r} - \frac{u}{r} = \omega - 2\frac{u}{r} = \omega - 2\Omega, \quad \text{and}$$

$$M = \oint a\tau \, \mathrm{d}s = \int_0^{2\pi} a^2 \tau \, \mathrm{d}\theta = 2\pi\nu a^2(\omega - 2\Omega).$$

From our earlier choice of non-dimensional quantities, we obtain $\omega = \Omega\omega^*$, so that

$$M^* = \frac{M}{2\pi a^2 \nu \Omega} = \omega_w^* - 2.$$

This can be evaluated numerically, with some effort. Consider the integrand for ω_w^*,

$$I(s) = \frac{K_0(\sqrt{s})}{K_1(\sqrt{s})} \frac{e^{st}}{\sqrt{s}}$$

On the contour C, this integrand oscillates with period $2\pi/t$ and decays slowly as $|s| \to \infty$, making the integral difficult to approximate by conventional methods. To avoid this, we choose a different contour. $I(s)$ has a branch cut on the negative real axis, and a singularity at $s = 0$, but otherwise is analytic. Consequently, the integral of $I(s)$ over the contour shown in Figure C-1 vanishes. As the height of this contour is allowed to grow, the vertical segment will become the desired integral. The behavior of $I(s)$ at $s = 0$ and at $|s| \to \infty$ in the left half plane is such that the integrals over the arcs will vanish. Consequently, the integral we seek can be determined from the integrals above and below the branch cut:

$$\omega_w^*(t) = \frac{1}{2\pi i} \int_{-\infty+i\epsilon}^{0+i\epsilon} I(s) \, \mathrm{d}s + \frac{1}{2\pi i} \int_{0-i\epsilon}^{-\infty-i\epsilon} I(s) \, \mathrm{d}s \, .$$

To simplify, let $s = u^2$, so that

$$\frac{K_0(\sqrt{s})}{K_1(\sqrt{s})} \frac{e^{st}}{\sqrt{s}} \, \mathrm{d}s = 2\frac{K_0(u)}{K_1(u)} e^{u^2 t} \, \mathrm{d}u \, .$$

The corresponding contour for u, conveniently, begins at $u = i\infty$, descends to $u = 0$, and then continues down to $u = -i\infty$; the square root "unwraps" the keyhole contour, so that

$$\omega_w^*(t) = \frac{1}{\pi i} \int_{i\infty}^{-i\infty} \frac{K_0(u)}{K_1(u)} e^{u^2 t} \, \mathrm{d}u = \frac{i}{\pi} \int_{-i\infty}^{i\infty} \frac{K_0(u)}{K_1(u)} e^{u^2 t} \, \mathrm{d}u \, .$$

Now let $u = ix$, so that

$$\omega_w^*(t) = -\frac{1}{\pi} \int_{-\infty}^{\infty} \frac{K_0(ix)}{K_1(ix)} e^{-x^2 t} \, \mathrm{d}x \, .$$

Because the K_0 and K_1 are analytic and nonzero on $(0, \infty)$, this integrand has an even real part on the real line and an odd imaginary part, giving

$$\omega_w^*(t) = -\frac{2}{\pi} \int_0^{\infty} \Re\left\{ \frac{K_0(ix)}{K_1(ix)} \right\} e^{-x^2 t} \, \mathrm{d}x \, .$$

This integrand is non-oscillatory, non-singular at $x = 0$, and decays as $e^{-x^2 t}$ as $x \to \infty$. Consequently, it can be evaluated to any desired degree of accuracy. If we wish to avoid complex arithmetic, $K_\alpha(ix)$ can be re-expressed as a combination of $J_\alpha(x)$ and $Y_\alpha(x)$, the Bessel functions of the first and second kind, giving

$$\omega_w^*(t) = -\frac{2}{\pi} \int_0^{\infty} \frac{J_0 Y_1 - J_1 Y_0}{J_1^2 + Y_1^2} e^{-x^2 t} \, \mathrm{d}x \, .$$

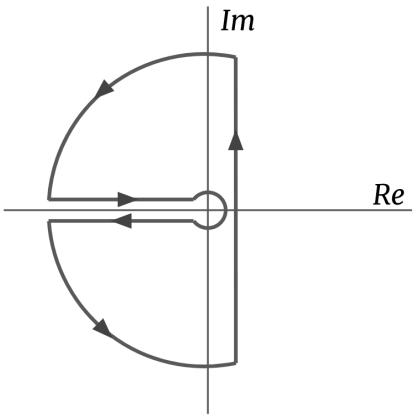


Figure C-1: Contour for the inverse Laplace transform.

However, depending on the numerical implementation of the Bessel functions, this real integrand may or may not evaluate faster than the previous complex integrand.