DISSERTATION

NUMERICAL ALGORITHMS FOR TWO-FLUID, WEAKLY-COMPRESSIBLE FLOWS

Submitted by

Erik Brodin

Department of Mechanical Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2024

Doctoral Committee:

Advisor: Stephen Guzik
Co-Advisor: Phillip Colella

Xinfeng Gao
Wade Troxell
Wolfgang Bangerth

ABSTRACT

NUMERICAL ALGORITHMS FOR TWO-FLUID, WEAKLY-COMPRESSIBLE FLOWS

A multifluid numerical method is developed for flows of two fluids in a single domain at low Mach numbers. An all-speed formulation of the Navier-Stokes equations governs the dynamics of both fluids and the level-set method defines the interface between them and the domain of each fluid. The algorithm represents velocity and pressure as single valued throughout the whole domain, and fluid dependant variables, density and bulk modulus, only in the domain of their respective fluid. The all-speed equations are not subject to the divergence-free velocity constraint through use of a redundant velocity equation, and are evolved in time using an implicit-explicit additive Runge-Kutta method resulting in a time step constrained only by the bulk fluid velocity. Each fluid is evolved conservatively with respect to the moving interface between them. Due to errors in the evolution in the interface, perturbations in the volume of each fluid, and thereby the density, can develop. A thermodynamically consistent correction is made to the position of the interface to reduce these unphysical perturbations. The algorithm developed here includes three novel contributions: (i) the use of a multifluid all-speed algorithm with a level-set method for evolution of the solution in time, (ii) a multifluid algorithm using the level-set to capture the interface in the weakly compressible regime that is thermodynamically consistent, and (iii) an initialization method for sharp corners in the level-set. Numerical tests have demonstrated that the algorithm exhibits the expected low Mach number behavior, achieves second order-accuracy, and ensures fluid volumes are bounded and convergent.

ACKNOWLEDGEMENTS

I would like to acknowledge Dr. Xinfeng Gao, Dr. Phillip Colella, and Dr. Stephen Guzik for their continued technical, professional, and inspiring advice, experience and support in guiding me through the technical and professional rigors of this program. A special thank you to my girlfriend Sarah, for her everlasting support and belief in me. I want to thank my friends Michael and Jordan for their enthusiasm and support during my time here. And my labmates Chris, Andy, Eli, Sean, Nate, and Josh for their friendship and help in the lab and out. I also want to acknowledge Dr. Todd Weisgraber with Lawrence Livermore National Laboratory who has sponsored this project and provided resources for its completion under the U.S. Department of Energy contract DE-AC52-07NA27344.

DEDICATION

I dedicate this dissertation to my mother, Mary.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivations and Objectives

Two-fluid flows, defined as fluid flows with two fluid species occupying distinct regions in space, is commonplace in many engineering applications and physical processes. Numerically, these problems are classified in general as multifluid problems, which represent an arbitrary number of fluids in a single domain. Such flows are seen in a wide range of applications including free surface flows [1, 2], such as waves in the ocean or liquid sloshing in tanks, bubbles rising though a liquid [3, 4], oil spills [5], liquid droplets [3, 4], atmospheric cloud modeling [6, 7], heliophysics [8, 9], and 3D printing. Providing a numerical method to simulate such flows increases the ability of engineers to analyze their important effects, design solutions around the analysis, and reduce expensive design iteration. It also provides scientists insight into physical processes that cannot easily be tested, as with heliophysics.

Many of these listed problems contains two common elements: (i) there are two fluids in the domain separated by an interface, and (ii) the flows are low Mach number, that is the fluid velocity is much smaller compared to the speed of sound ($M \ll 1$). While the second element is not universal and two fluids can flow transonically and supersonically, most practical engineering cases are in the low Mach number regime. In some cases it is important to describe the geometry of the interface, beyond simply its location, to accurately model some other property. An example is surface tension, which is an important factor in modeling bubbles and droplets. The geometry of the interface itself may be of prime importance to represent, like in direct ink writing (DIW) 3D printing.

Multiple fluids in the domain can be represented by a variety of methods, among the most popular are the Volume of Fluid (VOF) method, interface tracking, and interface capturing methods. The VOF method is widely used and tracks the total volume of each fluid in the domain by adding

an advection equation for each fluid to the set of governing equations making the evolution the domain of each fluid in time simple [10, 11]. However, this method does not give any information about the geometry of the interface between the fluids, particularly the curvature, it simply tracks the volume [12]. Reconstructing the interface between the fluids is possible but requires some complex methods, which is problematic if the geometry of the interface is of concern [13–15]. Interface tracking methods explicitly assign coordinate points to the interface and explicitly move them in time, and originally proposed by Harlow and Welch [16]. This class of methods can become cumbersome because it also requires some reconstruction of the interface geometry and is complicated further when there are topology changes to the interface and can become unstable in regions of high curvature [12, 13]. For these reasons, interface tracking is generally considered obsolete. Interface capturing methods avoid many of these pitfalls by representing the interface implicitly. The most widely used method in this class is the level-set method (LSM) of Osher and Sethian [12], where the interface is represented by the zero contour of an implicit function. The interface evolves over time with the local fluid velocity using an advection equation and naturally captures changes in topology. However, the primary drawback of this method is that it does not preserve the volumes of incompressible or weakly compressible fluids [17]. The LSM is the chosen method of this dissertation because of its ease of use, ability to accurately represent the interface without complex reconstruction, and its ability to capture topology changes.

The second primary concern of two-fluid flows in this dissertation is flow at low Mach numbers. All fluids at macroscopic scales, i.e. where the fluid is assumed to be a continuum and not a collection of discrete particles, are described by the Navier-Stokes equations. At low Mach numbers, fluids are typically treated as incompressible, which is an acceptable assumption, by applying the divergence free velocity constraint to the Navier-Stokes equations ($\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0$) [18, 19]. However, when using numerical methods to solve incompressible flow equations, the divergence free velocity constraint poses some difficulty in applying higher order ordinary differential equation (ODE) integrators, such as fourth order Runge-Kutta methods. This is because the operators associated with the divergence-free constraint are functions of space and time. Second order in time

integration methods for incompressible flows have been developed and are typically of a predictor-corrector type scheme that advances the solution without the divergence-free constraint and then corrects the velocity field so that it is divergence-free [20, 21]. These methods quickly become complex and are more unwieldy than traditional ODE integrators and have lower accuracy in time. Constructing a higher order in time scheme for incompressible flow only deepens the issue [22]. Numerically, the advantage of integrating the incompressible equations in time is that the stability constraint is placed on the bulk fluid velocity, not the speed of sound [20, 23]. To overcome the challenges of applying higher order in time methods to the equations of incompressible flow, a special formulation of the compressible flow equations, called the all-speed equations, are utilized. This allows the use of standard ODE integrators that are higher order in time by removing the divergence-free constraint on the velocity. Typically, the disadvantage of using compressible fluid equations is that the stability constraint of the ODE integrator depends on the speed of sound. For low Mach number flows this decreases the maximum stable time step and dramatically increases the computation time. However, the all-speed equations used herein are constructed in such a way and partnered with an additive Runge-Kutta (ARK) ODE integrator so that the stability constraint is based on the bulk fluid velocity, alleviating the primary issue caused by using compressible fluid equations. Therefore, using the all-speed equations, described in Chapter 2, allows the use of simpler and robust higher order in time ODE integrators with a stability constraint that is only dependent on the bulk fluid velocity at low Mach numbers.

These two major points lead to the overarching question this dissertation: how can a numerical algorithm be constructed to simulate two fluids in one domain at low Mach numbers with an accurate representation of geometry of the interface between them? To answer this question, the following specific objectives are achieved to build the numerical algorithm:

1. Implement and discretize a set of governing equations for low Mach-number flows.

2. Apply the level-set method to describe a two fluid system by capturing the interface between the fluids.

3. Develop a volume correction method to ensure each fluid's volume is consistent with its pressure equation of state.

Answering this question is the first step towards a multiphysics simulation of 3D printing.

1.2 Projection Methods and Low Mach Number Flow

To simulate the fluid behavior of two fluids, a set of governing equations that describe a physically consistent evolution of relevant fluid quantities must be implemented. The most important flow characteristic to consider, both physically and numerically, for formulating the governing equations is the fluid velocity relative to the speed of sound, the Mach number (M). . This leads to a set of governing equations that must remain stable at low Mach number flows and recovers the incompressible constraint in the limit as the Mach number goes to zero. Numerical techniques for incompressible flow, where the Mach number is identically zero, were first proposed by Chorin [24] utilizing projection. Two techniques for low Mach number flows are common today: pressure implicit methods and projection methods.

Pressure implicit methods generally utilize an artificial compressibility, a concept introduced by Chorin [25], to enforce the incompressibility condition on the solution. This can be done through a predictor-corrector type scheme, like SIMPLE (Semi-Implicit Method for Pressure Linked Equation) [26] and PISO (Pressure-Implicit with Splitting of Operators) [27]. In these methods the solution is advanced in time then the incompressibility condition is applied to the velocity, the pressure is then updated to be consistent with the velocity. This process is repeated until the solution is considered converged. The primary difference between these two methods is that the PISO algorithm updates pressure at each iteration, whereas SIMPLE uses an iterative method to converge the velocity and pressure to a consistent state. The SIMPLE algorithm is often used for steady state flows and PISO is a common algorithm for transient flows in commercial software like Fluent, OpenFOAM, and Converge. The primary drawbacks of these methods are increased iterations to convergence for more complex flows and their time accuracy for transient flows remains limited.

The zero Mach number projection is a technique expanding on traditional projection, built on the asymptotic expansion of the solution as the Mach number goes to zero. This technique was pioneered by Majda and Sethian [28] where they show that pressure can be split into a time varying part and a spatially varying part, with the spatially varying part scaling with M^2. Projection is used to enforce incompressibility in the analysis. Terms that scale with the Mach number are then eliminated and only Mach number independent terms are advanced. Thus, spatially varying pressure waves (acoustic waves) are completely removed from the solution while retaining thermal effects. Removing the acoustics also eliminates the time step restriction imposed by the speed of sound. This method is well suited to small domains where long wavelength acoustics cannot develop and the Mach number remains very small. If these conditions are not satisfied this method becomes insufficient.

A major advancement in projection methods was made by Bell, Colella, and Glaz (BCG) [20] where they introduced a second order accurate solver in time and space using a Godunov scheme for velocity advection applied to flows following the Boussinesq approximation. This method was improved upon by Bell and Marcus [21] where the method was extended to variable density flows. Lai [29] made another critical improvement using a marker and cell (MAC) discretization of the elliptic operator in the projection and applied it to variable density reacting flows.

This thesis draws largely on the work by Chaplin [30], who applied a MAC projection method to evolve the curl-free velocity alongside the full velocity and applied an ARK time integration to increase the time accuracy of the solution. The application of ARK to a set of weakly compressible equations allows the pressure waves to remain in the solution even at low M with the time step determined only by the fluid velocity. This is the set of governing equations is used herein.

1.3 Additive Runge-Kutta

To evolve the governing PDEs in time requires a time integration method. Including compressible physics in the governing equations imposes a stability constraint on the time step based on the speed of sound, which is overly restrictive for the time scales of 3D printing. The time inte-

gration method is chosen carefully to provide the correct numerical behavior for including weakly compressible physics into the governing equations without stability being restricted by the speed of sound. Additive Runge-Kutta (ARK) methods, introduced by Kennedy and Carpenter [31], are a generalized class of classical Runge-Kutta methods to integrate ordinary differential equations (ODEs) by splitting terms into groups that may be integrated by different Runge-Kutta methods. The most popular splitting is the implicit-explicit (IMEX), where one group of terms is evaluated using an explicit method and the other group is computed using an implicit method. This time integration method has been applied to great success in adaptive mesh methods for convection-diffusion equations [32] and in combustion applications, such as by Christopher [33]. The most attractive feature of ARK methods is the use of implicit solvers for stiff terms alongside traditional explicit methods for non-stiff terms. Using an implicit method for the stiff terms eliminates the restrictive stability constraint on the non-stiff terms. This is how compressible effects can be included in the solution without limiting the time step based on the speed of sound.

Applying an IMEX ARK method to the compressible Navier-Stokes equations is how pressure waves are retained in the solution without a prohibitively small time step, and is the approach taken by Chaplin [30]. Many other projection methods completely remove the pressure waves to achieve a larger time step. This ARK method is used to integrate fluxes associated with the acoustic waves implicitly while integrating fluxes associated with fluid velocity explicitly. Generally, the implicit fluxes are computed using a nonlinear solver, like that described in [31]. The implicit terms of the all-speed equations are arranged so that a computationally expensive nonlinear problem is replaced with a variable coefficient Helmholtz equation. This allows the use of fast multigrid methods for solving elliptic PDEs, bringing another benefit to using the ARK method.

1.4 The Level-Set Method

The level-set method (LSM) is the means to capture the interface between the two fluids. This technique developed by Osher and sethian [12] captures an interface whose topology changes over time. The LSM represents an interface as the zero contour of the signed distance function

(ψ). This method has many useful properties that make its application to flow fields very simple. The evolution of the level-set and the ψ field is given by a scalar advection equation, making it very simple to integrate in time. Changes in topology of the interface are handled naturally, meaning that the interface may split and merge without any special treatment. The definition of ψ is also not required over the entire computational domain, only in a few cells around the interface, greatly reducing computation. The position, orientation, and curvature of the interface is also easily computed through interpolation and standard finite differences, making the calculation of surface properties simple.

Another popular method for tracking two different fluids is the volume of fluid (VOF) method. This method is very useful and is widely employed in modeling free surface flows where the geometry of the free surface is not of primary concern. It works by tracking the volume fraction of a fluid in all cells, which is then used to compute the density in each cell and momentum is computed from this. The primary disadvantages of this method are: (i) it computes the solution over the entire domain, and (ii) it requires complex reconstructions to model the geometry of the interface. Solving for volumes across the whole domain is not necessary for representing the interface, only in cells around the interface. This is an advantage of the LSM, which only requires computation in a small region around the interface. Reconstructing the geometry of the interface with the VOF method requires some complex methods, while the LSM innately captures the geometry of the interface and evolves it over time without any extra methods.

The problem with using the LSM numerically is that it can break conservation of the governing equations due to ψ not being a conserved quantity and numerical errors in computing the ψ field. A volume correction method is applied so that mass may be conserved globally by redistributing mass around the level-set. With this correction the LSM can be coupled to the governing equations to track the interface between the two fluids conservatively.

1.5 Dissertation Organization

The rest of this dissertation is organized as follows. The all-speed system of equations are introduced and their discretization in space and time for a single fluid is described in Chapter 2 with results for verification. This will build the foundation required for extension to two fluids. Next, methods for two-fluids are described in Chapter 3 including the LSM and a level-set correction method. Results of the level-set and the two-fluid system are given in Chapter 4 and discussed. Chapter 5 presents a summary of the findings and draws conclusions from the presented work and gives directions for future work.

Chapter 2

All-Speed Navier Stokes Equations

This section defines the single fluid all-speed equations governing flow at low Mach numbers and their discretization. The two-fluid case utilizes the single fluid equations to evolve the single value quantities: velocity ($\boldsymbol{u}$), curl-free velocity ($\boldsymbol{u}_p$), and pressure (p) in the two-fluid case. There are a few options of governing equations when modeling low Mach number flow: (i) the compressible Navier-Stokes equations, (ii) the incompressible Navier-Stokes equations, (iii) Stokes flow equations, or (iv) the all-speed Navier-Stokes equations.

The fully compressible Navier-Stokes equations are typically employed in the $M > 0.3$ regime, where compressible effects become more significant [18, 34]. Numerical methods for solving the compressible Navier-Stokes equations are well established and used ubiquitously throughout this regime [23, 35–43]. A key feature of the compressible Navier-Stokes equations is the presence of the speed of sound in the characteristics, leading to the stability constraint of the ODE integrators dependent on the speed of sound [18, 23]. This is not a limitation for flows at higher Mach numbers because the fluid velocity is comparable to the speed of sound so it does not generally lead to a restrictive time step. When the Mach number is small ($M \ll 1$) the presence of the speed of sound in the stability constraint leads to an overly restrictive time step, making the time of computation intractable. An alternative would be to use implicit ODE integrators with a much larger stability range or apply complex preconditioners. Unfortunately, this requires careful treatment of the pressure and generally necessitates the use of computationally expensive nonlinear solvers, which mitigates the benefits of larger time steps [44]. For these reasons, the fully compressible Navier-Stokes equations are not suitable for simulating flows at low Mach numbers.

The incompressible Navier-Stokes equations are the second option to model flow at low Mach numbers. They are the limiting case of the compressible equations as the Mach number goes to zero ($M \to 0$) [18, 19], resulting in the divergence-free velocity constraint ($\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0$). At low Mach numbers ($M < 0.3$) this is a valid assumption and is commonly applied in engineering prac-

tice [18, 34]. With the divergence-free constraint, the stability constraint is only on the bulk velocity, not the speed of sound like the compressible equations [20, 23]. However, the divergence-free constraint poses new challenges for numerical integration. The divergence-free constraint must be imposed on the velocity field at all time steps for the solution to remain stable and consistent, resulting in difficult coupling constraints on the integrator [22]. Additionally, large variations in the density in space, a common occurrence in multifluid flows, are difficult to handle [21]. Therefore the incompressible equations posses the desired time stability constraint but introduce a new complexities that prevent the straightforward application of simple and robust higher order in time integrators [45].

Stokes flow is a subset of the incompressible Navier-Stokes equations for low Reynolds numbers ($Re \ll 1$), where viscous effects dominate over inertial effects. These equations do not posses the nonlinear velocity transport term ($\boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u}$), making the governing equations linear and simpler to implement numerically. However, like time-dependent incompressible flow, Stokes flow is subject to the incompressible constraint, $\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0$, which must be enforced at all time points of a time dependent flow field. Therefore, Stokes flow does not improve on the coupling constraints introduced by the incompressible Euler or Navier-Stokes equations [45]. Furthermore, assuming Stokes flow limits the application of this technology to flows where viscous effects dominate and there are no inertial effects. This description is valid for some applications, like the DIW printing filament of highly viscous materials. However, this may not be applicable to all printing materials and methods, like Melt Electro Writing, and is not suitable for two-fluid and mutlifluid methods because one fluid can have a much lower viscosity and not satisfy the conditions for Stokes flow.

The final option, and the one chosen for this dissertation, is the all-speed equations from Chaplin [30], which builds on the work of [20, 46]. These equations possess a time stability condition depending only on the bulk velocity and uses higher order ODE integrators for increased accuracy in time. This is achieved through the use of a redundant velocity equation and an implicit-explicit (IMEX) additive Runge-Kutta (ARK) ODE integrator. The redundant velocity equation retains compressibility effects in the all-speed system by separating the curl-free velocity field, $\boldsymbol{u}_p$,

from the total velocity field, $\boldsymbol{u}$, and evolving them independently, alleviating the difficulties the divergence-free velocity constraint imposes on ODE integrators. Integrating the system with the IMEX ARK method allows the pressure to be evolved implicitly by solving a Helmholtz equation, making the stability constrained only by the fluid velocity. The implicit system of the ARK integrator results in an elliptic PDE and can be solved quickly with multigrid methods. The all-speed system takes the advantages of both the compressible and incompressible systems with the limited cost of adding a single redundant equation for velocity. Long wavelength acoustics are preserved in this formulation as well because compressible effects are included, which is not immediately relevant in this application but can be important for other applications such as flow with chemical reactions.

At low Mach numbers the medium is generally considered incompressible in practical engineering use. Mathematically, incompressible flow is the limiting case of compressible flow as $M \to 0$, so the inclusion of compressible effects is mathematically valid. The inclusion of compressible effects is not to capture minute changes in pressure and density that are not relevant for most applications, but to remove the divergence-free constraint on the velocity at low Mach numbers. Without the divergence-free constraint, well known and mature ODE integrators can be used without modification to evolve the solution in time. If the divergence-free constraint is not removed, application of higher order ODE integrators is very challenging because the divergence-free constraint requires careful consideration of the coupling constraints built into the integrator.

The Navier-Stokes and Euler equations are built upon the continuum assumption, which assumes that at macroscopic scales the fluid can be modeled as a continuum which is infinitely divisible into smaller parts, instead of a collection of discrete particles. At the spatial, pressure, and temperature scales considered, this is a valid assumption for both liquids and gasses. This dissertation only considers flow at low Mach numbers, where the velocity field that satisfies the Euler and Navier-Stokes equations are considered smooth and continuous if the initial condition is smooth and continuous. For cases of a fluid with discontinuities in velocity at such low Mach numbers, it would be necessary to reevaluate the use of a continuum model. Discontinuities are

possible and permitted by the Euler, Navier-Stokes, and all-speed equations at higher Mach numbers ($M \geq 1$) and require specialized techniques to represent stably, but these flow conditions and discontinuity capturing methods are not in the scope of this dissertation.

The method of lines is applied to solve the governing equations, where a PDE is discretized in space to form an ODE in time. The ODE is integrated in time with an ODE integration method, in this case an ARK method, to approximate the solution to the PDE in space and time,

$$\frac{\partial \boldsymbol{\mathcal{U}}}{\partial t} = \mathcal{F}(\boldsymbol{\mathcal{U}}) \rightarrow \frac{\mathrm{d}\boldsymbol{U_i}}{\mathrm{d}t} = F_{\boldsymbol{i}}(\boldsymbol{U}), \tag{2.1}$$

where $\boldsymbol{\mathcal{U}}$ is the analytic solution vector to the PDE, $\mathcal{F}$ is the spatial differential operators, $\boldsymbol{U}$ is the discretized approximation of $\boldsymbol{\mathcal{U}}$, $F_{\boldsymbol{i}}$ is the approximation of the spatial operators of $\mathcal{F}$, and the subscript $\boldsymbol{i}$ indicates the location in the discrete domain. Spatial discretization is done with the Finite Volume Method (FVM), which divides the domain into smaller control volumes (cells) and approximates continuous variables as their average over each cell. The FVM is described in more detail in Section 2.2. The method of lines approach and the Finite Volume Method are mature and robust technologies used extensively for solving systems of conservation equations.

The high performance computational frameworks Chombo and Chord are utilized and expanded to discretize and implement the all-speed equations. Chombo is an open source library for high performance PDE solvers on structured grids provided by Lawrence Berkeley National Laboratory [47]. It provides fundamental data structures and solvers required for the all-speed equations. Chord is used to take advantage of its existing physics tools and is developed by the CFD & Propulsion Laboratory at Colorado State University. It is an FVM based solver designed for solving fully coupled compressible, reacting Navier-Stokes equations on structured Eulerian grids.

This chapter is organized as follows:

1. Definition and description of the inviscid and viscous All-Speed Navier-Stokes equations,

2. The discrete operators to evaluate the fluxes of the governing equations,

3. A description of the projection method and its discrete implementation,

4. The Additive Runge-Kutta (ARK) method and the solution method for the implicit evaluation,

5. Methods for enforcing the isothermal constraint and velocity constraints,

6. The boundary conditions for the governing equations,

7. Results for the single fluid all-speed equations.

2.1 All-Speed Governing Equations

The system of single fluid all-speed governing equations is derived from the convective form of the Navier-Stokes equations. The primary difference is the inclusion of a redundant equation for the curl-free velocity, $\boldsymbol{u}_p$. Adding a redundant equation allows the acoustic waves to be evolved separately from the bulk fluid flow. When the ARK solver is applied, pressure can be solved implicitly using the redundant $\boldsymbol{u}_p$ equation. This retains the acoustic physics without imposing a restrictive time step by including the acoustic waves in the explicit portion of the ARK solver. A more thorough description of this topic is given in Section 2.3.

The governing equations for the all-speed system are

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{u}) = 0, \tag{2.2}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u} + \frac{1}{\rho} \boldsymbol{\nabla} p - \frac{1}{\rho} \boldsymbol{\nabla} \cdot \boldsymbol{\tau} = \boldsymbol{0}, \tag{2.3}$$

$$\frac{\partial \boldsymbol{u}_p}{\partial t} - \mathbb{P}\left(\boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u} + \frac{1}{\rho} \boldsymbol{\nabla} p - \frac{1}{\rho} \boldsymbol{\nabla} \cdot \boldsymbol{\tau}\right) + \boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u} + \frac{1}{\rho} \boldsymbol{\nabla} p - \frac{1}{\rho} \boldsymbol{\nabla} \cdot \boldsymbol{\tau} = 0 \tag{2.4}$$

$$\frac{\partial p}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} p + K \boldsymbol{\nabla} \cdot \boldsymbol{u} = 0, \tag{2.5}$$

$$\boldsymbol{\tau} = \mu \left(\boldsymbol{\nabla} \boldsymbol{u} + (\boldsymbol{\nabla} \boldsymbol{u})^T\right) + \left(\zeta - \frac{2}{3}\mu\right) (\boldsymbol{\nabla} \cdot \boldsymbol{u}) \boldsymbol{I}, \tag{2.6}$$

where ρ, $\boldsymbol{u}$, and p are the density, velocity, and pressure, respectively, μ is the dynamic viscosity, K is the bulk modulus, ζ is the bulk viscosity, and $\mathbb{P}$ is the divergence-free projection operator. The $\mathbb{P}$ operator is discussed in more detail in Section 2.3.

Inviscid All-Speed System The single fluid governing equations for inviscid flow are obtained by setting $\mu = 0$ and assuming the Stokes hypothesis ($\zeta = 0$). Applying these assumptions and grouping the terms into groups evaluated explicitly and implicitly by the IMEX method gives

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \boldsymbol{u}\\ \boldsymbol{u}_p\\ p\end{bmatrix} = \begin{bmatrix}-\boldsymbol{\nabla}\cdot(\rho\boldsymbol{u})\\ -(\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}\\ \mathbb{P}\left(\frac{1}{\rho}\boldsymbol{\nabla}p + (\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}\right) - (\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}\\ -\boldsymbol{u}\cdot\boldsymbol{\nabla}p\end{bmatrix} + \begin{bmatrix}0\\ -\frac{1}{\rho}\boldsymbol{\nabla}p\\ -\frac{1}{\rho}\boldsymbol{\nabla}p\\ -K\boldsymbol{\nabla}\cdot\boldsymbol{u}_p\end{bmatrix}, \tag{2.7}$$

where the first group of terms on the right hand side is the explicit ARK flux and the second term is the implicit flux.

Viscous All-Speed System The viscous system for a single fluid has a similar splitting as the inviscid system,

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \boldsymbol{u}\\ \boldsymbol{u}_p\\ p\end{bmatrix} = \begin{bmatrix}-\boldsymbol{\nabla}\cdot(\rho\boldsymbol{u})\\ -(\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u} - \frac{1}{\rho}\boldsymbol{\nabla}p\\ \mathbb{P}\left((\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u} - \frac{1}{\rho}\boldsymbol{\nabla}\cdot\boldsymbol{\tau} + \frac{1}{\rho}\boldsymbol{\nabla}p\right) - (\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}\\ -\boldsymbol{u}\cdot\boldsymbol{\nabla}p\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{\rho}\boldsymbol{\nabla}\cdot\boldsymbol{\tau}\\ \frac{1}{\rho}\boldsymbol{\nabla}\cdot\boldsymbol{\tau} - \frac{1}{\rho}\boldsymbol{\nabla}p\\ -K\boldsymbol{\nabla}\cdot\boldsymbol{u}_p\end{bmatrix} \tag{2.8}$$

$$\boldsymbol{\tau} = \mu(\boldsymbol{\nabla}\boldsymbol{u} + \boldsymbol{\nabla}\boldsymbol{u}^T) + \left(\zeta - \frac{2}{3}\mu\right)(\boldsymbol{\nabla}\cdot\boldsymbol{u})\,\boldsymbol{I} \tag{2.9}$$

where the first group of terms on the right hand side is the explicit ARK flux and the second group is the implicit flux.

The $[1/\rho]\boldsymbol{\nabla}p$ term is moved from the implicit term to the explicit term in the velocity equation for the set of viscous equations. This allows the velocity to be solved implicitly in ARK by solving

another elliptic equation in the implicit term, keeping the stability constraint for the explicit solver on the bulk velocity.

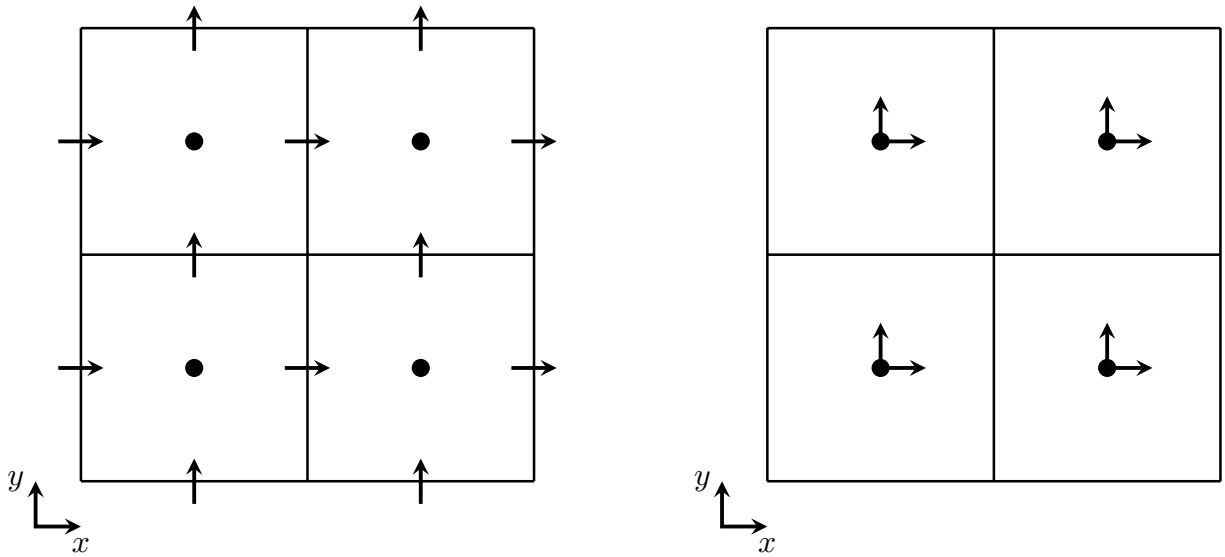
2.2 Spatial Discretization

The governing equations are discretized using the Finite-Volume (FV) discretization on a marker-and-cell (MAC) grid, where all primary dependent variables are given as cell averages except $\boldsymbol{u}_p$, which is given as a face average value. Vectors on faces of a MAC grid are described by the component normal to the face on that face, i.e. the x-normal component of $\boldsymbol{u}_p$ is on the x-normal grid faces, and similarly for the y-normal component. A diagram of the MAC grid and the discretization of vector values is given in Figure 2.1. All operators are described for second order spatial accuracy. The classic structured Cartesian FV discretization is used with control volumes $\Upsilon_{\boldsymbol{i}} = [\boldsymbol{i}h, (\boldsymbol{i}+\boldsymbol{I})h] \in \Upsilon$ where h is the grid spacing, $\boldsymbol{i} \in \mathbb{Z}^D$ is the cell index, $\boldsymbol{I} \in \mathbb{Z}^D$ is the vector with all entries equal to 1, and D is the number of spatial dimensions. Cell-averaged values are defined by

$$\varphi_{\boldsymbol{i}}(t) = \frac{1}{h^{\mathrm{D}}} \int_{\Upsilon_{\boldsymbol{i}}} \varphi(\boldsymbol{x}, t)\, d\boldsymbol{x}, \tag{2.10}$$

with $\boldsymbol{x} \in \mathbb{R}^D$ and t being time. The face between control volumes $\Upsilon_{\boldsymbol{i}}$ and $\Upsilon_{\boldsymbol{i}+\boldsymbol{e}^d}$ is $\Upsilon_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = [(\boldsymbol{i}+\boldsymbol{e}^d)h, (\boldsymbol{i}+\boldsymbol{I})h]$ where $\boldsymbol{e}^d$ is the unit vector in the d direction. Face-averaged values are defined as:

$$\varphi_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}(t) = \frac{1}{h^{\mathrm{D}-1}} \int_{\Upsilon_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}} \varphi(\boldsymbol{x}, t)\, d\boldsymbol{x}. \tag{2.11}$$



(a) Face average vector discretization on a MAC grid.

(b) Cell average discretization of a vector field on a MAC grid.

Figure 2.1: Diagrams of face average and cell average discretizations of vectors on a MAC grid.

Quantities are needed as both cell average and face average values so that projection can be performed or results of the projection can be made available to other operations. Quantities are moved between faces and cells at second order accuracy with a simple averaging operator, A:

$$A^{CF,d}(\varphi) = \frac{1}{2}\left(\varphi_{\boldsymbol{i}} + \varphi_{\boldsymbol{i}+\boldsymbol{e}^d}\right), \tag{2.12}$$

$$A^{CF}(\varphi) = \frac{1}{2}\left(\varphi_{\boldsymbol{i}} + \varphi_{\boldsymbol{i}+\boldsymbol{e}^d}\right) \forall d \in \mathrm{D}-1, \tag{2.13}$$

$$A^{FC,d}(\varphi) = \frac{1}{2}\left(\varphi_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} + \varphi_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}\right), \tag{2.14}$$

$$A^{FC}(\varphi) = \frac{1}{2}\left[\left(\varphi_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^0} + \varphi_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^0}\right), \ldots, \left(\varphi_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^{\mathrm{D}-1}} + \varphi_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^{\mathrm{D}-1}}\right)\right]^T \tag{2.15}$$

and easily extended to vectors,

$$A^{CF,d}(\boldsymbol{u}) = \frac{1}{2}\left(u^d_{\boldsymbol{i}} + u^d_{\boldsymbol{i}+\boldsymbol{e}^d}\right), \tag{2.16}$$

$$A^{FC,d}(\boldsymbol{u}) = \frac{1}{2}\left(u^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} + u^d_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}\right). \tag{2.17}$$

The velocity on faces is necessary for the computation of $\boldsymbol{\nabla} \cdot (\rho \boldsymbol{u})$ and the advective terms $(\boldsymbol{u} \cdot \boldsymbol{\nabla})p$ and $(\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u}$. The velocity on faces must obey the constraint $\boldsymbol{u} = \mathbb{P}(\boldsymbol{u}) + \boldsymbol{u}_p$ imposed by the projection. To impose this constraint, $\boldsymbol{u}$ is averaged to cell faces and then projected with the $\mathbb{P}$ projection and then summed with $\boldsymbol{u}_p$. Velocity components that are not normal to the face are averaged from other faces after the projection and sum with $\boldsymbol{u}_p$ is performed. This gives the total velocity vector on a face as,

$$u^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = \mathbb{P}\left(\mathcal{A}^{CF}(\boldsymbol{u})_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}\right) + u^d_{p,\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}, \tag{2.18}$$

$$u^{d'\neq d}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = \mathcal{A}^{CF}\left(\mathcal{A}^{FC}(u^{d'}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^{d'}})\right). \tag{2.19}$$

Derivatives of cell average and face average values are required and are approximated by simple finite differences. Cell average derivatives are formulated depending on whether the source data is face averaged or cell averaged:

$$\left(\frac{\partial \varphi}{\partial x^d}\right)_{\boldsymbol{i}} = \frac{1}{h}(\varphi_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \varphi_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}) + \mathcal{O}(h^2), \tag{2.20}$$

$$\left(\frac{\partial \varphi}{\partial x^d}\right)_{\boldsymbol{i}} = \frac{1}{2h}(\varphi_{\boldsymbol{i}+\boldsymbol{e}^d} - \varphi_{\boldsymbol{i}-\boldsymbol{e}^d}) + \mathcal{O}(h^2). \tag{2.21}$$

Face average derivatives normal to the face are given by

$$\left(\frac{\partial \varphi}{\partial x^d}\right)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = \frac{1}{h}(\varphi_{\boldsymbol{i}+\boldsymbol{e}^d} - \varphi_{\boldsymbol{i}}) + \mathcal{O}(h^2) \tag{2.22}$$

In the computation of the velocity advection the full gradient is required on faces. The non-normal derivative on a face is found by

$$\left(\frac{\partial \varphi}{\partial x^{d'\neq d}}\right)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = A^{CF}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}\left(A^{FC,d'}_{\boldsymbol{i}}\left(\frac{\partial \varphi}{\partial x^{d'}}\right)\right) + \mathcal{O}(h^2). \tag{2.23}$$

Two types of gradients are defined, the cell centered gradient and the MAC gradient. The cell centered gradient is the standard 2D centered finite difference operator and the MAC gradient takes

the normal gradient of cell values across a face,

$$G^0(\varphi) = \frac{1}{2h}\left(\varphi_{\boldsymbol{i}+\boldsymbol{e}^d} - \varphi_{\boldsymbol{i}-\boldsymbol{e}^d}\right) \forall d = \boldsymbol{\nabla}\varphi + \mathcal{O}(h^2), \tag{2.24}$$

$$G^{\text{MAC}}(\varphi) = \frac{1}{h}\left(\varphi_{\boldsymbol{i}+\boldsymbol{e}^d} - \varphi_{\boldsymbol{i}}\right) = \boldsymbol{\nabla}\varphi + \mathcal{O}(h^2). \tag{2.25}$$

The MAC gradient, G^{MAC}, is consistent with and necessary for the definition of the projection operators discussed in Section 2.3. The gradient is easily extensible to a vector quantity with the result being a second order tensor. The MAC gradient returns the face normal vector components on their respective normal direction faces. The divergence operator is defined only as a cell average quantity, given as

$$D^{\text{MAC}}(\boldsymbol{u}) = \frac{1}{h}\sum_{d=0}^{\text{D}-1}\left(u^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - u^d_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}\right) = \boldsymbol{\nabla}\cdot\boldsymbol{u} + \mathcal{O}(h^2). \tag{2.26}$$

The divergence and gradient operators are constructed to be self-adjoint, consistent with the projection method.

The product of the gradient with a scalar is required as both cell averages and face averages. These products are defined as

$$\left(\frac{1}{\rho}\boldsymbol{\nabla}p\right)_{\boldsymbol{i}} = \frac{1}{\rho_{\boldsymbol{i}}}G^0(p), \tag{2.27}$$

$$\left(\frac{1}{\rho}\boldsymbol{\nabla}p\right)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = \frac{1}{A^{CF}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}(\rho)}G^{\text{MAC}}(p). \tag{2.28}$$

With the inverse of density on faces given as the inverse of the cell centered density averaged to faces,

$$\left(\frac{1}{\rho}\right)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = \frac{1}{A^{CF}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}(\rho)}. \tag{2.29}$$

Scalar advection occurs in the form of divergence of a scalar and vector and the dot product of a vector with the gradient of a scalar. The divergence with a scalar multiple is given as

$$D^{\text{MAC}}(\varphi \boldsymbol{u}) = \frac{1}{h}\sum_{d=0}^{\text{D}-1}\left((\varphi u^d)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - (\varphi u^d)_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}\right) = \boldsymbol{\nabla}\cdot(\varphi\boldsymbol{u}) + \mathcal{O}(h^2). \tag{2.30}$$

The advection of the gradient of a scalar term is given by

$$(\boldsymbol{u}\cdot\boldsymbol{\nabla}\varphi)_{\boldsymbol{i}} = D^{\text{MAC}}(\varphi\boldsymbol{u}) - \varphi_{\boldsymbol{i}} D^{\text{MAC}}(\boldsymbol{u}_p) + \mathcal{O}(h^2). \tag{2.31}$$

And vector advective transport is an extension of the scalar,

$$(\boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u})_{\boldsymbol{i}} = D^{\text{MAC}}(\boldsymbol{u}\boldsymbol{u})_{\boldsymbol{i}} - \boldsymbol{u}_{\boldsymbol{i}} D^{\text{MAC}}(\boldsymbol{u}) + \mathcal{O}(h^2), \tag{2.32}$$

$$= \frac{1}{h}\sum_{d=0}^{D-1}\left[(u^d\boldsymbol{u})_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - (u^d\boldsymbol{u})_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}\right] - \boldsymbol{u}_{\boldsymbol{i}} D^{\text{MAC}}(\boldsymbol{u}_p) + \mathcal{O}(h^2), \tag{2.33}$$

recalling that the velocity at faces must be obtained by (2.18) and (2.19). For example, for the x component of the velocity, the advection is

$$\boldsymbol{u}\cdot\boldsymbol{\nabla}u^0 = \frac{1}{h}\sum_{d=0}^{\text{D}-1}\left[(u^d u^0)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - (u^d u^0)_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}\right] - \boldsymbol{u}_{\boldsymbol{i}} D^{\text{MAC}}(\boldsymbol{u}_p) + \mathcal{O}(h^2). \tag{2.34}$$

Application of viscous effects distinguish the Navier-Stokes system in (2.8) from the inviscid Euler system in (2.7). The shear stress $\boldsymbol{\tau}$ is to be evaluated on faces so the divergence can be taken. Recall,

$$\boldsymbol{\tau} = 2\mu\boldsymbol{s} + \left(\zeta - \frac{2}{3}\mu\right)\boldsymbol{I}\boldsymbol{\nabla}\cdot\boldsymbol{u}, \tag{2.35}$$

$$\boldsymbol{s} = \frac{1}{2}\left(\boldsymbol{\nabla}\boldsymbol{u} + \boldsymbol{\nabla}\boldsymbol{u}^T\right). \tag{2.36}$$

with $\boldsymbol{s}$ being the strain rate tensor, μ being the dynamic viscosity of the fluid, and ζ being the bulk viscosity. The strain rate tensor, $\boldsymbol{s}$, is given on cell faces as,

$$\boldsymbol{s}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = \frac{1}{2}\left(G^{\text{MAC}}(\boldsymbol{u}) + G^{\text{MAC}}(\boldsymbol{u}^T)\right), \tag{2.37}$$

and is used to compute the stress tensor, $\boldsymbol{\tau}$, also on cell faces. The divergence of a rank two tensor is given as

$$D^{\text{MAC}}(\boldsymbol{\tau}) = \frac{1}{h}\sum_{d=0}^{D-1}\left(\tau^d_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \tau^d_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}\right), \tag{2.38}$$

where τ^d indicates the d column vector of the tensor $\boldsymbol{\tau}$. The divergence of the shear stresses is also required on the faces and given by averaging

$$(\boldsymbol{\nabla}\cdot\boldsymbol{\tau})_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = A^{CF}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}(D^{\text{MAC}}(\boldsymbol{\tau})). \tag{2.39}$$

2.3 Projection Methods

The purpose of the projection operation is to decompose a vector field into a curl-free part and divergence free part. Paired with the IMEX integration method, this allows the compressible portion of the velocity, $\boldsymbol{u}_p$, to be integrated implicitly so that the stability is only determined by the bulk velocity. The use of the projection method and IMEX together make this solution method possible.

This method was originally introduced to fluid flows by Chorin [24,48] where the fluid velocity was advanced in time with the divergence-free part and the pressure gradient was computed with the curl-free part. In the analytical case, this is also known mathematically as the Helmholtz or Hodge decomposition [49, 50]. The theory of the Helmholtz decomposition is reviewed here and some important properties are noted that are important for implementing this method numerically.

The Helmholtz decomposition states that any continuous vector field, $\boldsymbol{u}$, on a simply connected domain can be decomposed into the sum of the curl-free component $\mathcal{Q}(\boldsymbol{u})$ and divergence-free component $\mathcal{P}(\boldsymbol{u})$:

$$\boldsymbol{u} = \mathcal{P}(\boldsymbol{u}) + \mathcal{Q}(\boldsymbol{u}). \tag{2.40}$$

so that the decomposed vectors have the properties

$$\boldsymbol{\nabla} \cdot \mathcal{P}(\boldsymbol{u}) = 0, \tag{2.41}$$

$$\boldsymbol{\nabla} \times \mathcal{Q}(\boldsymbol{u}) = \boldsymbol{0}. \tag{2.42}$$

The curl-free field, $\mathcal{Q}(\boldsymbol{u})$, can be represented by the gradient of a scalar field, φ,

$$\mathcal{Q}(\boldsymbol{u}) = \boldsymbol{\nabla}\varphi, \tag{2.43}$$

and is curl-free due to the vector identity $\boldsymbol{\nabla} \times (\boldsymbol{\nabla}\varphi) = \boldsymbol{0}$. Taking the divergence of the entire $\boldsymbol{u}$ field gives

$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = \boldsymbol{\nabla} \cdot \mathcal{P} + \boldsymbol{\nabla} \cdot \boldsymbol{\nabla}\varphi, \tag{2.44}$$

$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = \boldsymbol{\nabla} \cdot \boldsymbol{\nabla}\varphi. \tag{2.45}$$

Written into a more useful form, this is

$$\Delta\varphi = \boldsymbol{\nabla} \cdot \boldsymbol{u} \in \Omega; \frac{\partial\varphi}{\partial n} = \boldsymbol{u} \cdot \boldsymbol{n} \in \partial\Omega, \tag{2.46}$$

where Δ is the Laplace operator, which has a unique solution up to a constant as long as $\int_{\partial\Omega} (\boldsymbol{u} \cdot \boldsymbol{n})\ dA = 0$. The divergence-free projection of $\boldsymbol{u}$ can then be written as

$$\mathcal{P}(\boldsymbol{u}) = \boldsymbol{u} - \boldsymbol{\nabla} \left(\Delta^{-1}(\boldsymbol{\nabla} \cdot \boldsymbol{u}) \right), \tag{2.47}$$

$$\mathcal{P}(\boldsymbol{u}) = \boldsymbol{u} - \mathcal{Q}(\boldsymbol{u}), \tag{2.48}$$

$$\mathcal{Q}(\boldsymbol{u}) = \boldsymbol{\nabla} \left(\Delta^{-1}(\boldsymbol{\nabla} \cdot \boldsymbol{u}) \right), \tag{2.49}$$

where $\Delta^{-1}(\boldsymbol{\nabla} \cdot \boldsymbol{u})$ indicates the solution φ of the Poisson equation (2.46). It can be shown that this decomposition is orthogonal [49], that is $\mathcal{P}(\boldsymbol{u}) \cdot \mathcal{Q}(\boldsymbol{u}) = 0$. Only simply connected domains are considered herein.

The discretized curl-free and divergence-free operators, $\mathbb{Q}(\boldsymbol{u})$ and $\mathbb{P}(\boldsymbol{u})$ follow from the analytical operators and can be implemented in two flavors: approximate and exact. Approximate projection methods, developed by Almgren [51], are simpler because they do not require a MAC type grid and use the cell centered definitions of the divergence and gradient finite differences. However, they decouple the grid so that the L operator has a null-space with rank 2D, with D being the number of spatial dimensions, resulting in a "checkerboard" effect [19, 29, 52]. This projection method is known to cause issues in the solution, particularly in the evolution of pressure [29, 52]. Exact projection does not have this decoupling issue, guaranteeing that the divergence of the $\mathbb{P}$ projection is zero (to machine precision), and can use the standard $2\text{D} + 1$ cell centered Laplace operator. Exact projection is achieved by replicating the idempotency and stability properties of the analytic projection operators [29, 46], that is

$$\mathcal{Q}^2(\boldsymbol{u}) = \mathcal{Q}(\boldsymbol{u}), \quad \mathcal{P}^2(\boldsymbol{u}) = \mathcal{P}(\boldsymbol{u}), \tag{2.50}$$

$$||\mathcal{Q}(\boldsymbol{u})|| \leq ||\boldsymbol{u}||, \quad ||\mathcal{P}(\boldsymbol{u})|| \leq ||\boldsymbol{u}||. \tag{2.51}$$

To replicate these properties, the divergence and gradient operators must be skew adjoint, $D = -G^T$, and the Laplace operator must be the composition of the divergence and gradient operators $L = D \circ G$. The disadvantage is that exact projection typically requires the use of a marker-

and-cell (MAC) type grid, where some quantities are discretized at cell faces and others at cell centers, complicating the coordination and storage of data. Exact projections on cell centered grids are possible but result in Laplace operators that are much more complicated [29]. Exact projection using a MAC grid is used for the projection operators in this dissertation as the increase in accuracy and use of fast standard Laplace solvers outweighs the drawbacks of a MAC grid. This method of projection is used in Chaplin [30] and Colella and Pao [46].

The divergence and gradient operators that form the discrete Laplace operator are the D^{MAC} and G^{MAC} operators described in (2.25) and (2.26), giving L the standard $2\text{D}+1$ centered Laplace operator

$$L_{\boldsymbol{i}}(\varphi_{\boldsymbol{i}}) = \frac{1}{h^2}\sum_{d=0}^{\text{D}-1}(\varphi_{\boldsymbol{i}+\boldsymbol{e}^d} - 2\varphi_{\boldsymbol{i}} + \varphi_{\boldsymbol{i}-\boldsymbol{e}^d}) = \Delta\varphi + \mathcal{O}(h^2). \tag{2.52}$$

Using this standardized Laplace operator enables the use of fast Poisson and multigrid solvers in Chombo [47]. Following the form of the continuous operators, the discrete projection operators are

$$\mathbb{Q}(\boldsymbol{u}) = G^{\text{MAC}}\left(L^{-1}(D^{\text{MAC}}(\boldsymbol{u}))\right), \tag{2.53}$$

$$\mathbb{P}(\boldsymbol{u}) = \boldsymbol{u} - \mathbb{Q}(\boldsymbol{u}), \tag{2.54}$$

where $L^{-1}(D^{\text{MAC}}(\boldsymbol{u}))$ represents the solution, φ, to the discrete Poisson equation, $L(\varphi) = D^{\text{MAC}}(\boldsymbol{u})$, that comes from discretizing (2.46).

2.4 Additive Runge-Kutta

The PDEs in (2.7) and (2.8) are integrated in time using an IMEX ARK method, specifically the 2-ARK$_3$(3)6L[2]SA method from Kennedy [31] is used here. The use of the IMEX method is critical for the inclusion of acoustic waves in the incompressible limit without limiting the time step of the integration. Terms associated with the acoustic waves are included in the implicit portion of the IMEX method so that any time step results in stable integration. The amplitude of the

acoustic waves is also damped by the implicit method as the Mach number decreases, recapturing the expected behavior in the zero Mach number limit.

The inviscid and viscous fluxes are split between the explicit and implicit ARK evaluations. The terms are arranged in such a way that a general nonlinear solver is replaced by a variable coefficient Helmholtz equation for pressure that may be solved with fast multigrid methods. This design dramatically reduces the computation required by the implicit solver because it eliminates an expensive matrix inversion required by the nonlinear solver. There are two formulations for the implicit solution, one for the inviscid system and one for the viscous system.

Inviscid ARK Solver The variables ρ, and ψ have only explicit fluxes, so their evaluation does not require a special solution procedure, and their solution at ARK stage (i) is given by

$$\rho^{(i)} = R_{\rho}^{(i)} \tag{2.55}$$

where $R_{\varphi}^{(i)}$ is the explicitly calculated update of variable φ at stage i:

$$R_{\varphi}^{(i)} = \varphi^n + \Delta t \sum_{j=1}^{i-1} \left(\alpha_{ij}^{[E]} F_{\varphi}^{[E]}(\boldsymbol{U}^{(j)}) + \alpha_{ij}^{[I]} F_{\varphi}^{[I]}(\boldsymbol{U}^{(j)}) \right). \tag{2.56}$$

with $\boldsymbol{U}^{(j)} = [\rho^{(j)}, \boldsymbol{u}^{(j)}, \boldsymbol{u}_p{}^{(j)}, p^{(j)}]$ being the solution vector at ARK stage j, and φ^n being the value at the previous time step, $t = n\Delta t$. The functions $F^{[E]}$ and $F^{[I]}$ indicate the explicit and implicit flux terms of the governing equations, shown as the first and second vectors, respectively, in (2.7). The $\alpha^{[E]}$ and $\alpha^{[I]}$ terms are the explicit and implicit ARK stage coefficients given in Appendix A. The inviscid system contains two variables with implicit fluxes: $\boldsymbol{u}_p$ and p. The ARK stage solution

for $\boldsymbol{u}_p$ and p is

$$\boldsymbol{u}^{(i)} = R_{\boldsymbol{u}_p}^{(i)} - \tilde{\alpha}_{ii}\frac{1}{\rho^{(i)}}\boldsymbol{\nabla}p^{(i)}, \tag{2.57}$$

$$\boldsymbol{u}_p{}^{(i)} = R_{\boldsymbol{u}_p}^{(i)} - \tilde{\alpha}_{ii}\frac{1}{\rho^{(i)}}\boldsymbol{\nabla}p^{(i)}, \tag{2.58}$$

$$p^{(i)} = R_p^{(i)} - \tilde{\alpha}_{ii}K\boldsymbol{\nabla}\cdot\boldsymbol{u}_p{}^{(i)}, \tag{2.59}$$

where $\tilde{\alpha}_{ii} = \alpha_{ii}^{[I]}\Delta t$.

Equations (2.58) and (2.59) can be rearranged into a single variable coefficient Helmholtz equation for pressure,

$$\left[\frac{1}{K} - \tilde{\alpha}_{ii}^2\boldsymbol{\nabla}\cdot\frac{1}{\rho^{(i)}}\boldsymbol{\nabla}\right]p^{(i)} = \frac{R_p^{(i)}}{K} - \tilde{\alpha}_{ii}\boldsymbol{\nabla}\cdot R_{\boldsymbol{u}_p}^{(i)} \in \Omega, \qquad \frac{\partial p^{(i)}}{\partial \boldsymbol{n}} = 0 \in \partial\Omega, \tag{2.60}$$

where K is the bulk modulus of the fluid, assumed to be constant. Note that $\rho^{(i)}$ is known because it is an entirely explicit evaluation. This equation is solved with a multigrid method. The solution to (2.60) is unique up to a constant because the Neumann boundary conditions are specified, however this is not an issue in the ARK stage solution of the governing equations because only $\boldsymbol{\nabla}p$ is required, which is determined uniquely. The solution at the time step p^{n+1} is still uniquely determined because its change only depends on $\boldsymbol{\nabla}p$, which is unique from the Helmholtz solution, and is given an initial value. Once the solution for $p^{(i)}$ is found, $\boldsymbol{u}^{(i)}$ and $\boldsymbol{u}_p{}^{(i)}$ are found by substituting $p^{(i)}$ into (2.57) and (2.58).

Viscous ARK Solver In the viscous system the shear stresses appear in the velocity implicit flux term forming another elliptic equation for velocity at stage (i). The ARK stage solution for

velocity in the viscous system is

$$\rho^{(i)} = R_{\rho}^{(i)} \tag{2.61}$$

$$\boldsymbol{u}^{(i)} = R_{\boldsymbol{u}}^{(i)} + \tilde{\alpha}_{ii}\frac{1}{\rho_{\boldsymbol{i}}^{(i)}}\left(\boldsymbol{\nabla}\cdot\boldsymbol{\tau}^{(i)}\right), \tag{2.62}$$

$$\boldsymbol{u}_p{}^{(i)} = R_{\boldsymbol{u}_p}^{(i)} + \tilde{\alpha}_{ii}\left(\frac{1}{\rho^{(i)}}\boldsymbol{\nabla}\cdot\boldsymbol{\tau}^{(i)} - \frac{1}{\rho}\boldsymbol{\nabla}p^{(i)}\right), \tag{2.63}$$

$$p^{(i)} = R_{p}^{(i)} - \tilde{\alpha}_{ii}K\boldsymbol{\nabla}\cdot\boldsymbol{u}_p{}^{(i)}. \tag{2.64}$$

Expressing $\boldsymbol{\tau}^{(i)}$ in terms of velocity, and rearranging (2.62) yields

$$\left[\rho^{(i)} - \tilde{\alpha}_{ii}\boldsymbol{\nabla}\cdot\left(\mu(\boldsymbol{\nabla}+\boldsymbol{\nabla}^T) + \mu\left(\bar{\zeta} - \frac{2}{3}\right)\boldsymbol{I}\boldsymbol{\nabla}\cdot\right)\right]\boldsymbol{u}^{(i)} = \rho^{(i)}R_{\boldsymbol{u}}^{(i)} \in \Omega, \qquad \boldsymbol{u}^{(i)} = \boldsymbol{u}|_{\partial\Omega} \in \partial\Omega, \tag{2.65}$$

where μ is the dynamic viscosity, and $\bar{\zeta} = \zeta/\mu$ is the ratio of the bulk viscosity to the dynamic viscosity at the initial temperature. Both the dynamic viscosity and bulk viscosity are assumed to be constant. This equation is solved using the multigrid method. The application of the Dirichlet condition to the velocity Helmholtz equations guarantees that the velocity is uniquely determined. The velocity boundary condition for the Helmholtz equation must be equivalent to the velocity boundary condition for the system of governing equations. The velocity solution is determined first before computing the pressure.

There is no change to the stage solution for pressure, but the stage solution of $\boldsymbol{u}_p$ now includes viscous stresses, resulting in a change to the pressure Helmholtz equation,

$$\left[\frac{1}{K} - \bar{\alpha}_{ii}^2\boldsymbol{\nabla}\cdot\left(\frac{1}{\rho^{(i)}}\boldsymbol{\nabla}\right)\right]p^{(i)} = \frac{R_p^{(i)}}{K} - \tilde{\alpha}_{ii}\boldsymbol{\nabla}\cdot\left(R_{\boldsymbol{u}_p}^{(i)} + \tilde{\alpha}_{ii}\frac{1}{\rho^{(i)}}\boldsymbol{\nabla}\cdot\boldsymbol{\tau}^{(i)}\right), \quad \left.\frac{\partial p}{\partial \boldsymbol{n}}\right|_{\partial\Omega} = 0, \tag{2.66}$$

where $\boldsymbol{\tau}^{(i)}$ is the viscous stress tensor calculated with the $\boldsymbol{u}^{(i)}$ velocity from the solution of (2.65). Like the inviscid case, the $p^{(i)}$ solution is used to compute the (i) stage implicit flux for $\boldsymbol{u}_p{}^{(i)}$ to complete the stage evaluation.

Time Step One of the primary purposes of using an IMEX ARK method is to use a time step that is only constrained by the bulk fluid velocity. In both the inviscid and viscous cases the time step is given by

$$\Delta t = \sigma \frac{h}{\sum_{d=0}^{D-1} |u^d|}, \tag{2.67}$$

where $\sigma \leq 1$ is the Courant-Friedrichs-Lewy (CFL) number.

2.5 Constraint Enforcement

The governing equations are subject to two major constraints: the system is isothermal and the velocity must be the sum of the divergence-free and curl-free projection. The solution may stray from these constraints due to numerical errors and must be enforced at the end of every time step. The pressure is relaxed to be consistent with the isothermal condition. Once the correction is made, the constraint that $\boldsymbol{u} = \mathbb{P}(\boldsymbol{u}) + \boldsymbol{u}_p$ is enforced.

When p is relaxed to the isothermal constraint, it must also be reflected in $\boldsymbol{u}_p$. To enforce this, the relationship between p and $\boldsymbol{u}_p$ is taken from the governing equations. The velocity, $\boldsymbol{u}$, is also affected by the change in pressure but this change will be applied when enforcing the consistency between $\boldsymbol{u}$ and $\boldsymbol{u}_p$, shown next. This gives the equations for the relaxation as

$$\delta \boldsymbol{u}_p{}^{n+1} = -\frac{\Delta t}{\rho^{n+1}} \boldsymbol{\nabla} \delta p^{n+1}, \tag{2.68}$$

$$\delta p^{n+1} = -K \Delta t \boldsymbol{\nabla} \cdot \delta \boldsymbol{u}_p{}^{n+1} + \eta \Delta t (p_0^{n+1} - \tilde{p}^{n+1}) \tag{2.69}$$

where $\tilde{p}^{n+1}$ represents the pressure after the ARK integration to time step $n+1$, T_0 is the initial isothermal temperature calculated from the initial pressure and density, R is the specific gas constant, and $\eta = 1/(2\Delta t)$ is a relaxation parameter. Rearranging these equations into a Helmholtz

system gives

$$\left[\frac{1}{K} - \Delta t^2 \boldsymbol{\nabla} \cdot \left(\frac{1}{\rho^{n+1}} \boldsymbol{\nabla}\right)\right] \delta p^{n+1} = \frac{1}{K} \eta \Delta t (p_0^{n+1} - \tilde{p}^{n+1}) \in \Omega, \qquad \frac{\partial p^{n+1}}{\partial \boldsymbol{n}} = 0 \in \partial\Omega. \quad (2.70)$$

The pressure correction is used to find the $\delta \boldsymbol{u}_p$ term. The velocity and pressure variables are then updated and velocity constraint applied as

$$\boldsymbol{u}_p{}^{n+1} = \mathbb{Q}(\tilde{\boldsymbol{u}}_p{}^{n+1} + \delta \boldsymbol{u}_p{}^{n+1}), \tag{2.71}$$

$$p^{n+1} = \tilde{p}^{n+1} + \delta p^{n+1}, \tag{2.72}$$

$$\boldsymbol{u}^{n+1} = A_{\boldsymbol{i}}^{FC}(\boldsymbol{u}_p{}^{n+1}) + \mathbb{P}(\tilde{\boldsymbol{u}}^{n+1}), \tag{2.73}$$

where $\tilde{\boldsymbol{u}_p}{}^{n+1}$, $\tilde{p}^{n+1}$, and $\tilde{\boldsymbol{u}}^{n+1}$ represent the uncorrected solutions from the ARK integration at time step $n+1$. By updating the velocity with $\boldsymbol{u}_p{}^{n+1}$ containing the correction, $\boldsymbol{u}$ now contains the pressure correction as well. These constraints are imposed at the end of each time step.

2.6 Initial and Boundary Conditions

Solid wall boundary conditions are used for all solutions obtained in this thesis, except for the viscous shear layer test which uses periodic boundary conditions. To achieve this, the homogeneous Neumann condition is applied to the density,

$$\frac{\partial \rho}{\partial \boldsymbol{n}} = 0 \in \partial\Omega. \tag{2.74}$$

For inviscid problems, the walls are considered slip walls with the boundary conditions

$$\boldsymbol{u} \cdot \boldsymbol{n} = 0, \quad \frac{\partial \boldsymbol{u}_{\parallel}}{\partial \boldsymbol{n}} = 0 \in \partial\Omega, \tag{2.75}$$

where $\boldsymbol{u}_{\parallel}$ indicates components of the velocity parallel to the faces. Periodic boundary conditions are applied to viscous problems. The application of solid walls results in a homogeneous Neumann

condition for the pressure,

$$\frac{\partial p}{\partial \boldsymbol{n}} = 0 \in \partial\Omega. \tag{2.76}$$

To initialize $\boldsymbol{u}_p$ on the faces, the face normal component of $\boldsymbol{u}$ is initialized on all faces and then projected with $\mathbb{Q}$, that is

$$\boldsymbol{u}_{p,\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}(t=0) = \mathbb{Q}(\boldsymbol{u}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}(t=0)) \tag{2.77}$$

This is opposed to initializing at cell centers, averaging to faces and then projecting, which does not reproduce the identically zero curl-free velocity of the initial conditions.

2.7 Results

Inviscid Vortex The inviscid vortex tests the inviscid all-speed system at low M. The initial condition is a potential flow vortex:

$$\rho = 1, \tag{2.78}$$

$$\boldsymbol{u} = \begin{bmatrix} 2\sin^2(\pi x)\sin(\pi y)\cos(\pi y) \\ -2\sin^2(\pi y)\sin(\pi x)\cos(\pi x) \end{bmatrix}, \tag{2.79}$$

$$p = 1\times 10^6, \tag{2.80}$$

these initial conditions give $M \approx 8.2\times 10^{-4}$. The slip wall boundary condition is applied and the system is solved on the domain $\Omega = [0,1]$ in 2D. Richardson extrapolation is used to determine the error. Tests were run at $\sigma = 0.5$ and the Richardson extrapolation is performed on data from $t = 0.5$. Images of the solution are shown at $N = 128$ cells at $t = 0.5$.

The inviscid system shows second order convergence for the solution variables u, v, p as expected as well as the divergence-free velocity components, u_d and v_d. The divergence-free velocity components are of more interest for this case as it contains the vast majority of the contribution to

Table 2.1: The L_∞ norm convergence of the inviscid vortex showing $\mathcal{O}(h^2)$ order convergence.

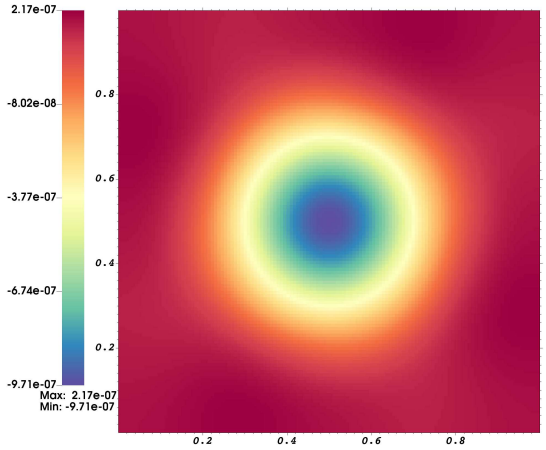
	$N = 16$	Rate	$N = 32$	Rate	$N = 64$	Rate	$N = 128$
ρ	5.678E-8	2.062	1.360E-8	1.984	3.437E-9	0.052	3.315E-9
u	4.865E-2	2.010	1.208E-2	1.960	3.105E-3	1.978	7.881E-4
v	4.865E-2	2.010	1.208E-2	1.960	3.105E-3	1.978	7.881E-4
p	5.204E-2	1.939	1.357E-2	1.986	3.424E-3	2.059	8.217E-4
u_d	4.865E-2	2.010	1.208E-2	1.960	3.105E-3	1.978	7.881E-4
v_d	4.865E-2	2.010	1.208E-2	1.960	3.105E-3	1.978	7.881E-4

Table 2.2: The L_1 norm convergence of the inviscid vortex showing $\mathcal{O}(h^2)$ order convergence.

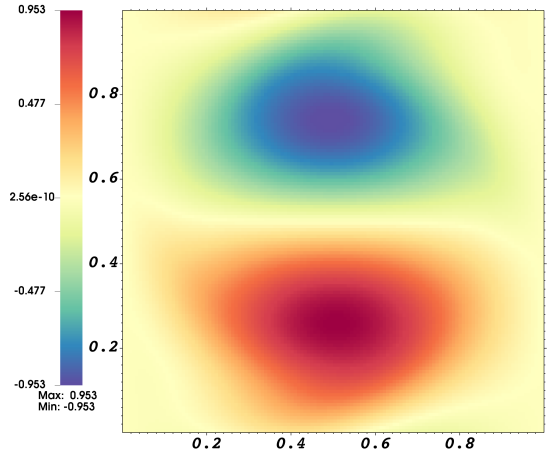
	$N = 16$	Rate	$N = 32$	Rate	$N = 64$	Rate	$N = 128$
ρ	1.514E-8	2.321	3.029E-9	1.836	8.484E-10	-0.153	9.432E-10
u	1.442E-2	2.055	3.472E-3	2.013	8.601E-4	2.002	2.146E-4
v	1.442E-2	2.055	3.472E-3	2.013	8.601E-4	2.002	2.146E-4
p	1.001E-2	1.988	2.523E-3	2.000	6.306E-4	1.922	1.664E-4
u_d	1.442E-2	2.055	3.472E-3	2.013	8.601E-4	2.002	2.147E-4
v_d	1.442E-2	2.055	3.472E-3	2.013	8.601E-4	2.002	2.147E-4

$\boldsymbol{u}$. There is a significant drop from second order convergence for ρ. This is because the changes in density are due purely to the divergence of $\boldsymbol{u}_p$, whose accuracy is limited by the tolerance of the multigrid solution to projection operations and Helmholtz solves. Near the incompressible limit, variations in the density are so small they run up against the machine precision. This is reflected by the low value of the L_p error norms in Table 2.1–Table 2.3. It is also seen from Table 2.1–Table 2.3 that the solution displays $\mathcal{O}(h^2)$ accuracy for all L_p error norms.

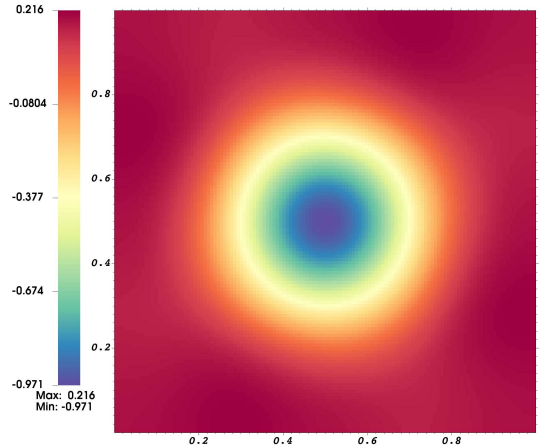
The all-speed system recovers the expected $\mathcal{O}(M^2)$ behavior at low Mach numbers as shown in Figure 2.3, where the normalized pressure difference at $t = 0.5$ is shown. This is corroborated by the pressure difference shown in Figure 2.2c, where the difference is very small compared to the initial pressure, $p(t = 0) = 1 \times 10^6$. There is a transition around $M \approx 8 \times 10^{-3}$, where the limiting behavior switches from $\mathcal{O}(M)$ to $\mathcal{O}(M^2)$. This is expected behavior where there is a transition from higher M asymptotics to lower M asymptotics.



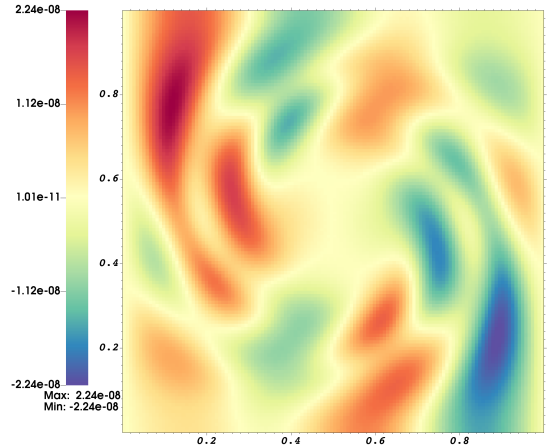
(a) $\rho(t = 0.5) - \rho(t = 0)$



(b) $u(t = 0.5)$



(c) $p(t = 0.5) - p(t = 0)$



(d) $u_p(t = 0.5)$

Figure 2.2: End state of the inviscid vortex.

Table 2.3: The L_2 norm convergence of the inviscid vortex showing $\mathcal{O}(h^2)$ order convergence.

	$N = 16$	Rate	$N = 32$	Rate	$N = 64$	Rate	$N = 128$
ρ	2.045E-8	2.270	4.241E-9	1.876	1.155E-9	-0.026	1.176E-9
u	1.790E-2	2.030	4.384E-3	2.009	1.089E-3	2.003	2.718E-4
v	1.790E-2	2.030	4.384E-3	2.009	1.089E-3	2.003	2.718E-4
p	1.442E-2	2.006	3.590E-3	2.001	8.970E-4	1.986	2.264E-4
u_d	1.790E-2	2.030	4.384E-3	2.009	1.089E-3	2.003	2.718E-4
v_d	1.790E-2	2.030	4.384E-3	2.009	1.089E-3	2.003	2.718E-4

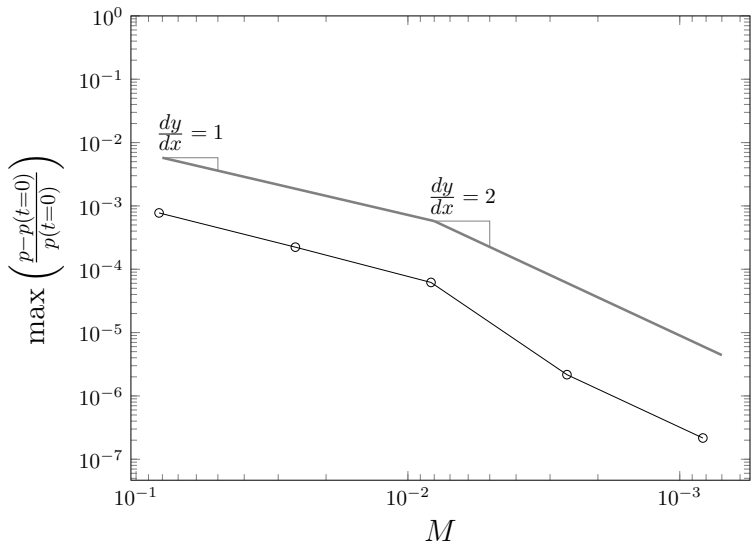


Figure 2.3: Asymptotic behavior of p with the Mach number at $N = 128$ and $t = 0.5$.

Viscous Shear Layer The viscous shear layer tests the viscous All-Speed system for the behavior and convergence of the viscous all-speed system for a single fluid. The initial condition generates two shear layers with high vorticity:

$$\rho = 1, \tag{2.81}$$

$$u = \begin{cases} \tanh\left(30(y - 0.25)\right) & y \leq 0.5 \\ \tanh\left(30(0.75 - y)\right) & \text{otherwise} \end{cases} \tag{2.82}$$

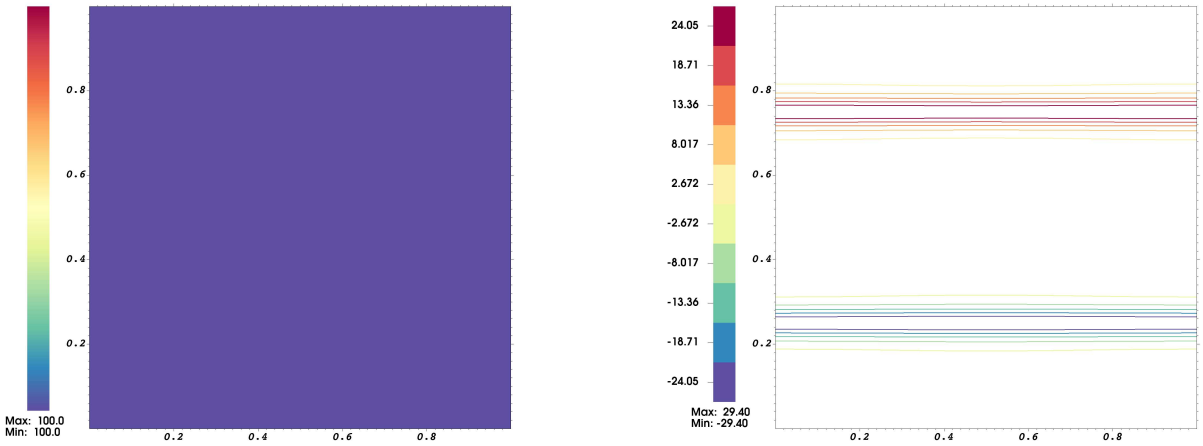
$$v = 0.05 \sin\left(2\pi x\right) \tag{2.83}$$

$$p = 100, \tag{2.84}$$

These initial conditions provide a Reynolds number of approximately $1/\mu$. Periodic boundary conditions are applied to all sides and the system is solved on the domain $\Omega = [0, 1]$ in 2D and run until the end time $t = 0.75$ with $\sigma = 0.5$ where Richardson extrapolation is used to determine the error. Images of the solution are shown at $N = 128$ cells.

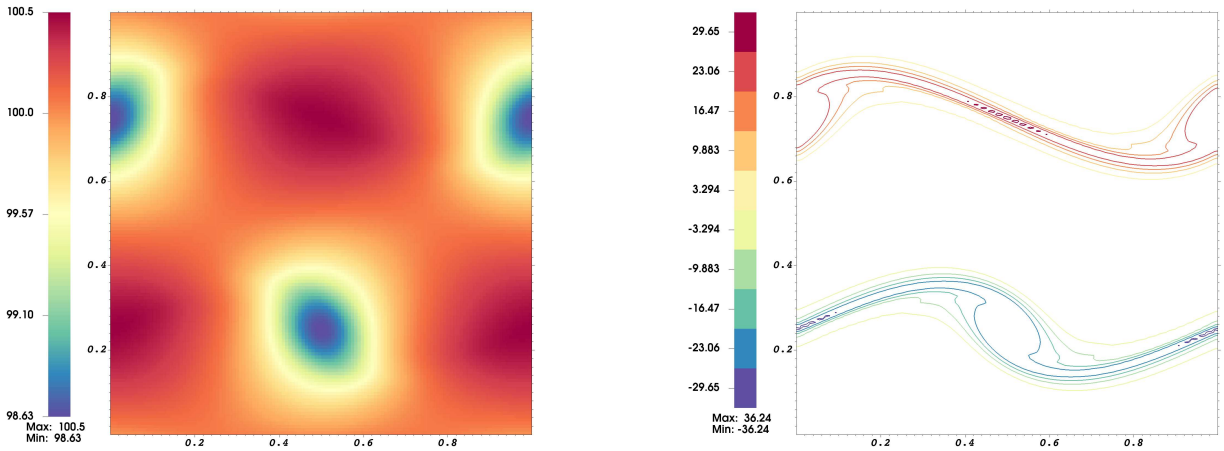
The convergence rate of the solution variables is less than two at coarser discretizations before reaching second order between $N = 128$ and $N = 256$. This lag is due to under-resolution of the vortices at lower discretizations. Once a fine enough discretization is reached the effects of viscosity are sufficiently resolved and second order is achieved. These errors and convergence are reported in Table 2.4–Table 2.6 for the case with $\mu = 1 \times 10^{-4}$.

The effect of varying viscosity is apparent by comparing the vorticity contours. Increasing the viscosity dissipates the velocity, resulting in the decrease of vorticity magnitude and the dissipation of the vorticity contours as shown in Figure 2.5–Figure 2.7. This is consistent with the behavior of viscous fluids, confirming that the viscous all-speed system captures the expected physical behavior.



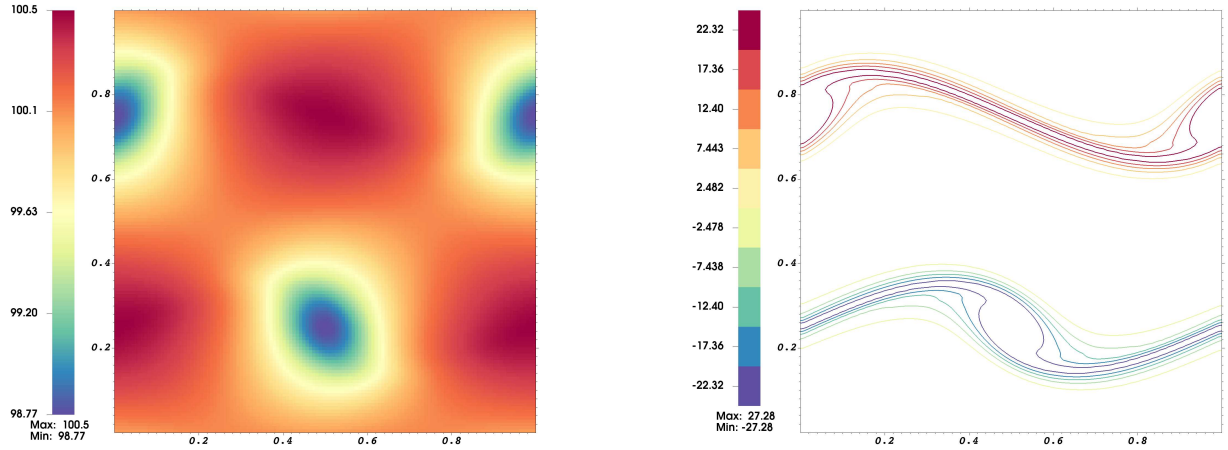
(a) $p(t = 0)$ **(b)** $\omega(t = 0)$

Figure 2.4: Initial pressure and vorticity for the shear layer.



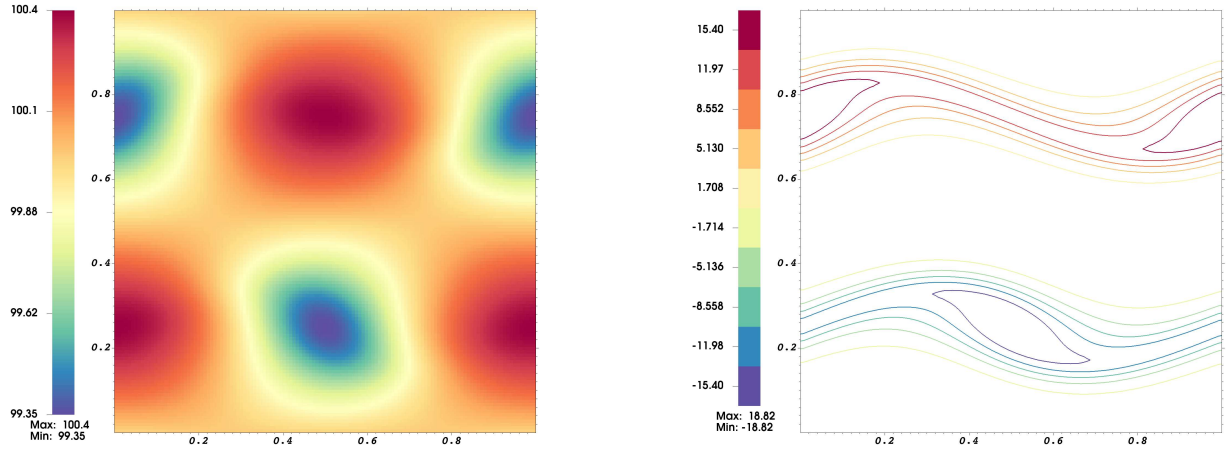
(a) $p(t = 0.75)$ **(b)** $\omega(t = 0.75)$

Figure 2.5: Pressure and vorticity for the shear layer at $t = 0.75$ and $\mu = 0$.



(a) $p(t = 0.75)$ **(b)** $\omega(t = 0.75)$

Figure 2.6: Pressure and vorticity for the shear layer at $t = 0.75$ and $\mu = 1 \times 10^{-4}$.



(a) $p(t = 0.75)$ **(b)** $\omega(t = 0.75)$

Figure 2.7: Pressure and vorticity for the shear layer at $t = 0.75$ and $\mu = 1 \times 10^{-3}$.

Table 2.4: The L_∞ values and convergence rates for the shear layer test at $\mu = 1\times10^{-4}$ showing $\mathcal{O}(h^2)$ order convergence.

	$N = 32$	Rate	$N = 64$	Rate	$N = 128$	Rate	$N = 256$
ρ	3.262E-03	0.713	1.991E-03	1.679	6.218E-04	1.966	1.591E-04
u	7.157E-01	1.348	2.812E-01	1.485	1.005E-01	1.899	2.694E-02
v	3.506E-01	0.731	2.112E-01	1.531	7.308E-02	1.878	1.988E-02
p	4.556E-01	0.714	2.778E-01	1.679	8.674E-02	1.966	2.220E-02
u_d	7.161E-01	1.348	2.814E-01	1.486	1.005E-01	1.899	2.694E-02
v_d	3.509E-01	0.732	2.113E-01	1.531	7.313E-02	1.880	1.987E-02

Table 2.5: The L_1 values and convergence rates for the shear layer test at $\mu = 1\times10^{-4}$ showing $\mathcal{O}(h^2)$ order convergence.

	$N = 32$	Rate	$N = 64$	Rate	$N = 128$	Rate	$N = 256$
ρ	5.707E-04	1.352	2.235E-04	1.820	6.329E-05	1.973	1.612E-05
u	1.078E-01	1.696	3.325E-02	1.909	8.853E-03	2.026	2.174E-03
v	1.054E-01	1.604	3.468E-02	1.900	9.293E-03	2.020	2.291E-03
p	7.986E-02	1.353	3.126E-02	1.821	8.850E-03	1.973	2.254E-03
u_d	1.077E-01	1.696	3.324E-02	1.909	8.849E-03	2.026	2.173E-03
v_d	1.054E-01	1.604	3.467E-02	1.899	9.296E-03	2.022	2.288E-03

Table 2.6: The L_2 values and convergence rates for the shear layer test at $\mu = 1\times10^{-4}$ showing $\mathcal{O}(h^2)$ order convergence.

	32	Rate	64	Rate	128	Rate	256
ρ	8.298E-04	1.113	3.837E-04	1.715	1.169E-04	1.965	2.993E-05
u	1.814E-01	1.697	5.595E-02	1.870	1.531E-02	2.009	3.805E-03
v	1.279E-01	1.282	5.261E-02	1.767	1.546E-02	1.981	3.915E-03
p	1.160E-01	1.114	5.360E-02	1.716	1.632E-02	1.964	4.182E-03
u_d	1.813E-01	1.696	5.595E-02	1.870	1.531E-02	2.009	3.804E-03
v_d	1.280E-01	1.283	5.261E-02	1.766	1.546E-02	1.981	3.916E-03

Chapter 3

Two-Fluid Methods

This chapter describes the numerical representation of the interface geometry using the level-set (LS) method and a novel method to correct the volume of each fluid. The two fluids are separated by an interface that is evolved in time according to this method, allowing the shape of the interface to change over time with the flow. Without a way to track the interface between the fluids only a single fluid or a diffuse mixture of fluids can be represented.

The interface is represented implicitly as the level-set of the signed distance function. Representing the interface in space and evolving it over time requires a few methods. The evolution process changes the position and geometry with the fluid velocity using an advection equation. Marching methods are used to compute the signed distance function in space in a narrow band around the LS. Finally, a correction is applied to the LS for thermodynamic consistency.

A correction method is necessary because numerical error in the evaluation and evolution of the level-set can, and often does, result in the volume of a fluid changing over time. This is unphysical behavior for nearly incompressible fluids moving at low Mach numbers, so the volume of the fluid must be constrained with a correction. A novel volume correction method that utilizes a conservative density update from the level-set, a pressure equation of state, and the level-set evolution is developed to satisfy this constraint. This chapter is organized in the following order:

1. A description of the level-set method and the signed distance function,
2. Interpolation and root finding methods important to the numerical evaluation of the signed distance function,
3. The description of marching methods to evaluate the signed distance function in space,
4. The evolution of the level-set in time,
5. The novel volume correction method to the level-set.

3.1 Two-Fluid Governing Equations

For a two-fluid system, the governing equations from Chapter 2 are slightly modified and the evolution equation for ψ is included. For the two-fluid system, density is a double valued quantity, meaning that two densities are defined over the domain, one for each fluid. This is the multifluid description. The equation for the density of each fluid is listed separately from the rest of the governing equations because it is not part of the ARK time integration. A separate update is made for the density because the evolution depends on the position of the level-set and must incorporate time varying geometries in the flux calculation to evolve the density conservatively. This topic is discussed in detail in Section 3.6.2.

For the weakly compressible multifluid algorithm, only the inviscid system is used in this research. The inviscid system is sufficient to demonstrate the novel thermodynamically consistent level-set method as viscous effects do not contribute to the method. The governing equations for the inviscid two-fluid system are

$$\frac{\partial}{\partial t}\rho^j = -\boldsymbol{\nabla}\cdot\left(\rho^j\boldsymbol{u}\right), \tag{3.1}$$

$$\frac{\partial}{\partial t}\begin{bmatrix}\boldsymbol{u}\\ \boldsymbol{u}_p\\ p\\ \psi\end{bmatrix} = \begin{bmatrix}-(\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}\\ \mathbb{P}\left(\frac{1}{\rho}\boldsymbol{\nabla}p + (\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}\right) - (\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u}\\ -\boldsymbol{u}\cdot\boldsymbol{\nabla}p\\ -s\end{bmatrix} + \begin{bmatrix}-\frac{1}{\rho}\boldsymbol{\nabla}p\\ -\frac{1}{\rho}\boldsymbol{\nabla}p\\ -K\boldsymbol{\nabla}\cdot\boldsymbol{u}_p\\ 0\end{bmatrix}, \tag{3.2}$$

with the superscript j in the density evolution denoting an individual fluid, and the first term on the right hand side representing explicit flux terms in ARK and the second term being implicit ARK fluxes, as before. In the evolution of $\boldsymbol{u}$, $\boldsymbol{u}_p$, and p, the density and bulk modulus must be single valued and defined over the whole domain to obtain a unique solution. This is done by using a

weighted average of the two fluid densities and a weighted harmonic average of the bulk moduli,

$$\rho(\boldsymbol{x},t) = \rho^a(\boldsymbol{x},0)\kappa^a(\boldsymbol{x},t) + \rho^b(\boldsymbol{x},0)(1-\kappa^a(\boldsymbol{x},t)), \tag{3.3}$$

$$K(\boldsymbol{x},t) = \left(\frac{\kappa^a(\boldsymbol{x},t)}{K^a} + \frac{(1-\kappa^a(\boldsymbol{x},t))}{K^b}\right)^{-1}, \tag{3.4}$$

where the a and b superscripts denote the fluid, and κ^a is the volume fraction of fluid a and is explained in detail in Section 3.6.1.

3.2 The Level-Set Method

The level-set (LS) method is the means to represent and evolve the interface between the two fluidsand distinguishes fluids in the multifluid method. The LSM captures an interface, Γ, as the zero contour of some arbitrary function ϕ. The simple requirements placed on ϕ are that it is positive on one side of the interface, Ω^a, and negative on the other side of the interface, Ω^b.

Though any smooth function (ϕ) can be used to represent the level-set, it is convenient to use the signed distance function, ψ. The signed distance function is attractive because it is smooth for a smooth interface, and $||\boldsymbol{\nabla}\psi|| = 1$ everywhere. Defining $\Gamma = \{\boldsymbol{x} : \psi(\boldsymbol{x}) = 0\}$ as the level-set of ψ and $\boldsymbol{x}_\Gamma \in \Gamma$, then ψ is

$$\psi(\boldsymbol{x},t) = \begin{cases} \min|\boldsymbol{x}-\boldsymbol{x}_\Gamma| & \Omega^a \\ 0 & \Gamma \\ -\min|\boldsymbol{x}-\boldsymbol{x}_\Gamma| & \Omega^b \end{cases}, \tag{3.5}$$

with the sign of ψ being arbitrarily assigned to one of the fluids, Ω^a or Ω^b and Γ being the interface between the fluids. The unit normal vector and the curvature of Γ are useful in the numerical

evaluation of ψ and are defined, respectively, as

$$\boldsymbol{n} = \boldsymbol{\nabla}\psi, \tag{3.6}$$

$$\mathcal{C} = \boldsymbol{\nabla} \cdot \boldsymbol{n} = \Delta\psi. \tag{3.7}$$

Initialization and marching methods are utilized to compute the value of ψ in space and are described in Section 3.4. Once ψ is defined over a set of points in the domain, the LS can be evolved using the evolution method described in Section 3.5. At the end of a time step, the signed distance function is altered to correct the position of the LS using the method in Section 3.6.

The change in the LS over time is found with the chain rule

$$\frac{\mathrm{d}}{\mathrm{d}t}(\psi)\bigg|_{\psi=0} = \left(\frac{\partial\psi}{\partial t} + \frac{\partial\psi}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{\partial\psi}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t} + \frac{\partial\psi}{\partial z}\frac{\mathrm{d}z}{\mathrm{d}t}\right)\bigg|_{\psi=0} = 0. \tag{3.8}$$

Noting the time derivatives of position (x, y, z) at the interface is the fluid velocity $\boldsymbol{u}$, the above equation can also be written as:

$$\frac{\partial\psi}{\partial t}\bigg|_{\Gamma} + (\boldsymbol{u} \cdot \boldsymbol{\nabla}\psi)|_{\Gamma} = 0. \tag{3.9}$$

3.3 Interpolation and Root Finding

Root finding and interpolation are utilized both in the ψ re-initialization procedure and in the evolution of ψ. For convenience, these operators are introduced first. Root finding is done from some cell centered point $\boldsymbol{x_i}$ in the direction of $\boldsymbol{\nabla}\psi_{\boldsymbol{i}}$. This gives a point along that ray as

$$\boldsymbol{p}(d, \boldsymbol{\nabla}\psi_{\boldsymbol{i}}) = \left(\boldsymbol{i} + \frac{1}{2}\Upsilon\right) h - dq_{\boldsymbol{i}}\frac{\boldsymbol{\nabla}\psi_{\boldsymbol{i}}}{||\boldsymbol{\nabla}\psi_{\boldsymbol{i}}||} \in \mathbb{R}^{\mathrm{D}}, \tag{3.10}$$

with $q_{\boldsymbol{i}}$ being the sign of ψ at $\boldsymbol{i}$, and d being the distance from $\boldsymbol{x_i}$. This turns a root finding problem in a $\mathbb{R}^{\mathrm{D}}$ to a problem in $\mathbb{R}$ for the root finding algorithm. Next, a method is needed to evaluate ψ at $\boldsymbol{p}(d, \boldsymbol{\nabla}\psi_{\boldsymbol{i}})$ for the root finder, this is done by Lagrangian interpolation. The basis polynomials

of the Lagrangian interpolation are formed by the cell centered values of ψ that bound the point $\boldsymbol{p}(d, \boldsymbol{\nabla}\psi)$, thus the nodes of the Lagrangian interpolation are

$$\boldsymbol{L} = \{\boldsymbol{i}' : \left| \left(\boldsymbol{i}' + \frac{1}{2}\Upsilon \right) - \boldsymbol{p}(d, \boldsymbol{\nabla}\psi_{\boldsymbol{i}}) \right|_{\infty} < 1\}. \tag{3.11}$$

where $\boldsymbol{i}'$ is a cell index. This gives a second order accurate, piecewise continuous interpolation of ψ. During the reinitialization method it is also necessary to interpolate $\boldsymbol{\nabla}\psi$. This interpolation is done using the same Lagrange interpolation strategy. The Lagrangian interpolation of a value π at a point $\boldsymbol{x}$ is denoted

$$\pi_{\boldsymbol{x}} = \mathcal{Q}(\pi, \boldsymbol{x}). \tag{3.12}$$

Brent's root solver, $\mathcal{D}$, is employed as the root finding algorithm determining the root d in (3.10) that evaluates to $\psi = 0$ using Lagrangian interpolation. This gives the value of ψ at the center of cell $\boldsymbol{i}$ as the distance d determined as the root by Brent's root solver, denoted

$$\psi = \mathcal{D}(\boldsymbol{\nabla}\psi_{\boldsymbol{i}}) \tag{3.13}$$

$$= d : (\mathcal{Q}(\psi, \boldsymbol{p}(d, \boldsymbol{\nabla}\psi_{\boldsymbol{i}})) = 0)\,. \tag{3.14}$$

It is also useful in the re-initialization methods to use the root point $\boldsymbol{p}(d, \boldsymbol{\nabla}\psi)$ denoted

$$\boldsymbol{x} = \mathcal{R}(\boldsymbol{\nabla}\psi_{\boldsymbol{i}}) \tag{3.15}$$

$$= \boldsymbol{p}(\psi, \boldsymbol{\nabla}\psi_{\boldsymbol{i}}) : (\mathcal{Q}(\psi, \boldsymbol{p}) = 0)\,. \tag{3.16}$$

3.4 Marching Methods

Analytical solutions to the signed distance functions are rare so numerical methods are necessary to compute it in space. The procedure for evaluating ψ in space consists of two steps: re-initialization and extension. Due to numerical errors in evolution the implicit function may

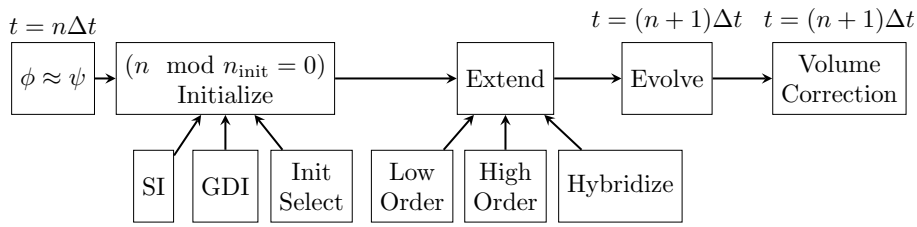
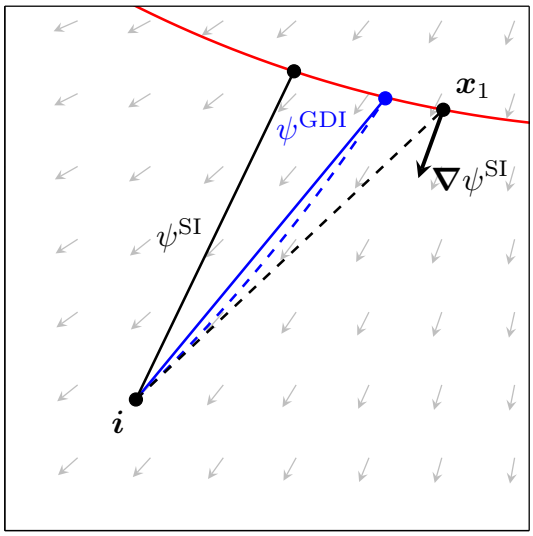


Figure 3.1: The steps of the complete marching method taking an implicit function ϕ that may or may not be ψ, converting it to ψ, extending and evolving it to the next time step. Note that initialization is only performed every select number of time steps, n_{init}.

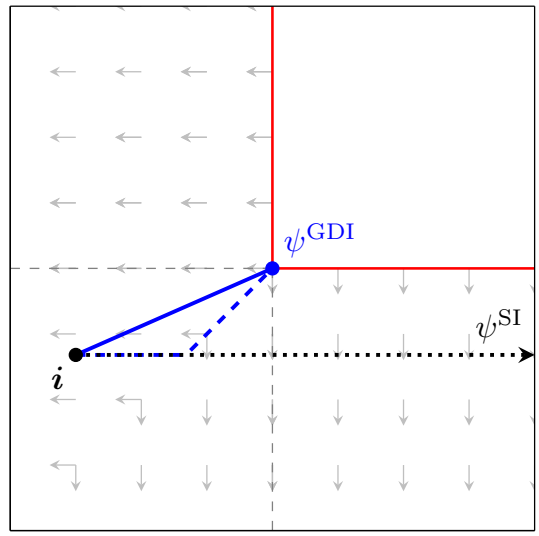
stray from the true signed distance function, the re-initialization method reconstructs an approximation of ψ from an implicit function field. Two methods for re-initialization are used, the gradient descent initialization (GDI) method for initialization around sharp corners and the Schwartz initialization (SI) around smooth regions in Γ. Re-initialization is only necessary every few time steps with the interval chosen by the user. The next step to compute ψ in space is an extension method which expands the region where the signed distance function outward from the level-set. Extension methods include a low order estimate, a higher order estimate, and a hybridization between the two based on the curvature of the signed distance function. This is necessary because the numerical evolution procedure consumes values of ψ on the outer region where it is defined so it must be refreshed. After these steps are completed as necessary, ψ is evolved in time with the evolution procedure. After the evolution, the position of the level-set is corrected so that the volume of each fluid is thermodynamically consistent between pressure and volume. The order of these operations within a single time step is outlined in Figure 3.1.

3.4.1 Initialization

The initialization process transforms an arbitrary implicit function ϕ to the signed distance function ψ. This step is necessary because the implicit representation of the level-set may drift from ψ to ϕ due to numerical errors in the extension and evolution procedures. The signed distance function is generated around the zero level-set up to a small distance of $\epsilon = 2\sqrt{2}h$ away from the level-set, initializing a sufficient band of cells to bootstrap the marching method. This initialization distance ensures at least 2 cells will be initialized in any direction from the level-set. From this



(a) Diagram of the Schwartz and Gradient Descent initialization methods around a smooth level-set.



(b) Diagram of the Schwartz and Gradient Descent initialization methods around a sharp corner in the level-set.

Figure 3.2: Comparison of initialization methods around smooth and sharp interfaces.

initial band, extension methods then expand the region where ψ is defined, which is explained in Section 3.4.2.

Two methods for initialization are shown. The first is the method developed by Schwartz et. al. [53], termed Schwartz Initialization (SI), and the second is a new method termed the gradient descent initialization (GDI). The SI method root finds in the direction of $\nabla\phi_{\boldsymbol{i}}$ to initialize ψ. The GDI method constructs an integral curve from $\boldsymbol{i}$ to Γ to initialize ψ which is useful around sharp corners and high curvature regions. The differences between the SI method and the GDI method are shown graphically bellow in Figure 3.2a on a smooth level-set and around a sharp corner in Figure 3.2b.

Schwartz Initialization

The Schwartz [53] method is used to initialize ψ around smooth regions of Γ. Let $G(\phi)$ indicate the standard second-order centered-stencil finite-difference approximation of the gradient, having $2\mathrm{D}+1$ points. Here ϕ is a general implicit representation of the zero level-set, and may or may not

be equal to ψ. The SI method begins by computing the gradient at the cell $\boldsymbol{i}$ as

$$\widehat{\boldsymbol{\nabla}\phi}_{\boldsymbol{i}} = \frac{G(\phi)_{\boldsymbol{i}}}{||G(\phi)_{\boldsymbol{i}}||}. \tag{3.17}$$

Then root finding is done to find the intersection point of the gradient $\widehat{\boldsymbol{\nabla}\phi}$ with Γ

$$\boldsymbol{x}_1 = \mathcal{R}(\boldsymbol{\nabla}\psi_{\boldsymbol{i}}), \tag{3.18}$$

the gradient is approximated at the same location giving the SI gradient,

$$\boldsymbol{\nabla}\psi_{\boldsymbol{i}}^{\text{SI}} = \mathcal{Q}(\boldsymbol{\nabla}\phi, \boldsymbol{x}_1). \tag{3.19}$$

The value ψ^{SI} is determined by root finding in the $\boldsymbol{\nabla}\psi_{\boldsymbol{i}}^{\text{SI}}$ direction,

$$\psi^{\text{SI}} = \mathcal{D}(\boldsymbol{\nabla}\psi_{\boldsymbol{i}}^{\text{SI}}). \tag{3.20}$$

Gradient Descent Initialization

The gradient descent initialization (GDI) method was created to handle initialization around sharp corners in the zero level-set of ψ where $\boldsymbol{\nabla}\phi$ is not unique. In short, the gradient descent method is able to initialize ψ because it utilizes an integral curve to create a unique displacement vector. This is opposed to the Schwartz method which interpolates $\boldsymbol{\nabla}\phi$ at the intersection of the zero level-set with $\boldsymbol{\nabla}\phi_{\boldsymbol{i}}$, where $\boldsymbol{\nabla}\phi_{\Gamma}$ may not be unique or when a root may not exist in the $\boldsymbol{\nabla}\phi_{\boldsymbol{i}}$ direction.

The GDI method begins at cell center of $\boldsymbol{i}$ and constructs a point $\boldsymbol{x}$ in the direction of the gradient distance h away

$$\boldsymbol{x} = h\left(\boldsymbol{i} + \frac{1}{2}\Upsilon\right) - q_{\boldsymbol{i}}\frac{G(\phi)_{\boldsymbol{i}}}{||G(\phi)_{\boldsymbol{i}}||}h, \tag{3.21}$$

where $q_{\boldsymbol{i}}$ is the sign of ϕ at cell $\boldsymbol{i}$. Then the value of the implicit function ϕ is interpolated at $\boldsymbol{x}$,

$$\phi_{\boldsymbol{x}} = \mathcal{Q}(\phi, \boldsymbol{x}). \tag{3.22}$$

Then check if the $\phi_{\boldsymbol{x}}\phi_{\boldsymbol{i}} < 0$. If this is true, then skip to Eqs. 3.26 – 3.27. Otherwise, continue the evaluation by reevaluating the gradient along the gradient again,

$$\boldsymbol{\nabla}\phi_{\boldsymbol{x}} = \mathcal{Q}(\boldsymbol{\nabla}\phi, \boldsymbol{x}) \tag{3.23}$$

$$\boldsymbol{x} = \boldsymbol{x} - q_{\boldsymbol{i}}\frac{\boldsymbol{\nabla}\phi_{\boldsymbol{x}}}{||\boldsymbol{\nabla}\phi_{\boldsymbol{x}}||}h. \tag{3.24}$$

Repeat (3.23)–(3.24) until the condition $\phi_{\boldsymbol{x}}\phi_{\boldsymbol{i}} < 0$ is met, the point that satisfies this condition is called $\boldsymbol{x}_{\text{GDI}}$. This point may not lay on Γ but is on the other side of Γ from $\boldsymbol{i}$. Finally, compute $\boldsymbol{\nabla}\psi_{\boldsymbol{i}}^{\text{GDI}}$ and $\psi_{\boldsymbol{i}}^{\text{GDI}}$ as

$$\boldsymbol{y} = \left(\boldsymbol{i} + \frac{1}{2}\Upsilon\right)h - \boldsymbol{x}_{\text{GDI}}, \tag{3.25}$$

$$\boldsymbol{\nabla}\psi_{\boldsymbol{i}}^{\text{GDI}} = \frac{\boldsymbol{y}}{||\boldsymbol{y}||}, \tag{3.26}$$

$$\psi_{\boldsymbol{i}}^{\text{GDI}} = \mathcal{D}(\boldsymbol{\nabla}\psi_{\boldsymbol{i}}^{\text{GDI}}). \tag{3.27}$$

Choosing the Initialization Solution

When initializing a cell, the SI and GDI solutions are computed first. Then choose the solution for cell $\boldsymbol{i}$ as the solution with the lowest magnitude in that cell:

$$(\psi_{\boldsymbol{i}}, \boldsymbol{\nabla}\psi_{\boldsymbol{i}}) = \begin{cases} \psi_{\boldsymbol{i}}^{\text{GDI}}, \boldsymbol{\nabla}\psi_{\boldsymbol{i}}^{\text{GDI}} & |\psi_{\boldsymbol{i}}^{\text{GDI}}| < |\psi_{\boldsymbol{i}}^{\text{SI}}| \\ \psi_{\boldsymbol{i}}^{\text{SI}}, \boldsymbol{\nabla}\psi_{\boldsymbol{i}}^{\text{SI}} & \text{otherwise} \end{cases}, \tag{3.28}$$

where ψ^{GDI} and ψ^{SI} are the gradient descent and Schwartz initialization solutions, respectively.

3.4.2 Extension

The extension methods expand the region where ψ is defined, called Ω^{valid}. Expanding Ω^{valid} allows ψ to be evolved multiple times before reinitialization and extending again, reducing the overall computational expense of computing ψ. Extension is best suited to this task because it evaluates the Eikonal equation that represents $\boldsymbol{\nabla}\psi$ over the entire domain and is insensitive to distance from the level-set. The extension method used is a modification of Kim's Global Marching Method [54] as described in Schwartz [53]. It involves a first-order least-squares approximation, a second order approximation, and a hybridization of the two.

In extension, ψ and $\boldsymbol{\nabla}\psi$ are extended iteratively to some distance R starting from the set of ψ values from the re-initialization method. The extension distance at each iteration is $r = \epsilon + n\sigma h$, where $\sigma = 1/(2\sqrt{5})$ and n is the extension iteration number. At the beginning of each iteration, define Ω^{ν} as the set of all cells surrounding Ω^{valid}, given as

$$\Omega^{\nu} = \bigcup_{\boldsymbol{e}^d : 1 \leq d \leq \mathrm{D}} (\Omega^{\text{valid}} + \boldsymbol{e}^d) - \Omega^{\text{valid}}, \tag{3.29}$$

with $\boldsymbol{e}^d$ being the unit vector in the d direction. An illustratino of Ω^{valid} and Ω^{ν} are shown in Figure 3.3 by the blue and green regions, respectively. The signed distance and its gradient are extended into Ω^{ν} from values in Ω^{valid} if possible and incorporated into Ω^{valid} and removed from Ω^{ν} at the end of the iteration. Repeat the evaluation on the modified Ω^{ν} until ψ and $\boldsymbol{\nabla}\psi$ are not extended into any cells or Ω^{ν} is empty. Then increase r and repeat the process until $r > R$.

Low-Order Approximation

It also provides a stable solution in regions of high curvature around the level-set, where the high-order solution would lead to instabilities.

A low-order solution is used as an initial guess in computing the high-order solution. The low-order approximation is denoted by

$$(\psi^L_{\boldsymbol{i}}, \boldsymbol{\nabla}\psi^L_{\boldsymbol{i}}) = \mathscr{E}^L(\psi, h, \Omega^{\text{valid}}, \boldsymbol{i}), \tag{3.30}$$

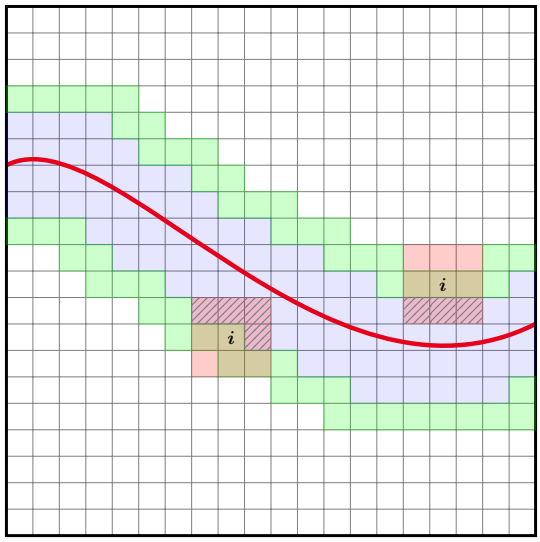


Figure 3.3: Diagram of cell sets in low-order extension, Ω^{valid} is the blue cells, Ω^{ν} is the green cells. Each low-order evaluation location, $\boldsymbol{i}$, has red cells, $\boldsymbol{U}$, and $\boldsymbol{U} \cap \Omega^{\text{valid}}$, the hashed cells.

and is based on a least-squares solution of ψ and $\boldsymbol{\nabla}\psi$ from all cells in Ω^{valid} adjacent to cell $\boldsymbol{i}$. At each $\boldsymbol{i} \in \Omega^{\nu}$ some sets and values must be defined to compute the low-order evaluation,

$$\boldsymbol{U} = \{\boldsymbol{j} : |\boldsymbol{j} - \boldsymbol{i}|_{\infty} \leq 1\} \tag{3.31}$$

$$\boldsymbol{P} = \{\boldsymbol{j} - \boldsymbol{i} : \boldsymbol{j} \in (\boldsymbol{U} \cap \Omega^{\text{valid}})\}, \tag{3.32}$$

$$\Psi = (\psi_{\boldsymbol{i}+\boldsymbol{p}_1},\ \psi_{\boldsymbol{i}+\boldsymbol{p}_2},\ \ldots,\ \psi_{\boldsymbol{i}+\boldsymbol{p}_n})^T, \tag{3.33}$$

$$A = (\boldsymbol{p}_1,\ \boldsymbol{p}_2,\ \ldots,\ \boldsymbol{p}_n)^T, \tag{3.34}$$

$$\Upsilon = (1,\ 1,\ \ldots,\ 1)^T, \tag{3.35}$$

where $\boldsymbol{j} \in \mathbb{Z}^D$ is a cell position vector, $\boldsymbol{U}$ is all cells immediately adjacent to $\boldsymbol{i}$ including $\boldsymbol{i}$, $\boldsymbol{P}$ is the set of displacement vectors between $\boldsymbol{i}$ and the intersection of $\boldsymbol{U}$ with Ω^{valid}, $\boldsymbol{p}$ is a vector in $\boldsymbol{P}$, and n is the number of members in $\boldsymbol{P}$. These sets are displayed graphically in Figure 3.3.

The relationship between ψ and $\boldsymbol{\nabla}\psi$ at points $\boldsymbol{i}$ and $\boldsymbol{p}$ is

$$\frac{1}{h}\left(\psi_{\boldsymbol{i}} - \psi_{\boldsymbol{p}}\right) = -\boldsymbol{p} \cdot \boldsymbol{\nabla}\psi_{\boldsymbol{i}} + \mathcal{O}(h). \tag{3.36}$$

In the extension, $\psi_{\boldsymbol{i}}$ and $\boldsymbol{\nabla}\psi_{\boldsymbol{i}}$ are unknown so they are replaced by the estimates $\tilde{\psi} \approx \psi_{\boldsymbol{i}}$, $\boldsymbol{v} \approx \boldsymbol{\nabla}\psi_{\boldsymbol{i}}$. This relationship is posed for all $\boldsymbol{p} \in \boldsymbol{P}$ and rearranged as a least squares problem,

$$A\boldsymbol{v} = -\frac{1}{h}\left(\tilde{\psi}\Upsilon - \Psi\right), \tag{3.37}$$

to give $\boldsymbol{v}$ as the least squares solution

$$\boldsymbol{v} = -\left(A^T A\right)^{-1} A^T \frac{1}{h}\left(\tilde{\psi}\Upsilon - \Psi\right) \tag{3.38}$$

$$= \frac{\tilde{\psi} - \bar{\psi}}{l}\boldsymbol{n} - \left(\boldsymbol{\omega}_2 - (\boldsymbol{\omega}_2 \cdot \boldsymbol{n})\boldsymbol{n}\right). \tag{3.39}$$

The $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ terms in (3.39) are found by solving their own least squares problems and then used to compute the other terms,

$$A\boldsymbol{\omega}_1 = -\frac{1}{h}\Upsilon, A\boldsymbol{\omega}_2 = -\frac{1}{h}\Psi, \tag{3.40}$$

$$l = \frac{1}{||\boldsymbol{\omega}_1||}, \tag{3.41}$$

$$\boldsymbol{n} = \boldsymbol{\omega}_1 l, \tag{3.42}$$

$$\bar{\psi} = (\boldsymbol{\omega}_1 \cdot \boldsymbol{\omega}_2)l^2 = \psi_{\boldsymbol{i}-l\boldsymbol{n}} + \mathcal{O}(h^2). \tag{3.43}$$

Enforcing the $||\boldsymbol{\nabla}\psi|| = 1$ property of the signed distance function to Eq. 3.39 produces the quadratic equations

$$\left(\tilde{\psi} - \bar{\psi}\right)^2 - l^2\,||\boldsymbol{\omega}_2 - (\boldsymbol{\omega}_2 \cdot \boldsymbol{n})\boldsymbol{n}||^2 = l^2 \tag{3.44}$$

that can be solved for $\tilde{\psi}$ as an estimate to $\psi_{\boldsymbol{i}}$. If $\boldsymbol{\omega}_1 = \boldsymbol{0}$ then the lower order solution at cell $\boldsymbol{i}$ is not defined on this iteration of the marching method. However, the solution at $\boldsymbol{i}$ may be attempted again on the next iteration.

The low order $\psi_{\boldsymbol{i}}$ solution is taken as one of the roots of Eq. 3.44. There are two conditions for these roots. If there are two real roots, then take the root where $|\tilde{\psi}| > |\bar{\psi}|$, otherwise set $\tilde{\psi} = \bar{\psi}$:

$$\{\psi_a, \psi_b\} = \text{roots}\left((\psi_{\boldsymbol{i}} - \bar{\psi})^2 + l^2||(\boldsymbol{\omega}_2 - (\boldsymbol{\omega}_2 \cdot \boldsymbol{n})\boldsymbol{n})||^2 - l^2 = 0\right), \tag{3.45}$$

$$\mathscr{L}(\psi, \boldsymbol{i}, \boldsymbol{P}) = \begin{cases} \max\left(|\psi_a|, |\psi_b|\right) & \psi_a, \psi_b \in \mathbb{R} \\ \bar{\psi} & \text{otherwise} \end{cases}. \tag{3.46}$$

In the condition where the curvature $K < 0$ at cell $\boldsymbol{i}$, $\boldsymbol{\nabla}\psi_{\boldsymbol{i}}$ is not unique, resulting in multiple possible values of ψ. The correct solution is the ψ and $\boldsymbol{\nabla}\psi$ giving $\min|\psi|$ over all $\boldsymbol{\nabla}\psi$ characteristics intersecting $\boldsymbol{i}$. To find this solution, (3.46) is solved over the set $\boldsymbol{B}$, the set of all mutually adjacent pairs of cells (2×2 blocks of cells in 3D) in $\boldsymbol{P}$. Each unordered pair of mutually adjacent cells (or unordered 2×2 blocks in 3D) is described as

$$\boldsymbol{b} = \{\boldsymbol{j}_1, \ldots, \boldsymbol{j}_{2^{\mathrm{D}-1}} : |\boldsymbol{j}_a - \boldsymbol{j}_{b \neq a}|_1 = 1 \wedge |\boldsymbol{j}_a - \boldsymbol{i}|_\infty = 1\}, \tag{3.47}$$

where each $\boldsymbol{b}$ contains $2^{\mathrm{D}-1}$ cell indices. A unique $\boldsymbol{b}$ only depends on its members, not their order (an unordered pair/set), i.e. $\{\boldsymbol{j}_1, \boldsymbol{j}_2\} = \{\boldsymbol{j}_2, \boldsymbol{j}_1\}$ in 2D. The set $\boldsymbol{B}$ is the collection of all $\boldsymbol{b}$ intersecting Ω^{valid}:

$$\boldsymbol{B} = \left\{\boldsymbol{b} - \boldsymbol{i} : \forall \boldsymbol{b} \in \Omega^{\text{valid}}\right\}, \tag{3.48}$$

so that $\boldsymbol{B}$ contains displacement vectors, like $\boldsymbol{P}$. Now the solution for $\psi_{\boldsymbol{i}}^L$ is fully described by Eq. 3.49 for $K \geq 0$ and $K < 0$ cases as:

$$\psi_{\boldsymbol{i}}^L = \begin{cases} q_{\boldsymbol{i}} \min_{\boldsymbol{B}} |\mathscr{L}(\psi, \boldsymbol{i}, \boldsymbol{B})| & \mathcal{C}_{\boldsymbol{i}} < 0 \text{ or } \omega_1 = 0 \\ \mathscr{L}(\psi, \boldsymbol{i}, \boldsymbol{P}) & \mathcal{C}_{\boldsymbol{i}} \geq 0 \text{ and } \omega_1 \neq 0 \end{cases}. \tag{3.49}$$

The low order $\psi_{\boldsymbol{i}}$ solution can then be applied to Eq. 3.39

$$\boldsymbol{\nabla}\psi_{\boldsymbol{i}}^L = \begin{cases} -(A_{\min}A_{\min}^T)^{-1}A_{\min}^T\frac{1}{h}(\psi_{\boldsymbol{i}}^L\Upsilon - \Psi_{\min}) & \mathcal{C}_{\boldsymbol{i}} < 0 \text{ and } \boldsymbol{\omega}_1 \neq \boldsymbol{0} \\ -(AA^T)^{-1}A^T\frac{1}{h}(\psi_{\boldsymbol{i}}^L\Upsilon - \Psi) & \mathcal{C}_{\boldsymbol{i}} \geq 0 \text{ or } \boldsymbol{\omega}_1 = \boldsymbol{0} \end{cases}, \tag{3.50}$$

$$K_{\boldsymbol{i}} = \min_{\boldsymbol{T}} q_{\boldsymbol{i}}(L(\psi))_{\boldsymbol{t}} \tag{3.51}$$

$$\boldsymbol{T} = \left\{\boldsymbol{j} : (|\boldsymbol{j} - \boldsymbol{i}|_\infty \leq 2) \cap \Omega^{\text{valid}}\right\}, \tag{3.52}$$

The curvature, $\mathcal{C}$, is evaluated over $\boldsymbol{T}$, the set of cells up to 2 cells away from $\boldsymbol{i}$, using the standard second-order cell-centered discrete Laplace operator, L, with $2\text{D} + 1$ points. The sets A, Ψ, and $\boldsymbol{\omega}_1$ are defined in the regions where $\mathcal{C} \geq 0$ by (3.34), (3.33), and (3.40), respectively. In regions where $\mathcal{C} < 1$ the sets $A_{\min}$, $\Psi_{\min}$, and $\boldsymbol{\omega}_{1,\min}$ are the corresponding sets giving the minimum value of ψ^L over all members of $\boldsymbol{B}$, $\boldsymbol{b}_{\min}$.

High-Order Approximation

The high-order solution provides a second order accurate estimation of ψ and $\boldsymbol{\nabla}\psi$ using initial data from the low-order solution. This solution is used when the magnitude of the curvature is low enough that instability is not introduced, as determined by the hybridization in Section 3.4.2. The high-order approximation is denoted by

$$(\psi_{\boldsymbol{i}}^H, \boldsymbol{\nabla}\psi_{\boldsymbol{i}}^H) = \mathscr{E}^H(\psi, h, \Omega^{\text{valid}}, \boldsymbol{i}). \tag{3.53}$$

It is a series of quadratic interpolations on $\boldsymbol{\nabla}\psi$ and ψ using the low-order approximation results $\boldsymbol{\nabla}\psi^L$ and ψ^L as initial guesses. For the high-order approximation to exist, the low-order approximation also must exist. There is no additional restriction on the existence of the high-order solution.

To build the high-order approximation at cell $\boldsymbol{i} \in \Omega^\nu$, a quadratic interpolation of ψ and $\boldsymbol{\nabla}\psi$ is made from an interpolation point in an adjacent cell $\boldsymbol{j} \in \Omega^{\text{valid}}$. First, $\bar{\boldsymbol{x}}_{\boldsymbol{i}}$ is established as the

interpolation point which determines $\boldsymbol{j}$,

$$\bar{\boldsymbol{x}}_{\boldsymbol{i}}(\boldsymbol{\nabla}\psi) = \left(\boldsymbol{i} + \frac{1}{2}\Upsilon\right) h - q_{\boldsymbol{i}} \frac{\boldsymbol{\nabla}\psi_{\boldsymbol{i}}}{|(\boldsymbol{\nabla}\psi_{\boldsymbol{i}})_{\max}|} h, \tag{3.54}$$

$$\boldsymbol{j} = \left\lfloor \frac{\bar{\boldsymbol{x}}_{\boldsymbol{i}}}{h} - \frac{1}{2}\left(\Upsilon - \boldsymbol{e}^{d_{\max}}\right) \right\rfloor, \tag{3.55}$$

$$\boldsymbol{y} = \bar{\boldsymbol{x}}_{\boldsymbol{i}} - \left(\boldsymbol{j} + \frac{1}{2}\Upsilon\right) h, \tag{3.56}$$

where $(\boldsymbol{\nabla}\psi_{\boldsymbol{i}})_{\max}$ denotes the component of $\boldsymbol{\nabla}\psi_{\boldsymbol{i}}$ that has the greatest magnitude. Defining $\boldsymbol{j}$ in this fashion eliminates the need for derivatives in the $d_{\max}$ direction because the displacement, $y_{d_{\max}}$, is zero. Cell $\boldsymbol{j}$ is the closest cell in the $d_{\max}$ direction. These points are shown on a sample grid in Figure 3.4.

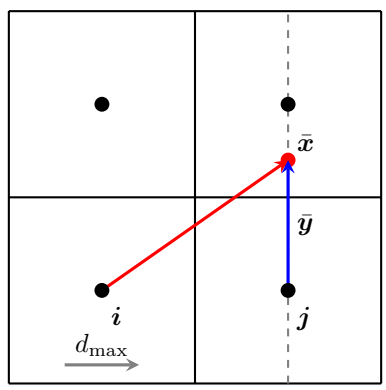


Figure 3.4: Diagram of the higher-order approximation of ψ.

The desired output, π, which can be either ψ or $\boldsymbol{\nabla}\psi$, is then given by

$$Q(\pi, \boldsymbol{y}) = \pi_{\boldsymbol{j}} + \sum_{d \neq d_{\max}} \left(\left.\frac{\partial \pi}{\partial x_d}\right|_{\boldsymbol{j}} y_d + \frac{1}{2} \left.\frac{\partial^2 \pi}{\partial x_d^2}\right|_{\boldsymbol{j}} y_d^2 \right) + \beta, \tag{3.57}$$

where $d_{\max}$ is the component index of $\boldsymbol{\nabla}\psi_{\max}$, and β is a mixed derivative described below. The partial derivatives for the interpolation are computed at the center of cell $\boldsymbol{j}$ using standard second order cell center finite differences. Notice that the sum in Eq. 3.57 is only over the directions that are not $d_{\max}$. This is because the $d \neq d_{\max}$ directions have a stronger effect on the direction of

$\boldsymbol{\nabla}\psi$ than the $d_{\max}$ direction. For example, if $\boldsymbol{\nabla}\psi$ points entirely in the vertical direction then its derivative in the y direction will be zero while the x derivative will contain all the variation in $\boldsymbol{\nabla}\psi$.

The mixed derivative β is only calculated when $\mathrm{D} = 3$ and is approximated as an average of standard second order centered mixed differences around $\boldsymbol{j}$,

$$\beta = \left.\frac{\partial^2 \pi}{\partial x_{d_1} \partial x_{d_2}}\right|_{\boldsymbol{j}} y_{d_1} y_{d_2}, \tag{3.58}$$

$$\left.\frac{\partial^2 \pi}{\partial x_{d_1} \partial x_{d_2}}\right|_{\boldsymbol{j}} \approx \frac{1}{N} \sum (\mathscr{D}^2_{d_1,d_2} \pi)_{\boldsymbol{j}+\boldsymbol{s}/2}, \tag{3.59}$$

$$\boldsymbol{s} = \{\alpha_1 \boldsymbol{e}^{d_1} + \alpha_2 \boldsymbol{e}^{d_2} : \alpha_1 = \pm 1, \alpha_2 = \pm 1\}, \tag{3.60}$$

$$(\mathscr{D}^2_{d_1,d_2} \pi)_{\bar{\boldsymbol{j}}+\boldsymbol{s}/2} = \frac{1}{h^2} \left[\alpha_1 \alpha_2 \left(\pi_{\bar{\boldsymbol{j}}+\alpha_1 \boldsymbol{e}^{d_1}+\alpha_2 \boldsymbol{e}^{d_2}} - \pi_{\bar{\boldsymbol{j}}+\alpha_2 \boldsymbol{e}^{d_2}}\right) - \alpha_1 \alpha_2 \left(\pi_{\bar{\boldsymbol{j}}+\alpha_1 \boldsymbol{e}^{d_1}} - \pi_{\bar{\boldsymbol{j}}}\right)\right], \tag{3.61}$$

with N being the number of terms in the sum for which $(\mathscr{D}^2_{d_1,d_2})$ is defined, that is all of the $\boldsymbol{j} + \boldsymbol{s} \in \Omega^{\text{valid}}$, and the directions $d_1 \neq d_2 \neq d_{\max}$. The $\mathscr{D}$ operator approximates the second order mixed derivative at cell vertices, which are then averaged from all available vertices to get the second order mixed derivative at the $\boldsymbol{j}$ cell center.

The higher-order approximation $\boldsymbol{\nabla}\psi^H$ is computed using two iterations of the quadratic interpolation. The first iterations generates an intermediate estimate of the gradient, $\overline{\boldsymbol{\nabla}\psi}$, from the low-order gradient, $\boldsymbol{\nabla}\psi^L$,

$$\bar{\boldsymbol{x}}_a = \bar{\boldsymbol{x}}_{\boldsymbol{i}}(\boldsymbol{\nabla}\psi^L), \quad \overline{\boldsymbol{\nabla}\psi} = Q(\boldsymbol{\nabla}\psi^L, \bar{\boldsymbol{x}}_a). \tag{3.62}$$

Repeat the interpolation using $\overline{\boldsymbol{\nabla}\psi}$ as the input to get the high-order gradient $\boldsymbol{\nabla}\psi^H$,

$$\bar{\boldsymbol{x}}_b = \bar{\boldsymbol{x}}_{\boldsymbol{i}}\left(\overline{\boldsymbol{\nabla}\psi}\right), \quad \boldsymbol{\nabla}\psi^H_{\boldsymbol{i}} = Q\left(\overline{\boldsymbol{\nabla}\psi}, \bar{\boldsymbol{x}}_b\right). \tag{3.63}$$

Next, approximate the signed distance, $\hat{\psi}$, at $\bar{\boldsymbol{x}}_c$ using ψ^L and $\boldsymbol{\nabla}\psi^H$,

$$\bar{\boldsymbol{x}}_c = \bar{\boldsymbol{x}}_{\boldsymbol{i}}(\boldsymbol{\nabla}\psi^H), \quad \hat{\psi} = Q(\psi^L, \bar{\boldsymbol{x}}_c). \tag{3.64}$$

There is no second interpolation for ψ^H because it would share the same interpolation point, as $\boldsymbol{\nabla}\psi^H$ has not changed. Finally, the distance between $\boldsymbol{i}$ and $\bar{\boldsymbol{x}}_c$ is added to $\hat{\psi}$ because $\hat{\psi}$ is interpolated at the point $\bar{\boldsymbol{x}}_c$,

$$\psi_{\boldsymbol{i}}^H = \hat{\psi} + q_{\boldsymbol{i}} h \frac{||\boldsymbol{\nabla}\psi_{\boldsymbol{i}}^H||}{|(\boldsymbol{\nabla}\psi_{\boldsymbol{i}}^H)_{\max}|}. \tag{3.65}$$

An adjustment is not needed to $\boldsymbol{\nabla}\psi^H$ because it is constant along the ray $\boldsymbol{i} - \bar{\boldsymbol{x}}_c$.

Hybridization

Hybridization is crucial to the stability of the extension, because the high-order method can lead to unstable solutions in time around areas of high curvature of Γ. Accordingly, the low order and high-order solutions are hybridized so that high accuracy is retained in low curvature regions and stability from the low-order method is favored around sharper curves. The hybridization coefficient η gauges the curvature relative to the grid discretization and performs the scaling between low and high-order solutions. The hybridization scheme for the final solution of $\psi_{\boldsymbol{i}}$ and $\boldsymbol{\nabla}\psi_{\boldsymbol{i}}$ is

$$(\psi, \boldsymbol{\nabla}\psi)_{\boldsymbol{i}} = \left((1-\eta)\psi^H + \eta\psi^L,\ (1-\eta^2)\boldsymbol{\nabla}\psi^H + \eta^2\boldsymbol{\nabla}\psi^L\right)_{\boldsymbol{i}} \tag{3.66}$$

$$\eta_{\boldsymbol{i}} = \begin{cases} 1 & h|L(\psi)|_{\max} > 1 \\ h|L(\psi)|_{\max} & \text{otherwise} \end{cases} \tag{3.67}$$

$$|L(\psi)|_{\max} = \max_{\boldsymbol{T}} |(L(\psi))_{\boldsymbol{t}}|, \tag{3.68}$$

where L is the standard 2D$+1$ point discrete Laplace operator and $\boldsymbol{T}$ is the set where the Laplacian is computed as given in Eq. 3.52. The hybridization constant $\eta = h|L(\psi)|_{\max}$ is the maximum change in $\boldsymbol{\nabla}\psi$ relative to the spatial discretization.

3.5 Level-Set Evolution

To evolve the LS in time, ψ must be evolved on its valid domain because the LS is represented implicitly. However, the evolution of the LS given by (3.9) is defined strictly on the LS so it must be extended to the domain Ω. By taking advantage of the $||\boldsymbol{\nabla}\psi|| = 1$ property of the signed distance function, (3.9) can be rewritten as

$$\frac{\partial \psi}{\partial t} + \boldsymbol{u}_\Gamma \cdot \boldsymbol{\nabla}\psi = \frac{\partial \psi}{\partial t} + s = 0, \tag{3.69}$$

$$\boldsymbol{\nabla}\psi \cdot \boldsymbol{\nabla} s = 0 \in \Omega, \tag{3.70}$$

$$s = \boldsymbol{u} \cdot \boldsymbol{\nabla}\psi \in \Gamma. \tag{3.71}$$

The value of s is constant along a ray σ, that is $s(\boldsymbol{x}(\sigma), t) = s(\boldsymbol{x}(\sigma = 0), t)$, where $\frac{\partial \boldsymbol{x}}{\partial \sigma} = \boldsymbol{\nabla}\psi(\sigma, t)$. Therefore extensions of s can be computed in Ω to evolve the value of ψ in the domain, evolving Γ implicitly.

The numerical method of computing s is simple. Begin by finding the intersection of $\boldsymbol{\nabla}\psi_{\boldsymbol{i}}$ with Γ using the root finding described in (3.3),

$$\boldsymbol{x}_\Gamma = \mathcal{R}(\boldsymbol{\nabla}\psi_{\boldsymbol{i}}). \tag{3.72}$$

The velocity is interpolated at this point using the interpolation described by (3.12),

$$\boldsymbol{u}_{\boldsymbol{x}} = \mathcal{Q}(\boldsymbol{u}, \boldsymbol{x}_\Gamma). \tag{3.73}$$

Finally, the extension of s is simply

$$s_{\boldsymbol{i}} = \boldsymbol{u}_{\boldsymbol{x}_\Gamma} \cdot \boldsymbol{\nabla}\psi_{\boldsymbol{i}}. \tag{3.74}$$

With this method to compute s, ψ is integrated in time using the ARK method.

3.6 Volume Correction Methods

This section discusses the LS correction method that enforces thermodynamic consistency between the pressure, volume, and mass of each fluid. This is necessary because of errors in the evolution of the signed distance function. These errors result in a position of the level-set that causes unphysical expansion or compression of fluid volumes. Through the correction method, the position of the level-set is restored so that the volume of each fluid is consistent with its thermodynamic relationship to mass and pressure.

This section begins by describing cell geometries necessary for the computation of the conservative evolution of density and the correction. Next, the cut cell geometries are used to update the density conservatively in time. Finally, the correction method updates the position of the level-set so that the volumes of each fluid are thermodynamically consistent.

3.6.1 Cut Cell Geometries

A few definitions of geometric quantities of the discretized domain required for the density update are described here. The computational domain is defined as Ω, with the domain occupied by fluid phase a denoted Ω^a and its boundary $\partial\Omega^a$. Recalling definitions of control volumes from Section 2.2 and domains of each fluid from Section 3.2, define

$$V^a = \text{vol}(\Omega^a), \tag{3.75}$$

$$V^a_{\boldsymbol{i}} = \Omega^a \cap \Upsilon_{\boldsymbol{i}}, \tag{3.76}$$

$$A^a_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = \partial\Upsilon_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} \cap \Omega^a, \tag{3.77}$$

as the volume of the cell $\boldsymbol{i}$ occupied by fluid phase a and the area of the $\boldsymbol{i} + \frac{1}{2}\boldsymbol{e}^d$ face intersecting fluid a, respectively. Additionally, define the volume fraction and face fraction, respectively, as

$$\kappa_{\boldsymbol{i}} = \frac{|V^a_{\boldsymbol{i}}|}{h^{\mathrm{D}}}, \tag{3.78}$$

$$\alpha^a_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = \frac{A^a_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}}{h^{\mathrm{D}-1}}. \tag{3.79}$$

These are similarly defined for the other fluid phase, b, with $\Upsilon_{\boldsymbol{i}} = V^a_{\boldsymbol{i}} \cup V^b_{\boldsymbol{i}}$ and $\partial\Upsilon_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = A^a_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} \cup A^b_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$.

Introducing a moving interface, i.e. the advecting level-set, creates time dependent geometries that must be accounted for in flux calculations. We extend the volumetric and area quantities defined earlier to the space-time domain by integrating them over time,

$$\bar{\alpha}^a = \frac{1}{\Delta t}\int_{n\Delta t}^{(n+1)\Delta t} \alpha^a(t)\, dt, \tag{3.80}$$

defined similarly for phase b. These space-time quantities are displayed on a 2D control volume integrated in time in Figure 3.5.

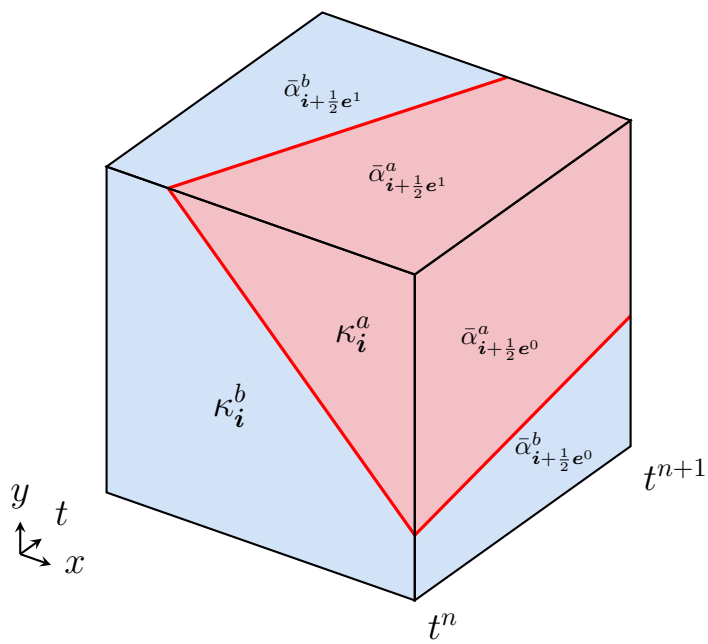


Figure 3.5: Space-time geometries and points on a 2D in space control volume integrated one time step.

The area or volume κ and $\bar{\alpha}$ are computed using the recursive moment calculation described in Schwartz et. al. [55]. This method computes volumes up to three dimensions and an arbitrary order of accuracy using the location of the intersection along one dimensional edges and derivatives of the normal direction of the interface.

3.6.2 Density Update

Here, the conservative update to the density of each fluid is described. A hybridized density update using conservative and non-conservative updates maintains global conservation and avoids the small cell instability. The small cell instability is a well known problem in cut cell and embedded boundary methods [56, 57]. Global conservation is achieved by redistributing mass around the small cells.

The density updated with this method is used in the next section to compute a pressure equation of state that will determine the correction to the LS. In the governing equations, the densities of each fluid are used to provide the density for the evaluation of the flux. Calculations are shown for fluid a but are equivalent for fluid b and are performed on both.

Applying the FV method to the continuity equation with a moving interface, it is assumed that there is no flux across the interface, which is a reasonable assumption in this case because the interface is defined to move with the velocity normal to it. This gives a space-time integral of the continuity equation as:

$$\frac{1}{h^{\mathrm{D}}}\int_{\mathscr{V}_{\boldsymbol{i}}^{a,(n,n+1)}}\left(\frac{\partial\rho}{\partial t}+\boldsymbol{\nabla}\cdot\boldsymbol{F}\right)\,dV\,dt \approx (\kappa\rho)_{\boldsymbol{i}}^{a,n+1}-(\kappa\rho)_{\boldsymbol{i}}^{a,n}+\frac{\Delta t}{h}\sum_{d=0}^{\mathrm{D}-1}\left((\bar{\alpha}^a F_d^a)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}-(\bar{\alpha}^a F_d^a)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}\right), \quad (3.81)$$

where $\mathscr{V}_{\boldsymbol{i}}^{a,(n,n+1)}=\{(\boldsymbol{x},t):\boldsymbol{x}\in V_{\boldsymbol{i}}^a(t),\ n\Delta t\le t\le(n+1)\Delta t\}$ and $\boldsymbol{F}^a=\rho^{a,n}\boldsymbol{u}^{n+1/2}$ with $\boldsymbol{u}^{n+1/2}=\boldsymbol{u}(t=(n+1/2)\Delta t)$ calculated by averaging or Runge-Kutta time interpolation from dense output. The flux divergence term in (3.81) gives the conservative update to ρ^a. Dividing this term by $\kappa^{a,n+1}$, the conservative flux term is denoted

$$(\boldsymbol{\nabla}\cdot\boldsymbol{F}^a)_{\boldsymbol{i}}^C=\frac{1}{\kappa_{\boldsymbol{i}}^{a,n+1}h}\sum_{d=0}^{\mathrm{D}-1}\left((\bar{\alpha}^a F_d^a)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}-(\bar{\alpha}^a F_d^a)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}\right). \quad (3.82)$$

The conservative update for the density in cell $\boldsymbol{i}$ can then be expressed as

$$(\kappa\rho)_{\boldsymbol{i}}^{a,n+1} = (\kappa\rho)_{\boldsymbol{i}}^{a,n} - \Delta t \kappa_{\boldsymbol{i}}^{a,n+1} (\boldsymbol{\nabla} \cdot \boldsymbol{F}^a)_{\boldsymbol{i}}^C. \tag{3.83}$$

This update suffers from the small cell instability, as $\kappa^{a,n+1} \to 0$ then $\rho^{a,n+1} \to \infty$, so this update cannot be applied directly. Instead, an initial hybridized update that is stable but not conservative is made:

$$\tilde{\rho}_{\boldsymbol{i}}^a = \rho_{\boldsymbol{i}}^{a,\text{ref}} - \Delta t \left(\eta_{\boldsymbol{i}} (\overline{\boldsymbol{\nabla} \cdot \boldsymbol{F}^a})_{\boldsymbol{i}}^C + (1 - \eta_{\boldsymbol{i}}) (\boldsymbol{\nabla} \cdot \boldsymbol{F}^a)_{\boldsymbol{i}}^{NC} \right). \tag{3.84}$$

Here $\rho^{a,\text{ref}} = \rho^{a,n}$ is a reference value, $\eta_{\boldsymbol{i}}$ is the hybridization value, $(\boldsymbol{\nabla} \cdot \boldsymbol{F}^a)^{NC}$ is the non-conservative flux of fluid a, and $(\overline{\boldsymbol{\nabla} \cdot \boldsymbol{F}^a})_{\boldsymbol{i}}^C$ is the modified conservative flux. Assuming no flux across the boundary, these fluxes are

$$(\boldsymbol{\nabla} \cdot \boldsymbol{F}^a)^{NC} = \frac{1}{\Delta t} \left(\rho^{a,\text{ref}} - \rho^{a,n} \right) = 0, \tag{3.85}$$

$$(\overline{\boldsymbol{\nabla} \cdot \boldsymbol{F}^a})_{\boldsymbol{i}}^C = \frac{1}{\kappa_{\boldsymbol{i}}^{a,n+1} \Delta t} \left(\kappa_{\boldsymbol{i}}^{a,n+1} \rho_{\boldsymbol{i}}^{a,\text{ref}} - (\kappa\rho)_{\boldsymbol{i}}^{a,n} \right) + \frac{1}{\kappa_{\boldsymbol{i}}^{a,n+1} h} \sum_{d=0}^{\text{D}-1} \left((\bar{\alpha}^a F_d^a)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - (\bar{\alpha}^a F_d^a)_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d} \right). \tag{3.86}$$

Choosing $\eta_{\boldsymbol{i}} = \kappa_{\boldsymbol{i}}^{a,n+1}$ cancels the volume fraction in the denominator, removing the instability as $\kappa^{a,n+1} \to 0$. The reference value is chosen specifically to make the non-conservative flux identically zero. This choice is stable and reflects no compression or expansion of the fluid. Constructing $(\boldsymbol{\nabla} \cdot \boldsymbol{F}^a)^{NC}$ in this way ensures $\rho^{a,\text{ref}}$ is canceled in the hybridized update and preserves the limit $\tilde{\rho}^a = \rho^{a,n}$ as $\eta_{\boldsymbol{i}} \to 0$.

To make the update conservative, the remaining mass left out by the initial hybridized update, δM, must be returned. Subtracting the hybridized update, (3.84), from the conservative update,

(3.83), gives the missing mass,

$$\kappa_{\boldsymbol{i}}^{a,n+1}\rho_{\boldsymbol{i}}^{a,n+1} = \kappa_{\boldsymbol{i}}^{a,n+1}\rho_{\boldsymbol{i}}^{a,\text{ref}} - \Delta t\kappa_{\boldsymbol{i}}^{a,n+1}(\overline{\boldsymbol{\nabla}\cdot\boldsymbol{F}^a})_{\boldsymbol{i}}^{C}, \tag{3.87}$$

$$- \quad \kappa_{\boldsymbol{i}}^{a,n+1}\tilde{\rho}_{\boldsymbol{i}}^{a} = \kappa_{\boldsymbol{i}}^{a,n+1}\rho_{\boldsymbol{i}}^{a,\text{ref}} - \Delta t\kappa_{\boldsymbol{i}}^{a,n+1}\left(\eta_{\boldsymbol{i}}(\overline{\boldsymbol{\nabla}\cdot\boldsymbol{F}^a})_{\boldsymbol{i}}^{C} + (1-\eta_{\boldsymbol{i}})(\boldsymbol{\nabla}\cdot\boldsymbol{F}^a)_{\boldsymbol{i}}^{NC}\right), \tag{3.88}$$

$$\delta M_{\boldsymbol{i}}^{a} = -\Delta t\kappa_{\boldsymbol{i}}^{a,n+1}(1-\eta_{\boldsymbol{i}})\left((\overline{\boldsymbol{\nabla}\cdot\boldsymbol{F}^a})_{\boldsymbol{i}}^{C} - (\boldsymbol{\nabla}\cdot\boldsymbol{F}^a)_{\boldsymbol{i}}^{NC}\right). \tag{3.89}$$

The missing mass is redistributed to the cells surrounding and including $\boldsymbol{i}$ that intersect Ω^a. The redistribution is necessary to spread out the remaining mass, $\delta M_{\boldsymbol{i}}$, to surrounding cells so that ρ^a remains globally conservative. The redistributed is done according to the scheme

$$\rho_{\boldsymbol{i}'}^{a,n+1} = \tilde{\rho}_{\boldsymbol{i}'}^{a} + w_{\boldsymbol{i},\boldsymbol{i}'}^{a}\delta M_{\boldsymbol{i}}^{a}, \tag{3.90}$$

$$w_{\boldsymbol{i},\boldsymbol{i}'}^{a} = \frac{1}{\sum_{\boldsymbol{i}'\in N(\boldsymbol{i})}\kappa_{\boldsymbol{i}'}^{a,n+1}}, \tag{3.91}$$

$$N(\boldsymbol{i}) = \{\boldsymbol{i}' : |\boldsymbol{i}'-\boldsymbol{i}|_\infty \leq 1\}, \tag{3.92}$$

where $\boldsymbol{i}'$ and $\boldsymbol{i}$ are cell indices. The redistribution is conservative as the weights satisfy the identity

$$w_{\boldsymbol{i},\boldsymbol{i}'}^{a} \geq 0, \tag{3.93}$$

$$\sum_{\boldsymbol{i}'\in N(\boldsymbol{i})} w_{\boldsymbol{i},\boldsymbol{i}'}^{a}\kappa_{\boldsymbol{i}'}^{a,n+1} = 1. \tag{3.94}$$

The entire density update is done similarly for fluid b.

3.6.3 Level-Set Correction

This section describes the correction made to the level-set after the density update from the previous section has been made. This correction enforces the thermodynamic consistency between the double valued description of density and the single valued description of pressure in a two-

fluid system. Perturbations in density due to errors in the level-set are transformed into level-set corrections to relax the fluid volumes towards the thermodynamically consistent state.

The approach taken here is to use the conservative density of each phase, computed by the method in the previous section, to compute a pressure for each fluid phase based on equation of state defined by the bulk modulus of the phase. A single pressure for the domain is also computed based the individual fluid pressures. The difference between the pressure of each fluid and the average pressure is translated into a volume difference for each fluid. Then, the volume difference is transformed into a correction to the level-set to adjust the volumes of each phase. Finally, a correction to the all-speed system from the equations of state is described to ensure the pressure in the all-speed system and the pressure from each fluid's equation of state is consistent.

If all fluids in the domain are initially at an equilibrium pressure of $p^{0,0}$ with the constant initial density of a fluid j being $\rho^{j,0}$, then isothermal changes in density of fluid j are equivalent to changes in pressure of the same fluid as described by bulk modulus,

$$p^{\text{EOS},a} = p^{0,0} + K^a \frac{\rho^a - \rho^{a,0}}{\rho^{a,0}}, \tag{3.95}$$

with ρ^a being the conservative density computed with the method in the previous section. This pressure reflects any perturbations in density due to expansion or compression caused by the level-set.

The pressure and volume of fluid j are related by the bulk modulus, K^j. The same relation is used to state a total pressure, p^{EOS}, that is in equilibrium with all fluids. The total pressure represents the single valued pressure state. Normalizing the bulk modulus to the reference pressure $\bar{K}^j = K^j/\hat{p}^{\text{EOS},j}$, gives the relations

$$\frac{\delta V^j}{V^j} = -\frac{1}{\bar{K}^j}\frac{\delta p^{\text{EOS},j}}{\hat{p}^{\text{EOS},j}}, \tag{3.96}$$

$$\frac{\delta V}{V} = -\frac{1}{\bar{K}}\frac{\delta p^{\text{EOS}}}{p^{\text{EOS}}}, \tag{3.97}$$

where V is the total volume of both fluids, $\hat{p}^{\text{EOS},j}$ is some reference pressure for fluid j, δV and δp^{EOS} are arbitrarily small isothermal changes in the total volume and pressure, and δV^j and $\delta p^{\text{EOS},j}$ are arbitrarily small isothermal changes to fluid j volume and pressure relative to the reference. Expressions for $\bar{K}$ and p^{EOS} need to be determined.

The difference between the single fluid pressure and the pressure of fluid j is $\delta p^{\text{EOS},j} = p^{\text{EOS}} - \hat{p}^{\text{EOS},j}$ with the reference pressure p^{EOS}. With these definitions, (3.96) becomes

$$\frac{\delta V^j}{V^j} = \frac{1}{\bar{K}^j} \frac{p^{\text{EOS},j} - p^{\text{EOS}}}{p^{\text{EOS}}}. \tag{3.98}$$

This describes the relationship between fluid j and the single fluid description and is used for the level-set correction.

Next, expressions for the total pressure and bulk modulus are needed. Two constraints are applied to derive these: (i) If the fluids are in equilibrium, then $p^{\text{EOS}} = p^{\text{EOS},j}$ and $\delta p^{\text{EOS},j} = \delta p^{\text{EOS}}$ for each fluid j, and (ii) the total volume change is equal to the sum of all fluid volume changes. Applying these conditions to (3.96) gives

$$\delta V = \sum^{j} \delta V^j = -\sum^{j} \frac{V^j}{\bar{K}^j} \frac{\delta p^{\text{EOS}}}{p^{\text{EOS}}}, \tag{3.99}$$

where the sum is over both fluids in the domain. Combining this statement with (3.97) gives the expression for the total normalized bulk modulus

$$\bar{K} = \frac{V}{\sum^j \frac{V^j}{\bar{K}^j}} = \frac{1}{\sum^j \frac{\kappa^j}{\bar{K}^j}}. \tag{3.100}$$

Considering the volume of the total computational domain Ω is constant, $\delta V = \sum^j \delta V^j = 0$, then (3.98) is summed over all j and set to zero to give

$$p^{\text{EOS}} = \bar{K} \sum^{j} \kappa^j \frac{p^{\text{EOS},j}}{\bar{K}^j}. \tag{3.101}$$

Now all quantities for the correction are defined.

For a system of only two fluids, the equations of state are

$$p^{\text{EOS}} = \left(\kappa^a \frac{p^{\text{EOS},a}}{\bar{K}^a} + \kappa^b \frac{p^{\text{EOS},b}}{\bar{K}^b}\right) \bar{K} \tag{3.102}$$

$$\bar{K} = \frac{1}{\frac{\kappa^a}{\bar{K}^a} + \frac{\kappa^b}{\bar{K}^b}}, \tag{3.103}$$

$$\bar{K}^j = \frac{K^j}{p^{\text{EOS},j}}, \tag{3.104}$$

$$p^{\text{EOS},j} = p^{0,0} + K^j \frac{(\rho^j - \rho^{j,0})}{\rho^{j,0}}, \tag{3.105}$$

with j being either fluid a or b.

The volume difference for a fluid described by (3.98) needs to be converted into a distance correction for the SD function. To do this, the volume change is converted into a velocity normal to the interface and the SD function is corrected using the ψ evolution equation. The volume of fluid b can be defined as

$$V^b = \int_\Omega H(\Gamma)\, dV, \tag{3.106}$$

$$H(\Gamma) = \begin{cases} 1 & \boldsymbol{x} \in (\Gamma \cup \Omega^b) \\ 0 & \text{otherwise} \end{cases} \tag{3.107}$$

where $H(\Gamma)$ is the Heaviside function with respect to the LS. Then the change in volume over time is

$$\frac{\partial V^b}{\partial t} = \frac{\partial}{\partial t} \int_\Omega H(\Gamma)\, dV = \int_\Omega \frac{\mathrm{d}H(\Gamma)}{\mathrm{d}\Gamma} \frac{\partial \Gamma}{\partial t}\, dV. \tag{3.108}$$

Applying the evolution equation of the interface from (3.71) and recognizing $\delta(\Gamma) = \frac{\mathrm{d}H(\Gamma)}{\mathrm{d}\Gamma}$, where $\delta(\Gamma)$ is the Dirac delta function, gives

$$\frac{\partial V^b}{\partial t} = \int_\Omega \delta(\Gamma)(\boldsymbol{u} \cdot \boldsymbol{n})\, dV, \tag{3.109}$$

$$\frac{\partial V^b}{\partial t} = \int_\Gamma (\boldsymbol{u} \cdot \boldsymbol{n})\, dS, \tag{3.110}$$

If $\boldsymbol{u}$ and $\boldsymbol{n}$ are assumed constant along the interface though the control volume Υ, then

$$\frac{\partial V^b}{\partial t} = \ell\, (\boldsymbol{u} \cdot \boldsymbol{n})|_\Gamma + \mathcal{O}(\ell^2), \tag{3.111}$$

with ℓ being the area of the interface Γ in the cell.

Integrating (3.111) from $t = n\Delta t$ to $t = (n+1)\Delta t$ to approximate the left hand side of (3.98), and recalling the definition of s from (3.71), gives the correction δs as

$$\delta s = \frac{1}{\ell \Delta t} \frac{V^b}{\bar{K}^b} \frac{p^{\text{EOS},b} - p^{\text{EOS}}}{p^{\text{EOS}}}. \tag{3.112}$$

Discretizing the s correction is straightforward, by recognizing that $V_{\boldsymbol{i}}^b = \kappa_{\boldsymbol{i}}^b V_{\boldsymbol{i}}^\Upsilon$, where $V_{\boldsymbol{i}}^\Upsilon$ is the volume of a control volume, (3.112) is given in discrete form as

$$\delta s_{\boldsymbol{i}} = \frac{1}{\Delta t} \left(\frac{\kappa^b V^\Upsilon}{\bar{K}^b \ell} \frac{p^{\text{EOS},b} - p^{\text{EOS}}}{p^{\text{EOS}}} \right)_{\boldsymbol{i}}. \tag{3.113}$$

The correction s is extended by finding the intersection of $\boldsymbol{\nabla}\psi_{\boldsymbol{i}}$ with Γ and using the s value of the cell at the intersection for the source cell $\boldsymbol{i}$. Then ψ can be evolved using

$$\psi^{n+1} = \tilde{\psi}^{n+1} + \sigma \delta s \Delta t, \tag{3.114}$$

with $\sigma = 1/2$ being a relaxation parameter and $\tilde{\psi}^{n+1}$ is the value of ψ after the time integration to step $n+1$.

To summarize, the process of the correction is performed in the following order,

1. Advance the governing equations one time step to time $t^0 + \Delta t$

2. Compute the cell geometries in Section 3.6.1

3. Advance ρ^a, ρ^b to time $t^0 + \Delta t$ using the hybridized update and redistribution method in Section 3.6.2,

4. Correct the pressure and ψ as described in this section,

5. Reevaluate ρ^a and ρ^b using the method in Section 3.6.2 with cell geometries recomputed with the new position of the LS with methods in Section 3.6.1.

After the steps above are completed, p^{EOS} is reevaluated and single valued variables are updated to be consistent with the two-fluid description. Using the same relations between p and $\boldsymbol{u}_p$ from the single fluid case, (2.68)–(2.69), a Helmholtz equation for the pressure correction is formed

$$\left[\frac{1}{K} - \Delta t^2 \boldsymbol{\nabla} \cdot \left(\frac{1}{\rho^{n+1}} \boldsymbol{\nabla}\right)\right] \delta p^{n+1} = \frac{1}{K} \eta \Delta t (p^{\text{EOS}} - \tilde{p}^{n+1}) \in \Omega \quad \frac{\partial p}{\partial \boldsymbol{n}} = 0 \in \partial\Omega, \tag{3.115}$$

$$\delta \boldsymbol{u}_p{}^{n+1} = -\Delta t \frac{1}{\rho^{n+1}} \boldsymbol{\nabla} \delta p^{n+1}. \tag{3.116}$$

The corrections are applied as they were in the single fluid case:

$$p^{n+1} = \tilde{p}^{n+1} + \delta p^{n+1}, \tag{3.117}$$

$$\boldsymbol{u}_p{}^{n+1} = \mathbb{Q}\left(\tilde{\boldsymbol{u}_p}{}^{n+1} + \delta \boldsymbol{u}_p{}^{n+1}\right), \tag{3.118}$$

$$\boldsymbol{u}^{n+1} = A^{FC}(\boldsymbol{u}_p{}^{n+1}) + \mathbb{P}(\tilde{\boldsymbol{u}}^{n+1}), \tag{3.119}$$

where $\tilde{p}^{n+1}$, $\tilde{\boldsymbol{u}_p}{}^{n+1}$, and $\tilde{\boldsymbol{u}}^{n+1}$ are the values from the ARK time integration before the correction.

Chapter 4

Two-Fluid Results

This chapter displays the results of numerical tests for the level-set method and the two-fluid method in Chapter 3. Three tests for the signed distance function marching methods method show the accuracy of interface geometry:

1. Evaluating the signed distance function around a box with sharp corners,
2. The signed distance function around a circle level-set,
3. The signed distance function around a star shaped level-set.

Another three tests are shown for the two-fluid system, all representing the interface between the two fluids as an ellipse:

1. An initially motionless velocity field with the density of each fluid slightly perturbed from equilibrium,
2. A constant uniform initial fluid velocity and equivalent density,
3. The inviscid vortex initial conditions from Section 2.7.

4.1 Marching Methods Results

Results for numerical tests of the marching methods and the volume correction method are presented. First the gradient descent initialization method is tested and compared to the Schwartz initialization on a box geometry in 2D. The box is chosen because it has a simple geometry with sharp corners where ψ can be computed exactly. Next the hybridized extension method is tested on circle and star geometries. The circle is used because the exact ψ solution is known and the star is chosen because it has changing curvature with concave and convex regions. The L_p norms of

the error are reported,

$$L_p = \left(h^{\mathrm{D}} \sum_{\boldsymbol{i}} |e|^p \right)^{1/p} \tag{4.1}$$

$$\tag{4.2}$$

with e being the estimated error. All tests are run on the domain $[-10, -10] \cup [10, 10]$ with discretizations $h = \{1/5, 1/10, 1/20, 1/40\}$. The initialization and extension distances are set as $\epsilon = 0.5$ and $R = 1.2$, respectively. The marching parameter is set as $\sigma = 1/(2\sqrt{5})$. It is important in measuring the convergence rates that these distances do not depend on the discretization so that area where the error is calculated is constant. In practice, ϵ and R would be based on h to save on computation.

Box

The novel GDI method presented in Section 3.4.1 is tested on a box with the level-set to be defined as

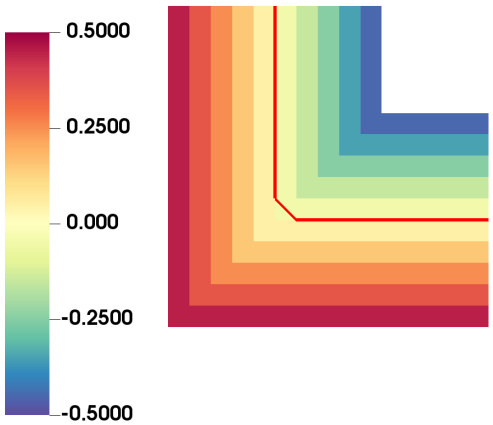
$$\phi = 0 : \{y = \pm 5, -5 \leq x \leq 5\} \cup \{x = \pm 5, -5 \leq y \leq 5\}. \tag{4.3}$$

Error is evaluating using the exact solution of ψ.

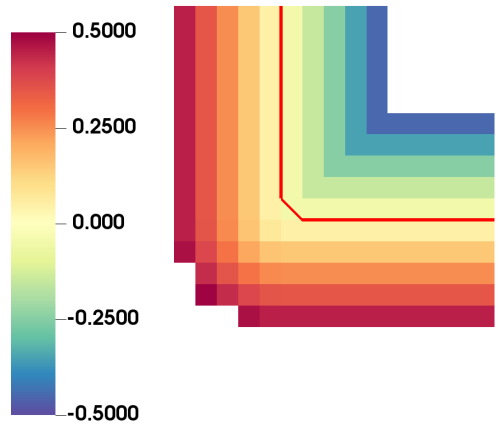
As shown in Figure 4.1a, the ϕ function results in sharp corners on the exterior and interior of the level-set whereas the exact ψ has rounded exterior corners, as shown in Figure 4.1b. The initialized ψ field using SI is shown in Figure 4.2a and is unable to initialize ψ around the corner. It is only capable of initializing on the diagonal because the finite gradient, $G(\phi)$, intersects the corner of the box. The use of integral curves in the GDI method allows initialization throughout the entire exterior corner, as shown in Figure 4.2b.

The convergence of the initialization followed by hybridized extension is reported in shown in Figure 4.4 and specific values reported in Table 4.1 and Table 4.2. Although the GDI method is only first order accurate, it shows a clear convergence pattern in Figure 4.4b as opposed to

the Schwartz initialization which has no convergence pattern as shown in Figure 4.4a. This is a substantial improvement over the SI method for initialization around sharp corners.

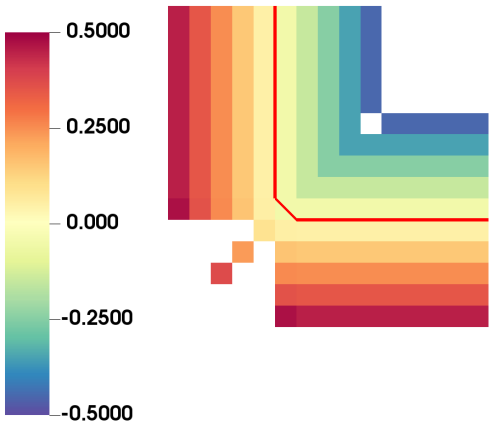


(a) Implicit representation ϕ of the box before initialization and extension.

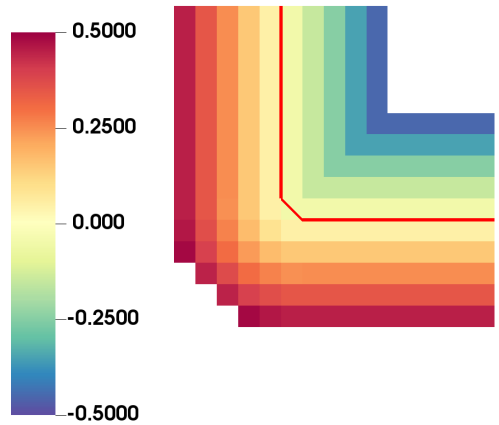


(b) Exact ψ of the Box.

Figure 4.1: Implicit function representing Γ, shown as the red line, for the box geometry.

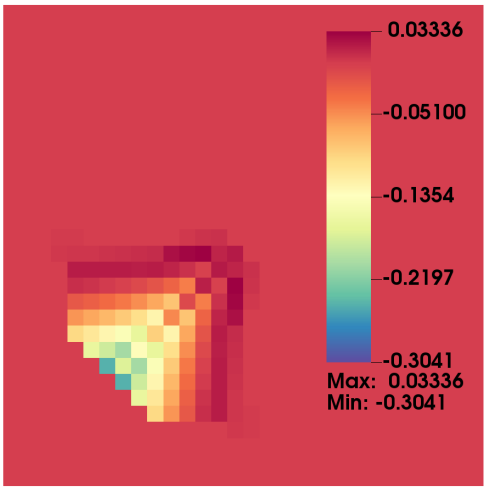


(a) Initialization of ϕ to ψ using the Schwartz method.

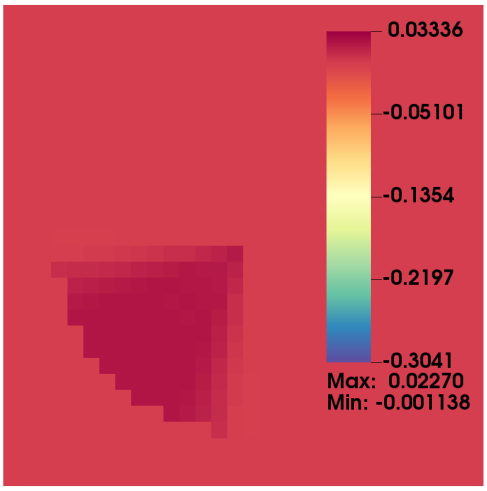


(b) Initialization of ϕ to ψ using the GDI method.

Figure 4.2: The initialization of ψ using the SI method and the GDI method on the box level-set.

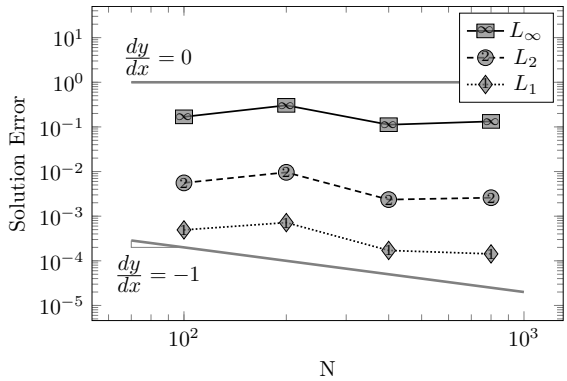


(a) The error after hybridized extension using the SI method.

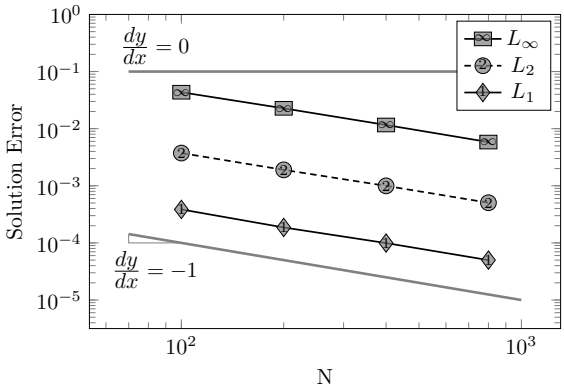


(b) The error after hybridized extension using the GDI method.

Figure 4.3: Error distribution after hybridized extension on the box using both SI and GDI methods.



(a) Error convergence of the hybridized extension with the SI method.



(b) Error convergence of the hybridized extension with the GDI method.

Figure 4.4: Convergence plots of the SI and GDI methods after hybridized extension on the box geometry.

Table 4.1: The error L_p norm of the hybridized extension of ψ with GDI on the box.

	$N = 100$	Rate	$N = 200$	Rate	$N = 400$	Rate	$N = 800$
L_1	0.0003854	1.048	0.0001864	0.909	0.00009932	0.991	0.00004998
L_2	0.003741	0.977	0.001901	0.931	0.0009971	0.981	0.0005051
L_∞	0.04344	0.936	0.0227	0.969	0.0116	0.993	0.005828

Table 4.2: The error L_p norm of the hybridized extension of ψ with SI on the box.

	$N = 100$	Rate	$N = 200$	Rate	$N = 400$	Rate	$N = 800$
L_1	4.927E-4	-0.543	7.180E-4	2.072	1.708E-4	0.257	1.429E-4
L_2	5.568E-3	-1.447	9.603E-3	3.736	2.352E-3	-0.257	2.591E-3
L_∞	1.682E-1	-1.087	3.041E-1	1.831	1.121E-1	-0.311	1.328E-1

Circle

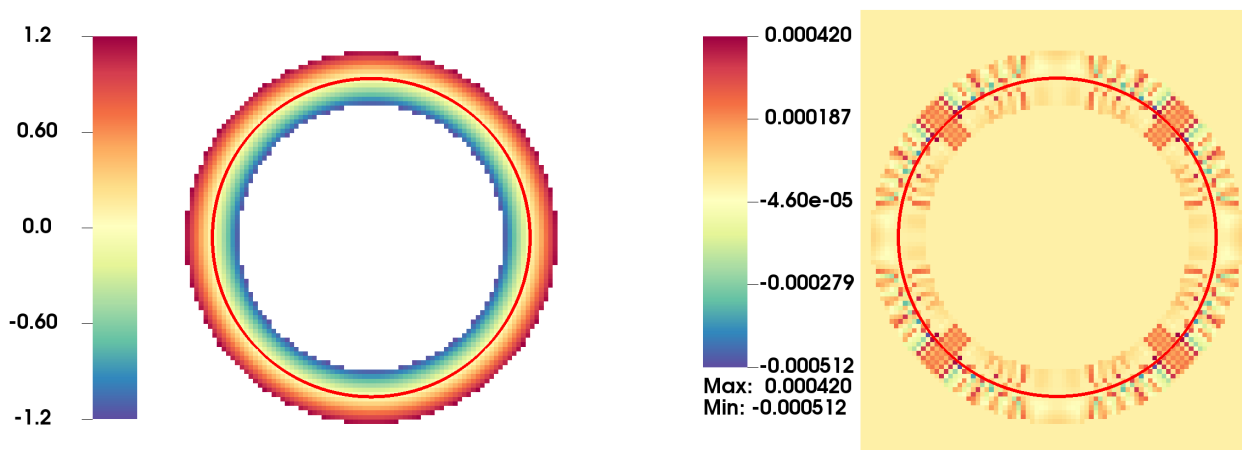
The circle case is useful because ψ is known exactly, so the hybridized extension algorithm can be compared to the exact solution. The level-set of the circle in Cartesian coordinates is

$$\psi = \sqrt{x^2 + y^2} - r, \quad \boldsymbol{\nabla}\psi = \begin{bmatrix} x/\sqrt{x^2+y^2} \\ y/\sqrt{x^2+y^2} \end{bmatrix} \quad , \tag{4.4}$$

with r being the desired radius of the circle, in this case $r = 7$. The level-set and ψ field computed from the extension method are shown in Figure 4.5a.

The most striking feature of the error is its distribution around the level-set where the error magnitude is periodic every half radian as shown in Figure 4.5b. This is due to the number of cells in the evaluated set P in different regions of the circumference. Larger error magnitudes occurs where the are fewer cells in P.

The expected convergence rate for the hybridized solution are achieved and shown in Table 4.3 and Figure 4.6. The L_∞ rate is slightly lower due to the hybridization reducing the accuracy in a few particular cells, this effect is more apparent in the star test in Figure 4.1



(a) ψ for the circle geometry after hybridized extension from the initialized band.

(b) The error between the hybridized extension of ψ and the exact solution.

Figure 4.5: The hybridized extension of ψ and its error on the circle.

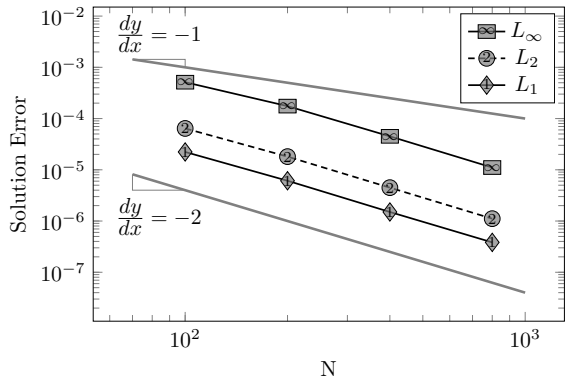


Figure 4.6: Error convergence of the hybridized circle extension.

Table 4.3: The error L_p norm of the hybridized extension of ψ on the circle.

	$N = 100$	Rate	$N = 200$	Rate	$N = 400$	Rate	$N = 800$
L_1	2.229E-5	1.868	6.109E-6	2.024	1.502E-6	1.971	3.832E-7
L_2	6.416E-5	1.828	1.808E-5	2.012	4.482E-6	2.011	1.112E-6
L_∞	5.122E-4	1.550	1.749E-4	1.960	4.497E-5	2.021	1.108E-5

Star

The star tests the marching methods around a level-set with varying curvature, providing a good test of the hybridization. This test was also presented in Schwartz [53] and is a useful comparison. The initial implicit function defining the level-set is

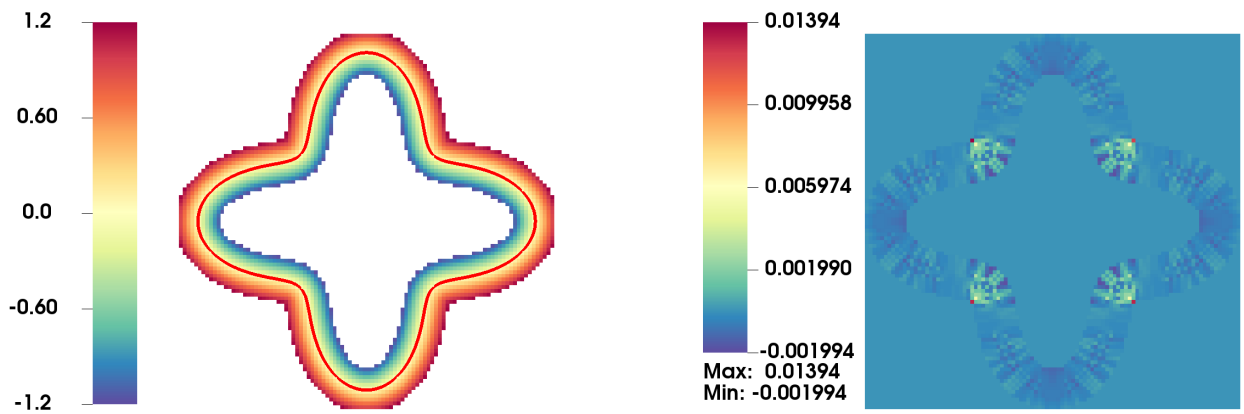
$$\phi = \sqrt{x^2 + y^2} + 16\frac{x^4}{(x^2 + y^2)^2} + 16\frac{x^2}{x^2 + y^2} - 9, \tag{4.5}$$

or more conveniently in polar coordinates as

$$r = 2\cos 2\theta + 7. \tag{4.6}$$

The error in ψ is estimated via Richardson extrapolation. The ψ field and the estimated error are shown in Figure 4.7.

This test shows that expected second order accuracy is achieved using the hybridized extension as shown in Table 4.4 and Figure 4.8. Notice that the L_∞ norm follows $\mathcal{O}(h)$ accuracy because of the hybridization. The effect of this can be observed in (4.7b) where the largest error occurs on the exterior corners of the Star where the magnitude of the curvature becomes very large and is negative. The same distribution of the error is observed in Schwartz [53].



(a) The hybridized extension of ψ on the star.

(b) The error of the hybrid extension on the star.

Figure 4.7: The signed distance field and error for the star geometry.

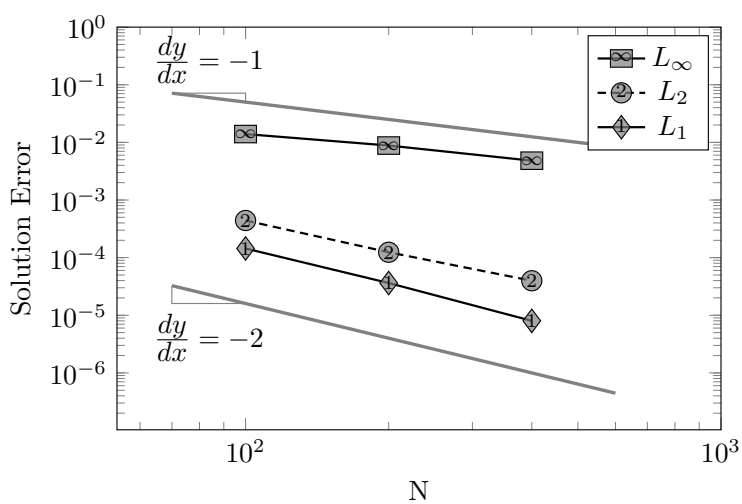


Figure 4.8: Error convergence of the hybridized star extension.

Table 4.4: The error L_p norm of the hybridized extension of ψ on the star.

	$N = 100$	Rate	$N = 200$	Rate	$N = 400$
L_∞	1.394E-2	0.661	8.818E-3	0.873	4.814E-3
L_1	1.442E-4	1.990	3.629E-5	2.181	8.004E-6
L_2	4.410E-4	1.829	1.241E-4	1.642	3.976E-5

4.2 Two-Fluid Results

The volume correction methods are tested in three conditions, all with an initially elliptic LS. The first is an initially quiescent flow field where the initial densities of both fluids are perturbed from equilibrium, showing that the correction method alters the position of the LS to restore the volume of each fluid to its equilibrium density. The second is uniform flow with uniform initial density, which tests that the volume of each fluid is preserved in an incompressible flow field without deformation. Finally, the inviscid vortex field from Chapter 2 is applied to test volume preservation in a weakly compressible field with a deforming LS. The results are measure with the volume of fluid a over time normalized to the initial volume, $V^a/V^a(t=0)$.

Density Ellipse The purpose of this experiment is to show that the correction method will change the volume of the fluids to restore pressure and density equilibrium. The density inside the ellipse is initialized slightly lower than the equilibrium density and slightly higher outside the ellipse. The fluid has zero initial velocity so the only changes to the velocity and pressure field will come from the correction method. The expectation is that the ellipse will shrink slightly and an increase in the volume of fluid a will be observed. The conditions for this test given explicitly are

$$\rho^{a,0} = \rho^{b,0} = 1, \tag{4.7}$$

$$\rho^a(t=0) = \rho^{a,0} + \epsilon, \quad \rho^b(t=0) = \rho^{b,0} - \epsilon, \quad \boldsymbol{u} = \mathbf{0}, \tag{4.8}$$

$$p^{00} = p(t=0) = 1\times10^6, \quad \epsilon = [1\times10^{-4},\ 1\times10^{-3},\ 1\times10^{-2}]. \tag{4.9}$$

A refinement study was done with $N = [64,\ 128,\ 256]$.

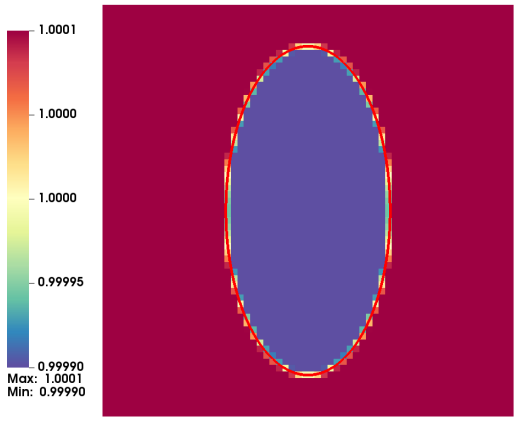


Figure 4.9: Initial ρ of the density ellipse with the level-set shown by the red contour.

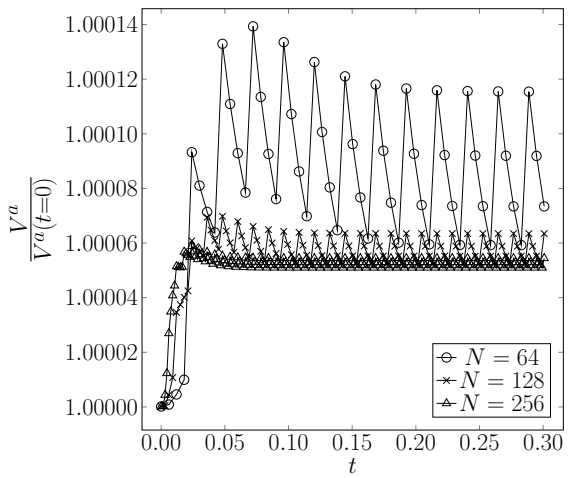


Figure 4.10: The normalized volume of fluid a at different levels of refinement.

It is clear from Figure 4.10 that the volume of fluid a has increased and approaches an equilibrium value as expected. The refinement of the discretization also shows a convergence to an equilibrium pressure. To test that greater density differences result in greater volume changes, the density perturbation, ϵ, is increased. The results in Figure 4.11 show that greater ϵ in the initial condition results in larger volume changes to achieve equilibrium. This is consistent with the physical expectation that larger density differences produce larger volume changes.

The outstanding feature of these results is the oscillations in the volume around the equilibrium occurring every four time steps. This is due to the reinitialization interval of the level-set, n_{init}, which is set to every four time steps. It is shown in Figure 4.12 that changing n_{init} changes how often these perturbations occur. Changing the n_{init} interval correlates directly with the frequency of the volume spikes, indicating this effect is not a consequence of the correction method but a perturbation induced by the reinitialization method. This happens because the reinitialization method cannot perfectly preserve the position of the LS and thereby changes the volume. This also shows that the correction method is functioning as desired by restoring perturbations in the volume

to equilibrium. While this is not an ideal result, the magnitude of these perturbations is bounded and the volume still converges. A more accurate reinitialization method would reduce this issue.

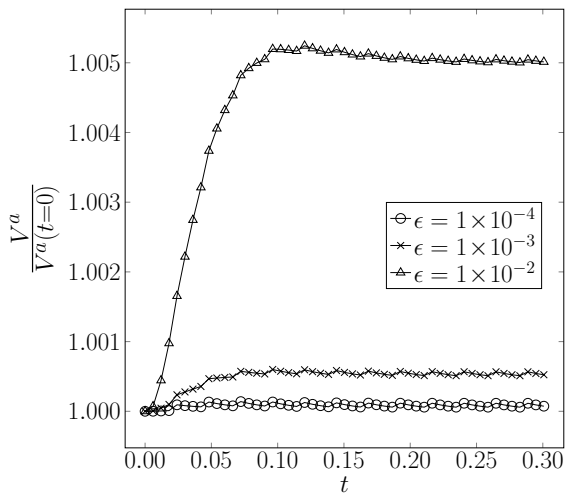


Figure 4.11: Normalized fluid a volume over time with different initial density perturbations, ϵ.

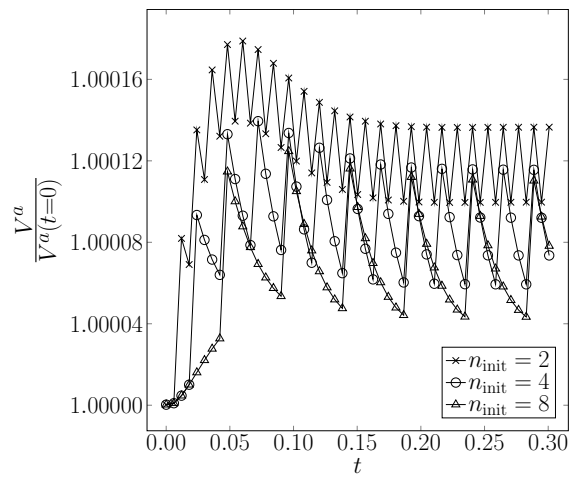


Figure 4.12: Normalized fluid a volume over time with different LS reinitialization intervals.

Uniform Flow In this test, the ellipse level-set is advected by a uniform constant velocity field where the interface does not undergo deformation due to the flow field. All densities and equilibrium densities are initialized to be equal. The expectation is the volume of each fluid is conserved around the initial value. For this test, the x-normal boundaries are periodic and the y-normal boundaries are slip walls. The simulation is run to $t = 0.25$ at a CFL $= 0.5$. Conditions for this test are given explicitly as:

$$\rho^{a,0} = \rho^{b,0} = 1, \quad \rho^a(t=0) = \rho^{a,0}, \quad \rho^b(t=0) = \rho^{b,0}, \tag{4.10}$$

$$\boldsymbol{u}(t=0) = [1,\ 0]^T, \quad p^{00} = p(t=0) = 1\times10^6. \tag{4.11}$$

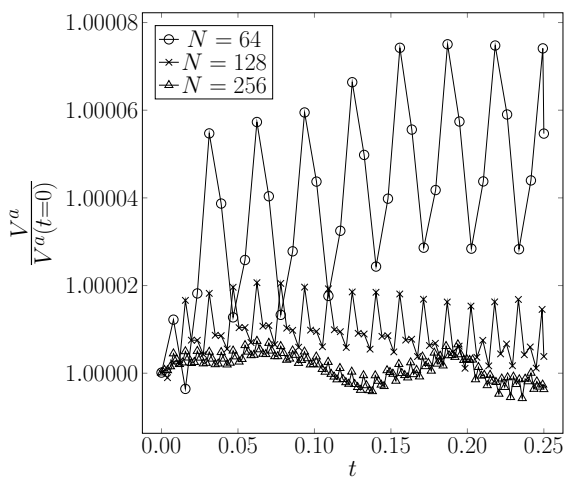


Figure 4.13: Normalized volume of fluid a over time at different spatial discretizations.

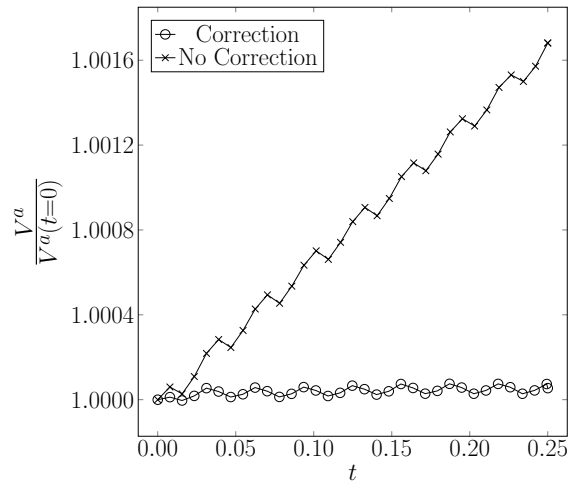


Figure 4.14: Normalized volume of uniform flow with and without the volume correction.

The volume of fluid a is expected to remain close to constant in this experiment, as the Mach number is approximately 8.4×10^{-4}. Although some small deviations are observed in Figure 4.13, the volume converges towards the initial volume when the spatial discretization is refined. The convergence towards the initial volume in this test shows that the correction is consistent with the expected physical result.

Comparing the volume of the ellipse with the correction to the ellipse without at $N = 64$ in Figure 4.14, it is clear that the correction prevents unbounded growth of fluid a volume. Without the correction, the volume of fluid a continues to increase without bound and straying from the weakly compressible constraint. This achieves the desired result and shows that volume is constrained under translation of the LS.

Vortex Flow The final set of tests is done with a vortex potential flow field. The purpose of this test is to test the volume preservation of the correction method when the interface is deformed under the motion of the fluid. The initial velocity is a vortical potential flow field that rotates counter clockwise. All densities are set to be in equilibrium like the uniform flow test, giving the

conditions for this test as

$$\rho^{a,0} = \rho^{b,0} = 1, \tag{4.12}$$

$$\rho^{a}(t=0) = \rho^{a,0}, \tag{4.13}$$

$$\rho^{b}(t=0) = \rho^{b,0}, \tag{4.14}$$

$$\boldsymbol{u}(t=0) = \begin{bmatrix} 2\sin^2 \pi x_0 \sin \pi y_0 \cos \pi y_0 \\ -2\sin^2 \pi y_0 \sin \pi x_0 \cos \pi x_0 \end{bmatrix}, \tag{4.15}$$

$$p^{00} = p(t=0) = 1 \times 10^6, \tag{4.16}$$

where $x_0 = x - 1/2$ and $y_0 = y - 1/2$ are the coordinates from the center of the vortex. Slip wall boundary conditions are applied to the vortex. This test is run until $t = 0.5$ at CFL $= 0.5$

The volume of fluid a is expected to remain close to constant in this test as well, with the maximum Mach number being approximately 8.8×10^{-4}. The data in Figure 4.15 show that the final volume of fluid a converges to its volume as the spatial refinement is increased. The convergence towards the initial volume again show that this method constrains the volume and is consistent with the expected physical result under a deforming interface.

The effect of the volume correction on the ellipse in the vortex is shown in Figure 4.16. It is clear that the volume of fluid a grows unbounded without the volume correction, inconsistent with the behavior of low speed flows. Figure 4.15 shows that the volume approaches perfect conservation as the discretization is refined. This confirms that the volume of each fluid is constrained in a flow field that deforms the LS.

Density is greatly affected by the correction as seen in Figure 4.17–Figure 4.18, showing the average density ($\rho = \rho^a \kappa^a + \rho^b \kappa^b$) at the end time of the simulation. Without the correction density variations tend to increase, particularly around regions of high curvature in the level-set due to the reduced accuracy of the level-set in these regions. These errors are still present with the correction as shown in Figure 4.18 but are significantly reduced by approximately an order of magnitude.

While, variations in density still exist with the correction the accuracy of the solution is greatly improved over the level-set method alone.

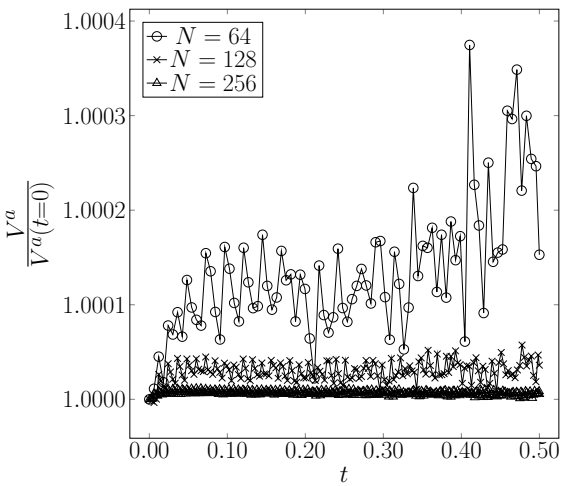


Figure 4.15: Normalized volume of fluid a over time in vortex flow at different discretizations.

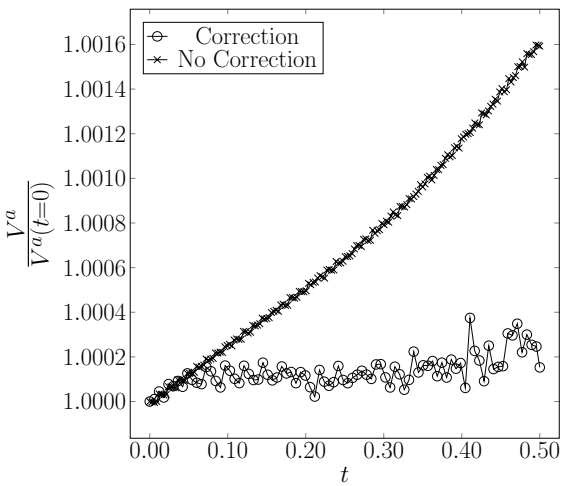


Figure 4.16: Normalized volume of fluid a in vortex flow with and without correction.

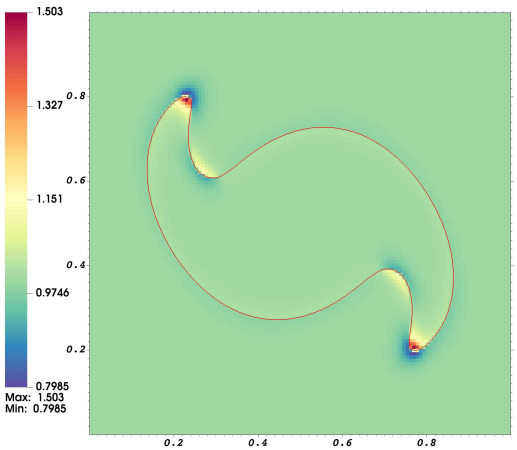


Figure 4.17: Two-fluid vortex $\rho^a\kappa^a + \rho^b\kappa^b$ without the level-set correction.

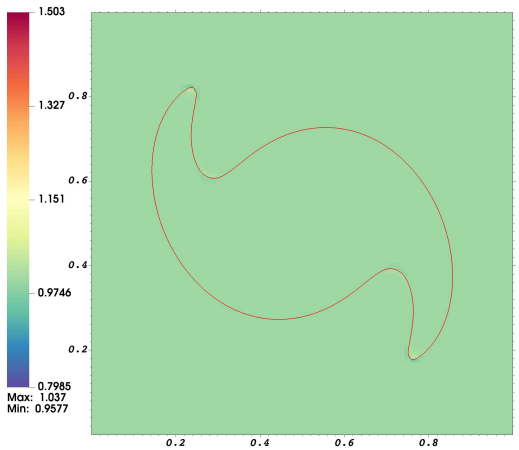


Figure 4.18: Two-fluid vortex $\rho^a\kappa^a + \rho^b\kappa^b$ with the level-set correction.

4.3 Discussion

The results in this chapter show that the signed distance function, and therefore the interface geometry, is represented up to second order accuracy through the initialization and extension methods. This is the reason why the all-speed equations are only discretized to second order accuracy in space. In situations where the magnitude of the curvature is high, the solution error increases, as seen in the star test. This is by design of the marching method where the extension method drops to first order accuracy around regions of high curvature for stability purposes, leading to a first order convergence in the L_∞ error norm. Additionally, using the level-set method, the curvature of the interface can also be estimated for applications with interface properties that are curvature dependent such as the surface tension. Around sharp corners, the novel GDI method is shown to be first order accurate after extension of the signed distance function. This is a substantial improvement over other methods such as the method from Schwartz [53] which cannot initialize around sharp corners and does not have a clear convergence pattern.

The density perturbed ellipse, the first multifluid test, is meant to show how the correction method can restore perturbations in density by changing the volume of each fluid. In that test, the density of the fluid outside of the ellipse is slightly above its equilibrium value so the volume of the outer fluid is expected to slightly increase to lower the density, precisely what is observed. After the fluid volume has changed it remains approximately constant, as would be expected. The volume of the exterior fluid is shown to converge towards an equilibrium as shown by the refinement study. Because there are only two fluids in this model, if one fluid volume remains constant then the other does as well. This is a major improvement over the level-set method alone, resulting in an unbounded increase in volume of the fluid on the exterior of the interface, which is clearly unphysical. As the discretization is refined the accuracy of the geometry and correction increase, therefore this convergence behavior is expected. When the magnitude of the perturbation changes the change in volume is also expected to change and is observed, further verification of the expected physical behavior. The volume follows a four time step periodic behavior, which is not physical. This is a result of the reinitialization interval, n_{init}, as shown by altering the initialization interval.

When the initialization interval is increased to eight time steps the period of this oscillation changes to eight time steps as well, clear evidence of this relationship. This is not unexpected because of the error associated with the reinitialization of the signed distance function, altering the position of the interface and resulting in volume changes. Results from this test indicate the level-set correction method recovers physically expected behavior when density is perturbed from equilibrium.

The uniform flow test verifies that the volume of each fluid remains approximately constant when the fluids densities are initially in equilibrium and flow is identically divergence-free initially. This expected behavior is observed when plotting the normalized volume of the exterior fluid. Though the presence of error is apparent, the volume converges towards the initial volume with increased refinement, similarly to the density perturbed case. Furthermore, the volume of the fluid is bounded unlike the level-set alone. This serves as further verification that the level-set with the correction converges towards the physically expected behavior.

The final test is the two-fluid vortex, whose initial condition is identical to the single fluid vortex for the single valued variables. This tests the performance of the correction in a weakly compressible case with a deforming interface. Like the uniform flow test, the volume of the fluids in this test is also not expected to change, as the Mach number is very low ($M \approx 8.8 \times 10^{-4}$) and the velocity field is initially divergence-free. The results reflect the expected behavior, showing approximately constant volume of the external fluid and equivalently the internal fluid. Volume is shown to converge towards the initial volume as the discretization is refined. The correction shows a major improvement over the level-set method alone, which has unbounded volume increase in this test. Density is massively effected by this correction, shown by the comparison between the average density with and without the level-set correction. While density perturbations still exist with the correction, they are greatly reduced resulting in improved solution accuracy. The errors are greatest in the higher curvature regions where the accuracy of the level-set is reduced to first order. This effect is expected from the hybridized formulation of the extension method in Section 3.4.2. These tests show the correction is necessary to achieve greater physical accuracy and capture the

expected behavior of low Mach number flows for multifluid flows with the interface captured by the level-set method.

Chapter 5

Conclusions

This dissertation displays a novel method for simulating flows with two fluids at low Mach numbers with an accurate representation of the interface geometry using the all-speed equations and the level-set method with a thermodynamically consistent correction to the level-set. To answer this driving question, the work in this dissertation has completed the stated objectives:

1. Implement a set of governing equations for low Mach-number flows.

2. Apply the level-set method to describe a two fluid system by defining the interface between the fluids.

3. Develop a volume correction method to ensure each fluid's volume is consistent with its pressure equation of state.

The weakly compressible all-speed system for inviscid and viscous flow is implemented as the governing equations for low Mach number flows. Using the weakly compressible system allows for high order accuracy in time with the ARK method without the formulation of complicated divergence-free constraints in space and time that would be required for a purely incompressible flow. Tests in Chapter 2 confirm this system is discretized to second order accuracy and reproduces the expected scaling of the pressure at low Mach numbers. This is achieved using constant coefficient projection operators, a redundant equation for the curl-free velocity, and an IMEX ARK time integrator. The system of governing equations is combined with the level-set method to provide a multifluid description of the two-fluid system.

The level-set method is applied to track the interface between the two fluids and distinguish their domains in a multifluid system. It provides superior representation of the geometry and describes smooth changes to the geometry over time naturally. The level-set is represented by the signed distance function which is constructed using initialization and extension methods up

to second order accuracy, as shown in Chapter 3. Initialization methods are enhanced with the introduction of the novel gradient descent intialization method, which is able to initialize the signed distance function around sharp corners in the level set with first order accuracy.

The multifluid description of the system is completed with the novel volume correction method. The novel correction method serves two purposes: (i) it enforces agreement between the single valued description of the pressure and the double valued description of density, and (ii) relaxes errors in the level-set position towards the thermodynamically consistent state. This correction method is shown in Chapter 3 to correct for initial density perturbations and conserve volume in low Mach number flows. Without this correction, inconsistencies between the pressure and density in the equation of state will accumulate and generate unphysical results, such as the unbounded growth of one volume as shown in Chapter 4. Therefore, the level-set correction is a necessary and novel addition to accurately represent two-fluid flows with the level-set at low Mach numbers.

In the completion of these objectives, the following primary novel contributions to the field have been made:

1. A multifluid algorithm using an all-speed method to solve the fluid equations.

2. A multifluid algorithm using the level-set to capture the interface in the weakly compressible regime that is thermodynamically consistent.

3. The gradient descent initialization method to initialize the signed distance function around sharp corners in the level-set.

Multifluid algorithms have so far been implemented for compressible or strictly incompressible flows. The extension to weakly-compressible flows can greatly improve upon the weaknesses of these previously separate regimes. Larger time steps than those allowed by fully compressible algorithms can be used for more efficient computing. And including acoustic physics important in gas-gas chemical reactions can be included, which is not possible in an incompressible algorithm.

The level-set method has previously been restricted to incompressible flow with some recent work done with high Mach number compressible flow. Most multifluid algorithms utilize the VOF

method and add a complex algorithm to reconstruct the interface. This work bridges the gap by applying the level-set to fluid velocities beyond the incompressible and below supersonic using currently available level-set methods and adding a correction.

The novel gradient descent initialization method addresses a problem in the application of the level-set method to sharp geometries. The typical method is to intentionally smooth the geometry to avoid the initialization issue around sharp corners. Instead this method retains the original geometry with a method that is first order accurate.

This dissertation lays the foundation towards the simulation of DIW printing by describing and implementing a set of governing equations and numerical methods for the fluid dynamics of the DIW filament in a multifluid system with the ambient domain, utilizing the level-set method to track the printing filament, and applying a novel volume correction method to enforce consistency between volume and equations of state. Future work should be done to apply adaptive mesh refinement (AMR) to the all-speed equations around the level-set. This would greatly improve the accuracy of the interface without requiring refinement of the whole computational domain, which is much more computationally expensive. Towards simulating DIW printing, future work should be on thermal effects, fluid to solid phase transition, and solid mechanics for the hardened filament.

There are limits to the methods presented in this dissertation. Higher order accuracy with the all-speed equations has been demonstrated by Chaplin [30] but is limited to second order here because the level-set can only be represented to second order accuracy. The physics are limited to the isothermal case but can be expanded with some modification to the constraints and level-set correction. The methods here are only applied to two-fluid systems and limited to such applications. Although, this work can be generalized with multiple level-sets to represent three or more fluid phases. The volume correction method is also only applied to two phases, but can be easily extended to an arbitrary number of fluids.

Bibliography

[1] Dartzi Pan and Chih-Hao Chang. The capturing of free surfaces in incompressible multi-fluid flows. *International Journal for Numerical Methods in Fluids*, 33(2):203–222, 2000.

[2] FS De Sousa, Norberto Mangiavacchi, Luis Gustavo Nonato, Antonio Castelo, Murilo F Tomé, Valdemir Garcia Ferreira, José Alberto Cuminato, and Sean Mckee. A front-tracking/front-capturing method for the simulation of 3d multi-fluid flows with free surfaces. *Journal of Computational Physics*, 198(2):469–499, 2004.

[3] Mark Sussman and Elbridge Gerry Puckett. A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows. *Journal of computational physics*, 162(2):301–337, 2000.

[4] Mark Sussman, Peter Smereka, and Stanley Osher. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational physics*, 114(1):146–159, 1994.

[5] Guangtao Duan, Bin Chen, Ximin Zhang, and Yechun Wang. A multiphase mps solver for modeling multi-fluid interaction with free surface and its application in oil spill. *Computer Methods in Applied Mechanics and Engineering*, 320:133–161, 2017.

[6] Detlef Hinneburg and Oswald Knoth. Non-dissipative cloud transport in eulerian grid models by the volume-of-fluid (vof) method. *Atmospheric Environment*, 39(23-24):4321–4330, 2005.

[7] Daniel Shipley, Hilary Weller, Peter A Clark, and William A McIntyre. Two-fluid single-column modelling of rayleigh–bénard convection as a step towards multi-fluid modelling of atmospheric convection. *Quarterly Journal of the Royal Meteorological Society*, 148(742):351–377, 2022.

[8] LL Williams, DT Hall, HL Pauls, and GP Zank. The heliospheric hydrogen distribution: a multifluid model. *The Astrophysical Journal*, 476(1):366, 1997.

[9] D Alexashov and V Izmodenov. Kinetic vs. multi-fluid models of h atoms in the heliospheric interface: a comparison. *Astronomy & Astrophysics*, 439(3):1171–1181, 2005.

[10] W. F. Noh and Paul Woodward. SLIC (Simple Line Interface Calculation). In Adriaan I. van de Vooren and Pieter J. Zandbergen, editors, *Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics June 28 – July 2, 1976 Twente University, Enschede*, pages 330–340, Berlin, Heidelberg, 1976. Springer.

[11] C. W Hirt and B. D Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201–225, January 1981.

[12] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

[13] David J Benson. Volume of fluid interface reconstruction methods for multi-material problems. *Applied Mechanics Reviews*, 55(2):151–165, April 2002.

[14] J López, J Hernández, P Gómez, and F Faura. A volume of fluid method based on multidimensional advection and spline interface reconstruction. *Journal of Computational Physics*, 195(2):718–742, April 2004.

[15] James Edward Pilliod and Elbridge Gerry Puckett. Second-order accurate volume-of-fluid algorithms for tracking material interfaces. *Journal of Computational Physics*, 199(2):465–502, September 2004.

[16] Francis H Harlow and J Eddie Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The physics of fluids*, 8(12):2182–2189, 1965.

[17] Mark Sussman, Ann S Almgren, John B Bell, Phillip Colella, Louis H Howell, and Michael L Welcome. An adaptive level set approach for incompressible two-phase flows. *Journal of Computational Physics*, 148(1):81–124, 1999.

[18] John D. Anderson. *Modern compressible flow: with historical perspective*. McGraw-Hill series in aeronautical and aerospace engineering. McGraw-Hill, New York, 2nd ed. edition, 1990.

[19] Dimitris Drikakis and William Rider. *High-resolution methods for incompressible and low-speed flows*. Springer Science & Business Media, 2005.

[20] John B Bell, Phillip Colella, and Harland M Glaz. A second-order projection method for the incompressible navier-stokes equations. *Journal of computational physics*, 85(2):257–283, 1989.

[21] John B Bell and Daniel L Marcus. A second-order projection method for variable-density flows. *Journal of Computational Physics*, 101(2):334–348, 1992.

[22] Benjamin Sanderse and Barry Koren. Accuracy analysis of explicit runge–kutta methods applied to the incompressible navier–stokes equations. *Journal of Computational Physics*, 231(8):3041–3063, 2012.

[23] Charles Hirsch. *Numerical computation of internal and external flows: The fundamentals of computational fluid dynamics*. Elsevier, 2007.

[24] Alexandre Joel Chorin. Numerical solution of the navier-stokes equations. *Mathematics of computation*, 22(104):745–762, 1968.

[25] Alexandre Joel Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of computational physics*, 135(2):118–125, 1997.

[26] Suhas V Patankar and D Brian Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. In *Numerical prediction of flow, heat transfer, turbulence and combustion*, pages 54–73. Elsevier, 1983.

[27] Raad I Issa. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of computational physics*, 62(1):40–65, 1986.

[28] A. Majda and J. A. Sethian. The derivation and numerical solution of the equations for zero Mach number combustion. *Combust. Sci. Technol.*, 42:185–205, 1985.

[29] Mindy Fruchtman Lai. A projection method for reacting flow in the zero mach number limit, 1993.

[30] Christopher Michael Chaplin. *An improved all-speed projection algorithm for low Mach number flows*. PhD thesis, UC Berkeley, 2018.

[31] C. Kennedy and M. Carpenter. Additive Runge-Kutta schemes for convection-diffusion-reaction equations. *Applied Numerical Mathematics*, 44:139–181, 2003.

[32] Q. Zhang, H. Johansen, and P. Colella. A fourth-order accurate finite-volume method with structured adaptive mesh refinement for solving the advection-diffusion equation. *SIAM J. Sci. Comput.*, 34:B179–B201, 2012.

[33] J. Christopher, X. Gao, and S.Guzik. High-order implicit-explicit additive runge-kutta schemes for numerical combustion with adaptive mesh refinement. J. Comput. Phys., 2021.

[34] R. C. Hibbeler. *Fluid mechanics*. YBP Print DDA. Pearson, Boston, 2015.

[35] Amiram Harten, Peter D Lax, and Bram van Leer. On upstream differencing and godunov-type schemes for hyperbolic conservation laws. *SIAM review*, 25(1):35–61, 1983.

[36] A Harten, B Engquist, S Osher, and S Chakravarthy. Uniformly high order essentially non-oscillatory schemes iii. *J. Comput. Phys.*, 71:231–303, 1987.

[37] SK Godunov. A finite-difference method for the numerical computation of discontinuous solutions of the equations of fluid dynamics. *Mat. Sb.*, 47:271, 1959.

[38] Phillip Colella. Volume-of-fluid methods for partial differential equations. In *Godunov Methods: Theory and Applications*, pages 161–177. Springer, 2001.

[39] P. Colella. Multidimensional upwind methods for hyperbolic conservation laws. *J. Comput. Phys.*, 87(1):171–200, 1990.

[40] P. Colella and P. Woodward. The piecewise parabolic method for gas-dynamical simulations. *J. Comput. Phys.*, 54:174–201, 1984.

[41] X Gao and S. M. J. Guzik. A fourth-order scheme for the compressible Navier-Stokes equations. AIAA 2015-0298, 53rd AIAA Aerospace Sciences Meeting, 2015.

[42] X. Gao, L. D. Owen, and S. M. J. Guzik. A parallel adaptive numerical method with generalized curvilinear coordinate transformation for compressible Navier-Stokes equations. *Int. J. Numer. Meth. Fluids*, 82:664–688, 2016.

[43] S. M. Guzik, X. Gao, and C. Olschanowsky. A high-performance finite-volume algorithm for solving partial differential equations governing compressible viscous flows on structured grids. *Comput. Math Appl.*, 72:2098–2118, 2016.

[44] IJ Keshtiban, F Belblidia, and MF Webster. Compressible flow solvers for low mach number flows-a review. *Int. J. Numer. Methods Fluids*, 23:77–103, 2004.

[45] Jean-Luc Guermond, Peter Minev, and Jie Shen. An overview of projection methods for incompressible flows. *Computer methods in applied mechanics and engineering*, 195(44-47):6011–6045, 2006.

[46] Phillip Colella and Karen Pao. A projection method for low speed flows. *Journal of Computational Physics*, 149(2):245–269, 1999.

[47] M. Adams, P. Colella, D. T. Graves, J. N. Johnson, H. S. Johansen, N. D. Keen, T. J. Ligocki, D. F. Martin, P. W. McCorquodale, D. Modiano, P. O. Schwartz, T. D. Sternberg, and B. Van Straalen. Chombo software package for AMR applications—design document. Technical Report LBNL-6616E, Lawrence Berkeley National Laboratory, Berkeley, California, USA, 2015.

[48] Alexandre Joel Chorin. On the convergence of discrete approximations to the navier-stokes equations. *Mathematics of computation*, 23(106):341–353, 1969.

[49] Alexandre Joel Chorin and Jerrold E. Marsden. *A mathematical introduction to fluid mechanics*. Texts in applied mathematics ; 4. Springer-Verlag, New York, 3rd ed. edition, 1993.

[50] George Gabriel Stokes. On the dynamical theory of diffraction. *Transactions of the Cambridge Philosophical Society*, 9:1–48, 1849.

[51] Ann S Almgren, John B Bell, and William G Szymczak. A numerical method for the incompressible navier-stokes equations based on an approximate projection. *SIAM Journal on Scientific Computing*, 17(2):358–369, 1996.

[52] Christopher Radcliff Anderson. Derivation and solution of the discrete pressure equations for the incompressible navier-stokes equations. 1988.

[53] Peter Schwartz and Phillip Colella. A second-order accurate method for solving the signed distance function equation. *Communications in Applied Mathematics and Computational Science*, 5(1):81–97, 2010.

[54] Seongjai Kim. An \calo(n) level set method for eikonal equations. *SIAM journal on scientific computing*, 22(6):2178–2193, 2001.

[55] Peter Schwartz, Julie Percelay, Terry Ligocki, Hans Johansen, Daniel Graves, Dharshi Devendran, Phillip Colella, and Eli Ateljevich. High-accuracy embedded boundary grid generation using the divergence theorem. *Communications in Applied Mathematics and Computational Science*, 10(1):83–96, 2015.

[56] Phillip Colella, Daniel T Graves, Benjamin J Keen, and David Modiano. A cartesian grid embedded boundary method for hyperbolic conservation laws. *Journal of Computational Physics*, 211(1):347–366, 2006.

[57] D. T. Graves, P. Colella, D. Modiano, J. Johnson, B. Sjogreen, , and X. Gao. A cartesian grid embedded boundary method for the compressible Navier Stokes equations. *Comm. App. Math. Comp. Sci.*, 8(1):99–122, 2013.

Appendix A

ARK Weights

A.1 Explicit Weights

Stage weights of the ERK (explicit) portion of the 2-ARK$_4$(3)6L[2]SA method as given by Kennedy [31].

Table A.1: The stage weights $a_{ij}^{[E]}$ and b_i for the ERK portion of the 2-ARK$_4$(3)6L[2]SA ARK method.

c_i	$a_{ij}^{[E]}$					
0	0	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0	0
$\frac{83}{250}$	$\frac{13861}{62500}$	$\frac{6889}{62500}$	0	0	0	0
$\frac{31}{50}$	$\frac{-116923316275}{2393684061468}$	$\frac{-2731218467317}{15368042101831}$	$\frac{9408046702089}{11113171139209}$	0	0	0
$\frac{17}{20}$	$\frac{-451086348788}{2902428689909}$	$\frac{-2682348792572}{7519795681897}$	$\frac{12662868775082}{11960479115383}$	$\frac{3355817975965}{11060851509271}$	0	0
1	$\frac{647845179188}{3216320057751}$	$\frac{73281519250}{8382639484533}$	$\frac{552539513397}{3454668386233}$	$\frac{3354512671639}{8306763924573}$	$\frac{4040}{17871}$	0
b_i	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$\frac{-2260}{8211}$	$\frac{1}{4}$

A.2 Implicit Weights

Stage weights of the ESDIRK (implicit) portion of the 2-ARK$_4$(3)6L[2]SA method as given by Kennedy [31].

Table A.2: The stage weights $a_{ij}^{[I]}$ and b_i for the ESDIRK portion of the 2-ARK$_4$(3)6L[2]SA ARK method.

c_i	$a_{ij}^{[I]}$					
0	0	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0
$\frac{83}{250}$	$\frac{8611}{62500}$	$\frac{-1743}{31250}$	$\frac{1}{4}$	0	0	0
$\frac{31}{50}$	$\frac{5012029}{34652500}$	$\frac{-654441}{2922500}$	$\frac{174375}{388108}$	$\frac{1}{4}$	0	0
$\frac{17}{20}$	$\frac{15267082809}{155376265600}$	$\frac{-71443401}{120774400}$	$\frac{730878875}{902184768}$	$\frac{2285395}{8070912}$	$\frac{1}{4}$	0
1	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$\frac{-2260}{8211}$	$\frac{1}{4}$
b_i	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$\frac{-2260}{8211}$	$\frac{1}{4}$

Appendix B

Pseudocodes

B.1 Single Fluid Algorithms

Algorithm 1 Inviscid all-speed single fluid algorithm for a single time step.

1: Initialize the state $\boldsymbol{U}^{(1)} = \boldsymbol{U}^n$
2: **for** $i = 2$ **to** $\mathrm{N} = 6$ **do**
3: Evaluate the RHS of the governing equations
4: Compute the explicit ARK stage solution

$$\boldsymbol{R}^{(i)} = \boldsymbol{U}^n + \Delta t \sum_{j=1}^{i-1} \left(\alpha_{i,j}^{[E]} \boldsymbol{F}^{[E]}(\boldsymbol{U}^{(j)}) + \alpha_{i,j}^{[I]} \boldsymbol{F}^{[I]}(\boldsymbol{U}^{(j)}) \right) \tag{B.1}$$

5: Set the density stage solution $\rho^{(i)} = R_\rho^{(i)}$
6: Evaluate the $p^{(i)}$ solution by solving the Helmholtz equation

$$\left(\frac{1}{K} - (\alpha_{ii}^{[I]} \Delta t) \boldsymbol{\nabla} \cdot \frac{1}{\rho^{(i)}} \right) p^{(i)} = \frac{R_p^{(i)}}{K} - \alpha_{ii}^{[I]} \Delta t \frac{1}{\rho} \boldsymbol{\nabla} \cdot R_{\boldsymbol{u}_p}^{(i)} \tag{B.2}$$

7: Compute and add the stage implicit flux to $\boldsymbol{u}$ and $\boldsymbol{u}_p$

$$\boldsymbol{u}_p{}^{(i)} = R_{\boldsymbol{u}_p}^{(i)} - \alpha_{ii}^{[I]} \Delta t \frac{1}{\rho^{(i)}} \boldsymbol{\nabla} p^{(i)} \tag{B.3}$$

$$\boldsymbol{u}^{(i)} = R_{\boldsymbol{u}}^{(i)} - \alpha_{ii}^{[I]} \Delta t \frac{1}{\rho^{(i)}} \boldsymbol{\nabla} p^{(i)} \tag{B.4}$$

8: **end for**
9: Compute the solution at $t = (n+1)\Delta t$

$$\boldsymbol{U}^{n+1} = \boldsymbol{U}^n + \Delta t \sum_{i=1}^{6} \left(b_i^{[E]} \boldsymbol{F}^{[E]}(\boldsymbol{U}^{(i)}) + b_i^{[I]} \boldsymbol{F}^{[E]}(\boldsymbol{U}^{(i)}) \right) \tag{B.5}$$

10: Enforce the pressure and velocity constraints

$$\left[\frac{1}{K} - \Delta t^2 \boldsymbol{\nabla} \cdot \left(\frac{1}{\rho^{n+1}} \boldsymbol{\nabla} \right) \right] \delta p^{n+1} = \frac{1}{K} \eta \Delta t (p_0^{n+1} - \tilde{p}^{n+1}) \tag{B.6}$$

$$\boldsymbol{u}_p{}^{n+1} = \mathbb{Q}(\tilde{\boldsymbol{u}_p}{}^{n+1} + \delta \boldsymbol{u}_p{}^{n+1}), \tag{B.7}$$

$$p^{n+1} = \tilde{p}^{n+1} + \delta p^{n+1}, \tag{B.8}$$

$$\boldsymbol{u}^{n+1} = A_{\boldsymbol{i}}^{FC}(\boldsymbol{u}_p{}^{n+1}) + \mathbb{P}(\tilde{\boldsymbol{u}}^{n+1}). \tag{B.9}$$

Algorithm 2 Inviscid all-speed single fluid algorithm for a single time step.

1: Initialize the state $\boldsymbol{U}^{(1)} = \boldsymbol{U}^n$
2: **for** $i = 2$ **to** $\mathrm{N} = 6$ **do**
3: Evaluate the RHS of the governing equations
4: Compute the explicit ARK stage solution

$$\boldsymbol{R}^{(i)} = \boldsymbol{U}^n + \Delta t \sum_{j=1}^{i-1} \left(\alpha_{i,j}^{[E]} \boldsymbol{F}^{[E]}(\boldsymbol{U}^{(j)}) + \alpha_{i,j}^{[I]} \boldsymbol{F}^{[I]}(\boldsymbol{U}^{(j)}) \right) \tag{B.10}$$

5: Set the density stage solution $\rho^{(i)} = R_\rho^{(i)}$
6: Compute the stage solution for velocity by solving the equation

$$\left[\rho^{(i)} - \tilde{\alpha}_{ii} \boldsymbol{\nabla} \cdot \left(\mu(\boldsymbol{\nabla} + \boldsymbol{\nabla}^T) + \mu \left(\bar{\zeta} - \frac{2}{3} \right) \boldsymbol{I} \boldsymbol{\nabla} \cdot \right) \right] \boldsymbol{u}^{(i)} = \rho^{(i)} R_{\boldsymbol{u}}^{(i)} \tag{B.11}$$

7: Evaluate the $p^{(i)}$ solution by solving the Helmholtz equation

$$\left[\frac{1}{K} - \bar{\alpha}_{ii}^2 \boldsymbol{\nabla} \cdot \left(\frac{1}{\rho^{(i)}} \boldsymbol{\nabla} \right) \right] p^{(i)} = \frac{R_p^{(i)}}{K} - \tilde{\alpha}_{ii} \boldsymbol{\nabla} \cdot \left(R_{\boldsymbol{u}_p}^{(i)} + \tilde{\alpha}_{ii} \frac{1}{\rho^{(i)}} \boldsymbol{\nabla} \cdot \boldsymbol{\tau}^{(i)} \right) \tag{B.12}$$

8: Compute and add the stage implicit flux to $\boldsymbol{u}_p$

$$\boldsymbol{u}_p{}^{(i)} = R_{\boldsymbol{u}_p}^{(i)} - \alpha_{ii}^{[I]} \Delta t \frac{1}{\rho^{(i)}} \boldsymbol{\nabla} p^{(i)} \tag{B.13}$$

9: **end for**
10: Compute the solution at $t = (n+1)\Delta t$

$$\boldsymbol{U}^{n+1} = \boldsymbol{U}^n + \Delta t \sum_{i=1}^{6} \left(b_i^{[E]} \boldsymbol{F}^{[E]}(\boldsymbol{U}^{(i)}) + b_i^{[I]} \boldsymbol{F}^{[E]}(\boldsymbol{U}^{(i)}) \right) \tag{B.14}$$

11: Enforce the pressure and velocity constraints

$$\left[\frac{1}{K} - \Delta t^2 \boldsymbol{\nabla} \cdot \left(\frac{1}{\rho^{n+1}} \boldsymbol{\nabla} \right) \right] \delta p^{n+1} = \frac{1}{K} \eta \Delta t (p_0^{n+1} - \tilde{p}^{n+1}) \tag{B.15}$$

$$\boldsymbol{u}_p{}^{n+1} = \mathbb{Q}(\tilde{\boldsymbol{u}}_p{}^{n+1} + \delta \boldsymbol{u}_p{}^{n+1}), \tag{B.16}$$

$$p^{n+1} = \tilde{p}^{n+1} + \delta p^{n+1}, \tag{B.17}$$

$$\boldsymbol{u}^{n+1} = A_{\boldsymbol{i}}^{FC}(\boldsymbol{u}_p{}^{n+1}) + \mathbb{P}(\tilde{\boldsymbol{u}}^{n+1}). \tag{B.18}$$

B.2 Two-Fluid Algorithms

Algorithm 3 Gradient descent initialization.

if $\phi_{\boldsymbol{i}} \neq 0$ **then**
  while $\phi_{\boldsymbol{i}}\phi_{\boldsymbol{x}} > 0$ **do**
    $\boldsymbol{x} = \boldsymbol{x} - q_{\boldsymbol{i}} \frac{\boldsymbol{\nabla}\phi_{\boldsymbol{x}}}{||\boldsymbol{\nabla}\phi_{\boldsymbol{x}}||} h$
    $\boldsymbol{\nabla}\phi_{\boldsymbol{x}} = \mathcal{Q}(\boldsymbol{\nabla}\phi, \boldsymbol{x})$
    $\phi_{\boldsymbol{x}} = \mathcal{Q}(\phi, \boldsymbol{x})$
  end while
  if $\phi_{\boldsymbol{i}} > 0$ **then**
    $\boldsymbol{a} = h\left(\boldsymbol{i} + \frac{1}{2}\Upsilon\right),\ \boldsymbol{b} = \boldsymbol{x}$
  else
    $\boldsymbol{a} = \boldsymbol{x},\ \boldsymbol{b} = h\left(\boldsymbol{i} + \frac{1}{2}\Upsilon\right)$
  end if
  $\boldsymbol{y} = (\boldsymbol{a} - \boldsymbol{b})$
  $\boldsymbol{\nabla}\psi_{\boldsymbol{i}}^{\text{GDI}} = \frac{\boldsymbol{y}}{||\boldsymbol{y}||}$
  $\psi_{\boldsymbol{i}}^{\text{GDI}} = \mathcal{D}(\boldsymbol{\nabla}\psi_{\boldsymbol{i}}^{\text{GDI}})$
end if

Algorithm 4 Kim's global marching method extending ψ from some initial band.

```
Ω^X = ∅, Ω^ν = ∅
σ = 1/(2√5)
r = ε + σΔx
n = 0
while r ≤ R do
    Ω^X = ⋃_{s:|s_d|≤1} (Ω^valid + s) − Ω^valid
    for i ∈ Ω^ν do
        if ℰ(ψ, Δx, Ω^valid, i) ∃ then
            (ψ_i, ∇ψ_i) = ℰ(ψ, Δx, Ω^valid, i)
            if |ψ_i| ≤ r then
                Ω^ν = Ω^ν ∪ i
                n += 1
            end if
        end if
    end for
    Ω^valid = Ω^valid ∪ Ω^ν
    Ω^X = ∅, Ω^ν = ∅
    if n = 0 then
        r += σΔx
    end if
    n = 0
end while
```

Algorithm 5 Extension of s for the evolution of ψ.

```
for i ∈ Ω^valid do
    x_Γ = ℛ(∇ψ_i)
    u_{x_Γ} = 𝒬(u, x_Γ)
    s_i = u_{x_Γ} · ∇ψ_i
end for
```

Algorithm 6 Multifluid density update algorithm.

1: **for** $j = \text{fluid}a, \text{fluid}b$ **do**
2: **for** $\boldsymbol{i} \in \Omega$ **do**
3: Compute $\kappa_{\boldsymbol{i}}^{j}$ and $\bar{\alpha}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}^{j}$
4: Evaluate the conservative and non-conservative fluxes

$$(\boldsymbol{\nabla} \cdot \boldsymbol{F}^j)^{NC} = \frac{1}{\Delta t}\left(\rho^{j,\text{ref}} - \rho^{j,n}\right) = 0, \tag{B.19}$$

$$(\overline{\boldsymbol{\nabla} \cdot \boldsymbol{F}^j})_{\boldsymbol{i}}^{C} = \frac{1}{\kappa_{\boldsymbol{i}}^{j,n+1}\Delta t}\left(\kappa_{\boldsymbol{i}}^{j,n+1}\rho_{\boldsymbol{i}}^{j,\text{ref}} - (\kappa\rho)_{\boldsymbol{i}}^{j,n}\right) + \frac{1}{\kappa_{\boldsymbol{i}}^{j,n+1}h}\sum_{d=0}^{\text{D}-1}\left((\bar{\alpha}^j F_d^j)_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - (\bar{\alpha}^j F_d^j)_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}\right) \tag{B.20}$$

5: Compute the initial hybridized density update, $\tilde{\rho}_{\boldsymbol{i}}^{j}$

$$\tilde{\rho}_{\boldsymbol{i}}^{j} = \rho_{\boldsymbol{i}}^{j,\text{ref}} - \Delta t\left(\eta_{\boldsymbol{i}}(\overline{\boldsymbol{\nabla} \cdot \boldsymbol{F}^j})_{\boldsymbol{i}}^{C} + (1-\eta_{\boldsymbol{i}})(\boldsymbol{\nabla} \cdot \boldsymbol{F}^j)_{\boldsymbol{i}}^{NC}\right) \tag{B.21}$$

6: Find the mass difference between the conservative and hybridized upate

$$\delta M_{\boldsymbol{i}}^{j} = -\Delta t \kappa_{\boldsymbol{i}}^{j,n+1}(1-\eta_{\boldsymbol{i}})\left((\overline{\boldsymbol{\nabla} \cdot \boldsymbol{F}^j})_{\boldsymbol{i}}^{C} - (\boldsymbol{\nabla} \cdot \boldsymbol{F}^j)_{\boldsymbol{i}}^{NC}\right) \tag{B.22}$$

7: Redistribute the mass increment $\delta M_{\boldsymbol{i}}^{j}$

$$\rho_{\boldsymbol{i}'}^{j,n+1} = \tilde{\rho}_{\boldsymbol{i}'}^{j} + w_{\boldsymbol{i},\boldsymbol{i}'}^{j}\delta M_{\boldsymbol{i}}^{j}, \tag{B.23}$$

$$w_{\boldsymbol{i},\boldsymbol{i}'}^{j} = \frac{1}{\sum_{\boldsymbol{i}' \in N(\boldsymbol{i})} \kappa_{\boldsymbol{i}'}^{j,n+1}}, \tag{B.24}$$

$$N(\boldsymbol{i}) = \{\boldsymbol{i}' : |\boldsymbol{i}' - \boldsymbol{i}|_\infty \leq 1\} \tag{B.25}$$

8: **end for**
9: **end for**

Algorithm 7 Multifluid correction to pressure and the level-set

1: Compute the density update with Algorithm 6
2: **for** $\boldsymbol{i} \in \Omega$ **do**
3: **for** $j = \{\text{fluid}a, \text{fluid}b\}$ **do**
4: Compute the thermodynamic pressure and normalized bulk modulus

$$p^{\text{EOS},j} = p^{0,0} + K^j \frac{(\rho^j - \rho^{j,0})}{\rho^{j,0}}, \tag{B.26}$$

$$\bar{K}^j = \frac{K^j}{p^{\text{EOS},j}} \tag{B.27}$$

5: **end for**
6: Compute the total thermodynamic pressure and normalized bulk modulus

$$\bar{K} = \frac{1}{\frac{\kappa^a}{\bar{K}^a} + \frac{\kappa^b}{\bar{K}^b}}, \tag{B.28}$$

$$p^{\text{EOS}} = \left(\kappa^a \frac{p^{\text{EOS},a}}{\bar{K}^a} + \kappa^b \frac{p^{\text{EOS},b}}{\bar{K}^b} \right) \bar{K} \tag{B.29}$$

7: Evaluate the signed distance correction

$$\delta s_{\boldsymbol{i}} = \frac{1}{\Delta t} \left(\frac{\kappa^b V^{\Upsilon}}{\bar{K}^b \ell} \frac{p^{\text{EOS},b} - p^{\text{EOS}}}{p^{\text{EOS}}} \right)_{\boldsymbol{i}} \tag{B.30}$$

8: **end for**
9: **for** $\boldsymbol{i} \in \Omega$ **do**
10: Extend the δs correction using Algorithm 5
11: Evolve ψ with the correction

$$\psi^{n+1} = \tilde{\psi}^{n+1} + \sigma \delta s \Delta t, \tag{B.31}$$

12: Correct the single fluid pressure with the total multifluid pressure

$$\left[\frac{1}{K} - \Delta t^2 \boldsymbol{\nabla} \cdot \left(\frac{1}{\rho^{n+1}} \boldsymbol{\nabla} \right) \right] \delta p^{n+1} = \frac{1}{K} \eta \Delta t (p^{\text{EOS}} - \tilde{p}^{n+1}), \tag{B.32}$$

$$\delta \boldsymbol{u}_p{}^{n+1} = -\Delta t \frac{1}{\rho^{n+1}} \boldsymbol{\nabla} \delta p^{n+1}, \tag{B.33}$$

$$p^{n+1} = \tilde{p}^{n+1} + \delta p^{n+1}, \tag{B.34}$$

$$\boldsymbol{u}_p{}^{n+1} = \mathbb{Q} \left(\tilde{\boldsymbol{u}}_p{}^{n+1} + \delta \boldsymbol{u}_p{}^{n+1} \right), \tag{B.35}$$

$$\boldsymbol{u}^{n+1} = A^{FC}(\boldsymbol{u}_p{}^{n+1}) + \mathbb{P}(\tilde{\boldsymbol{u}}^{n+1}), \tag{B.36}$$

13: **end for**
14: Reevaluate ρ^a and ρ^b using Algorithm 6