NUMERICAL METHODS FOR SIMULATING MULTIPHASE FLOWS WITH A FOCUS ON ATOMIZATION

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Mark Francis Owkes

August 2014

NUMERICAL METHODS FOR SIMULATING MULTIPHASE FLOWS WITH A FOCUS ON ATOMIZATION

Mark Francis Owkes, Ph.D.

Cornell University 2014

In this dissertation, numerical methods useful for the simulation of gas-liquid multiphase flows are presented. Multiphase flows are commonly found throughout nature, human life, and engineering devices. As a result, accurate and predictive simulations of such flows will improve our understanding of these complex systems and aid in the development of more efficient engineering devices that exploit multiphase dynamics.

The majority of multiphase flows dynamics occur at the gas-liquid interface. For example, many quantities (e.g., density and species concentrations) are discontinuous at the interface and surface tension is a singular force that acts at the interface. Therefore, accurately tracking the location of the interface, sharply handling discontinuities, and computing accurate interface curvature are critical components for predictive simulations of multiphase flows.

In this work, two novel interface tracking strategies are proposed and tested. The methods extend the capabilities of both level set and volume-of-fluid (VOF) methods, which are commonly used interface capturing methodologies. A discretely consistent methodology is presented to transport VOF and additional quantities that may be discontinuous at the phase interface. By using the same transport scheme for the phase interface and the quantities, discrete conservation and second-order solution of the conservation laws is achieved. An improvement is proposed to the height function method, which is often used to compute the cur-

vature in VOF simulations. Additionally, the height function method is extended to compute the curvature in the context of a conservative level set.

These methods are used to simulate atomization, an important process in the combustion of liquid fuels. Namely, a liquid jet in cross-flow, an air-blast atomizer, and an electrically charged spray, are simulated and the results are compared to available experimental data. Qualitative comparisons of the spray appearance as well as quantitative measures of the spray penetration, drop size distributions, and droplet velocity distributions show that the simulations are capable of predicting the spray characteristics and are a viable tool in the engineering design process. Furthermore, the simulations provide a wealth of data that is useful for improving our understanding of multiphase flow systems.

BIOGRAPHICAL SKETCH

Mark Owkes grew up in Munnsville, a small town in central New York. Mark went to Clarkson University in Potsdam, NY to study Mechanical Engineering at the Bachelor level. While there he participated in an Honor's Program research project looking at the feasibility of contra-rotating vertical axis wind turbine under the direction of Dr. Kenneth Visser. After earning his Bachelor of Science from Clarkson in 2008, Mark headed west and joined the Department of Mechanical Engineering at the University of Colorado at Boulder. While there he worked under Dr. Olivier Desjardins on developing numerical methods to simulate multiphase flows. Mark earned his Master of Science degree from CU Boulder in 2011. At that point, he followed his advisor back east to Cornell University in Ithaca, NY to finish his doctoral program. In 2014, Mark will earn his Ph.D. from the Sibley School of Mechanical and Aerospace Engineering. Mark plans to continue his academic career as an assistant professor at Montana State University in Bozeman, MT.

Mark's research interests include the development of numerical methods for capturing gas-liquid interfaces in multiphase flow simulations. He is interested in simulations of primary atomization to gain insight into the physical phenomena important in the break-up of a liquid jet into droplets. Notably, he has developed both a level set and a volume-of-fluid interface capturing scheme that improve the accuracy and conservation properties of such methods.

This dissertation is dedicated to Denali and all other young scientists.

May your curiosity never wane.

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the support, commitment, and effort of a number of important people in my life. I would like to thank everyone who has contributed in so many ways. In particular I would like to thank my committee members Dr. Olivier Desjardins, Dr. Philip Liu, and Dr. Jane Wang. Additionally this work would not be possible without the financial support of the National Science Foundation CBET 1034506, Office of Naval Research SBIR Ph.1 N68335-10-C-0263, and General Electric Company.

I have had the opportunity to be the first Ph.D. student of Dr. Olivier Desjardins. Even though Olivier was an assistant professor working toward tenure while I was under his tutelage, he never pressured me to work harder just for the sake of producing results. Additionally, Olivier continually strives to develop the best numerical methods to study multiphase flows. This created an atmosphere that pushed me to the state-of-the-art of numerical methods, allowed me to develop as an independent researcher, and investigate tangent ideas in my research. I have really enjoyed working under Olivier and thank him for all his help and guidance over the past six years.

My Ph.D. work was made significantly easier due to the support of a great research group. I want to thank Jeremy for always being there when I needed someone to chat with and work though a problem. Jesse for bringing so much enthusiasm to the group. Bret for pushing all of us to do our best and to get involved with the graduate program. Peter for so many fruitful discussions about tetrahedra. Sunil, John, Houssem, Stephanie, Sheng, and Neola, thanks for bringing new ideas and interest into the research group.

During my Ph.D. I got married and had a daughter. This has been such a wonderful part of my life. I couldn't ask for anything more from my wife Kathleen.

She has been so supportive and allowed me to do the work that had to get done while always encouraging us to enjoy the outdoors and other parts of life. The last year, when we became parents has been particularly amazing. Thank you Kathleen for being at my side and I look forward to the next step of our lives.

I would like to thank my family for their support. Even though going to graduate school took me across the country, my parents were always encouraging and supportive that I was pursuing higher education. Luckily for them, my adviser brought me to Cornell and close to my family and we have had a lot of fun during the past two years.

Throughout my graduate work I have had a terrific network of friends. When I moved to Colorado to go to graduate school I had never played Catan or gone backpacking. Now I can say I have done a lot of both of those things and many others. And it is all because I had a group of friends that wanted to get up and go and invited me along for the ride. I would particularly like to thank Tim and Kathleen who invited me to play volleyball when I first moved to Colorado and then became very close friends. I also want to thank Russel for being interested and excited about everything, Rebecca for enjoying life, Scott for diving for every volleyball, Shanon for amazing dinners, and Lauren and Kirk for being the best dog sitters. I also want to thank Brad, Paul, and Pearl for many great board games. To all of you and to the countless others that are my friends: "Thank you".

TABLE OF CONTENTS

Biographical Sketch . . . iii
Dedication . . . iv
Acknowledgements . . . v
Table of Contents . . . vii
List of Tables . . . x
List of Figures . . . xi

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Overview of previous experimental work . . . 3
1.3 Overview of previous numerical work . . . 5
1.4 Contributions . . . 9
1.5 Organization of this document . . . 10

2 Discontinuous Galerkin Conservative Level **12**
2.1 Introduction . . . 12
2.2 Mathematical formulation . . . 16
2.2.1 Classical level set . . . 16
2.2.2 Conservative level set . . . 17
2.3 Discontinuous Galerkin implementation . . . 20
2.3.1 DG formulation . . . 20
2.3.2 DG level set transport . . . 23
2.3.3 DG level set reinitialization . . . 25
2.3.4 Numerical stability . . . 29
2.3.5 Minimum/maximum preserving limiter . . . 30
2.4 Level set time advancement . . . 34
2.5 Interface normal and curvature . . . 36
2.5.1 Spurious velocities . . . 42
2.6 Validation . . . 43
2.6.1 Zalesak's disk . . . 43
2.6.2 Two-dimensional deformation . . . 46
2.6.3 Standing wave . . . 51
2.6.4 Kelvin-Helmholtz instability . . . 54
2.7 Conclusions . . . 62

3 Conservative Second-Order Geometric Volume-of-Fluid Method **64**
3.1 Introduction . . . 64
3.2 Mathematical formulation . . . 69
3.2.1 Problem setup and notations . . . 70
3.2.2 Flux velocity . . . 75
3.2.3 Liquid volume fraction transport . . . 75
3.3 Computational geometry toolbox . . . 76

3.3.1 Interface reconstruction . . . 76
3.3.2 Discrete representation of the flux volume . . . 78
3.3.3 Construction of conservative fluxes . . . 89
3.3.4 Parallelization . . . 94
3.3.5 Extension to unstructured meshes . . . 95
3.3.6 Implementation . . . 96
3.4 Verification tests . . . 99
3.4.1 Zalesak's disk . . . 99
3.4.2 Two-dimensional deformation . . . 103
3.4.3 Three-dimensional deformation . . . 106
3.4.4 Droplet in homogeneous isotropic turbulence . . . 109
3.5 Conclusions . . . 111
3.6 Additional algorithms . . . 111

4 Transport of Quantities With Discontinuities 114
4.1 Mathematical formulation . . . 114
4.2 Numerical methods . . . 115
4.2.1 Convective fluxes . . . 116
4.2.2 Additional fluxes . . . 120
4.2.3 Source term . . . 121
4.2.4 Implicit formulation . . . 122
4.3 Verification tests . . . 122
4.3.1 Discontinuous scalar transport test . . . 122
4.3.2 Discontinuous scalar diffusion test . . . 124
4.4 Conclusions . . . 126
4.A Analytic solution to diffusion in a cylinder . . . 127

5 Height Function Interface Curvature Calculation 130
5.1 Introduction . . . 130
5.2 Methodology . . . 135
5.3 Verification tests . . . 139
5.3.1 Circle test case . . . 139
5.3.2 Sphere test case . . . 148
5.4 Validation tests . . . 150
5.4.1 Solution of the Navier-Stokes equations . . . 151
5.4.2 Spurious-currents test case . . . 152
5.4.3 Standing-wave test case . . . 155
5.5 Conclusions . . . 157

6 Simulations of primary atomization 158
6.1 Liquid jet in cross-flow . . . 159
6.1.1 Introduction . . . 159
6.1.2 Simulation setup . . . 159
6.1.3 Simulation results . . . 162

6.2 Air-blast n-dodecane atomization 170
6.2.1 Geometry and Numerical setup 170
6.2.2 Shear instability results 172
6.2.3 Drop characteristics 174
6.3 Electrohydrodynamic assisted atomization 176
6.3.1 Mathematical Formulation 178
6.3.2 Numerical methods 181
6.3.3 Simulation results 183
6.4 Conclusions 189

7 Conclusions **190**
7.1 Future work 192

Bibliography **195**

LIST OF TABLES

1.1 Percentage of energy supply by fuel type in United States [1]. . . . 2

2.1 Capillary numbers for different Laplace numbers in spurious velocities test 43
2.2 Effect of mesh on capillary number in spurious velocities test . . . 43
2.3 Non-dimensional numbers used to setup the four cases used in the study of the Kelvin-Helmholtz instability. 60

3.1 Error norms for the transport of Zalesak's disk simulations. 103
3.2 Error norms for the two-dimensional deformation test. 106
3.3 Comparison of proposed scheme and EMFPA of López et al. [2] . . 106
3.4 Comparison of proposed scheme and Hernández et al. [3] 108
3.5 Error norms for the droplet in homogeneous isotropic turbulence test case. 111

5.1 Capillary number and time per time-step for various Laplace numbers. 154
5.2 Capillary number for Laplace number of 12,000 on various meshes using the combined method. 154

6.1 Non-dimensional properties for liquid jet in cross-flow 162
6.2 Properties of n-dodecane and nitrogen. 171
6.3 Flow parameters for the test case. 172
6.4 Non-dimensional numbers used in the charged kerosene jet simulations. 184
6.5 Physical parameters in charged kerosene jet simulations. 185

LIST OF FIGURES

2.1 Comparison of conservative and classical level set with liquid volume fraction . . . 18
2.2 Stencils used to compute curvature . . . 38
2.3 Convergence of interface normal and curvature . . . 41
2.4 Zalesak's disk on various meshes . . . 44
2.5 Zalesak's disk using different DG orders . . . 45
2.6 Zalesak's disk with different amounts of reinitialization . . . 45
2.7 Zalesak's disk after 50 rotations . . . 46
2.8 Time series of deformation test case . . . 47
2.9 Deformation test case with different amounts of reinitialization . . . 47
2.10 Deformation test case on various meshes . . . 48
2.11 Deformation test case with different DG orders . . . 49
2.12 Deformation test case, mass versus time . . . 50
2.13 Standing wave with unity density ratio . . . 53
2.14 Standing wave with density ratio of 1000 . . . 53
2.15 Geometry used for Kelvin-Helmholtz test case . . . 55
2.16 Growth-rates for Kelvin-Helmholtz . . . 61
2.17 Convergence of Kelvin-Helmholtz test case with mesh refinement . . . 62

3.1 Methods used to compute geometric fluxes . . . 67
3.2 Example geometry used to construct flux volumes . . . 71
3.3 Flux volume associated with cell face . . . 74
3.4 Volume-of-Fluid (VOF) representation of interface . . . 77
3.5 Picewise linear interface calculation (PLIC) reconstruction of interface . . . 77
3.6 Partition of two-dimensional fluxes into simplices . . . 79
3.7 Partition of three-dimensional fluxes into simplices . . . 80
3.8 Example of signed simplices representing fluxes . . . 83
3.9 Ordering of vertices used to construct simplices . . . 85
3.10 Shared faces of flux volumes between neighboring cells . . . 86
3.11 Steps used to calculate the liquid volume fraction within a simplex that crosses multiple planes. . . . 88
3.13 Correction for two-dimensional solenoidal flux . . . 94
3.14 Correction for three-dimensional solenoidal flux . . . 94
3.15 Simplex cut by plane . . . 99
3.16 Zalesak's disk on various meshes . . . 102
3.17 Convergence of E_{shape} for Zalesak's disk . . . 103
3.18 Two-dimensional deformation test on various meshes . . . 104
3.19 Shape error for the two-dimensional deformation test . . . 105
3.20 Three-dimensional deformation test on various meshes . . . 107
3.21 E_{shape} for the three-dimensional deformation test . . . 108
3.22 Droplet in homogeneous isotropic turbulence on various meshes . . . 110

3.23 E_{shape} for droplet in homogeneous isotropic turbulence 110

4.1 Initial electric charge density used in discontinuous scalar transport test. 124
4.2 Transported electric charge density within liquid phase for discontinuous scalar transport test case 124
4.3 L_2 error for discontinuous scalar transport test case 125
4.4 Solution for diffusion test case with time 126
4.5 L_2 error for diffusion test case 126

5.1 Example of heights used to compute interface curvature. 133
5.2 Example of mesh-decoupled columns and heights. 135
5.3 Partitioning of a two-dimensional column 137
5.4 Partitioning of a three-dimensional column 137
5.5 The nine columns used to compute curvature in three-dimensions. . 138
5.6 Convergence of curvature error for circle test case. 141
5.7 Convergence of curvature errors for circle test case with analytic heights. 142
5.8 Dependency of curvature error on angular position. 143
5.9 Convergence of curvature error for circle test case using mesh-decoupled and standard methods. 144
5.10 Convergence of curvature error for circle test case with combined method and method of Popinet. 146
5.11 Example of a droplet where heights and widths are not well defined. 147
5.12 Example of how curvature within the shaded cell is computed using the proposed scheme. 147
5.13 Percentage of cells without well-defined heights using the standard method. 147
5.14 Convergence of curvature errors for sphere test case. 149
5.15 Percentage of cells in sphere test case without well-defined heights. 149
5.16 Convergence of curvature error for sphere test case with smaller stencils. 150
5.17 Curvature error on surface of sphere using different methods. . . . 150
5.18 Convergence of capillary number for spurious currents test case. . . 154
5.19 Time evolution of capillary number for spurious currents test case. 154
5.20 Standing wave test case with $\rho_l/\rho_g = 1$. 156
5.21 Standing wave test case with $\rho_l/\rho_g = 1000$ 156
5.22 Convergence of amplitude error for standing wave test case. 156

6.1 Injector geometries used in liquid jet in cross-flow simulations. Liquid flows from bottom to top. 160
6.2 Velocity field at the exit plane (left) and on a cut-plane through (right) the round-edged and sharp-edged injectors. 161

6.3 Rendering of liquid jet in cross-flow produced by the sharp-edged injector . . . 163
6.4 Velocity field within liquid jet in cross-flow produced by the sharp-edged injector . . . 164
6.5 Snapshot of the gas-liquid interface for the liquid jet in cross-flow from the two injector geometries. . . . 165
6.6 Liquid jet in cross-flow penetration. Red line shows experimental correlation for outermost edge from Gopala [4]. . . . 166
6.7 AMD . . . 168
6.8 SMD . . . 168
6.9 Vertical velocity . . . 168
6.10 Spanwise velocity . . . 169
6.11 Streamwise velocity . . . 169
6.12 Scatter plot of eccentricity of droplets versus droplet size. . . . 169
6.13 Geometry of air-blast atomizer. . . . 170
6.14 Air-blast injector dimensions. . . . 171
6.15 Comparison of jet from (a) experiment and (b) simulation. . . . 173
6.16 Example of nozzle wetting and the effect on break-up process. . . . 173
6.17 Measurement of shear instability using photos at two different times during the experiment (a,b) and a rendering of simulation data (c). 175
6.18 Probability density function of (a) drop size and (b) drop velocity using experimental and simulation results. . . . 176
6.19 EHD enhanced atomization . . . 177
6.20 Geometry for simulations of EHD enhanced kerosene atomization. . 184
6.21 Snapshots of the uncharged kerosene jet (top) and the EHD enhanced atomizing jet (bottom) computed using the small computational domain. . . . 185
6.22 Snapshots of EHD enhanced kerosene atomization simulation. . . . 187
6.23 Snapshot of the charged EHD enhanced jet computed on the large domain. . . . 188
6.24 Electric charge density for EHD enhanced kerosene atomization simulation. . . . 188

CHAPTER 1

INTRODUCTION

1.1 Motivation

In a recent report, the International Panel on Climate Change (IPCC) states [1]:

> "Warming of the climate system is unequivocal, and since the 1950s, many of the observed changes are unprecedented over decades to millennia. The atmosphere and ocean have warmed, the amounts of snow and ice have diminished, sea level has risen, and the concentrations of greenhouse gases have increased."

This unprecedented warming and these changes to Earth's climate have clearly been linked with human influences through greenhouse gas emissions [1]. Therefore, society needs to act quickly to curb greenhouse gas emissions or face the consequences of global warming.

Projections show that energy use in the United States will increase from 96 quad Btu to 108 quad Btu in 2040 [5]. The increase is expected in all end-use sectors (industrial, commercial, residential, and transportation) and will be fulfilled using a variety of energy sources. Table 1.1 shows the distribution of energy use by fuel in 2011 and projections in 2040 [5]. The largest fuel type for the foreseeable future is petroleum. Petroleum is a liquid fuel that is converted into mechanical energy through chemical conversion in a combustion chamber. In addition to petroleum, biofuels are also liquid fuels utilized in similar combustion systems. Therefore, improving the efficiency of liquid fuel combustion systems is of utmost importance

if society is going to reduce greenhouse gas emissions while meeting our growing demand for energy [1].

Table 1.1: Percentage of energy supply by fuel type in United States [1].

	2011	2040
Natural gas	26	26
Renewables	8	11
Nuclear	8	9
Coal	20	19
Petroleum	36	32
Biofuels	1	2

The conversion of liquid fuels to mechanical or thermal energy is performed through combustion. Fuel is injected into the combustion chamber and undergoes atomization, which is the process that breaks the coherent fuel jet into droplets. Small droplets evaporate and the fuel vapor undergoes combustion. The atomization process is of great importance since it controls the size and spatial distribution of fuel droplets, consequently their evaporation rate, and therefore the efficiency of the entire combustion process. As a result, improvements to fuel atomization has the potential to significantly reduce the production of greenhouse gases and other harmful pollutants from liquid fuel combustion systems.

Atomization systems have been studied experimentally and with simulations, however the physical processes that control the atomization dynamics are currently not adequately understood. This understanding is needed to make *a priori* estimates of spray dynamics for a new fuel injector. The work in this dissertation aims at advancing numerical techniques such that realistic atomization systems can be studied and probed at a level of detail not attainable with currently available experimental techniques.

In addition to atomization systems, gas-liquid multiphase flows are ubiquitous in many aspects of our lives and throughout nature. Our everyday interaction with liquids almost always involves a gas-liquid interface. Drinking, washing, and swimming are a few common activities that involve multiphase flows. Additionally, many engineering applications depend on multiphase flows such as heat transfer by boiling or condensing. Therefore, while the focus of this dissertation is on atomizing jets, the numerical methodologies are applicable to a wide range of applications.

1.2 Overview of previous experimental work

Atomization is inherently difficult to investigate experimentally. By definition, atomizing systems produce a large number of optically opaque droplets that hinder optical access to the break-up phenomena. As a result, experiments often focus on measuring global spray characteristic such as penetration length and spray angle. Droplets are often spatially separated downstream and far from the injector. In this region, a variety of droplet imaging techniques have been employed to measure the droplet size and velocities. The techniques include particle image velocimetry (PIV) [6] which measures droplet size and velocity through successive images of an region of the spray illuminated with a laser, laser Doppler velocimetry (LDV) [7] is a technique for measuring droplet velocity using the Doppler effect [8]. phase Doppler particle analyzers (PDPA) measure both size and velocity simultaneously and are based on LDV systems. All these spray diagnostic systems are limited when measuring dense sprays where droplets are highly concentrated and cause multiple light scattering events and are likely to have non-spherical shapes [9].

Recent advancements in experimental techniques using ultra high speed X-rays

have allowed research to probe inside the dense region of an atomizing jet [10, 11]. This technique uses X-rays combined with, for example, phase contrast imaging, which exploits differences in the refractive index of different fluids [12] for visualization or small angle X-ray scattering (SAXS) to measure droplet size and velocity [10]. These experiment are performed at facilities like the Advanced Photon Source at Argonne National Laboratory [13]. This facility was funded by the U.S. Department of Energy and cost $497 million [13]. Using this state-of-the-art facility, experiments of atomizing liquid fuels are still challenging. For example, there are logistical challenges to studying atomization under realistic pressurized conditions within expensive government owned facilities.

Experiments of simplified systems have also been studied. For example, Marmottant and Villermaux [14] captured beautiful images of atomizing jets that have significantly lower Reynolds and Weber numbers compared to jets used in fuel injection systems. As a result, the jets have coherent structures that have been related to a progression of instabilities computed using linear stability analysis. Shear between the liquid jet and the surrounding air causes the formation of a Kelvin-Helmholtz type instability. As this instability grows, waves are produced that push against the air causing a Rayleigh-Taylor type instability to form. As this instability grows, ligaments are formed that ultimately break under a Rayleigh-Plateau instability. Measurements of the instability length scales have been compared with predictions from linear stability analysis and reasonable agreement was found. For realistic atomizing flows found in fuel injection systems, turbulence is an important and present flow feature. As a result, linear stability analysis will likely not be able to predict the general features of the flow. Faeth et al. [15] provides a review of the multiphase flow phenomena relevant to spray combustion. In particular they focus on the structure of the dense region near the injector and properties of primary

and secondary breakup. They found the density ratio has a significant effect on the onset of breakup through and the role of aerodynamic phenomena that can enhance breakup.

1.3 Overview of previous numerical work

With continually increasing computational resources and advancements to computational methods, computational fluid dynamics (CFD) is a promising alternative to experiments for studying multiphase flow systems. However, performing multiphase simulations of atomizing liquid jets is challenging for many reasons. A wide range of length scales are present, extending from the large coherent motions down to the smallest droplets. The large scales dictate the spray angle, penetration length, and the initial interface perturbation that can initiate the atomization process. The small scales are important in combustion applications since the small droplets will evaporate quickly and have a significant effect on the combustion dynamics. Simulating these flows requires sufficient resolution to capture small scale features and large enough domains to capture the large scales. The resulting simulations tend to be very large with billions of computational cells.

Other challenges in multiphase simulations arise at the phase interface. Discontinuities in material properties and a discontinuity in pressure due to the surface tension force require special treatment. Many numerical methods have been developed to handle these discontinuities. For example, discontinuities are handled in the continuum surface force (CSF) [16] by smearing them over multiple computational cells. The ghost fluid method (GFM) [17] is an alternative that sharply accounts for discontinuities.

These methods all share the common thread that they require knowledge of where the interface is located. During an atomization process, the phase interface undergoes many topology changes such as deformation, breakup, and merging. Despite these complexities, they can be described by

$$\frac{D\rho}{Dt} = \frac{\partial \rho}{\partial t} + \boldsymbol{u} \cdot \nabla \rho = 0, \tag{1.1}$$

where ρ is the density, $\boldsymbol{u}$ is the velocity field, and t is time[1]. This simple advection equation is challenging to solve because the density is discontinuous at the phase interface.

A variety of numerical techniques have been developed to locate the phase interface. These methods can be broadly classified as interface tracking or interface capturing. The former uses either a mesh that deforms with the interface known as arbitrary Lagrangian-Eulerian (ALE) methods [18–21] or marker and cell (MAC) methods that track the interface with Lagrangian particles [22]. Both of these approaches, while highly accurate, require significant re-meshing or re-seeding when the interface deforms substantially, which is a common occurrence in simulations of atomizing jets. Alternatively, interface capturing methods can be used, such as level-set [23, 24] and volume-of-fluid (VOF) [25–27] schemes that implicitly represent the interface and have been used to simulate atomizing flows.

Level set methods represent the interface as an iso-surface of the level set function [23, 24]. In its basic formulation the level set is defined as a signed distance function. Changing the representation of the interface from the discontinuous density in Eq. 1.1 to the smooth level set function is advantageous. The level set func-

[1] Note that interface topology changes can occur on the molecular scale where the continuum assumption that used to derive the continuity equation is not valid. Furthermore, some interface dynamics, such contact line motion, are not described by the continuity equation. Nonetheless, the continuity equation describe most interface motions very well since it captures all but the very smallest interface processes.

tion can easily be transported using standard discretization methodologies such as finite difference schemes. However, because the level set function is arbitrary and has no physical significance, the method has no inherent mass conservation properties. The introduction of the conservative level set [28–30] greatly improved the conservation properties of level set schemes. The conservative level set is built by choosing the level set function to more closely approximate the discontinuous density, thus recovering physical significance and the associated conservation properties. By using a smoothed step function, the level set function can be smooth enough to be transported easily while sharp enough to approximate the density field. Chapter 2 describes an advancement to the conservative level set method wherein a discontinuous Galerkin discretization is used. The method improves the accuracy of the level set function representation thereby further improving its conservation properties.

Even with the introduction of the conservative level set and the discontinuous Galerkin discretization, exact conservation is unattainable with the level set method. Contrarily, the volume-of-fluid method [31–33] can achieve discrete conservation. VOF methods store the fraction of each computational cell that is within the liquid phase. Using this information a sharp representation of the gas-liquid interface is constructed that can be transported using geometric operations [31, 34, 35]. This framework allows conservative schemes to be constructed. However, for three-dimensional problems the geometric transport operations can become complex. To alleviate this, the three-dimensional transport step is often split into a series of one-dimensional steps [25], but this introduces a splitting error. Un-split schemes deal with complex geometry and were first introduced for two-dimensional simulations [2, 35]. The extension to three-dimensions is more complicated and, to the best of our knowledge, three-dimensional un-split geomet-

ric schemes have only recently been proposed [3,36]. Chapter 3 describes the first conservative, second-order, un-split VOF scheme. The complex geometry is dealt with by reducing the problem to a series of systematic geometric operations that greatly simplifies the implementation.

In many multiphase flow systems additional quantities that are discontinuous at the phase interface are present. In Chapter 4, a conservative second-order methodology is proposed to solve conservation laws for these quantities. The method is built such that discrete consistency is maintained with the interface transport. This is critical in order to avoid spurious over/undershoots and conservation errors at the phase interface.

One of the ongoing challenges of interface tracking schemes is computing an accurate and convergent interface curvature [37]. The curvature is related to the pressure jump at the interface due to the surface tension force and needs to be computed accurately to avoid spurious velocities near the phase interface. Brackbill et al. [16] developed the continuum surface force (CSF) method, which is one of the first methodologies to compute the curvature and handle the surface tension force in a VOF scheme. In the CSF method, the interface curvature is computed from a mollified (smoothed) approximation of the liquid volume fraction. More recently, the height function method [38–40] has been developed that provides a sharp calculation of the curvature. Chapter 2 provides a novel application of the height function methodology to the conservative level set. Chapter 5 describes an improvement to the height function method that allows for accurate and robust curvature calculations on an under-resolved interface where the standard height function method often fails.

1.4 Contributions

The contributions in this dissertation provide advancements to numerical methods to study gas-liquid multiphase flows and the application of those methods to relevant engineering applications. These contributions are the following:

- The mass conservation properties of the level set method is improved by introducing a discontinuous Galerkin discretization. This discretization allows for an arbitrarily high-order representation of the level set function without the need of a large computational stencil. The small stencil ensures the method is highly parallelizable.
- The curvature calculation of the conservative level set method is improved by applying the height function methodology to the conservative level set. The height function method is commonly used in the context of VOF methods but this work shows it can also be used to compute converging, second-order curvatures in the context of the conservative level set.
- A three-dimensional, un-split, geometric VOF method has been developed. This method leverages two key ideas that makes it straightforward to implement. The first is the use of simplices (e.g., triangles or tetrahedra) to greatly simplify the representation of complex shapes. The second is a simple sign convention that identifies the direction of fluxes. The scheme achieves discrete conservation and boundedness of the VOF field and is second-order accurate.
- A discretization for conservation laws of quantities that are discontinuous at the phase interface is proposed and tested. The method is constructed to be discretely consistent with the VOF interface transport scheme ensuring

transport even near the discontinuities is robust, second-order accurate, and conservative.

- A mesh-decoupled interface curvature method has been developed that improves calculations for small interface structures where the standard height function method fails. The method leverages the geometric operations used in the VOF transport and can potentially be implemented easily in a geometric VOF codes that uses computational geometry routines.

1.5 Organization of this document

Chapters 2, 3, and 5 are pre-print copies of journal papers. Each chapter is self-contained and can be read separately. The chapters describe the discontinuous Galerkin discretization of the level set, the three-dimensional conservative second-order VOF method, and the mesh-decoupled height function method, respectively. Presently, the work in Chapters 2 and 3 has been accepted for publication in the Journal of Computational Physics (see [41] and [42]). The work in Chapter 5 has been submitted for publication to the Journal of Computational Physics.

Chapter 4 describes a solution strategy for solving conservation laws for scalars that are discontinuous at the phase interface. The chapter builds on the ideas presented in Chapter 3 but can be read as a standalone document. A journal article describing this work is currently under preparation.

Chapter 6 presents a series of numerical simulations that were performed using the numerical methods described in Chapters 2 – 5. The simulation results have been disseminated through a paper published in Atomization and Sprays [43] and a series of conference papers. The conferences include the International Confer-

ence on Liquid Atomization and Spray Systems (ILASS), the American Institute of Aeronautics and Astronautics (AIAA), and the American Physical Society's Division of Fluid Dynamics (APS-DFD).

CHAPTER 2

DISCONTINUOUS GALERKIN CONSERVATIVE LEVEL

2.1 Introduction

In simulations of multiphase flows, discontinuities at the interface arise from different fluid properties and a jump in pressure due to surface tension. The discontinuities make discretizing the Navier-Stokes equations challenging, consequently numerical methods have been developed to handle these singularities including the continuum surface force (CSF) approach [16] and the ghost fluid method (GFM) [17]. Both the CSF method and the GFM are based on the assumption that the interface location is known accurately. The discontinuous Galerkin conservative level set (DG-CLS) method, presented herein, provides an accurate interface location needed for the CSF method, the GFM, or other chosen method.

Commonly, two approaches are used to locate the interface: interface tracking and interface capturing. Interface tracking schemes typically use either arbitrary Lagrangian-Eulerian (ALE) methods based on a mesh that deforms with the interface [18–21] or marker and cell (MAC) methods that advect Lagrangian particles that define a given fluid by their locations [22]. The main problem with interface tracking schemes occurs when the interface deforms substantially or when the interface disconnects and reconnects. Significant re-meshing or re-seeding of particles is needed to account for the large interface changes.

Interface capturing methods include volume of fluid (VOF) and level set methods. VOF methods capture the interface using the volume fraction of fluid within each grid cell [25–27]. While VOF schemes have excellent mass conservation prop-

erties, they suffer from the challenge of reconstructing the interface location using only the cell integral volume fraction. Level set methods advect a function defined such that the interface is represented by an iso-surface of a scalar field called the level set [23,24]. Level set methods alleviate the problem found with VOF methods of having to reconstruct an interface since the interface is explicitly defined by the function. Although mass conservation is problematic with the classical level set method, the accurate conservative level set (ACLS) [28] offers good mass conservation and a well-defined interface location. Details of the classical and conservative level set methods are given in Section 2.2.

Spatial discretization of the conservative level set, used in the ACLS method, can be achieved with finite difference operators [28]; however, the discontinuous Galerkin (DG) method was chosen in this work for its high accuracy and compact stencil [44, 45]. High accuracy is obtained by projecting the solution onto high-order discontinuous polynomials, similar to finite element methods. Compactness is a result of the local nature of the polynomials. Since the polynomials are defined on each grid cell, updates do not need global information but rather only information from nearest grid cell neighbors. This small stencil results in minimal communication requirements and a highly parallelizable code.

The conservative level set method includes a transport equation that describes the convection of the level set due to the velocity field and a reinitialization equation that maintains the shape of the level set. Cockburn and Shu [46] provide a DG discretization of the transport equation with an accurate temporal integration method and appropriate definition of fluxes. The DG discretization was applied to the classical level set by Marchandise et al. [47]. A quadrature-free implementation was used wherein all the integrals that appear in the weak form of the equations

were precomputed to improve computational efficiency. Marchandise et al. [48] reinitialized the classical level set using a recursive contouring algorithm with a fast search tree method to find the smallest distance to the interface which for the classical level set method is also the value of the level set. However, when the conservative level set is used the level set is not a signed distance function and a different reinitialization method is used. Following the steps of the ACLS method a convective-diffusion equation is solved to reinitialize the level set and maintain the level set profile. We propose to discretize the convective-diffusion equation using DG in order to maintain the high order accuracy of the level set for both the transport and reinitialization steps. Details of the DG implementation are given in Section 2.3 which includes background information on our DG formulation in Section 2.3.1 and the particulars of the spatial discretization of the transport and reinitialization equations in Sections 2.3.2 and 2.3.3, respectively.

The most straightforward method to integrate the transport and reinitialization equations in time is an explicit scheme such as a Runge-Kutta (RK) method. An explicit scheme does not require global communications thereby maintaining the highly parallelizable nature of the DG spatial discretization. Cockburn and Shu [46] provide a description of the RK methods and show many are stable when high order polynomials are used to approximate the solution. The total variation diminishing third order RK (TVD-RK3) method is used in this work; details are provided in Section 2.4.

As described previously, the interface curvature and normal have direct effects on the solution; therefore, the methods used to calculate these interface properties should be accurate and converge under mesh refinement. Obtaining convergence is difficult with the conservative level set because the level set profile is a relatively

sharp approximation of a step function. The sharp profile is used because the smallest mass losses are achieved when the level set profile is the sharpest. Therefore, the sharpest resolvable profile is used resulting in a fixed number of grid cells across the profile. When the mesh is refined, the number of cells across the profile does not change but rather the profile is sharpened. The fixed number of points across the profile makes it difficult to obtain convergence of level set gradients used in the calculation of interface normal and curvature. To acquire a converging normal and curvature, Marchandise et al. [48] proposed a least squares method for the classical level set. The method was latter applied by Desjardins et al [28] to the ACLS method that uses the conservative level set. The least squares method showed second and first order convergence for the normal and curvature, respectively [28]. To improve the convergence of the interface curvature from first to second order, we applied the height function method commonly used in volume of fluid (VOF) methods [49] to the DG-CLS formulation as described in Section 2.5.

In Section 2.6, we provide details of numerical experiments conducted using the DG-CLS method. The tests include normal and curvature convergence, simulations that examine transport of the level set, tests of the level set method coupled with the Navier-Stokes solver, and a realistic application.

2.2 Mathematical formulation

2.2.1 Classical level set

The classical level set, G, represents the interface as the zero iso-surface of a signed distance function, i.e.

$$|G| = |\boldsymbol{x} - \boldsymbol{x}^I|, \tag{2.1}$$

where $\boldsymbol{x}^I$ is the location on the interface that is closest to the coordinate $\boldsymbol{x}$. The level set is defined to be positive on one side of the interface and negative on the other side. By defining the level set using a signed distance function, the interface is naturally represented by the zero iso-surface, $G(\boldsymbol{x}, t) = 0$.

Motion of the interface is achieved by solving the transport equation

$$\frac{\partial G}{\partial t} + \boldsymbol{U} \cdot \nabla G = 0, \tag{2.2}$$

where t is time and $\boldsymbol{U}$ is the velocity field. In addition to providing the interface location, the level set is also used to calculate the interface normal vector, $\boldsymbol{n}$, and curvature, κ, using

$$\boldsymbol{n} = \frac{\nabla G}{|\nabla G|} \tag{2.3}$$

and

$$\kappa = -\nabla \cdot \boldsymbol{n}, \tag{2.4}$$

respectively. Equations 2.3 and 2.4 provide an accurate result when the level set is smooth such as the signed distance function. However, transporting the interface using Eq. 2.2 will alter the smoothness of the level set when the velocity field is not uniform; therefore, a reinitialization step is added to restore the level set to a signed distance function. A variety of methods can be used to reinitialize the

level set to a signed distance function. Fast marching methods involve solving the Eikonal equation $|\nabla G| = 1$ from the interface outwards [24, 50]. Closest point algorithms that use a tree-based structure to determine the closest point that lies on the interface. Another common reinitialization method solves the Hamilton-Jacobi equation [51],

$$\frac{\partial G}{\partial \tau} + S(G)(|\nabla G| - 1) = 0, \tag{2.5}$$

in pseudo-time, τ, until steady state is achieved. In the previous equation, S is a modified sign function such as the function described by Sussman et al. [52].

The classical level set does not represent any physical quantity; therefore, conservation of the signed distance function will not provide conservation of mass or other useful result. When the classical level set is used, the mass of the fluid can change leading to significant errors especially for applications with complex velocity fields and frequent interface topology changes.

2.2.2 Conservative level set

In an effort to add conservation properties to the classical level set scheme, the signed distance function is replaced with a modified hyperbolic tangent function [28–30],

$$\Psi(\boldsymbol{x}, t) = \frac{1}{2}\left(\tanh\left(\frac{G}{2\,\varepsilon}\right) + 1\right), \tag{2.6}$$

where ε sets the thickness of the profile and G is the signed distance function. Note that the conservative level set function represents the interface using the $\Psi = 0.5$ iso-surface.

The conservative level set function mimics the liquid volume fraction which is a Heaviside function as shown in Fig. 2.1. In the limit that ε goes to zero, the

volume under the level set function is equal to the liquid volume fraction as shown by

$$\lim_{\varepsilon \to 0} \int_V \Psi(\boldsymbol{x}, t)\, \mathrm{d}v = \int_V H(\Psi(\boldsymbol{x}, t) - 0.5)\, \mathrm{d}v \tag{2.7}$$

where H is the Heaviside function. The left hand side of the previous equation is the volume under the level set and the right hand side represents the volume within the $\Psi = 0.5$ iso-surface which is the liquid volume fraction due to the definition of the level set function. A useful consequence of Eq. 2.7 is conservation of the conservative level set, Ψ, results in conservation of liquid volume fraction when ε goes to zero. Therefore, ε is chosen to be as small as possible while maintaining reasonable resolution of the level set function resulting in a balance between mass conservation errors and inaccuracies in representing an under-resolved function.

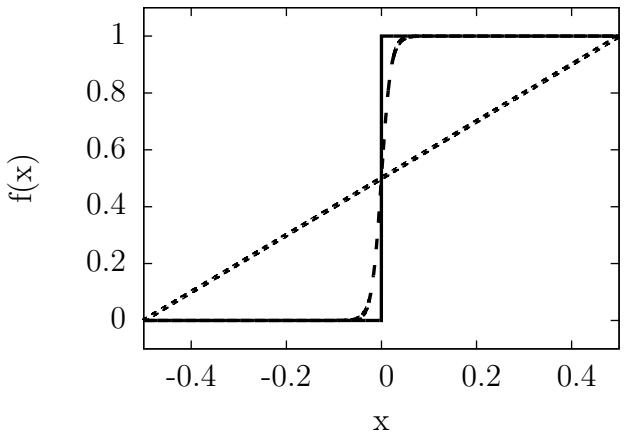


Figure 2.1: Functions showing liquid volume fraction, conservative level set, and classical level set represented by solid, dashed, and dotted lines, respectively.

The conservative level set can be transported using Eq. 2.2 by replacing G with Ψ. Furthermore, in the context of a solenoidal velocity field, i.e. $\nabla \cdot \boldsymbol{U} = 0$, the equation can be written in conservative form

$$\frac{\partial \Psi}{\partial t} + \nabla \cdot (\boldsymbol{U}\Psi) = 0, \tag{2.8}$$

which ensures that Ψ is discretely conserved.

Reinitialization is performed by solving

$$\frac{\partial \Psi}{\partial \tau} + \nabla \cdot (\Psi(1-\Psi)\boldsymbol{n}) = \nabla \cdot (\varepsilon(\nabla \Psi \cdot \boldsymbol{n})\boldsymbol{n}). \tag{2.9}$$

This equation is developed by recognizing that we want our level set function given by Eq. 2.6 as a steady state solution of an equation. In one-dimension we want the steady-state solution to be [29]

$$\frac{\partial \Psi}{\partial n} = \frac{\Psi(1-\Psi)}{\varepsilon}. \tag{2.10}$$

One way to extend the previous equation to three-dimensions is to write Eq. 2.9, which has the property that both the compressive term, shown on the left, and the diffusive term, shown on the right, only act in the direction normal to the interface restricting motion in the direction tangential to the interface. Reinitialization in the tangential direction results in non-physical motions that smooth the interface. Note that the normal, $\boldsymbol{n}$, that appears in Eq. 2.10 is calculated before it is solved and remains constant throughout the reinitialization step. Similarly to solving the Hamilton-Jacobi equation which can be used to reinitialize the classical level set, Eq. 2.9 is advanced in pseudo-time, τ. The steady-state solution is the hyperbolic tangent function described by Eq. 2.6 and occurs when the compressive term is equated by the diffusive term. Both Eqs. 2.8 and 2.9 are written in conservative form making the conservative level set, Ψ, a conserved quantity and minimizing mass conservation errors.

2.3 Discontinuous Galerkin implementation

DG schemes offer a variety of desirable properties when applied to the conservative level set method. DG methods allow for arbitrarily high orders of accuracy without the necessity of a large stencil, which results in a robust, accurate, and highly scalable scheme. The method presented here is separated into a background section on DG followed by two sections that describe the spatial discretization of the level set transport and reinitialization equations, respectively.

2.3.1 DG formulation

The physical domain is represented by Ω with closed boundary Γ. This domain is represented using a computational grid consisting of a collection of Q non-overlapping grid cells referred to as ω_q where $q = 1, \ldots, Q$. The union of all cells is equal to the physical domain, i.e.

$$\Omega = \omega_1 \cup \omega_2 \cup \cdots \cup \omega_Q. \tag{2.11}$$

Each grid cell ω_q has an associated closed boundary γ_q such that the set of all cell boundary parts that are a member of exactly one cell boundary is equal to the closed physical domain boundary, i.e.

$$\Gamma = (\gamma_1 \cup \gamma_2 \cup \cdots \cup \gamma_Q) \setminus (\gamma_1 \cap \gamma_2 \cap \cdots \cap \gamma_Q). \tag{2.12}$$

DG is used to spatially discretize a function onto the computational grid. The main idea is to representing the function using a finite linear combination of basis functions within each grid cell creating a piecewise continuous representation of the function. The basis functions, $\boldsymbol{\phi} = \{\phi_0, \phi_1, \ldots, \phi_P\}$, can be almost any set of basis

functions; however, orthogonal polynomials such as the Legendre polynomials, used in this work, are preferred. An example of a DG approximation of a function is shown below wherein the level set is approximated within the q^{th} grid cell using

$$\Psi(\boldsymbol{x}|\boldsymbol{x} \in \omega_q, t) \approx \Psi_{h,q}(\boldsymbol{x}, t) = \sum_{m=1}^{P} \psi_{m,q}(t)\phi_m(\boldsymbol{\chi}_q(\boldsymbol{x})) = \psi_{m,q}\phi_m, \tag{2.13}$$

where $\Psi_{h,q}$ is the DG approximation of Ψ within the q^{th} grid cell, P is the number of degrees of freedom (defined below), $\psi_{m,q}$ is the weight associated with the m^{th} basis function and q^{th} grid cell, and $\boldsymbol{\chi}_q$ is a coordinate system local to ω_q and defined, for a basis formed using Legendre polynomials, such that $\boldsymbol{\chi}_q = [-1, 1]^3$ within ω_q. By defining $\boldsymbol{\chi}_q$ this way, the orthogonality relations between the Legendre polynomials are maintained, i.e.

$$\int_{\omega_q} \phi_i \phi_j \, \mathrm{d}v = \int_{\omega_q} \phi_i^2 \, \mathrm{d}v \, \delta_{ij} \tag{2.14}$$

where δ_{ij} is the Kronecker delta.

When the weights, $\boldsymbol{\psi}$, need to be calculated, such as during initialization of simulation, the following equation can be solved:

$$\psi_{i,q} = \int_{\omega_q} \Psi \phi_i \, \mathrm{d}v \left(\int_{\omega_q} \phi_i^2 \, \mathrm{d}v \right)^{-1} \tag{2.15}$$

for $i = 1, \ldots, P$ and $q = 1, \ldots, Q$. The previous equation is derived by first writing the weak form of Eq. 2.13, which involves multiplying by the test function, ϕ_s, and integrating over the domain. Next, the weight, $\psi_{i,q}(t)$ is removed from the spatial integral and the orthogonality property shown by Eq. 2.14 is employed to simplify the integral containing the product of two basis functions. Finally, the equation is rearranged to get Eq. 2.15.

Because the basis functions and weights are local to each grid cell, discontinuities of the DG representation of the level set at grid cell boundaries can occur. In

other words, the approximated function can have different values at a boundary between cells depending on if the function is evaluated using data from the cell on one side of the cell boundary or the other. This is why the method is known as *discontinuous* Galerkin.

The following equation can be used to compute one-dimensional Legendre polynomials used in this work for the DG basis functions,

$$\phi_i = \frac{i!}{(2i)!} \frac{\mathrm{d}^i[(x^2-1)^i]}{\mathrm{d}x^i} \tag{2.16}$$

for $i = 0, 1, \ldots, \infty$. Multi-dimensional basis sets can be created by combining the functions from one-dimensional sets. Typically, the set is constrained such that the total order is less than a threshold called the total polynomial order, O. Given a total polynomial order of O provides a $O+1$ order accurate representation of the function being approximated. The number of Legendre basis functions and associated weights for a given O is referred to as the number of degrees of freedom, P, and can be calculated using,

$$P = \frac{1}{d!} \prod_{k=1}^{d} (O+k) = \frac{(O+d)!}{O!d!} \tag{2.17}$$

For reference, the ten Legendre basis functions within the q^{th} grid cell used in a second order ($O = 2$), three-dimensional ($d = 3$) implementation are

$$\boldsymbol{\phi} = [1,\ \chi_{q,1},\ \chi_{q,2},\ \chi_{q,3},\ \chi_{q,1}^2 - 1/3,\ \chi_{q,1}\,\chi_{q,2},\ \chi_{q,1}\,\chi_{q,3},\ \chi_{q,2}^2 - 1/3,\ \chi_{q,2}\,\chi_{q,3},\ \chi_{q,3}^2 - 1/3],$$

where $\boldsymbol{\chi}_q = (\chi_{q,1}, \chi_{q,2}, \chi_{q,3})^t$.

To solve an equation using DG the following steps are used. First, a weak form of the equation is constructed by multiplying by a test function and integrating over the domain. In the DG formulation the test function is taken from the set of basis functions, $\boldsymbol{\phi}$. Integration by parts is used leading to a collection of integrals

over cell volumes and surfaces. Next, the weak form of the equation is spatially discretized onto the grid by substituting the DG approximation, shown in Eq. 2.13 for the level set, into the equation. Because discontinuities can exist at the faces between the cell of interest and neighboring cells, a method that systematically provides one value that respects the physics of the problem must be developed. The result of this method is used in the surface integrals to construct the fluxes. Finally, if the basis functions are orthogonal, the P equations that result from the previous steps can be decoupled using the property shown in Eq. 2.14.

2.3.2 DG level set transport

Transport of the conservative level set is done by solving Eq. 2.8 using the quadrature-free DG method following the work of Marchandise et al. [47]. The first step to obtain the DG discretization of Eq. 2.8 is to write the weak form by multiplying by a test function ϕ_s, integrating over the domain Ω with boundary Γ, and performing a formal integration by parts. The result is

$$\int_\Omega \frac{\partial \Psi}{\partial t}\phi_s \,\mathrm{d}v - \int_\Omega \Psi U_j \frac{\partial \phi_s}{\partial x_j}\,\mathrm{d}v + \oint_\Gamma \Psi \phi_s U_j N_j \,\mathrm{d}s = 0 \tag{2.18}$$

for $s = 1, \ldots, P$, where N_j is the j^{th} component of the domain boundary normal vector $\boldsymbol{N}$ and $\mathrm{d}v$ and $\mathrm{d}s$ represent volume and surface integrals, respectively. Note, Einstein's summation notation is used throughout this paper for any indices that appear twice in a term unless the index is explicitly defined.

Spatial discretization introduces the DG approximation of the level set within each grid cell described using Eq. 2.13 and results in

$$\int_{\omega_q} \frac{\partial \psi_{m,q}\phi_m}{\partial t}\phi_s \,\mathrm{d}v - \int_{\omega_q} \psi_{i,q}\phi_i U_j \frac{\partial \phi_s}{\partial x_j}\,\mathrm{d}v + \oint_{\gamma_q} \widehat{\psi_{i,q}\phi_i U_j} N_j^c \phi_s \,\mathrm{d}s = 0, \tag{2.19}$$

for $s = 1, \ldots, P$ and $q = 1, \ldots, Q$, where $\boldsymbol{N}^c$ is the normal to the cell boundary with j^{th} component N_j^c. This equation is not fully defined since the approximate solution is discontinuous at the cell boundaries and a unique function does not exist when evaluating the surface integral. Therefore, an appropriate method to evaluate the flux, $\widehat{\psi_{i,q}\phi_i U_j} N_j^c$, must be chosen. For the transport equation, the flux can be up-winded based on the sign of $\left(\boldsymbol{U} \cdot \boldsymbol{N}^c\right)_{\gamma_q}$, which is the velocity at the cell boundary projected onto the cell boundary normal vector (interpolation may be necessary if the velocity or normal are not located at the center of the cell face). The flux can be written as

$$\widehat{U_j \psi_{i,q}\phi_i} N_j^c = \left(U_j N_j^c\right)_{\gamma_q} \left(\psi_{i,q}\phi_i\right)_{\text{up}} = \begin{cases} \left(U_j N_j^c\right)_{\gamma_q} \left(\psi_{i,q}\phi_i\right)^{\text{in}} & \text{if } \left(U_j N_j^c\right)_{\gamma_q} > 0 \\ \left(U_j N_j^c\right)_{\gamma_q} \left(\psi_{i,q}\phi_i\right)^{\text{out}} & \text{if } \left(U_j N_j^c\right)_{\gamma_q} < 0, \end{cases} \tag{2.20}$$

where $(*)^{\text{in}}$ is $(*)$ evaluated at the face using information from the cell of interest; analogously, $(*)^{\text{out}}$ is $(*)$ calculated at the face but using information from the neighboring cell. This flux is known as a Roe flux, but as described by Cockburn and Shu [53], any two-point Lipschitz continuous monotone flux is appropriate. Adding the properties $\boldsymbol{\psi} = \boldsymbol{\psi}(t)$, $\boldsymbol{\phi} = \boldsymbol{\phi}(\boldsymbol{\chi}_q)$, and $\boldsymbol{U} = \boldsymbol{U}(t)$ within the q^{th} cell, results in

$$\frac{\partial \psi_{m,q}}{\partial t} \int_{\omega_q} \phi_m \phi_s \, \mathrm{d}v - \psi_{i,q} u_j \int_{\omega_q} \phi_i \frac{\partial \phi_s}{\partial x_j} \, \mathrm{d}v + \left(u_j N_j^c\right)_{\gamma_q} \left(\psi_{i,q}\right)_{\text{up}} \oint_{\gamma_q} \left(\phi_i\right)_{\text{up}} \phi_s \, \mathrm{d}s = 0 \tag{2.21}$$

for $s = 1, \ldots, P$ and $q = 1, \ldots, Q$. The assumption of $\boldsymbol{U}$ being constant within each cell allows for it to be removed from the integral. This assumption is reasonable since a second order accurate Navier-Stokes solver is used in our code. In addition, the surface normal, $\boldsymbol{N}^c$, is a constant on regular grids wherein cell boundaries are composed of a collection of flat faces, and the surface integral is split into multiple integrals on each flat face.

Since our DG formulation is based on orthogonal basis functions as shown by Eq. 2.14, the P coupled equations in Eq. 2.21 can be decoupled and written as

$$\frac{\partial \psi_{m,q}}{\partial t} \int_{\omega_q} \phi_m^2 \, \mathrm{d}v - \psi_{i,q} U_j \int_{\omega_q} \phi_i \frac{\partial \phi_m}{\partial x_j} \, \mathrm{d}v + \left(U_j N_j^c\right)_{\gamma_q} (\psi_{i,q})_{\mathrm{up}} \oint_{\gamma_q} (\phi_i)_{\mathrm{up}} \phi_m \, \mathrm{d}s = 0 \tag{2.22}$$

for $m = 1, \ldots, P$ and $q = 1, \ldots, Q$. If the basis is not orthogonal, the system of P coupled equations described by Eq. 2.21 must be solved.

All the integrals in Eq. 2.22 are now a function of only the basis functions that are defined at the start of a simulation and can be computed once in an initialization routine. In practice on Cartesian meshes, one can compute the integrals over an arbitrary cell size, e.g. $[-1, 1]^3$, and then use a scaling factor to adjust the pre-computed integrals to the size of the cell being updated. Equation 2.22 is the quadrature-free discontinuous Galerkin form of the transport equation that can easily and efficiency be updated once a time integration scheme is chosen as described in Section 2.4.

2.3.3 DG level set reinitialization

Reinitialization is used to maintain the shape of the hyperbolic tangent profile and limit mass loss. Equation 2.9 is solved using a DG discretization applied in a similar fashion as described in the transport section. The steps include construction of a weak form of the reinitialization equation, discretization on the grid that supports the approximate solution, definition of proper fluxes that provide a unique value at faces where discontinuities exist, and finally, decoupling of the equations if an orthogonal set of basis functions is used.

The weak form of Eq. 2.9 is found by multiplying by a test function ϕ_s, inte-

grating over the domain, and performing a formal integration by parts, resulting in

$$\begin{aligned}\int_\Omega \frac{\partial \Psi}{\partial \tau}\phi_s \,\mathrm{d}v - \int_\Omega \big(\Psi - \Psi^2\big) n_j \frac{\partial \phi_s}{\partial x_j}\,\mathrm{d}v + \oint_\Gamma \big(\Psi - \Psi^2\big) n_j N_j \phi_s \,\mathrm{d}s \\ = -\int_\Omega \varepsilon n_j n_k \frac{\partial \Psi}{\partial x_j}\frac{\partial \phi_s}{\partial x_k}\,\mathrm{d}v + \oint_\Gamma \varepsilon \frac{\partial \Psi}{\partial x_j} n_j n_k N_k \phi_s \,\mathrm{d}s\end{aligned} \tag{2.23}$$

for $s = 1, \ldots, P$, where $\boldsymbol{n}$ is the interface normal vector, and $\boldsymbol{N}$ is the domain boundary normal vector. $\boldsymbol{n}$ and $\boldsymbol{N}$ have d dimensional components, and the j^{th} components are represented by n_j and N_j, respectively.

Discretization onto a grid involves integrating over cells and substituting in the DG approximation of the level set shown by Eq. 2.13. The spatially discretized form of Eq. 2.23 in the q^{th} cell is

$$\begin{aligned}\int_{\omega_q} \frac{\partial \psi_{m,q}\phi_m}{\partial \tau}\phi_s \,\mathrm{d}v - \int_{\omega_q} \psi_{i,q}\phi_i n_j \frac{\partial \phi_s}{\partial x_j}\,\mathrm{d}v + \oint_{\gamma_q} \widehat{\psi_{i,q}\phi_i n_j} N_j^c \phi_s \,\mathrm{d}s \\ + \int_{\omega_q} \psi_{i,q}\phi_i \psi_{k,q}\phi_k n_j \frac{\partial \phi_s}{\partial x_j}\,\mathrm{d}v - \oint_{\gamma_q} \widehat{\psi_{i,q}\phi_i \psi_{k,q}\phi_k} n_j N_j^c \phi_s \,\mathrm{d}s \\ = -\int_{\omega_q} \varepsilon n_j n_k \frac{\partial \psi_{i,q}\phi_i}{\partial x_j}\frac{\partial \phi_s}{\partial x_k}\,\mathrm{d}v + \oint_{\gamma_q} \varepsilon \widehat{\frac{\partial \psi_{i,q}\phi_i}{\partial x_j}} n_j n_k N_k^c \phi_s \,\mathrm{d}s\end{aligned} \tag{2.24}$$

for $s = 1, \ldots, P$ and $q = 1, \ldots, Q$, where the cell boundary normal is $\boldsymbol{N}^c$ with j^{th} component N_j^c. The fluxes, shown with hats, i.e. $\widehat{(*)}$, are not fully defined because the solution is discontinuous at the cell boundaries and must be constructed appropriately. The convective fluxes are dealt with similarly to the flux in the transport

equation and are up-winded using the Roe formulation

$$\widehat{\psi_{i,q}\phi_i n_j} N_j^c = (\psi_{i,q}\phi_i)_{\text{up}}(n_j N_j^c)_{\gamma_q}$$
$$= \begin{cases} (\psi_{i,q}\phi_i)^{\text{in}}(n_j N_j^c)_{\gamma_q} & \text{if} \quad (1-2\psi_{i,q}\phi_i)(n_j N_j^c)_{\gamma_q} > 0, \\ (\psi_{i,q}\phi_i)^{\text{out}}(n_j N_j^c)_{\gamma_q} & \text{if} \quad (1-2\psi_{i,q}\phi_i)(n_j N_j^c)_{\gamma_q} < 0, \end{cases} \tag{2.25a}$$

$$\widehat{\psi_{i,q}\psi_{k,q}\phi_i\phi_k} n_j N_j^c = (\psi_{i,q}\psi_{k,q}\phi_i\phi_k)_{\text{up}}(n_j N_j^c)_{\gamma_q}$$
$$= \begin{cases} (\psi_{i,q}\psi_{k,q}\phi_i\phi_k)^{\text{in}}(n_j N_j^c)_{\gamma_q} & \text{if} \quad (1-2\psi_{i,q}\phi_i)(n_j N_j^c)_{\gamma_q} > 0, \\ (\psi_{i,q}\psi_{k,q}\phi_i\phi_k)^{\text{out}}(n_j N_j^c)_{\gamma_q} & \text{if} \quad (1-2\psi_{i,q}\phi_i)(n_j N_j^c)_{\gamma_q} < 0, \end{cases} \tag{2.25b}$$

where $(n_j N_j^c)_{\gamma_q}$ is the interface normal $\boldsymbol{n}$ projected onto the cell face normal $\boldsymbol{N}^c$. The diffusive flux, $\widehat{\varepsilon\psi_{i,q}\frac{\partial\phi_i}{\partial x_j} n_j n_k N_k^c}$, is also required at a face where a discontinuity exists. However, unlike the convective fluxes, an up-wind approach is not appropriate. A logical and simple implementation is to take the arithmetic mean of diffusive fluxes calculated on the left and right sides of the face, but this method does not take into consideration the discontinuity at the face and, therefore, is inconsistent [54]. As a result, the reconstructed DG method of Luo et al. [54] is used to define a unique flux value at the face that is consistent with the DG solution. To do this, a reconstructed function, $R(\boldsymbol{\chi}, t) = \boldsymbol{r}(t)\cdot\tilde{\boldsymbol{\phi}}(\boldsymbol{\chi})$, is introduced in the discontinuous Galerkin space that spans the two cells containing the face of interest. The modified basis functions, $\tilde{\boldsymbol{\phi}}$, are members of the same set of functions as the previously described basis functions, $\boldsymbol{\phi}$, but are scaled such that they extend over the two cells. Furthermore, the set of modified basis functions $\tilde{\boldsymbol{\phi}}$ contain higher order functions as described below. To find the weights, $\boldsymbol{r}$, the following set of $2P$

constraints are applied in a least squares sense:

$$\int_{\omega_{q^-}} \psi_{i,q^-}\phi_i\tilde{\phi}_s \,\mathrm{d}v = \int_{\omega_{q^-}} r_i\tilde{\phi}_i\tilde{\phi}_s \,\mathrm{d}v \tag{2.26a}$$

$$\int_{\omega_{q^+}} \psi_{i,q^+}\phi_i\tilde{\phi}_s \,\mathrm{d}v = \int_{\omega_{q^+}} r_i\tilde{\phi}_i\tilde{\phi}_s \,\mathrm{d}v \tag{2.26b}$$

for $s = 1, \ldots, P$, where ω_{q^-} and ω_{q^+} are the volumes of the cells to the left and right of the face, respectively. Luo et al. [54] suggested the properties of the reconstruction function can be improved by adding an extra term of order $O + 1$ in the direction approximately normal to the interface. We add the term in the direction that has the largest component of the interface normal vector. For example, if the normal vector at the face of interest is in the x-direction, $\phi_{q,11} = \chi_{q,1}^3 - 3\chi_{q,1}/5$ would be added to the 10 basis shown in Eq. 2.18. This is possible because $2P$ ($P \geq 1$) constraints are available to find $P + 1$ unknowns, $\boldsymbol{r}$, when the extra term is added. Using the reconstructed function, R, the diffusive flux is written as

$$\widehat{\varepsilon\psi_{i,q}\frac{\partial\phi_i}{\partial x_j}n_j n_k N_k^c} = \varepsilon r_i \frac{\partial\tilde{\phi}_i}{\partial x_j}(n_j n_k N_k^c)_{\gamma_q}. \tag{2.27}$$

Combining Eq. 2.24 with fluxes from Eqs. 2.25a, 2.25b, and 2.27, using the properties $\boldsymbol{\psi} = \boldsymbol{\psi}(t)$, $\boldsymbol{\phi} = \boldsymbol{\phi}(\boldsymbol{\chi})$, $\boldsymbol{r} = \boldsymbol{r}(t)$, and $\tilde{\boldsymbol{\phi}} = \tilde{\boldsymbol{\phi}}(\boldsymbol{\chi})$, and applying the property shown in Eq. 2.14 for orthogonal basis functions leads to

$$\begin{aligned}
\frac{\partial\psi_m}{\partial\tau}\int_{\omega_q}\phi_m^2 \,\mathrm{d}v - \psi_{i,q}n_j\int_{\omega_q}\phi_i\frac{\partial\phi_m}{\partial x_j}\,\mathrm{d}v + (\psi_{i,q})_{\mathrm{up}}\big(n_j N_j^c\big)_{\gamma_q}\oint_{\gamma_q}(\phi_i)_{\mathrm{up}}\phi_m \,\mathrm{d}s \\
+ \psi_{i,q}\psi_{k,q}n_j\int_{\omega_q}\phi_i\phi_k\frac{\partial\phi_m}{\partial x_j}\,\mathrm{d}v - (\psi_{i,q}\psi_{k,q})_{\mathrm{up}}\big(n_j N_j^c\big)_{\gamma_q}\oint_{\gamma_q}(\phi_i\phi_k)_{\mathrm{up}}\phi_m \,\mathrm{d}s \\
= -\varepsilon\psi_{i,q}n_j n_k\int_{\omega_q}\frac{\partial\phi_i}{\partial x_j}\frac{\partial\phi_m}{\partial x_k}\,\mathrm{d}v + \varepsilon r_i(n_j n_k N_k^c)_{\gamma_q}\oint_{\gamma_q}\frac{\partial\tilde{\phi}_i}{\partial x_j}\phi_m \,\mathrm{d}s
\end{aligned} \tag{2.28}$$

for $m = 1, \ldots, P$ and $q = 1, \ldots, Q$. Similar to the transport equation, the discretized reinitialization equation shown above can be easily updated after a time

integration scheme is chosen. Also, all the integrals in the equaion only depend on the basis functions and can be pre-computed suggesting that a quadrature-free approach should work. However, numerical experiments have shown that using a quadrature-free implementation of the convective term is unstable. Details of the instability and our solution are described in Section 2.3.4.

2.3.4 Numerical stability

Numerical tests have identified two necessary adjustments to the DG implementation described heretofore. The adjustments include a restriction on when the reinitialization compressive term is applied and careful evaluation of the integrals in the reinitialization equation. The problem arises when oscillations develop on level set profile from numerical errors. In some situations the convective term will amplify the oscillation leading to new artificial interfaces that hinder robustness. To circumvent this issue, the convective term is only used when the level set is bounded such that $-\zeta \leq \Psi_h \leq 1+\zeta$ for some small ζ ($\approx 1\times 10^{-5}$). The restriction is applied to the volume and surface terms that come from the convective term. The diffusion term is always used and keeps oscillations from growing.

Equation 2.28 shows the equation that is solved to reinitialize the level set. In this equation, all of the integrals only depend on the basis functions that are chosen a priori and the integrals can be evaluated and stored in an initialization routine resulting in a quadrature-free approach. However, we have found that the quadrature-free approach is unstable because of the way the restriction on the convective term is applied. The DG representation of the level set allows for variations within the cell that can lead to some regions that exceed one or zero. If convective term is used, because the cell (or face) centered value is within the

threshold given above, the convective term can increase the value of the function in the region where the function is greater that one or less than zero and lead to an unstable situation. Therefore, a quadrature scheme is used and consists of the following steps for each face within the surface integrals and the volume integrals: 1) determine quadrature points needed to exactly evaluate the integral, 2) at each quadrature point, if $-\zeta \leq \Psi_h \leq 1 + \zeta$ is true then this point should be included in the integration of the convective and diffusive fluxes (If evaluating a surface integral, up-winding should be based on the value of Ψ_h at this point.); else, only the diffusive flux is included in the quadrature integration at this point. This approach has been shown to be stable for all of the test cases studied.

2.3.5 Minimum/maximum preserving limiter

The modifications described in the numerical stability section make the DG scheme stable and robust. However, numerical experiments showed significant overshoot and undershoot of the level set outside the interval $[0, 1]$ on which it is defined. The overshoot and undershoots result from the amplification of oscillations found in the high order terms used to represent the level set. To reduce this phenomenon we added the minimum/maximum preserving (MMP) limiter of Zhang and Shu [55]. The limiter was designed to maintain the order of the DG scheme while modifying the formulation such that the function stays within the interval $[m, M] = [0, 1]$.

Implementation of the limiter is straight forward for the transport and requires replacing the DG representation of the level set within cell q, $\Psi_{h,q}$, with a modified function $\widetilde{\Psi}_{h,q}$ given by,

$$\widetilde{\Psi}_{h,q} = \Theta_q\big(\Psi_{h,q} - \overline{\Psi}_{h,q}\big) + \overline{\Psi}_{h,q}, \tag{2.29}$$

where $\overline{\Psi}_{h,q}$ is the mean value of the DG representation of the level set in the q^{th} cell. Θ is a measure for how close the cell is to the interval bounds and is defined to be,

$$\Theta_q = \min\left\{\left|\frac{m - \overline{\Psi}_{h,q}}{m_q - \overline{\Psi}_{h,q}}\right|, \left|\frac{M - \overline{\Psi}_{h,q}}{M_q - \overline{\Psi}_{h,q}}\right|, 1\right\}, \tag{2.30}$$

where m_q and M_q are in the minimum and maximum of the DG representation of the level set within the q^{th} cell, respectively. To avoid finding the minimum and maximum of a high order polynomial, m_q and M_q can be by approximated by finding the minimum and maximum of the level set at quadrature points. We use an N-point Gauss-Lobatto quadrature rule in each direction within the grid cell of interest. N is chosen to be the smallest integer satisfying $2O < 2N - 3$ which is the quadrature needed to exactly integrate a polynomial of order $2O$.

Substituting the modified conservative level set, given in Eq. 2.29, into the DG discretized level set transport equation, Eq. 2.22, results in,

$$\begin{aligned}\frac{\partial \psi_{m,q}}{\partial t}\int_{\omega_q}\phi_m^2\,\mathrm{d}v - \Theta_q\psi_{i,q}U_j\int_{\omega_q}\phi_i\frac{\partial\phi_m}{\partial x_j}\,\mathrm{d}v + (\Theta_q - 1)\overline{\Psi}_q U_j\int_{\omega_q}\frac{\partial\phi_m}{\partial x_j}\,\mathrm{d}v \\ + (\Theta_q)_{\text{up}}\big(U_jN_j^c\big)_{\gamma_q}(\psi_{i,q})_{\text{up}}\oint_{\gamma_q}(\phi_i)_{\text{up}}\phi_m\,\mathrm{d}s \\ - \big((\Theta_q)_{\text{up}} - 1\big)(U_jN_j)_{\gamma_q}\big(\overline{\Psi}_q\big)_{\text{up}}\oint_{\gamma_q}\phi_m\,\mathrm{d}s = 0\end{aligned} \tag{2.31}$$

for $m = 1, \ldots, P$ and $q = 1, \ldots, Q$, where $(\Theta_q)_{\text{up}}$ and $\big(\overline{\Psi}_q\big)_{\text{up}}$ are up-winded using the same criteria used for the other terms in the transport equation defined in Eq. 2.20.

The modification of the transport equation consists of the following. When $\Theta = 1$ which occurs when the minimum and maximum of the grid cell are contained within the interval $[m, M]$, the scheme reverts back to the original DG scheme. However, when $\Theta = 0$ then the flux is based on only the mean of the DG representation instead of the entire DG polynomial. A value of Θ between 0 and 1

results in a weighted combination of the fluxes based on the mean and on the full DG representation.

Applying the MMP limiter to the reinitialization equation is more difficult because of the non-linear convective term. Furthermore, we do not apply the MMP limiter to the diffusion term because this term is designed to smooth the level set function and should not create an unbounded function. Our approach to implement the MMP limiter for the non-linear convective is to apply the MMP limiter to the entire portion of the non-linear term that depends on Ψ_h resulting in the following modified function,

$$\widetilde{\Psi_{h,q} - \Psi_{h,q}^2} = \Theta_q'\Big(\big(\Psi_{h,q} - \Psi_{h,q}^2\big) - \Big(\overline{\Psi}_{h,q} - \overline{\Psi}_{h,q}^2\Big)\Big) + \Big(\overline{\Psi}_{h,q} - \overline{\Psi}_{h,q}^2\Big) \tag{2.32}$$

with

$$\Theta_q' = \min\left\{\left|\frac{(m-m^2) - \Big(\overline{\Psi}_{h,q} - \overline{\Psi}_{h,q}^2\Big)}{\big(m_q - m_q^2\big) - \Big(\overline{\Psi}_{h,q} - \overline{\Psi}_{h,q}^2\Big)}\right|, \left|\frac{M - M^2 - \Big(\overline{\Psi}_{h,q} - \overline{\Psi}_{h,q}^2\Big)}{\big(M_q - M_q^2\big) - \Big(\overline{\Psi}_{h,q} - \overline{\Psi}_{h,q}^2\Big)}\right|, 1\right\}, \tag{2.33}$$

where m and M are the bounds on the interval and m_q and M_q are the minimum and maximum value of the DG representation of the level set with the q^{th} grid cell. Substituting Eq. 2.32 into the DG discretized reinitialization equation Eq. 2.28 for

$\Psi_{h,q} - \Psi_{h,q}^2$ results in,

$$\begin{aligned}
\frac{\partial \psi_m}{\partial \tau} \int_{\omega_q} \phi_m^2 \,\mathrm{d}v - \Theta_q' \psi_{i,q} n_j \int_{\omega_q} \phi_i \frac{\partial \phi_m}{\partial x_j} \,\mathrm{d}v + \Theta_q' (\psi_{i,q})_{\mathrm{up}} \big(n_j N_j^c\big)_{\gamma_q} \oint_{\gamma_q} (\phi_i)_{\mathrm{up}} \phi_m \,\mathrm{d}s \\
+ \Theta_q' \psi_{i,q} \psi_{k,q} n_j \int_{\omega_q} \phi_i \phi_k \frac{\partial \phi_m}{\partial x_j} \,\mathrm{d}v \\
- \Theta_q' (\psi_{i,q} \psi_{k,q})_{\mathrm{up}} \big(n_j N_j^c\big)_{\gamma_q} \oint_{\gamma_q} (\phi_i \phi_k)_{\mathrm{up}} \phi_m \,\mathrm{d}s \\
- \big(\Theta_q' - 1\big)\big(\overline{\Psi}_q - \overline{\Psi}_q^2\big) \int_{\omega_q} \frac{\partial \phi_m}{\partial x_j} \,\mathrm{d}v \\
+ \Big(\big(\Theta_q'\big)_{\mathrm{up}} - 1\Big)\Big(\big(\overline{\Psi}_q\big)_{\mathrm{up}} - \big(\overline{\Psi}_q\big)_{\mathrm{up}}^2\Big) \oint_{\gamma_q} \phi_m \,\mathrm{d}s \\
= - \varepsilon \psi_{i,q} n_j n_k \int_{\omega_q} \frac{\partial \phi_i}{\partial x_j} \frac{\partial \phi_m}{\partial x_k} \,\mathrm{d}v + \varepsilon r_i (n_j n_k N_k^c)_{\gamma_q} \oint_{\gamma_q} \frac{\partial \tilde{\phi}_i}{\partial x_j} \phi_m \,\mathrm{d}s
\end{aligned} \tag{2.34}$$

for $m = 1, \ldots, P$ and $q = 1, \ldots, Q$. The MMP limiter parameters appearing in the flux terms, $(\Theta_q)_{\mathrm{up}}$ and $\big(\overline{\Psi}_q\big)_{\mathrm{up}}$, are up-winded using the same criteria as the rest of the convective term which was defined in Eqs. 2.25a and 2.25b.

Numerical tests of the DG scheme with the MMP limiter have shown a reduction in overshoots and undershoot from more than 50% to less than 1%. The limiter is supposed to be a strict minimum/maximum preserving scheme for linear problems. The small overshoots and undershoots we observed came solely from the non-linear convective term found in the reinitialization equation. The reduction significantly improves the scheme because regions with overshoots are analogous with a region with increased density since more of the level set is in the region than should be. Furthermore, an increase in density is not physical in the context of an incompressible flow and any reduction in overshoots and undershoots should improve the physical accuracy of the results. All of the results shown in this paper used the aforementioned implementation of the MMP limiter.

2.4 Level set time advancement

As described previously, the level set is transported with the velocity field and then reinitialized using Eqs. 2.22 and 2.28, respectively. Temporal discretization is performed using a total variation diminishing third order Runge-Kutta (TVD-RK3) scheme. This scheme has been shown by Cockburn and Shu [53] to be stable with up to eighth order polynomial basis functions. Using the explicit TVD-RK3 method allows for the cells to be decoupled from each other and does not hinder the highly scalable properties of DG. The CFL constraint is approximately [46]

$$\mathrm{CFL} \leq \frac{1}{2O+1}, \tag{2.35}$$

where O is the highest order of the polynomials used as the basis functions.

The CFL constraint, Eq. 2.35 must be respected by the transport and reinitialization equation. The transport equation includes a convective term with corresponding CFL number

$$\mathrm{CFL}_{\mathrm{trans.}} = \frac{\max|\boldsymbol{U}|\Delta t_{\mathrm{trans.}}}{h}, \tag{2.36}$$

where $\max|\boldsymbol{U}|$ is the maximum of the velocity magnitude within the domain and h is the smallest characteristic mesh size taken to be $\min(\Delta\boldsymbol{x})$ for our Cartesian mesh. Plugging Eq. 2.36 into Eq. 2.35 provides an upper bound for the time-step size, $\Delta t_{\mathrm{trans.}}$.

The reinitialization equation contains a convective and a diffusive term resulting in a CFL conditions that includes two terms,

$$\mathrm{CFL}_{\mathrm{reinit.}} = \max\left(\frac{\max|U_{\mathrm{reinit.}}|\Delta\tau_{\mathrm{reinit.}}}{h}, \frac{4\varepsilon\Delta\tau_{\mathrm{reinit.}}}{h^2}\right). \tag{2.37}$$

Where the first term is the convective CFL that has a maximum convective velocity, $\max|U_{\mathrm{reinit.}}| = 1$. The second term is the diffusive CFL and contains ε which was

define previously and sets the thickness of the hyperbolic tangent profile. Equation 2.37 is combined with Eq. 2.35 to find an upper bound on the reinitialization time-step size in pseudo-time, $\Delta\tau_{\text{reinit.}}$.

Reinitialization maintains the shape of the profile but also introduces errors into the solution. The errors can be limited by choosing to reinitialize enough steps in pseudo-time to maintain the proper profile shape while avoiding unneeded reinitialization steps. To control the amount of reinitialization, we introduce the parameter F, known as the reinitialization factor. F typically varies between $F = 0$ and $F = 1$ which correspond to no reinitialization and an amount of reinitialization that can moves the level set the same distance as was done by the transport step, respectively. Choosing an appropriate value for F is dependent on the nature of the test case. In some flows, the transport step does not change the level set profile and very little or no reinitialization is needed; this type of flow includes uniform flow and Zalesak's disk test case that uses solid body rotation. Other flows, such as stagnation points and the vortex used in the deformation test case, have complex flow fields that deform the level set profile. This set of flows require more reinitialization in order to maintain high accuracy and low mass loss. Zalesak's disk and the deformation test case are provided in Section 2.6 where we analyzed the effect of F on the solution.

For completeness we include the procedure used to update the level set in time:

1. Velocity field is updated using the Navier-Stokes momentum equations over a time Δt.
2. The level set is transported using Eq. 2.31 for a total time of Δt. Substeps may be appropriate if $\Delta t_{\text{trans.}}$ given by the CFL constraint, Eq. 2.35, is smaller that Δt used for the flow solver.

3. Interface normal vectors and curvature are calculated using the procedure outlined in Section 2.5.

4. The level set is reinitialized using Eq. 2.34 by a total amount of pseudo-time equal to

$$\Delta\tilde{\tau}_{\text{reinit.}} = F \max|\boldsymbol{U}| / \max|U_{\text{reinit.}}|. \tag{2.38}$$

   Using this definition for $\Delta\tilde{\tau}_{\text{reinit.}}$ allows the reinitialization step to move the level set a distance equivalent to a fraction, F, of the maximum distance the level set moved during the transport step. If $\Delta\tilde{\tau}$ is larger than the maximum reinitialization time-step, $\Delta\tau$, calculated using the CFL constraint in Eq. 2.37 then multiple sub-steps will be required.

5. Go to step 1.

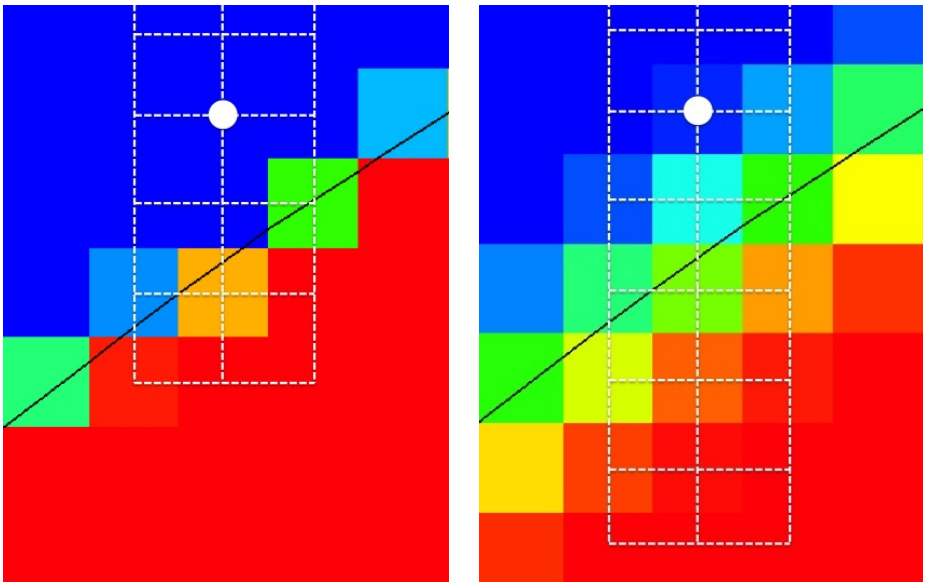
2.5 Interface normal and curvature

The normal to the interface is used for the reinitialization of the level set and must be calculated accurately. The curvature is used to determine the pressure jump that results from surface tension in the solution of the Navier-Stokes equations and has a direct effect on the solution. Therefore, having the normal vector and curvature converge under mesh refinement is necessary for mesh independence studies and predictive simulations. Convergence is difficult to obtain because as the mesh is refined, the thickness of the conservative level set function is also reduced so that the number of grid cells across the hyperbolic tangent profile is fixed. This problem is not unique to the normal and curvature. Whenever the calculation of a quantity does not become more accurate as the mesh is refined the same situation arises. Desjardins et al. [28] was confronted with this problem when

developing the ACLS method and applied a least squared approach proposed by Marchandise et al. [48] for the classical level set. The scheme calculates the normal and curvature from a least squares polynomial fit of a reconstructed distance level set over the cell of interest and its 26 nearest neighbors. The least squares method has been shown to converge with a second order normal and first order curvature when applied to the conservative level set [28]. Second order convergence of the normal vector is acceptable, but a first order curvature is sub-optimal. Therefore, we use the least squares approach to calculate the interface normal but not the interface curvature.

The proposed method to calculate interface curvature is to use a height function technique popular in volume of fluid (VOF) methods [49]. The scheme has been shown within VOF schemes to calculate a curvature that converges with second order accuracy. The idea is to integrate the volume fraction in a pseudo-normal direction forming a height in the cell of interest and the neighboring cells. The curvature is then calculated using finite difference operators on the heights. The pseudo-normal direction is defined as the direction (x, y, or z) with the largest component of the interface normal vector. In three dimensions, a stencil of $3 \times 3 \times 7$ cells is used with seven cells in the pseudo-normal direction and three cells in each of the tangential directions [49]. Seven cells is chosen for accuracy when the interface is at an angle as shown in Fig. 2.2(a). The nine heights are calculated over the 3×3 stencil and finite difference operators are used to calculate the curvature [49].

To demonstrate how the approach works, the curvature is calculated assuming the pseudo-normal direction has been determined to be the x-direction at the cell with coordinates i, j, k. The first step is to calculate nine heights over a 3×3 mesh



(a) Volume fraction and stencil used in VOF height function formulation

(b) First DG degree of freedom and stencil used in DG height function formulation

Figure 2.2: Stencils used to compute curvature at cell indicated with white circle. Stencil for VOF is shown by the white dotted lines in (a) and is a total of 3×7 ($3 \times 3 \times 7$ in 3D). Correspondingly, the stencil used for DG is shown in (b) and has a 3×11 stencil ($3 \times 3 \times 11$ in 3D).

by integrating in the x-direction at each location using

$$H_{j'k'} = \sum_{i'=i-3}^{i+3} f_{i'j'k'} \Delta x \quad \text{for} \quad \begin{cases} j' = j-1, j, j+1 \\ k' = k-1, k, k+1 \end{cases}, \tag{2.39}$$

where $f_{i'j'k'}$ is the volume fraction in cell i', j', k' and Δx is the width of the cell in the pseudo-normal direction. Using the heights, the curvature is calculated using second order finite difference operators that can be written as

$$\kappa = \frac{H_{yy} + H_{zz} + H_{yy}H_z^2 + H_{zz}H_y^2 - 2H_{yz}H_yH_z}{\left(1 + H_y^2 + H_z^2\right)^{3/2}} \left(\frac{\partial f_{ijk}/\partial x}{|\partial f_{ijk}/\partial x|} \right) \tag{2.40}$$

with

$$H_y = \frac{H_{j+1,k} - H_{j-1,k}}{2\Delta y} \tag{2.41a}$$

$$H_z = \frac{H_{j,k+1} - H_{j,k-1}}{2\Delta z} \tag{2.41b}$$

$$H_{yy} = \frac{H_{j+1,k} - 2H_{jk} + H_{j-1,k}}{\Delta y^2} \tag{2.41c}$$

$$H_{zz} = \frac{H_{j,k+1} - 2H_{jk} + H_{j,k-1}}{\Delta z^2} \tag{2.41d}$$

$$H_{yz} = \frac{H_{j+1,k+1} - H_{j+1,k-1} - H_{j-1,k+1} + H_{j-1,k-1}}{2\Delta x\ 2\Delta y}. \tag{2.41e}$$

To extend this method to the conservative level set, we modified the way the heights, H, are calculated. Instead of integrating volume fraction, the conservative level set, Ψ_h, is integrated. For example, if the pseudo-normal direction is still assumed to be in the x-direction, then

$$H_{j'k'} = \sum_{i'=i-\frac{S-1}{2}}^{i+\frac{S-1}{2}} f^{\mathrm{DG}}_{i'j'k'}\Delta x \quad \text{for} \quad \begin{cases} j' = j-1, j, j+1 \\ k' = k-1, k, k+1 \end{cases}, \tag{2.42}$$

with

$$f^{\mathrm{DG}}_{i'j'k'} = \int_{\omega_{q'}} \Psi_{h,q'}\,\mathrm{d}v = \psi_{i,q'} \int_{\omega_{q'}} \phi_i\,\mathrm{d}v \tag{2.43}$$

where q' is the index for the i', j', k' cell. For Legendre polynomials, Eq. 2.43 reduces only the first degree of freedom or $f^{\mathrm{DG}}_{i'j'k'} = \psi_{0,q'}$.

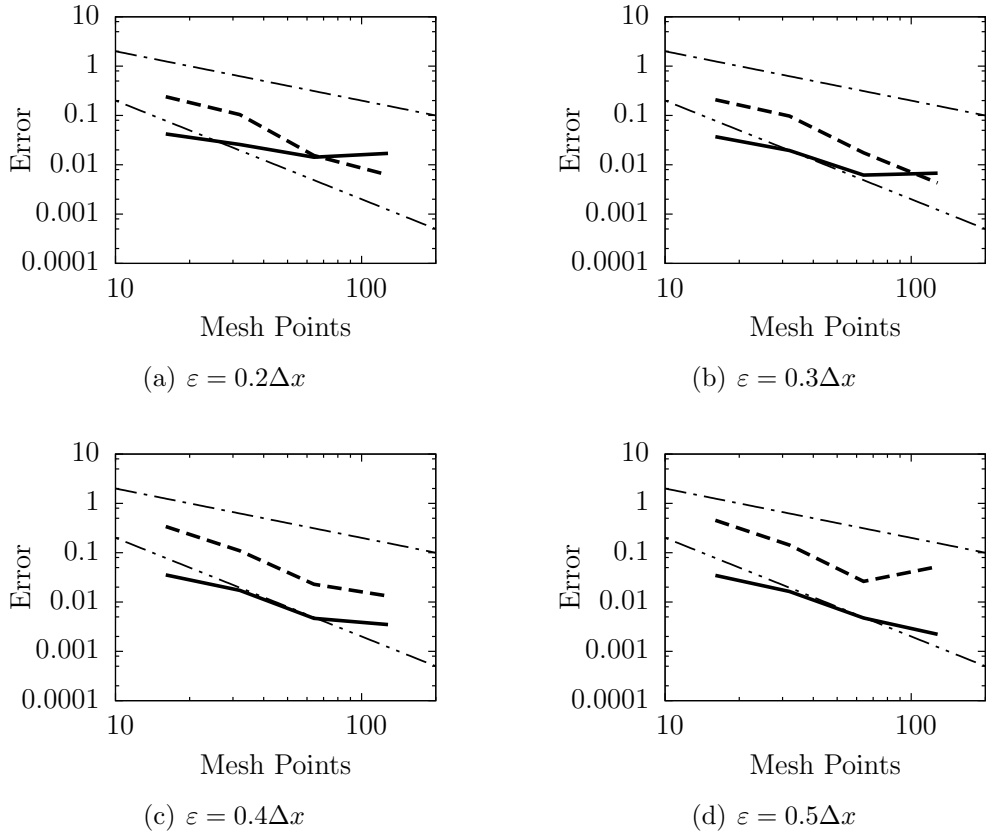
The stencil size has also been changed from $3 \times 3 \times 7$ to $3 \times 3 \times S$ where S is chosen to achieve high accuracy by capturing the width of the interface within the stencil and should be based on the thickness of the interface. For example, an interface thickness corresponding to $\varepsilon = 0.5\Delta x$ is captured on roughly four cells and results in $S = 11$ or a $3\times3\times11$ stencil as shown by Fig. 2.2(b). For other values of ε, the stencil can be determined by noting the profile thickness is approximately $8\varepsilon/\Delta x$ cells, which was determined using numerical tests. Therefore, the stencil should be seven cells plus the thickness of the profile or $S \approx 7 + 8\varepsilon/\Delta x$.

The large stencil leads to numerical difficulties in two circumstances that must be dealt with if accuracy and robustness are to be maintained. The first scenario manifests when two interfaces approach each other. This can occur when two liquid structures come close together or when a liquid entity becomes thin. To avoid mixing information from the two interfaces, the stencil size should be reduced in the pseudo-normal direction such that influences from the other interface are not used when calculating the heights. The second circumstance occurs when the structure has a large curvature, and the interface is at an angle with respect to the coordinate system. Defining a stencil that is aligned with one of the coordinate axis and captures the entire profile can be impossible to construct. Our solution to this problem is to give up on the height function approach because it is ill-posed and revert back to the least squares approach of Marchandise et al. [48] mentioned previously.

Convergence of the level set height function approach was studied by calculating the curvature of a circle. The problem was initialized with an exact DG level set field and then the curvature was calculated on various meshes ranging from 16^2 to 128^2 cells using different hyperbolic tangent thickness from $\varepsilon = 0.2\Delta x$ to $0.5\Delta x$. As the profile thickness is decreased, the level set approaches a step function representing the liquid volume fraction and the curvature calculation should be improved but the calculation of the interface normal is more prone to errors. Conversely, when the profile thickness is increased the level set becomes smoother and errors are reduced in the calculation of the normal vector but the curvature calculation deteriorates. Therefore, a balance must be obtained where good convergence of both the normal and curvature is achieved.

Figure 2.3 contains L_∞ convergence plots of the normal and curvature using

different profile thicknesses. Based on the results, we concluded that a profile thickness defined by $\varepsilon = 0.4\Delta x$ achieves second order convergence for the curvature and between first and second order convergence for the normal. When the profile thickness is thicker, $\varepsilon = 0.5\Delta x$, the function is smoother and a more accurate normal can be calculated using the least squares method but a less accurate curvature is found. Likewise, when $\varepsilon = 0.2\Delta x$ or $\varepsilon = 0.3\Delta x$ the profile is sharper and the curvature is calculated accurately; however, the normal is less accurate. As a result, $\varepsilon = 0.4\Delta x$ was used in all of the test cases shown in this paper.



(a) $\varepsilon = 0.2\Delta x$ (b) $\varepsilon = 0.3\Delta x$

(c) $\varepsilon = 0.4\Delta x$ (d) $\varepsilon = 0.5\Delta x$

Figure 2.3: Convergence of normal and curvature for different level set thicknesses set by ε. The thick solid line and the thick dashed line show L_∞ errors for the interface normal and curvature, respectively. First and second order convergence is shown with the dot-dashed and the dot-dot-dashed lines for reference.

2.5.1 Spurious velocities

Even though the curvature has been shown to have second order convergence, the errors could be large. Therefore, the following test gauges whether the curvature errors will lead to significant spurious velocities. The parasitic velocities result from errors in the curvature calculation propagating through the calculation of the surface tension force in the Navier-Stokes momentum equations. The test consists of simulating a two-dimensional drop of diameter, $D = 0.4$, inside of a unit box. The physical properties used in the simulation are density ratio set to unity or $\rho_1 = \rho_2 = \rho$, surface tension of $\sigma = 1$, and unity viscosity ratio with $\mu_1 = \mu_2 = 0.1$. The free parameter is the density of both fluids and is used to set the Laplace number, $La = 1/(Oh)^2 = \sigma\rho D/\mu^2$. After a non-dimensional time of $t\sigma/(\mu D) = 250$, the capillary number, $Ca = |\boldsymbol{u}|_{\max}\mu/\sigma$, is computed. The capillary number is a non-dimensional estimate of spurious velocities.

This test case uses the DG discretization of the conservative level set method described herein in conjunction with our multiphase CFD code. The numerical code, NGA [56], has developed for accurate simulations of turbulent reactive flows.

Table 2.1 shows the capillary number for varying Laplace numbers on 32^2 and 64^2 meshes. For all Laplace numbers and both meshes, the capillary number remains small. To investigate the mesh convergence properties of the spurious currents, a test was conducted with the Laplace number fixed at 12,000 and the mesh varied from 16^2 through 128^2. The results in Table 2.2 show convergence is obtained up to a 64^2 mesh and low capillary numbers are found for all meshes. The results indicate the curvature errors do not lead to excessive spurious velocities.

	Capillary Number	
Laplace number	32^2 mesh	64^2 mesh
12	1.10925E-03	7.30265E-06
120	6.46696E-04	7.82316E-06
1,200	1.05564E-04	7.79871E-06
12,000	9.36451E-05	6.45236E-06
120,000	5.41598E-05	9.55614E-06
1,200,000	1.66377E-06	2.46310E-06

Table 2.1: Capillary number observed at 10 time units for various Laplace numbers on a 32^2 mesh and a 64^2 mesh.

	Capillary Number			
Laplace number	16^2 mesh	32^2 mesh	64^2 mesh	128^2 mesh
12,000	2.11102E-04	9.36451E-05	6.45236E-06	7.48101E-06

Table 2.2: Capillary number observed at 10 time units for a Laplace numbers of 12,000 on various meshes.

2.6 Validation

2.6.1 Zalesak's disk

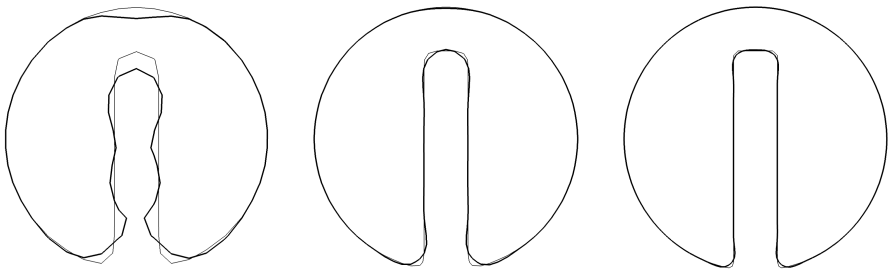
The first validation test case is known as Zalesak's disk [57] and tests the ability of the DG-CLS scheme to transport a complex geometry with sharp corners. The test consists of solid body rotation of a notched disk with radius 0.15, notch width of 0.05, and center at $(x, y) = (0, 0.25)$ within a square domain of size 1×1. The notched disk is subjected to rotations using the two-dimensional velocity field:

$$U = -2\pi y, \tag{2.44a}$$

$$V = +2\pi x. \tag{2.44b}$$

For this test, the disk's shape should remain unchanged. Figure 2.4 shows how mesh refinement affected the final shape of the notched disk. Meshes consisting of 50^2, 100^2, and 200^2 were tested. For reference, the 50^2 mesh has only two grid

cells across the notch and even with this very coarse mesh the notch in the circle is maintained after the disk is rotated one full time. When the mesh is refined more to the 100^2 and 200^2 cases we find very good results after the disk has been rotated.

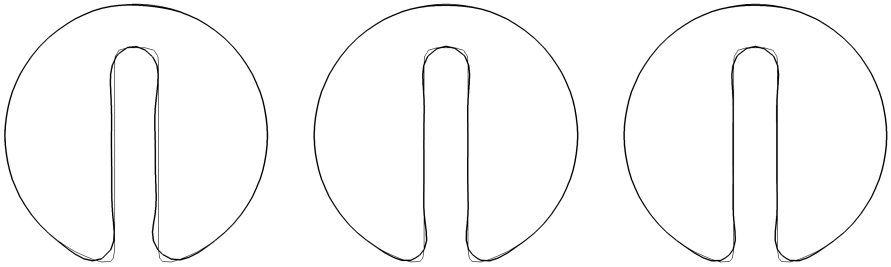


(a) Mesh: 50^2 (b) Mesh: 100^2 (c) Mesh: 200^2

Figure 2.4: The calculated solutions of Zalesak's disk after one full rotation using various meshes. Second order polynomials and reinitialization factor of $F = 0.5$ were used for all three cases. Solution is shown with the thick line and the exact solution (projected onto the mesh) is given with the thin line.

Next, the effect of the order of the polynomials used in the DG scheme to represent the level set was studied. Figure 2.5 shows results for polynomials with orders: $O = 1$, $O = 2$, and $O = 3$. Convergence towards the exact solution was shown when the polynomial basis order was increased. However, the difference between the solutions obtained using second and third order polynomials was relatively small suggesting other errors, such as errors from mesh resolution, are more dominate. Furthermore, the differences indicate that running a simulation with second order basis functions may be a good compromise between accuracy and cost.

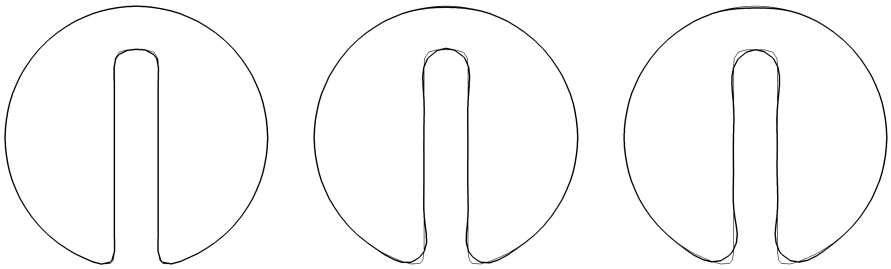
Figure 2.6 shows results for different amounts of reinitialization. The amount of reinitialization is characterized by the reinitialization factor described in Section 2.4 which was varied from 0 to 1. For this problem the velocity field is prescribed to produce solid body rotation of the notch circle and the DG level set scheme is



(a) DG order: 1 (b) DG order: 2 (c) DG order: 3

Figure 2.5: The calculated solutions of Zalesak's disk after one full rotation using various orders of polynomials in the DG representation of the level set. A mesh of 100^2 and reinitialization factor of $F = 0.5$ were used for all three cases. Solution is shown with the thick line and the exact solution (projected onto the mesh) is given with the thin line.

capable of transporting the notch circle without the need for reinitialization. As a result, the best results are obtained when the reinitialization factor is set to zero, $F = 0$. As more reinitialization is performed, additional errors are introduced and the final shape of the notched circle does not match the initial shape as well. The case with no reinitialization, $F = 0$, was tested further by increasing the number of rotations of the notch circle from 1 to 50 and reducing the mesh from 100^2 to 50^2. Results are shown in Fig. 2.7 which shown that even after 50 rotations and a very coarse mesh the DG scheme is able to maintain the notched circle very well.



(a) $F = 0$ (b) $F = 0.5$ (c) $F = 1.0$

Figure 2.6: The calculated solutions of Zalesak's disk after one full rotation different amounts of reinitialization. A mesh of 100^2 and second order polynomials were used for all three cases. Solution is shown with the thick line and the exact solution (projected onto the mesh) is given with the thin line.

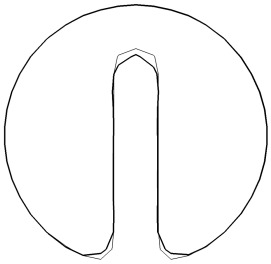


Figure 2.7: Zalesak's disk after 50 rotations. Parameters for simulation include reinitialization factor, $F = 0$, a 50^2 mesh, and second order polynomials.

2.6.2 Two-dimensional deformation

The two-dimensional deformation test case consists of the stretching and un-stretching of a drop in a vortex. The simulation was initialized with a two-dimensional drop of diameter 0.3 with center at $(x, y) = (0, 0.25)$ within a unit square domain. The drop is stretched by the velocity field until time is equal to four, $t = 4$; then, the velocity field reverses for another four time units and the liquid should return to its initial state. The velocity field used to achieve the stretching and un-stretching is

$$U = -2\sin(\pi x)^2 \sin(\pi y)\cos(\pi y)\cos(\pi t/8), \tag{2.45a}$$

$$V = +2\sin(\pi y)^2 \sin(\pi x)\cos(\pi x)\cos(\pi t/8). \tag{2.45b}$$

Figure 2.8 shows snapshots of the progression from initial state ($t = 0$), to the fully stretched state ($t = 4$), and back to the final state ($t = 8$). The exact solution for the final state is to perfectly match the initial state. However, when the liquid is in the fully stretched state, the tail will drop below the mesh resolution leading to a loss of mass. Because the conservative level set is designed to limit mass loss, the liquid is moved from the unresolvable tail into droplets resolvable on the mesh. This phenomenon is clearly visible in Fig. 2.8 at $t = 3$, 4, and 5 where the tail has been replaced by droplets. A result of the mass-conserving scheme is that the tail

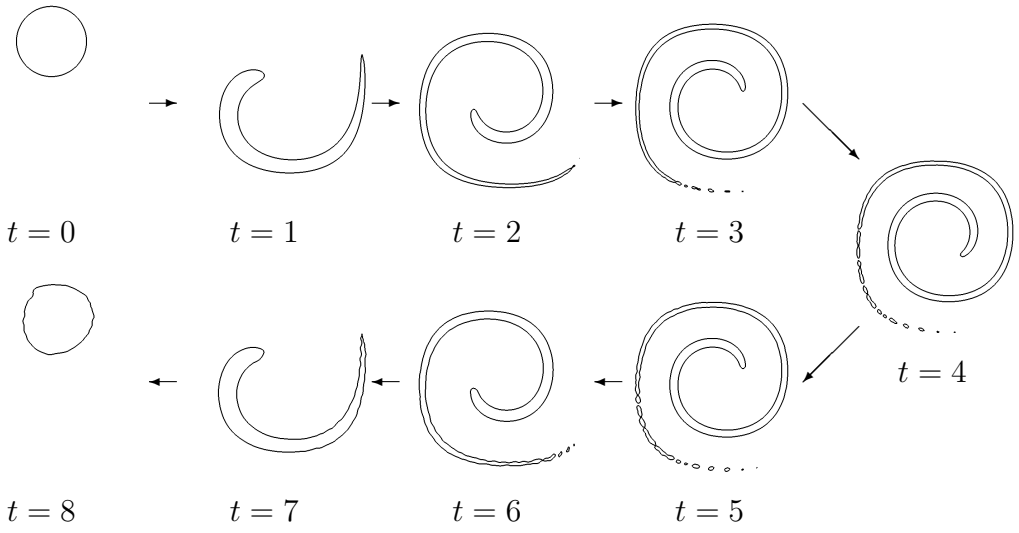


Figure 2.8: Snapshots of the two-dimensional deformation test case at various times. Results as a function of time using a 128^2 mesh and second order polynomial basis functions.

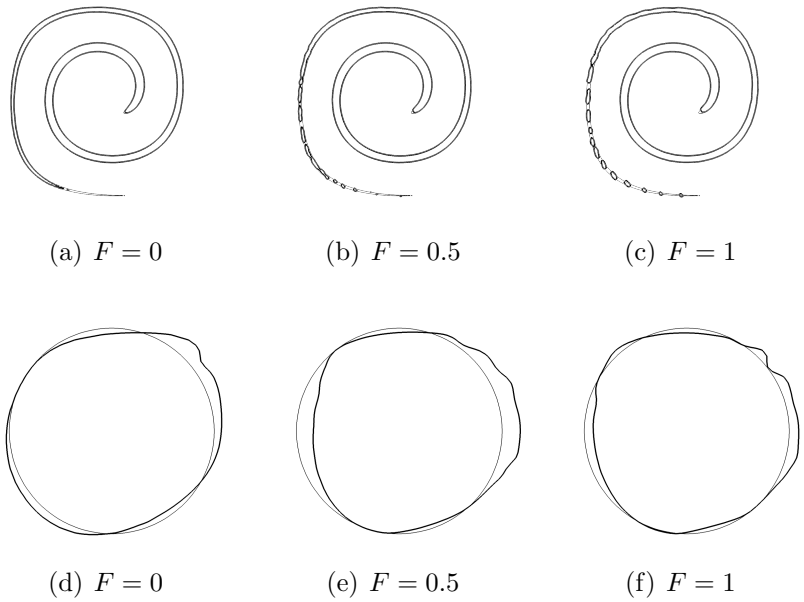
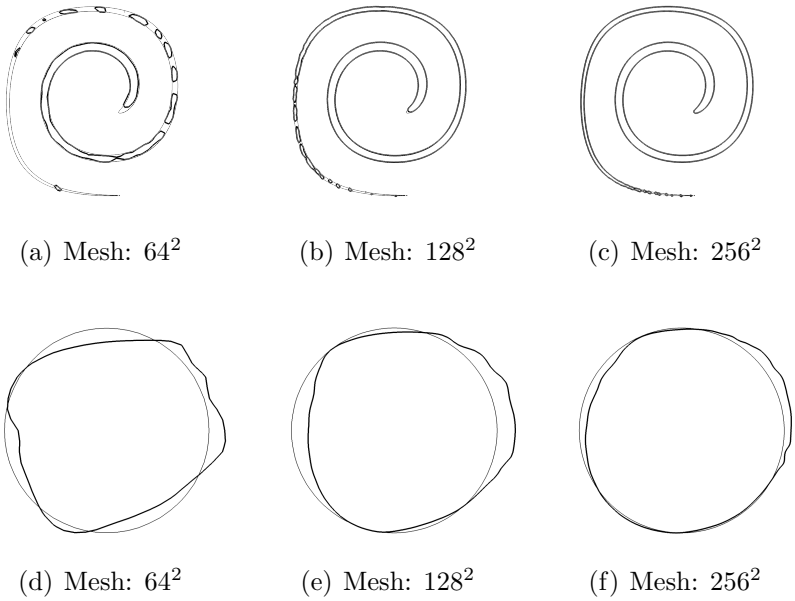


Figure 2.9: Interface location of deformation test case with various amounts of reinitialization. The reinitialization factor, F, was varied from 0 to 1. Figures (a), (b), and (c) show results obtained at maximum deformation, $t = 4$. Figures (d), (e), and (f) show results obtained at the end of the simulation, $t = 8$. The exact solution is shown with a thin line for reference. For all cases a 128^2 mesh and second order polynomials in the DG discretization were used.



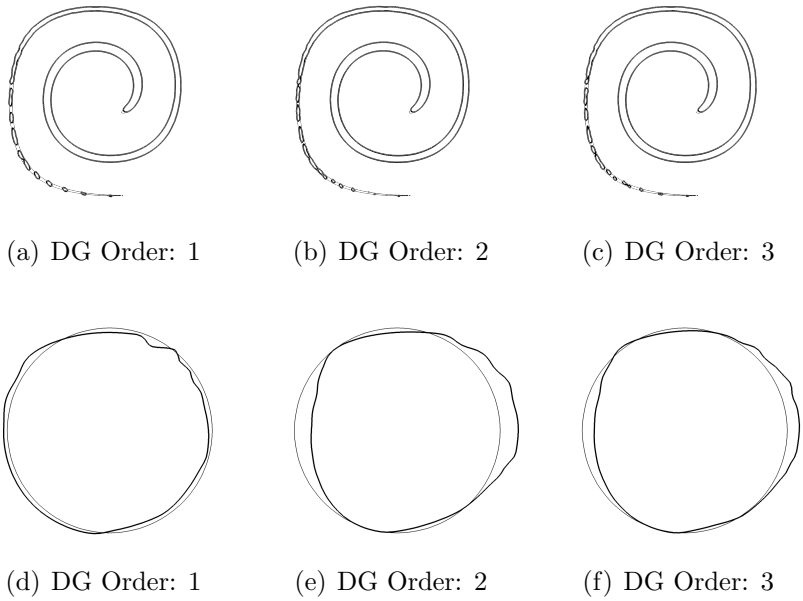
(a) Mesh: 64^2 (b) Mesh: 128^2 (c) Mesh: 256^2

(d) Mesh: 64^2 (e) Mesh: 128^2 (f) Mesh: 256^2

Figure 2.10: Interface location of deformation test case on different meshes, namely: 64^2, 128^2, and 256^2. Figures (a), (b), and (c) show results obtained at maximum deformation, $t = 4$. Figures (d), (e), and (f) show results obtained at the end of the simulation, $t = 8$. The exact solution is shown with a thin line for reference. For all cases second order polynomials were used in the DG discretization and the reinitialization factor was set to $F = 0.5$.

should be deformed and the final interface location may not match the expected exact solution.

To find a good balance of no reinitialization to excessive reinitialization, simulations were conducted of the two-dimensional deformation test case with varying amounts of reinitialization. The test used a mesh with 128^2 cells and second order polynomial basis functions. The amount of reinitialization is characterized by the reinitialization factor, F, which adjusts the amount of reinitialization from none ($F = 0$) to an amount wherein reinitialization can move the level set the same distance as the transport step ($F = 1$). Results are provided in Fig. 2.9 and show that as more reinitialization is performed the tail was broken into more droplets which resulted in a change in the final shape of the interface. For the case with no



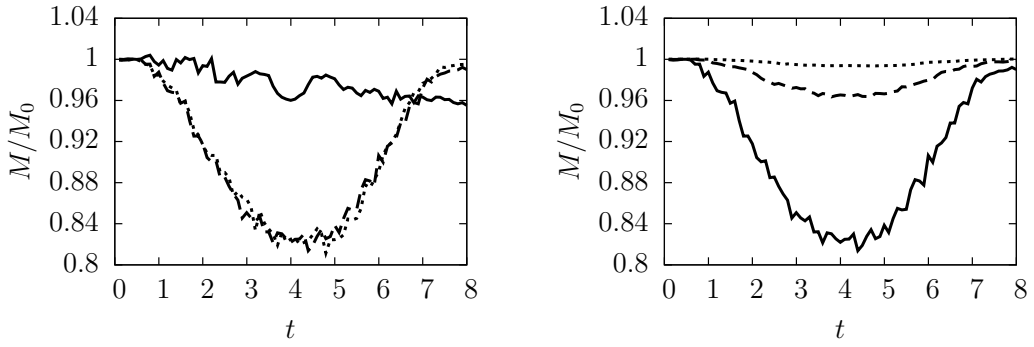
(a) DG Order: 1 (b) DG Order: 2 (c) DG Order: 3

(d) DG Order: 1 (e) DG Order: 2 (f) DG Order: 3

Figure 2.11: Interface location of deformation test case using different polynomials orders in the DG discretization of the level set. Figures (a), (b), and (c) show results obtained at maximum deformation, $t = 4$. Figures (d), (e), and (f) show results obtained at the end of the simulation, $t = 8$. The exact solution is shown with a thin line for reference. For all cases a mesh with 128^2 points was used and the reinitialization factor was set to $F = 0.5$.

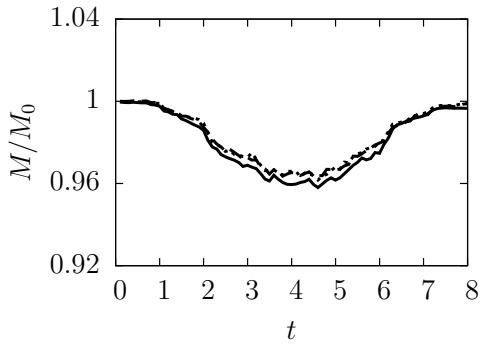
reinitialization the tail became under-resolved and the liquid within this portion of the tail was lost.

To study the convergence properties of the scheme, the mesh was refined and the order of the polynomial basis was varied. Figure 2.10 shows how the solution changes as the mesh was refined from 64^2 to 256^2 mesh points. When a finer mesh was used, the tail was almost totally captured; however, with the coarse mesh, the tail became under-resolved and broke into droplets. Because the liquid in the tail was moved into droplets the final solution does not match the expected exact solution. Similar results are shown in Fig. 2.11 wherein the effect of the DG order was analyzed. At the fully stretched state, the higher order polynomial had the ability to capture more of the tail region and less of the tail was broken into



(a) Reinitialization factor: $F = 0$ (solid), $F = 0.5$ (dashed), and $F = 1$ (dotted), Mesh=64^2, DG order=2

(b) Mesh: 64^2(solid), 128^2 (dashed), and 256^2 (dotted), $F = 0.5$, DG order=2



(c) DG Order: $O = 1$ (solid), $O = 2$ (dashed), and $O = 3$ (dotted), $F = 0.5$, Mesh=128^2

Figure 2.12: Mass of liquid normalized by initial mass plotted versus time for two-dimensional deformation test. Figure (a) shows results when the amount of reinitialization is varied. The effect of different meshes was plotted in Figure (b). The DG order was changed and the results are shown in Figure (c).

droplets. Also the difference between the higher order cases (2 and 3) is less than the difference between the lower order cases (1 and 2) indicating that 2^{nd} order polynomials may be a good compromise between cost and accuracy.

Conservation of mass was also studied. Figure 2.12(a) shows non-dimensionalized mass as a function of time for various amounts of reinitialization. Data from the case without reinitialization, $F = 0$, clearly shows that at the end of the simulation more than 4% of the mass was lost. When reinitialization was

used, $F = 0.5$ or $F = 1.0$, the amount of mass loss is significantly less. In the middle of the simulation the results indicate that there is significant mass loss for the cases that use reinitialization, this is not due to the mass being lost but rather arises due to the difference between the physical liquid volume fraction and our approximation of the quantity using the conservative level set. The difference between the two quantities results in an apparent loss of mass when the curvature of the interface changes dramatically like when the tail in the deformation test case is broken into droplets.

The effect of mesh refinement on the mass conservation properties of the scheme was studied and the results plotted in Fig. 2.12(b). As expected, even on the coarsest grid the amount of mass that was lost throughout the simulation was very small. On the finest grid the amount of the tail region that fell below mesh resolution and was transferred into resolvable droplets was the least and therefore the dip in the middle of the figure is the smallest. For the coarser meshes, more of the tail falls below mesh resolution and therefore more of the liquid is moved into droplets resulting in large curvature changes and the significant apparent loss of mass in the middle of the simulation.

Figure 2.12(c) shows the results from simulations with different DG orders. The differences in the results are very small indicating the order of the DG polynomials does not have a large effect on the solution for this test case.

2.6.3 Standing wave

The standing wave test case consists of the viscous damping of a surface wave and provides insight into problems that include significant interaction between

surface tension and viscous forces. The Navier-Stokes equations are solved using our CFD code known as NGA [56]. The test is two-dimensional with a domain of $[0, 2\pi] \times [0, 2\pi]$. Periodic boundary conditions are used in the x-direction and slip conditions on the top and bottom walls. Two fluids are placed in the domain separated by a flat interface perturbed by a sinusoidal wave. The initial interface location is given using the conservative level set, Ψ, by,

$$\Psi(x, y, t = 0) = \frac{1}{2}\left(\tanh\left(\frac{\pi - y + A_0 \cos(2\pi x/\lambda)}{2\varepsilon}\right) + 1\right), \tag{2.46}$$

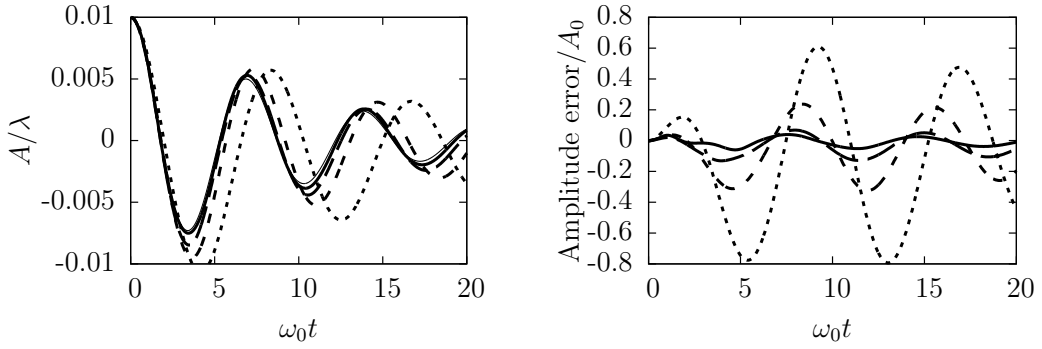
where λ is the perturbation wavelength which is set to 2π and A_0 is the initial amplitude of the wave chosen to be $A_0 = 0.01\lambda$. Prosperetti [58] derived an analytical solution to the evolution of the wave amplitude with time, provided the kinematic viscosity, ν, of both fluids are equal. For details of the analytical results the reader is referred to the paper by Prosperetti [58]. Here we only recall non-dimensionalization of time is performed using the inviscid oscillation frequency,

$$\omega_0 = \sqrt{\frac{\sigma}{\rho_l + \rho_g}}. \tag{2.47}$$

For our investigations the density ratio is set to either 1 or 1000 and three different meshes are tested, namely an 8^2 mesh, a 16^2 mesh, and a 32^2 mesh. Simulations were performed up to a non-dimensional time of $\omega_0 t = 20$ which captures approximately three interface oscillation periods. This parameter space was chosen to follow work done by Herrmann [59] and Desjardins et al. [28].

For the test cases with a density ratio of unity both fluid densities are set to 1. In the case with a density ratio of 1000, the liquid and gas densities were set to $\rho_l = 1000$ and $\rho_g = 1$, respectively. The non-dimensional surface tension coefficient was set to $\sigma = 2$ and the non-dimensional kinematic viscosity was set to $\nu = 0.064720863$ in both fluids. The time step was set to $\Delta t = 0.01$ for all of

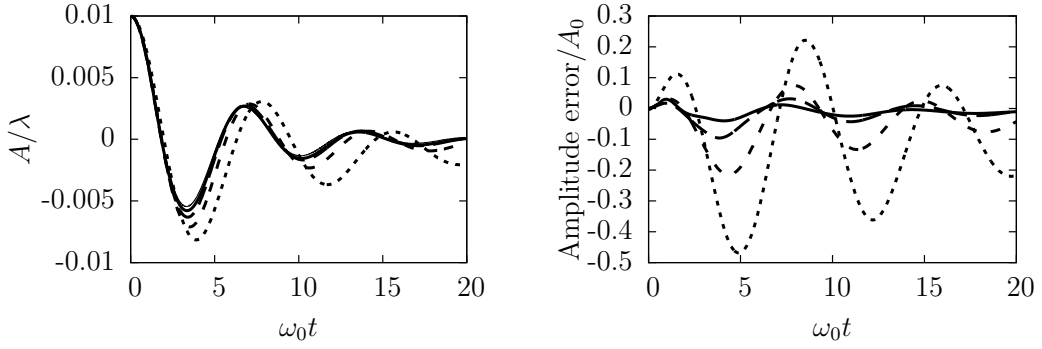
the mesh sizes considered. The DG parameters were fixed with the reinitialization factor $F = 0.5$ and second order DG polynomials.



(a) Non-dimensional wave amplitude versus non-dimensional time

(b) Non-dimensional wave amplitude error versus non-dimensional time

Figure 2.13: Standing wave test case with unity density ratio. 8×8 mesh shown with dotted line, 16×16 mesh given with dashed line, and thick solid line shows 32×32 mesh. Solid line in (a) provides the theoretical solution.



(a) Non-dimensional wave amplitude versus non-dimensional time

(b) Non-dimensional wave amplitude error versus non-dimensional time

Figure 2.14: Standing wave test case with density ratio of 1000. 8×8 mesh shown with dotted line, 16×16 mesh given with dashed line, and thick solid line shows 32×32 mesh. Solid line in (a) provides the theoretical solution.

Results are shown in Figs. 2.13 and 2.14 for a density ratio of 1 and 1000, respectively. The results include the wave amplitude versus time and the error between the computational and theoretical solutions as a function of time, shown in (a) and (b), respectively. Convergence to the theoretical solution with mesh

refinement is shown for both tests. The results are comparable to previous studies [28, 59, 60] and suggest that less than 16 cells are needed to accurately capture the wave physics.

2.6.4 Kelvin-Helmholtz instability

The Kelvin-Helmholtz (KH) instability can arise when velocity shear occurs in either a continuous fluid or at the interface of two fluids. The instability is important to a variety applications including atomization of liquid jets. In this test case we access the abilities of the DG scheme and the rest of our CFD code [56] to capture the KH instability and correctly predict the growth-rate of the instability with time. The problem we focused on is two-dimensional and multiphase with the gas on the top of the domain and the liquid on the bottom.

The frame of reference is defined so that the base flow velocity at the interface is zero leading to the following definitions of base flow velocity in the gas and liquid phases,

$$U_g(y) = U_{g,\infty}\text{erf}\left(\frac{y}{\delta_g}\right) \quad \text{for} \quad y > 0, \tag{2.48a}$$

$$U_l(y) = U_{l,\infty}\text{erf}\left(\frac{y}{\delta_l}\right) \quad \text{for} \quad y < 0. \tag{2.48b}$$

Note that gas and liquid are denoted by the subscripts g and l, respectively. $U_{g,\infty}$ and $U_{l,\infty}$ represent the asymptotic velocities away from the interface. The boundary layer thicknesses are represented by δ_g and δ_l. For reference, Fig. 2.15 shows a graphical representation of the parameters.

Equations 2.48a and 2.48b are not independent but are coupled by the continuity of shear stress at the interface. The four parameters $U_{g,\infty}$, $U_{l,\infty}$, δ_g, and δ_l

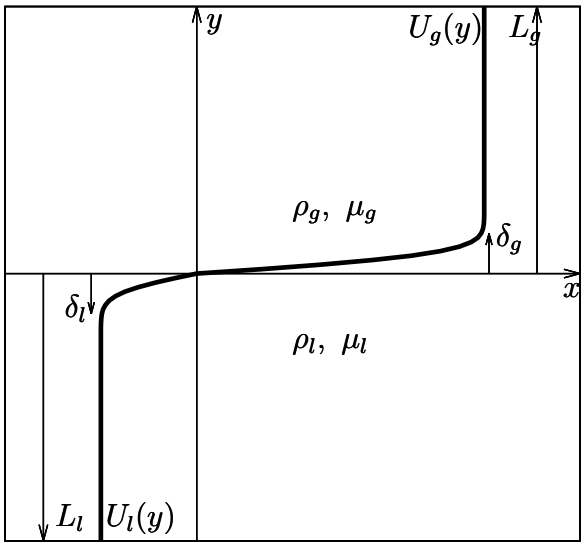


Figure 2.15: Geometry used for Kelvin-Helmholtz test case. Gas is in the top half of the domain and liquid in the bottom. The liquid and gas boundary layer thicknesses, δ_l and δ_g, are shown along with the distance from the interface to the top and bottom of the domain shown by L_g and L_l, respectively. The parallel base flow profile is depicted and labeled $U_g(y)$ in the gas and $U_l(y)$ in the liquid.

appearing in Eqs. 2.48a and 2.48b are related by,

$$\frac{\mu_g U_{g,\infty}}{\delta_g} = \frac{\mu_l U_{l,\infty}}{\delta_l}. \tag{2.49}$$

Therefore, only three of the four parameters can be chosen and the forth must be calculated using Eq. 2.49.

To analyze the growth-rate of a given disturbance to the base flow we utilize linear stability analysis, namely the Orr-Sommerfeld equations. As in the work of Bagué et al. [61], we write the disturbance to the base flow, u and v, in terms of the stream functions ψ_l and ψ_g using,

$$u_g = \frac{\partial \psi_g}{\partial y},\ v_g = \frac{\partial \psi_g}{\partial x}, \tag{2.50a}$$

$$u_l = \frac{\partial \psi_l}{\partial y},\ v_l\ = \frac{\partial \psi_l}{\partial x}. \tag{2.50b}$$

Note that there is no connection to $\boldsymbol{\psi}$ used in the DG representation of the conservative level set and ψ is used here to represent the stream functions for historical consistency. Since the domain is periodic in x and the base flow does not depend on time we may write the stream functions as

$$\psi_g = \phi_g(y)\exp(i\alpha(x-ct)) \quad \text{for} \quad y>0, \tag{2.51a}$$

$$\psi_l = \phi_l(y)\exp(i\alpha(x-ct)) \quad \text{for} \quad y<0, \tag{2.51b}$$

where ϕ_g and ϕ_l are the eigenfunctions within the gas and liquid, α is the real wave number, and the complex eigenvalue $c = c_r + ic_i$ provides the wave speed, c_r, and the growth-rate of the wave, αc_i. Throughout this section, the subscripts r and i refer to the real and imaginary parts of the variable, respectively.

The Orr-Sommerfeld equation describes the evolution of the linearized momentum equations due to the perturbation and are written in each phase,

$$U_g\phi_g'' - \alpha^2 U_g\phi_g - c\phi_g'' + c\alpha^2\phi_g - U_g''\phi_g = \frac{m}{r}\frac{1}{i\alpha \mathrm{Re}_l}\left(\phi_g^{(4)} - 2\alpha^2\phi_g'' + \alpha^4\phi_g\right) \text{ for } y>0, \tag{2.52a}$$

$$U_l\phi_l'' - \alpha^2 U_l\phi_l - c\phi_l'' + c\alpha^2\phi_l - U_l''\phi_l = \frac{1}{i\alpha \mathrm{Re}_l}\left(\phi_l^{(4)} - 2\alpha^2\phi_l'' + \alpha^4\phi_l\right) \quad \text{for } y<0, \tag{2.52b}$$

where $m = \mu_g/\mu_l$ is the viscosity ratio and $r = \rho_g/\rho_l$ is the density ratio. Re_l is the liquid Reynolds number and is defined, along with the other non-dimensional Reynolds and Weber numbers, as,

$$\mathrm{Re}_l = \frac{\rho_l U_{g,\infty}\delta_g}{\mu_l}, \quad \mathrm{Re}_g = \frac{\rho_g U_{g,\infty}\delta_g}{\mu_g}, \quad \mathrm{We}_l = \frac{\rho_l U_{g,\infty}^2\delta_g}{\sigma}, \quad \mathrm{We}_g = \frac{\rho_g U_{g,\infty}^2\delta_g}{\sigma}. \tag{2.53}$$

There are four boundary conditions for each forth order generalized eigenvalue problem. The four conditions (two for each equations) that are located at the

domain boundaries are,

$$\phi_g = \phi_g' = 0 \quad \text{for} \quad y = L_g, \tag{2.54a}$$

$$\phi_l = \phi_l' = 0 \quad \text{for} \quad y = -L_l, \tag{2.54b}$$

and provide a no slip conditions for the perturbations. The height of the domain is controlled by L_g and L_l which should be large enough to not impact the results. At the interface are the remaining four boundary conditions that ensure continuity of the normal and tangential components of the velocity as well as the normal and shear stresses. The boundary conditions can be written as,

$$\phi_g = \phi_l \qquad \text{for} \quad y = 0, \tag{2.55a}$$

$$\phi_g' + \frac{U_g'}{c}\phi_g = \phi_l' + \frac{U_l'}{c}\phi_l \qquad \text{for} \quad y = 0, \tag{2.55b}$$

$$-\frac{\alpha^2}{rc\mathrm{We}_l} = \frac{1}{r}(c\phi_l' + U_l'\phi_l) + \frac{1}{i\alpha r\mathrm{Re}_l}\left(\phi_l^{(3)} - 3\alpha^2\phi_l'\right) - \left(c\phi_g' + U_g'\phi_g\right) + \frac{m}{i\alpha r\mathrm{Re}_l}\left(\phi_g^{(3)} - 3\alpha^2\phi_g'\right) \qquad \text{for} \quad y = 0, \tag{2.55c}$$

$$\left(\phi_l'' + \alpha^2\phi_l + \frac{U_l''}{c}\phi_l\right) = m\left(\phi_g'' + \alpha^2\phi_g + \frac{U_g''}{c}\phi_g\right) \qquad \text{for} \quad y = 0. \tag{2.55d}$$

All of the boundary conditions except for the one given by Eq. 2.55c are linear with respect to the eigenvalue c. Because of the nonlinear condition, an iterative procedure needs to be performed to solve for the solution unless the boundary condition can be linearized. Boomkamp et al. [62], linearized Eq. 2.55c using Eqs. 2.55a and 2.55b resulting in,

$$\begin{aligned} -\frac{\alpha^2}{r\mathrm{We}_l}\frac{\phi_l' - \phi_g'}{U_l' - U_g'} = \frac{1}{r}(c\phi_l' + U_l'\phi_l) + \frac{1}{i\alpha r\mathrm{Re}_l}\left(\phi_l^{(3)} - 3\alpha^2\phi_l'\right) \\ -\left(c\phi_g' + U_g'\phi_g\right) + \frac{m}{i\alpha r\mathrm{Re}_l}\left(\phi_g^{(3)} - 3\alpha^2\phi_g'\right) \quad \text{for} \quad y = 0. \end{aligned} \tag{2.56}$$

The linearized form is appropriate when $U_l' \neq U_g'$. Our base flow meets this criteria and the linearized form of the boundary condition is used in this study to avoid the iterative procedure.

The Orr-Sommerfeld equation is solved numerically be approximating the eigenfunctions using a series of Chebyshev polynomials. The functions are orthogonal on an interval $[-1, 1]$. Therefore, we must transform the intervals $y = [0, L_g]$ and $y = [-L_l, 0]$ to $z = [-1, 1]$ using linear transforms. Using the transformed coordinate system the eigenfunctions can be approximated with the Chebyshev collocation method, namely $\phi_g(z) = \sum_{n=0}^{N} a_n T_n(z)$ and $\phi_l(z) = \sum_{n=0}^{N} b_n T_n(z)$, where T_n is the n^{th} Chebyshev polynomial of the first kind, and N is the number of polynomials used to approximate the eigenfunction. As described by Bagué et al. [61], N is an important parameter that is chosen so that a balance is achieved between errors that arise from a poor approximation of the eigenfunction and round off errors. A poor fit arises when a small number of Chebyshev polynomials are used and the series is not able to conform to the eigenfunctions. Round off errors dominate when a large number of polynomials are used. Therefore, the number of polynomials must be somewhere between the small number that causes a poor fit and the large number that cause round off errors. The approach we used to find a good value for N followed the method outlined by Bagué et al. [61] and consisted of increasing N until a range is found where the eigenvalue remains relatively constant. Then N is taken to be the beginning of this range.

Solution of the Orr-Sommerfeld equation results in the calculation of the eigenvalue, c, and the eigenvectors, ϕ_g and ϕ_l, which provide the theoretical growth-rate and the perturbed velocity field needed to initialize the simulation. The theoretical growth-rate is calculated from the imaginary part of the eigenvalue using, αc_i. The growth-rate is non-dimensionalized using the gas boundary layer thickness and the free stream gas velocity and can be written as $\alpha c_i \delta_g / U_{g,\infty}$. The initial perturbed velocity field used in simulations of the Kelvin-Helmholtz instability is the sum of the base flow and a perturbation that can be calculated from the eigenvectors

using,

$$U(x,y,t=0) = U_g(y) + \lambda\big(\phi'_{g,r}\cos(\alpha x) - \phi_{g,i}\sin(\alpha x)\big) \quad \text{for} \quad y>0, \tag{2.57a}$$

$$U(x,y,t=0) = U_l(y) + \lambda\big(\phi'_{l,r}\cos(\alpha x) - \phi_{l,i}\sin(\alpha x)\big) \quad \text{for} \quad y<0, \tag{2.57b}$$

$$V(x,y,t=0) = \lambda\alpha(\phi_{g,r}\sin(\alpha x) + \phi_{g,i}\cos(\alpha x)) \quad \text{for} \quad y>0, \tag{2.57c}$$

$$V(x,y,t=0) = \lambda\alpha(\phi_{l,r}\sin(\alpha x) + \phi_{l,i}\cos(\alpha x)) \quad \text{for} \quad y<0, \tag{2.57d}$$

where α is the amplitude of the perturbation which was set to $\alpha = 10^{-3}U_{g,\infty}$ for all of the test cases. The interface is also initially perturbed and the displacement of the level set is given using the classical level set,

$$\begin{aligned} G(x,y,t=0) = y + \frac{\lambda\alpha^2}{\alpha^2|c|^2}[c_i(\phi_{g,i}(y=0)\cos(\alpha x) + \phi_{g,r}(y=0)\sin(\alpha x)) \\ +c_r(\phi_{g,r}(y=0)\cos(\alpha x) - \phi_{g,i}(y=0)\sin(\alpha x))], \end{aligned} \tag{2.58}$$

where $\phi_{g,r}(y=0)$ and $\phi_{g,i}(y=0)$ are the real and imaginary parts of the gas eigenfunction evaluated at the interface where $y=0$. The conservative level set can be calculated from the classical level set using Eq. 2.6.

The growth-rate from the simulations was calculated from the temporal evolution of the perturbation. Initially, the disturbance is sinusoidal and during the linear growth period remains a sinusoidal function. Therefore, we found the amplitude of the perturbation by fitting $y = \beta\sin(2\pi x/L_x + \xi)$ through the interface, where β is the amplitude and ξ is the phase of the sinusoidal function. β and ξ are varied to fit the function to the interface. The growth-rate is calculated from the slope of $\log(\beta)$ plotted versus time. It is important to only calculate the growth-rate from the linear stability regime since non-linear growth of the stability is not described by the Orr-Sommerfeld equation. In order to define a systematic method to find the end of the linear stability regime we plot $\log(\beta)$ versus time at each time-step and calculate the coefficient of determination, R^2. If the value of R^2,

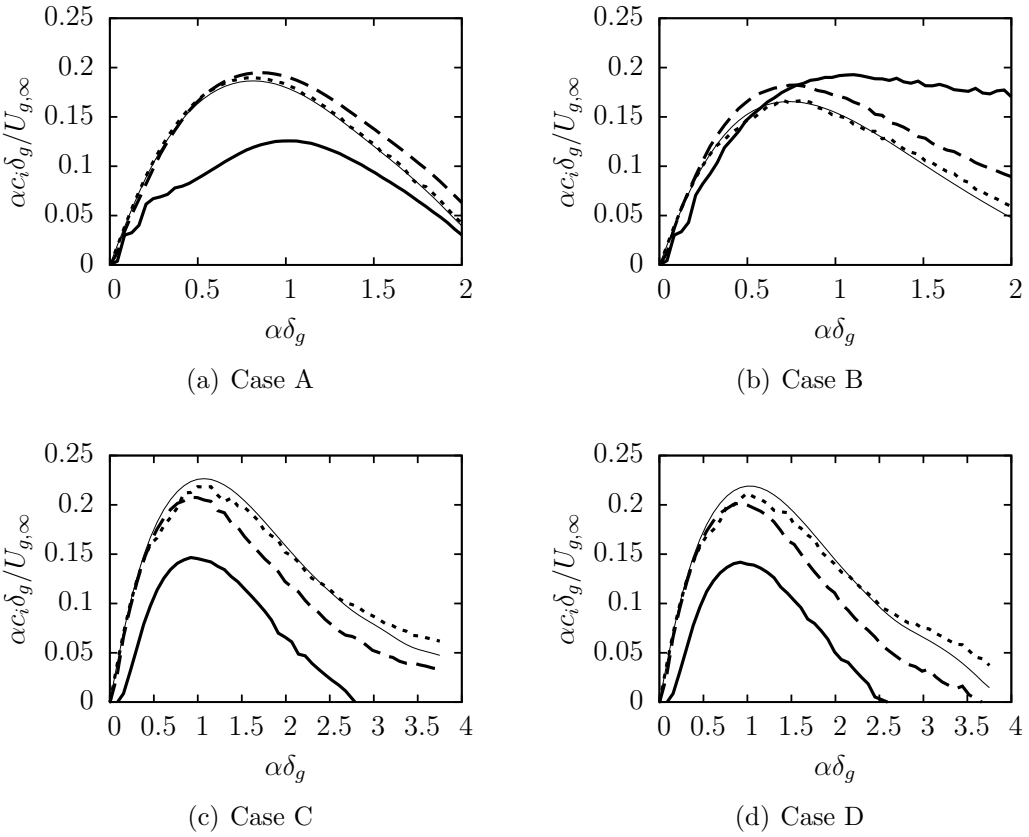
found using all previous values of β and the one from the current time step, is less than the previous value or R^2 then the new data point is an outlier and therefore is the beginning of the non-linear stability region. We should remark that the actual distinction between the linear and non-linear stability regimes is difficult to define and we use the aforementioned method as a systematics way to find the transition and not a strict mathematical definition of the transition.

Case	$r = \rho_g/\rho_l$	$m = \mu_g/\mu_l$	Re_g	Re_l	We_g	We_l
A	0.1	1	2000	200	∞	∞
B	0.1	1	2000	200	10	10
C	0.99	0.1	2000	19800	∞	∞
D	0.99	0.1	2000	19800	10	10

Table 2.3: Non-dimensional numbers used to setup the four cases used in the study of the Kelvin-Helmholtz instability.

Four different cases were considered following a previously investigation by Bagué et al. [61]. For all of the cases the vertical domain size was set by $L_g = 6\delta_g$ and $L_l = 6\delta_l$ and the boundary layer thicknesses were set to $\delta_g = \delta_l = 2.5 \times 10^{-3}\,\mathrm{m}$. The free stream velocity in the gas phase was a constant for all of the cases with a value of $U_{g,\infty} = 10\,\mathrm{m\,s^{-1}}$. The gas properties, $\rho_g = 1\,\mathrm{kg\,m^{-3}}$ and $\mu_g = 1.25 \times 10^{-5}\,\mathrm{kg\,m^{-1}\,s^{-1}}$, were also fixed throughout the study. Liquid properties were varied by changing the density ratio and viscosity ratio. Cases A and B have a density ratio of unity, $r = 1$, and a viscosity ratio of $m = 0.1$. The density and viscosity ratios are changed to $r = 0.1$ and $m = 0.99$ for case C and D. The surface tension coefficient is also varied and is set to zero in cases A and C and to $\sigma = 2.5 \times 10^{-3}\,\mathrm{J^2\,m^{-1}}$ in cases B and D. Table 2.3 shows the important non-dimensional numbers for the four test cases. The width of the domain, L_x, is set to the same size of the wavelength, α, so that one period of the disturbance fits within the computational space.

For the four test cases, numerous simulations were run with varying perturba-



(a) Case A

(b) Case B

(c) Case C

(d) Case D

Figure 2.16: Numerical and theoretical results for the four Kelvin-Helmholtz cases. Numerical results are shown on three different meshes namely 8^2 (thick solid), 32^2 (dashed), and 128^2 (dotted). The thin solid line shows the theoretical solution.

tion wavelengths. For each simulation a growth-rate was calculated. Figure 2.16 shows plots of the growth-rate versus the wavenumber. The process was repeated using different meshes and the results from 8^2, 32^2, and 128^2 are shown on the plots. The theoretical growth-rate was also calculated for each wavelength and plotted for reference. As the mesh is refined the numerical growth-rates approach the theoretical growth-rates for all of the test cases. The coarsest mesh, 8^2, is capable of predicting the general trend of the growth-rate as a function of wavelength. The simulations on the 32^2 mesh predict the growth-rates very well and could provide sufficient resolution for simulations that include Kelvin-Helmholtz type instabilities. Additional meshes were also tested, namely 16^2 and 64^2. Using

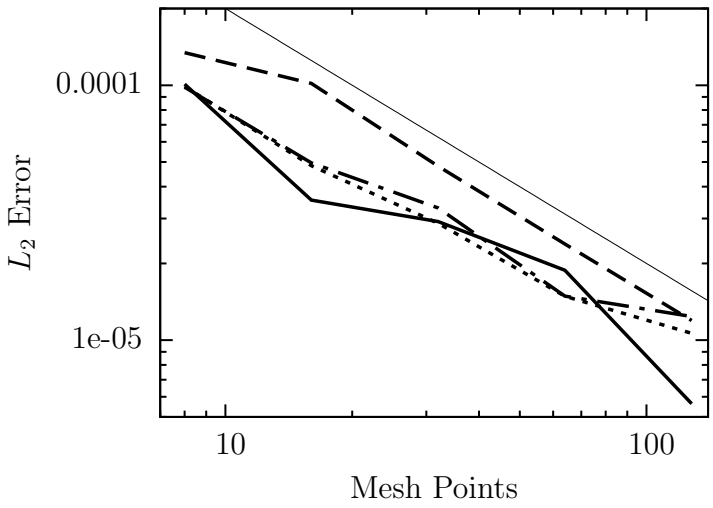


Figure 2.17: Mesh refinement convergence for the four test cases used in the Kelvin-Helmholtz test case. Figure shows L_2 error versus the number of mesh points for Case A (thick solid), Case B (dashed), Case C (dotted), and Case D (dot-dashed). The thin solid line shows first order convergence.

the results from all of the simulations a convergence plot, Fig. 2.17, was produced wherein the L_2 error between the numerical and theoretical growth-rates for all wavelengths is plotted. For all four test cases first order convergence was demonstrated suggesting the DG formulation and our CFD code are capable of capturing the Kelvin-Helmholtz instability and the quality of the results should improve with mesh refinement.

2.7 Conclusions

A discontinuous Galerkin discretization of the conservative level set has been developed. The conservative level set provides good mass conservation and DG offers an arbitrarily high order representation of the level set while only requiring communication with neighbors that share a face. This novel combination of the accurate conservative level set with a DG discretization leads to an accurate, highly

parallelizable scheme, with good mass conservation properties. The scheme was enhanced by the addition of a maximum/minimum preserving (MMP) limiter that works to maintain the boundedness of the level set. Calculation of the curvature was improved by using the height function approach commonly used on VOF formulations. The method results in a curvature that converges with second order accuracy and has low errors. The DG discretization and curvature calculation were applied to a variety of test cases including Zalesak's disk, a two-dimensional deformation test case, the viscous damping of a surface wave, and an investigation of the Kelvin-Helmholtz instability. The test cases highlighted the accuracy and mass conserving properties of the scheme under a wide range of parameters.

CHAPTER 3

CONSERVATIVE SECOND-ORDER GEOMETRIC VOLUME-OF-FLUID METHOD

3.1 Introduction

In simulations of multiphase flows, an accurate representation of interface motion is important for many interesting engineering applications with significant interface topology changes. Large discontinuities can exist at the interface, including large jumps in pressure, density, and viscosity. Therefore, various numerical schemes have been developed to track the interface location. These schemes can be broadly classified as interface tracking and interface capturing. Interface tracking schemes represent the interface explicitly using, for example, a mesh that deforms with the interface [18] or marker particles on the interface [22]. When the interface undergoes significant deformation and breaking or merging events, interface tracking schemes suffer from the need to frequently perform re-meshing or re-seeding of marker particles. Interface capturing schemes implicitly represent the interface and include level set methods [23] and volume-of-fluid (VOF) schemes [25]. Level set methods can be very accurate but suffer from the lack of discrete mass conservation. The conservation properties were improved through the development of the conservative level set [28, 29, 41, 63], but the method still lacks discrete conservation. VOF methods have the potential to provide discrete mass conservation and second-order accuracy, and are the basis for this work.

VOF methods track the interface by storing the ratio of liquid volume to cell volume for each computational cell, known as the liquid volume fraction. VOF methods were introduced in the early 1970's when DeBar [31], Nichols and Hirt [32],

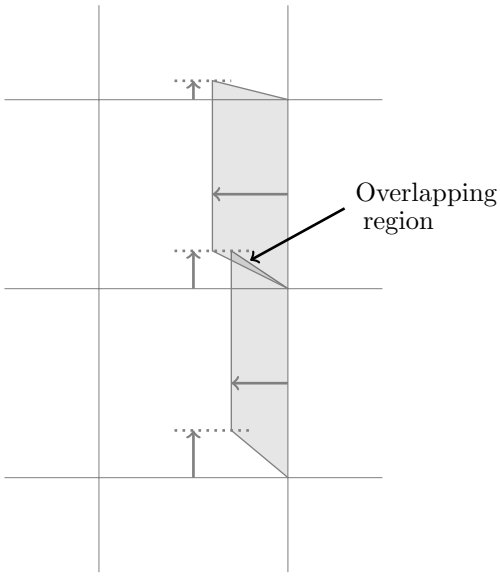
and Noh and Woodward [33] all developed variants of VOF within a short period of time. Many advancements have improved VOF schemes since their inception through improved representations of the liquid within the domain and improved advection schemes. Tryggvason et al. [37] provide a detailed history of VOF methods with descriptions of many of the significant contributions. In piecewise linear interface calculation (PLIC) methods, the interface is approximated within each cell using a straight line (2D) or plane (3D) [31, 34, 35]. PLIC methods mainly differ in the algorithm used to orient the linear function through the calculation of the interface normal vector. In this work we use the PLIC interface representation with an interface normal computed using the ELVIRA method [26], although the methodology can readily be used with other interface normal calculation strategies.

VOF schemes also differ in how the liquid volume fraction is transported. Early VOF schemes used flux splitting, wherein the multidimensional transport step is replaced by successive one-dimensional transport steps [25]. This approach, which is straightforward to implement, suffers from errors due to the flux splitting step. Alternatively, un-split schemes that avoid this source of error have been constructed. Pilliod and Puckett [26] developed a second-order un-split transport algorithm by computing fluxes by integrating over volumes formed by characteristics in space-time. The method is shown to produce superior results to split methods, however the extension to three dimensions is not provided.

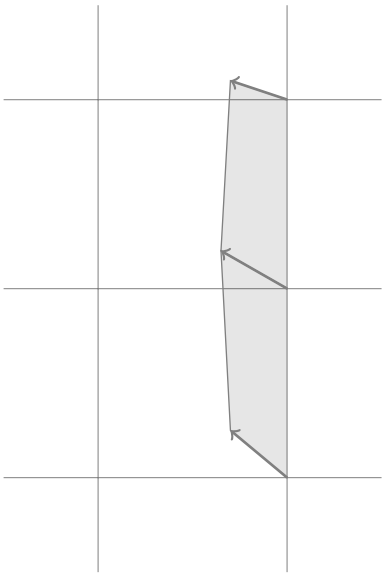
Geometric un-split transport schemes provide a framework for constructing fluxes that are consistent with the characteristics of the problem and are used in this work. Rider and Kothe [35] proposed an un-split geometric advection scheme that uses trapezoidal flux regions constructed using cell face velocities as shown in Fig. 3.1(a). In general, neighboring faces will have different velocities, which

results in regions that overlap and are fluxed twice. López et al. [2] developed EMFPA as an improvement to Rider and Kothe's method. EMFPA uses cell vertex velocities to create the flux region and avoids any overlapping regions as shown in Fig. 3.1(b). The present work is an extension of EMFPA to three dimensions and the two schemes produce very similar results in two dimensions. Note that EMFPA is more straightforward to implement than the proposed scheme if the reader is strictly interested in two dimensional problems. Hernández et al. [3] developed a three-dimensional flux calculation method known as FMFPA-3D that is based on the velocity on each edge of the cell face as shown in Fig. 3.1(c). The resulting fluxes can have overlapping regions that hinder the conservation properties of the scheme. The proposed method constructs three-dimensional flux regions using cell corner velocities as shown in Fig. 3.1(d). The resulting flux volumes do not overlap and provide a framework for un-split, conservative, bounded, three-dimensional transport. The approach is complicated due to the presence of non-flat faces on the flux region that are produced when the corner vertex velocity vectors do not lie in the same plane. Furthermore, the flux volumes can become crossed [64]. However, a straightforward, systematic approach is presented in this work to deal with the complex geometry.

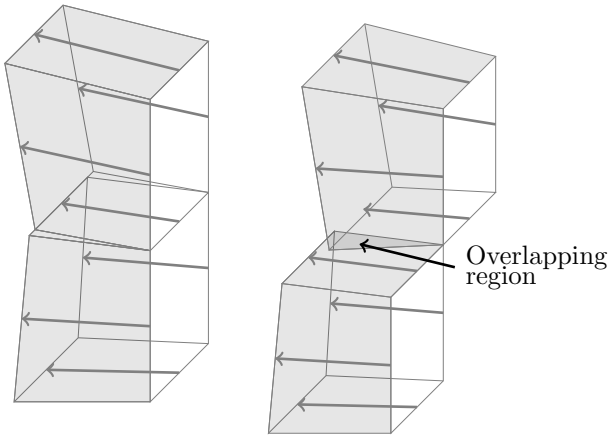
The proposed VOF scheme shares similarities with other geometric VOF methods. The HyLEM method of Le Chenadec and Pitsch [36] is a semi-Lagrangian transport scheme that updates the liquid volume fraction by projecting the cell forward and backward in time. HyLEM is not flux-based and is not discretely conservative. The proposed scheme can be seen as a flux based version of HyLEM, wherein the fluxes have been modified to ensure discrete conservation. If the fluxes are not modified, the two schemes are equivalent [65]. The voFoam method of Marić et al. [66] is a three-dimensional, un-split, conservative flux based VOF



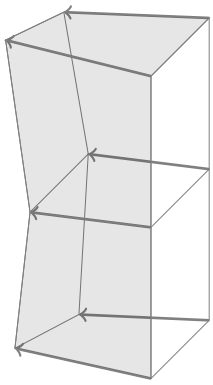
(a) 2D fluxes based on face velocities, Rider and Kothe [35]



(b) 2D fluxes based on vertex velocities, López et al. [2]



(c) 3D fluxes formed using planes based on edge velocities, Hernández et al. [3]



(d) 3D fluxes based on vertex velocities, proposed scheme

Figure 3.1: Example of methods used to compute geometric fluxes. Velocity vectors used to construct fluxes are shown with arrows. Fluxes are shaded.

scheme that is similar to the proposed method. voFoam is shown to work well, however it is unclear how crossed fluxes, wherein the flux moves liquid into and out of a cell through the same face, are dealt with. In three dimensions it is not obvious how to deal with the complex geometries that arise in such cases. Furthermore, it is unclear how the flux volumes are rescaled to construct conservative fluxes without introducing overlapping regions between neighboring flux volumes. The EMFPA-3D of Ivey and Moin [67] is similar to voFoam. Details are not provided on how crossed flux volumes are handled. Unfortunately, their approach to form conservative flux volumes introduces overlapping regions. Furthermore, their study fails to provide verification or validation test cases for the method.

In the proposed scheme, purely geometrical operations are used to calculate the fluxes. The flux geometry is systematically partitioned into a collection of simplices (triangles or tetrahedra in two or three dimensions, respectively). Simplices are easier to work with computationally and, when coupled with an appropriate sign convention, lead to a straightforward scheme that relies on a small number of geometric routines. The approach naturally handles crossed flux volumes. The method uses a correction to the flux volumes to ensure discrete conservation that does not introduce overlapping regions between neighboring flux volumes. The correction is performed using an analytic expression that avoids the need for an iterative procedure. With this framework, conservative un-split transport can be performed. In this paper, we apply this transport methodology to the VOF interface capturing strategy. The proposed method could be used to transport other quantities. For example, both momentum and the gas-liquid interface could be transported, forming a method similar to the scheme of Le Chenadec and Pitsch [65] but with the addition of discrete conservation of mass and momentum. This scheme will be considered in a future publication.

This paper begins with a detailed mathematical derivation of the method in Section 3.2. Section 3.3 provides a description of the computational building blocks that are used in the method. The section includes details on the reconstruction of the interface from the liquid volume fraction, the construction and discrete representation of the flux volumes, and the calculation of the fluxes. Canonical test cases including Zalesak's disk, two- and three-dimensional deformation tests, and the time evolution of a drop in synthetic homogeneous isotropic turbulence are presented in Section 3.4. Finally, conclusions are drawn in Section 3.5.

3.2 Mathematical formulation

In this section a detailed derivation of the conservation laws applied to a fixed control volume is presented. The derivation recasts a scalar advection partial differential equation (PDE) into a relation for the evolution of a scalar in a fixed control volume due to geometric fluxes. While the resulting equations, Eq. 3.13 for a general advected function and Eq. 3.18 for the liquid volume fraction, are similar to previously published un-split geometric transport advection equations, see for example [2,3,35,66,68,69], this derivation is useful in that it shows these are exact relations. The derivation highlights that exact fluxes to advect a function from time t^n to time t^{n+1} can be computed by constructing streak-tubes emitted from each face of the control volume and evaluating the advected function within the streak-tube at time t^n.

3.2.1 Problem setup and notations

The material evolution of a conserved scalar $f(\boldsymbol{x}, t)$ in a solenoidal velocity field is described by

$$\frac{\partial f}{\partial t} + \nabla \cdot (\boldsymbol{u} f) = 0, \tag{3.1}$$

where $\boldsymbol{x}$ is the spatial coordinate, t is time, and $\boldsymbol{u}$ is the velocity field that is assumed to be known. Integrating this equation over a discrete time-step $\Delta t = t^{n+1} - t^n$ and fixed control volume CV (e.g., a computational cell) with bounding surface CS and using Gauss' theorem on the second term allows us to write

$$\int_{CV} \left(f\left(\boldsymbol{x}, t^{n+1}\right) - f(\boldsymbol{x}, t^n) \right) \mathrm{d}V + \int_{t^n}^{t^{n+1}} \oint_{CS} f \boldsymbol{u} \cdot \boldsymbol{n}_{CV} \, \mathrm{d}S \, \mathrm{d}t = 0. \tag{3.2}$$

The term on the right is the flux through the surface of the control volume and depends on $f(\boldsymbol{x}, t)$ throughout the time-step, which is not usually known. Typically, a discrete representation of $f(\boldsymbol{x}, t^n)$ is known and an update equation is used to calculate $f(\boldsymbol{x}, t^{n+1})$. Therefore, we recast the flux so it depends solely on $f(\boldsymbol{x}, t^n)$. To do this we start by partitioning the surface of the control volume CS into sub-surfaces ∂CS_i (e.g., the faces of our computational cell) such that

$$\begin{aligned} &CS = \bigcup_{i=1}^{N_S} \partial CS_i \text{ and} \\ &\partial CS_i \cap \partial CS_j = \emptyset \text{ for } i \text{ and } j \in \{1, \ldots, N_S\} \text{ and } i \neq j. \end{aligned} \tag{3.3}$$

To each sub-surface ∂CS_i we associate the flux volume $\Omega_i(t)$ with bounding surface $\omega_i(t)$. $\Omega_i(t)$ is the signed volume that flows through the sub-surface ∂CS_i between time t and t^{n+1}. Hence, $\Omega_i(t^{n+1})$ is zero by definition and $\Omega_i(t^n)$ is the volume that flows through ∂CS_i during Δt. Figure 3.2 shows an example control volume with surface CS that has been partitioned with four sub-surfaces ∂CS_i for $i = 1, \ldots, 4$. The flux volume associated with each sub-surface is shown. The sign of the volume is defined to be negative if the volume moves through the sub-surface and into the

CV during the step and positive if the volume moves through the sub-surface and out of the CV. Formal definitions of the flux volumes and signs are provided in the derivation below.

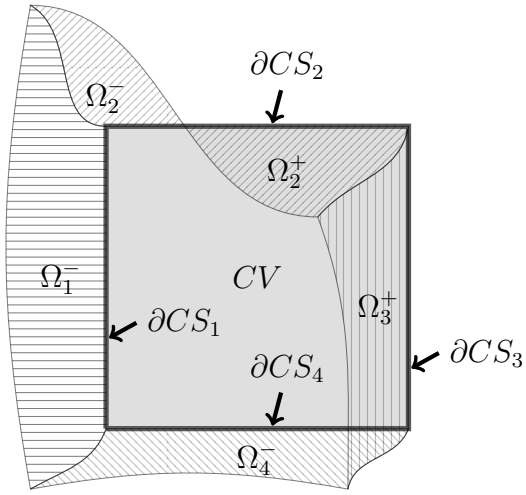


Figure 3.2: Example geometry used to construct the flux volumes. The control volume CV is shown as the shaded region. The bounding surface of CV has been partitioned into four sub-surfaces ∂CS_i with $i = 1, \ldots, 4$. Associated with each sub-surface is the flux volume Ω_i. Each flux volume is indicated with a different pattern of lines.

Integrating Eq. 3.1 over the flux volume Ω_i and using Gauss' theorem on the second term allows us to write

$$\int_{\Omega_i(t)} \frac{\partial f}{\partial t} \, \mathrm{d}V + \oint_{\omega_i(t)} f \boldsymbol{u} \cdot \boldsymbol{n}_{\Omega_i} \, \mathrm{d}S = 0. \tag{3.4}$$

In this equation, $\boldsymbol{n}_{\Omega_i}$ is the outward-pointing normal to $\Omega_i(t)$. We define ω_i such that part of it coincides with ∂CS_i, which is fixed in time. The rest of ω_i is defined to have a zero flux of f and thus must be a material surface that moves with the flow. Therefore, ω_i can be partitioned into two sub-regions, namely $\omega_{i,F} = \omega_i \cap \partial CS_i = \partial CS_i$ that is fixed and $\omega_{i,M} = \omega_i \setminus \partial CS_i$ that is a material surface. Integrating the previous equation over Δt and using this partition, we can

write

$$\int_{t^n}^{t^{n+1}} \int_{\Omega_i(t)} \frac{\partial f}{\partial t} \,\mathrm{d}V \,\mathrm{d}t + \int_{t^n}^{t^{n+1}} \int_{\omega_{i,M}(t)} f\boldsymbol{u} \cdot \boldsymbol{n}_{\Omega_i} \,\mathrm{d}S \,\mathrm{d}t + \int_{t^n}^{t^{n+1}} \int_{\partial CS_i} f\boldsymbol{u} \cdot \boldsymbol{n}_{\Omega_i} \,\mathrm{d}S \,\mathrm{d}t = 0. \tag{3.5}$$

The last term is very similar to the flux term in Eq. 3.2 and will provide the connection between the two equations. However, the first two terms in this equation are difficult to deal with in their current form but can be elucidated using Leibniz's rule which states

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega_i(t)} f \,\mathrm{d}V = \int_{\Omega_i(t)} \frac{\partial f}{\partial t} \,\mathrm{d}V + \int_{\omega_{i,M}(t)} f\boldsymbol{u} \cdot \boldsymbol{n}_{\Omega_i} \,\mathrm{d}S, \tag{3.6}$$

where we have used $\omega_i = \omega_{i,M} \cup \omega_{i,F}$, the property $\boldsymbol{u}_s = 0$ on $\omega_{i,F}$, and defined $\boldsymbol{u}_s = \boldsymbol{u}$ on $\omega_{i,M}$, which makes $\omega_{i,M}$ a material surface. At this point, it should be clear that $\Omega_i(t)$ is a streak-tube emitted backward in time from the surface ∂CS_i during the time period t to t^{n+1}. Integrating the previous equation over the time-step results in

$$\begin{aligned} \int_{\Omega_i(t^{n+1})} f\big(\boldsymbol{x}, t^{n+1}\big) \,\mathrm{d}V - \int_{\Omega_i(t^n)} f(\boldsymbol{x}, t^n) \,\mathrm{d}V \\ = \int_{t^n}^{t^{n+1}} \int_{\Omega_i(t)} \frac{\partial f}{\partial t} \,\mathrm{d}V \,\mathrm{d}t + \int_{t^n}^{t^{n+1}} \int_{\omega_{i,M}(t)} f\boldsymbol{u} \cdot \boldsymbol{n}_{\Omega_i} \,\mathrm{d}S \,\mathrm{d}t. \end{aligned} \tag{3.7}$$

By definition, $\Omega_i(t^{n+1})$ has a zero volume and the first term in the previous equation is zero. Henceforth, we will call $\Omega_i(t^n)$ the flux volume and adopt the notation $\Omega_i = \Omega_i(t^n)$. Subtracting Eq. 3.7 from Eq. 3.5 and using this notation leads to

$$\int_{t^n}^{t^{n+1}} \int_{\partial CS_i} f\boldsymbol{u} \cdot \boldsymbol{n}_{\Omega_i} \,\mathrm{d}S \,\mathrm{d}t = \int_{\Omega_i} f(\boldsymbol{x}, t^n) \,\mathrm{d}V, \tag{3.8}$$

which provides a simple relationship between the flux through the sub-surface ∂CS_i and the volume integral over Ω_i. The previous equation states that the flux of f through ∂CS_i during the time-step is equivalent to the integral of f in the flux volume at the beginning of the time-step.

The previous equation can almost be combined with Eq. 3.2, leading to a useful time advancement equation. However, the flux term in Eq. 3.8 contains $\boldsymbol{n}_{\Omega_i}$, the outward-pointing normal to Ω_i, while the normal in Eq. 3.2 is $\boldsymbol{n}_{CV}$, which is outward-pointing with respect to CV. Because the normal vectors are defined using the same surface ∂CS_i, they are either identical (i.e., $\boldsymbol{n}_{CV} \cdot \boldsymbol{n}_{\Omega_i} = +1$) or point in opposite directions (i.e., $\boldsymbol{n}_{CV} \cdot \boldsymbol{n}_{\Omega_i} = -1$). As a result, we partition the sub-surfaces ∂CS_i into two regions ∂CS_i^+ and ∂CS_i^- such that

$$\begin{aligned} &\partial CS_i = \partial CS_i^+ \cup \partial CS_i^- \text{ and} \\ &\partial CS_i^+ \cap \partial CS_i^- = \emptyset. \end{aligned} \tag{3.9}$$

The sub-surfaces are defined using

$$\partial CS_i^+ = \{\boldsymbol{x} \in \partial CS_i \mid \boldsymbol{n}_{CV}(\boldsymbol{x}) \cdot \boldsymbol{n}_{\Omega_i}(\boldsymbol{x}) = +1\} \tag{3.10}$$

and

$$\partial CS_i^- = \{\boldsymbol{x} \in \partial CS_i \mid \boldsymbol{n}_{CV}(\boldsymbol{x}) \cdot \boldsymbol{n}_{\Omega_i}(\boldsymbol{x}) = -1\}. \tag{3.11}$$

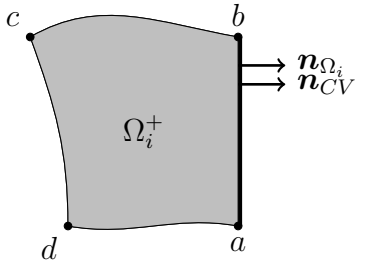
Furthermore, we associate to the sub-surfaces ∂CS_i^+ and ∂CS_i^- the flux volumes Ω_i^+ and Ω_i^-, respectively. The flux volumes are defined using the same methodology as described previously, and can therefore be thought of as streak-tubes. Figure 3.3 shows three example fluxes with positive, negative, and positive and negative flux volumes, respectively.

With these definitions, Eq. 3.8 can be written as

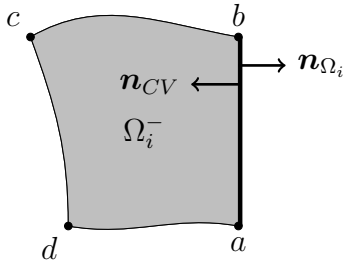
$$\int_{t^n}^{t^{n+1}} \int_{\partial CS_i} f\boldsymbol{u} \cdot \boldsymbol{n}_{CV} \,\mathrm{d}S \,\mathrm{d}t = \int_{\Omega_i^+} f(\boldsymbol{x}, t^n) \,\mathrm{d}V - \int_{\Omega_i^-} f(\boldsymbol{x}, t^n) \,\mathrm{d}V. \tag{3.12}$$

Finally, combining Eq. 3.2 with Eq. 3.12 leads to

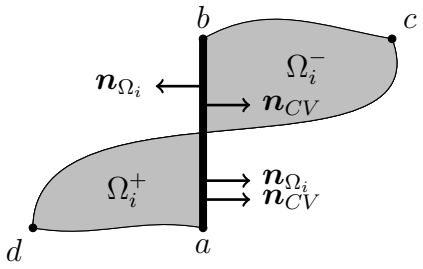
$$\int_{CV} \big(f\big(\boldsymbol{x}, t^{n+1}\big) - f(\boldsymbol{x}, t^n)\big) \,\mathrm{d}V + \sum_{i=1}^{N_S} \left[\int_{\Omega_i^+} f(\boldsymbol{x}, t^n) \,\mathrm{d}V - \int_{\Omega_i^-} f(\boldsymbol{x}, t^n) \,\mathrm{d}V \right] = 0. \tag{3.13}$$



(a) Simple example with positive flux volume, $\Omega_i^- = \emptyset$.



(b) Simple example with negative flux volume, $\Omega_i^+ = \emptyset$.



(c) Example with crossed flux volume with both positive and negative regions.

Figure 3.3: Example of the geometry used to define the flux volume Ω_i (shaded region) with outward-facing normal $\boldsymbol{n}_{\Omega_i}$. The flux volume is associated with the sub-surface ∂CS_i shown with the thick line with vertices a and b. $\boldsymbol{n}_{CV}$ the outward-facing normal to CV is also shown.

The term on the left of this equation describes the change in f within the control volume CV during the time-step. The change is due to fluxes into or out of the control volume, as described by the term on the right. The flux term has been recast into a volume integral over the flux volumes, Ω_i^+ and Ω_i^-, that move out of and into the control volume, respectively.

3.2.2 Flux velocity

Until now, we have only considered one control volume. However, the control volumes represent the computational cells used in a simulation, so it is necessary to extend the notation to describe a collection of control volumes. Let the number of control volumes used in the simulation be N_{CV} and the p^{th} control volume be denoted CV_p with bounding surface CS_p, which has been partitioned into sub-surfaces $\partial CS_{p,i}$ for $i = 1, \ldots, N_S$. Similarly, the flux volume associated with the sub-surface $\partial CS_{p,i}$ is indicated as $\Omega_{p,i}$.

We introduce useful quantities associated with $\partial CS_{p,i}$ and $\Omega_{p,i}$. We define the area $\mathcal{A}_{p,i} = \int_{\partial CS_{p,i}} \mathrm{d}S$ and the signed volume $\mathcal{V}_{p,i} = \int_{\Omega^+_{p,i}} \mathrm{d}V - \int_{\Omega^-_{p,i}} \mathrm{d}V$. From the area, volume, and the time interval Δt, a mean normal fluxing velocity $\mathcal{U}_{p,i}$ can be defined as

$$\mathcal{U}_{p,i} = \frac{\mathcal{V}_{p,i}}{\Delta t \mathcal{A}_{p,i}}. \tag{3.14}$$

To ensure discrete conservation, the flux velocities must respect the solenoidal condition $\sum_{i=1}^{N_s} \mathcal{U}_{p,i}\mathcal{A}_{p,i} = 0$.

3.2.3 Liquid volume fraction transport

To examine liquid volume fraction transport, we choose

$$f(\boldsymbol{x}, t) = \begin{cases} 1, & \text{if } \boldsymbol{x} \text{ is in the liquid at time } t, \\ 0, & \text{if } \boldsymbol{x} \text{ is in the gas at time } t, \end{cases} \tag{3.15}$$

and introduce the shorthand notations

$$\alpha_p(t) = \frac{1}{\mathcal{V}_p} \int_{CV_p} f(\boldsymbol{x}, t)\, \mathrm{d}V \tag{3.16}$$

where $\mathcal{V}_p = \int_{CV_p} \mathrm{d}V$ and

$$\alpha_{p,i} = \frac{1}{\mathcal{V}_{p,i}} \left(\int_{\Omega^+_{p,i}} f(\boldsymbol{x}, t^n)\, \mathrm{d}V - \int_{\Omega^-_{p,i}} f(\boldsymbol{x}, t^n)\, \mathrm{d}V \right), \tag{3.17}$$

which are the liquid volume fraction within the p^{th} control volume CV_p and signed flux volume $\Omega_{p,i}$, respectively.

With these notations, Eq. 3.13 can be written as

$$\frac{\alpha_p(t^{n+1}) - \alpha_p(t^n)}{\Delta t} + \frac{1}{\mathcal{V}_p} \sum_{i=1}^{N_S} (\alpha_{p,i} \mathcal{U}_{p,i} \mathcal{A}_{p,i}) = 0, \tag{3.18}$$

which describes the change in liquid volume fraction within the p^{th} computational cell due to fluxes, defined using streak-tubes, through the cell faces. No discretization choices have been made in the derivation of the previous equation and it is the equivalent of Eq. 3.1, where f is chosen to be the liquid volume fraction, written for an arbitrary control volume.

3.3 Computational geometry toolbox

The framework presented heretofore requires calculating the volume and liquid volume fraction associated with the flux volumes $\Omega_{p,i}$ using a scheme that is accurate, efficient, and implementable in three dimensions. In this section, we present the computational tools needed to make the necessary calculations.

3.3.1 Interface reconstruction

The interface reconstruction step corresponds to the process of calculating an approximation to the interface location from the liquid volume fraction. The interface

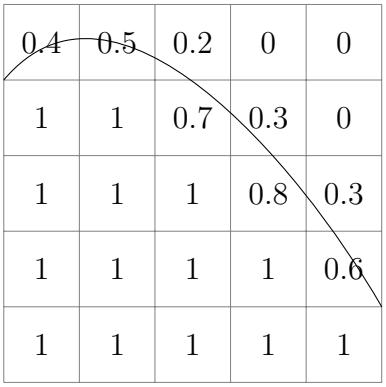


Figure 3.4: Representation of interface (solid line) using a VOF scheme. Liquid volume fraction α shown with numbers.

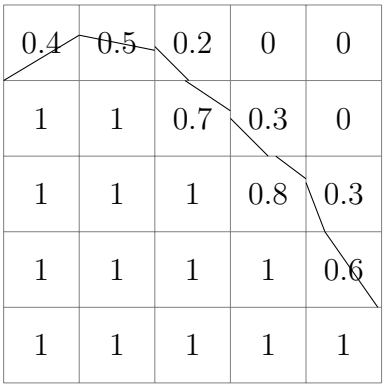


Figure 3.5: Example of PLIC reconstruction of interface from liquid volume fraction.

location is needed to determine $\alpha_{p,i}$, the liquid volume fraction within the flux volume $\Omega_{p,i}$. A variety of methods have been developed in order to perform that reconstruction step [25, 31, 33, 34]. We are using a three-dimensional extension of the piecewise linear interface calculation (PLIC) method [31, 34], wherein the interface is represented using a piecewise planar reconstruction within each computational cell. Figure 3.5 shows an example of how the reconstructed interface may look in two dimensions.

Within each computational cell, the plane is uniquely defined with the interface normal vector and the liquid volume fraction. The normal vector provides the direction of the plane and the liquid volume fraction constrains the location of the plane such that the cell volume on the liquid side of the plane equals $\alpha_p \mathcal{V}_p$. In this work, the second-order interface normal calculation method known as ELVIRA [26] is used. Scardovelli and Zaleski [70] developed analytical relationships that permit the calculation of the PLIC reconstruction plane from the normal and liquid volume fraction directly. The combination of the analytical relations, the interface normal, and the liquid volume fraction provides a unique piecewise planar representation

of the interface. Alternatively, an iterative approach such as Brent's method [71] could be used to form the PLIC reconstruction.

3.3.2 Discrete representation of the flux volume

Tessellation

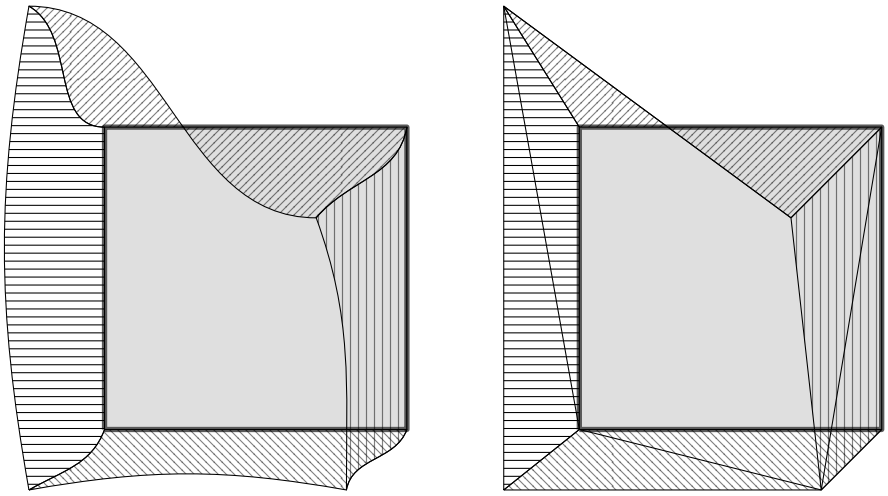
The flux volumes $\Omega_{p,i}$ are streak-tubes that generally do not have flat faces (even in two dimensions) and are non-convex with both positive and negative regions. In order to deal with these objects, we propose approximating the flux volumes with a collection of simplices (triangles in two dimensions and tetrahedra in three dimensions), denoted $\Delta_{p,i,j}$ for $j = 1, \ldots, N_{\text{sims}}$ where N_{sims} is the number of simplices such that

$$\Omega_{p,i} \approx \widetilde{\Omega}_{p,i} = \bigcup_{j=1}^{N_{\text{sims}}} \Delta_{p,i,j}. \tag{3.19}$$

In the previous equation, $\widetilde{\Omega}_{p,i}$ is the discrete representation of $\Omega_{p,i}$, which is simpler to manipulate computationally. Similarly, the discrete approximations of $\alpha_{p,i}$ and $\mathcal{V}_{p,i}$ are denoted with $\widetilde{\alpha}_{p,i}$ and $\widetilde{\mathcal{V}}_{p,i}$, respectively.

The number of simplices used in the tessellation depends on a balance between computational cost and accuracy, both of which increase with increasing number of simplices. In a second-order implementation, edges of the flux volume can be approximated with straight lines. In two dimensions, the resulting shape can therefore be represented using two simplices. This is shown in Fig. 3.6, wherein each of the flux volumes associated with a two-dimensional computational cell are represented using two simplices. A three-dimensional flux hexahedral volume with straight edges can be represented with five simplices. However, faces on

opposite sides of the flux volumes will be cut along different diagonals, which can result in overlaps between the tessellated faces of neighboring flux volumes. Using six simplices avoids this source of error since the diagonals on opposite sides of the flux volume go in the same direction. Figure 3.7 shows an example of the discretization of a three-dimensional flux volume using six simplices. Notice in the figure how the front face with vertices $\boldsymbol{a}^n\boldsymbol{a}^{n+1}\boldsymbol{b}^{n+1}\boldsymbol{b}^n$ is cut along the diagonal $\boldsymbol{a}^{n+1}\boldsymbol{b}^n$, which is in the same direction as the diagonal $\boldsymbol{d}^{n+1}\boldsymbol{c}^n$ that cuts the opposite face. In a five-simplex representation, the front face would still be cut by the $\boldsymbol{a}^{n+1}\boldsymbol{b}^n$ diagonal, but the opposite face would be cut along the $\boldsymbol{c}^{n+1}\boldsymbol{d}^n$ diagonal. Again, using more simplices improves the discrete representation of the flux volume but the computational cost also increases.

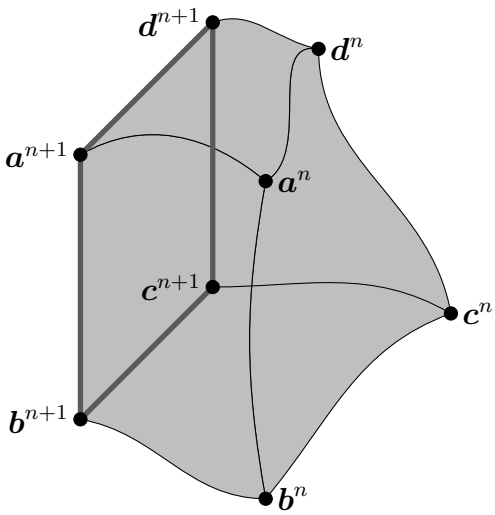


(a) Example of real geometry.

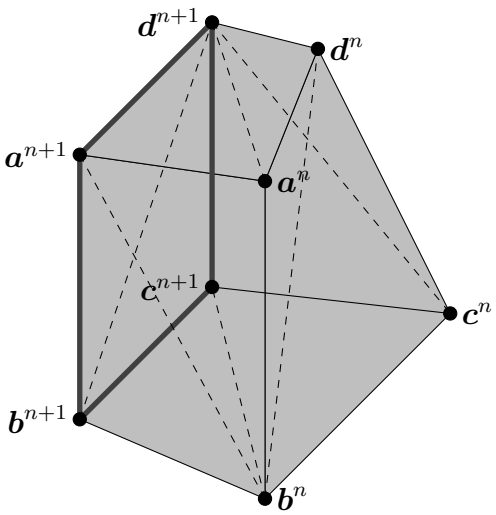
(b) Discrete representation of geometry.

Figure 3.6: Example geometry of the two-dimensional flux volume associated with a computational cell (shaded). (a) shows an example of a realistic geometry where the four flux volumes (indicated with lines) do not have straight edges. (b) shows how the real geometry can be approximated using two simplices per flux volume.

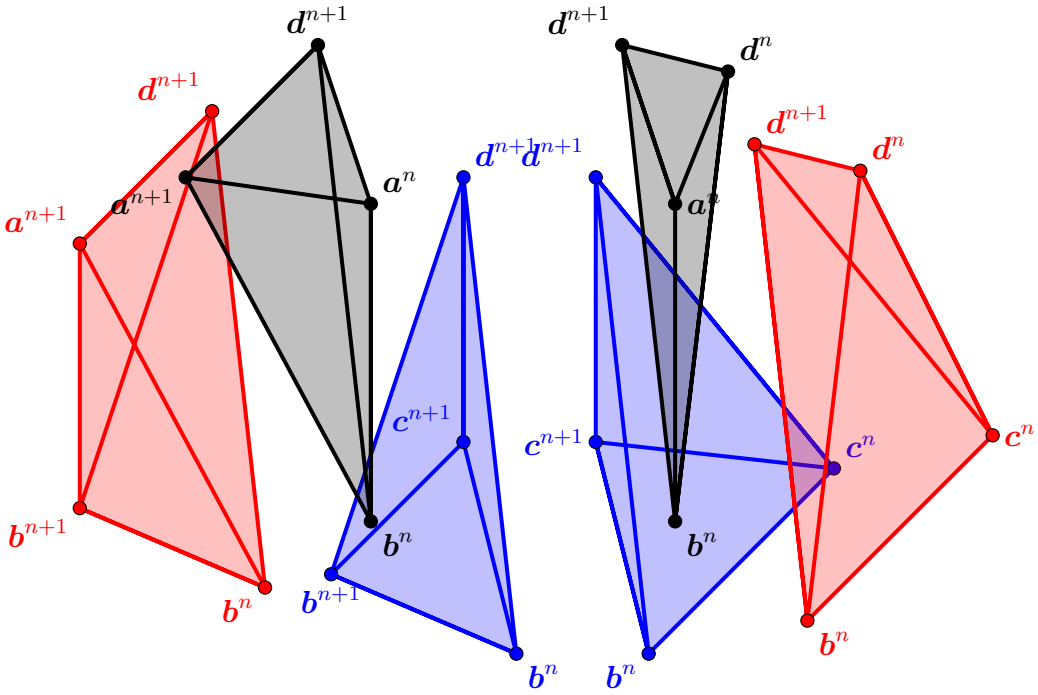
The simplices are created using a systematic approach that makes the implementation straightforward. Vertices that exist on the corners of the computational cell face are identified. Since the flux volume is a streak-tube, each vertex is trans-



(a) Example of a flux volume in three dimensions.



(b) Discrete representation of a flux volume using six simplices.



(c) Exploded view of discrete representation of a flux volume using six simplices.

Figure 3.7: Example of a three-dimensional flux volume associated with the computational cell face with vertices $(\boldsymbol{abcd})^{n+1}$. (a) shows the real shape of the flux volume. (b) and (c) show how the flux volume is discretized using simplices.

ported back in time along its streak-line. For example, in Fig. 3.7 the vertices with superscript $n+1$ are on the corners of $\partial CS_{p,i}$. These vertices are transported back in time to their locations with superscript n. The simplices can then be constructed from the location of the original and transported vertices.

Vertex transport along a streak-line can be described using

$$\begin{aligned} \frac{d\boldsymbol{x}_v(t)}{dt} &= \boldsymbol{u}(\boldsymbol{x}_v(t), t) \text{ and} \\ \boldsymbol{x}_v(t_0) &= \boldsymbol{x}_{v,0}, \end{aligned} \tag{3.20}$$

where $\boldsymbol{x}_v(t)$ is the position of the vertex at time t and $\boldsymbol{x}_{v,0}$ is the starting location of the vertex on the computational cell face at time t_0. This equation is integrated backwards in time to find $\boldsymbol{x}_v(t^n)$. To simplify the process, we assume the velocity field is time-invariant. This is a reasonable (second-order) approximation since the velocity and interface transport steps are staggered in time in our implementation. Many time integration strategies can be used to solve Eq. 3.20, however higher-order methods will result in a vertex more closely following its streak-line. We have tested a range of Runge-Kutta schemes from first to sixth order and found little difference in the computed solutions. A second-order Runge-Kutta method is consistent with the rest of the scheme and was chosen due to the low cost compared with higher-order methods.

Simplex construction and sign convention

In two dimensions, it is possible to list all of the different shapes the flux volume can take and to systematically break the shapes into two simplices with associated signs that indicate whether the flux is into or out of CV_p. However, this approach is tedious due to the large number of different flux volume shapes that exist. In three dimensions, a similar approach is even more difficult to implement. Therefore, we

propose a straightforward and systematic way of constructing the simplices and determining their sign.

We start by associating a sign to each of the simplices in the partition of $\widetilde{\Omega}_{p,i}$ based on the location and ordering of the simplices' vertices. The sign can easily be determined using

$$\text{sign}(\Delta) \equiv \text{sign}\Big((\boldsymbol{b}-\boldsymbol{a}) \times (\boldsymbol{c}-\boldsymbol{a})\Big) \tag{3.21}$$

for the two-dimensional simplex with vertices $\boldsymbol{abc}$ and

$$\text{sign}(\Delta) \equiv \text{sign}\left[\Big((\boldsymbol{b}-\boldsymbol{a}) \times (\boldsymbol{c}-\boldsymbol{a})\Big) \cdot \left(\boldsymbol{d} - \frac{1}{3}(\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c})\right)\right] \tag{3.22}$$

in three dimensions for the simplex with vertices $\boldsymbol{abcd}$. Clearly, the ordering of the vertices determines the sign and therefore it has to be chosen such that, when a simplex contributes a positive flux, it has a positive sign, and when the simplex contributes a negative flux, it has negative sign. The sign of the simplex determines whether it is part of $\widetilde{\Omega}^{+}_{p,i}$ or $\widetilde{\Omega}^{-}_{p,i}$ (the discretized counterparts to $\Omega^{+}_{p,i}$ and $\Omega^{-}_{p,i}$), i.e.,

$$\widetilde{\Omega}^{+}_{p,i} = \bigcup_{\substack{j=1 \\ \text{sign}(\Delta_{p,i,j})=+1}}^{N_{\text{sims}}} \Delta_{p,i,j} \qquad \text{and} \qquad \widetilde{\Omega}^{-}_{p,i} = \bigcup_{\substack{j=1 \\ \text{sign}(\Delta_{p,i,j})=-1}}^{N_{\text{sims}}} \Delta_{p,i,j}. \tag{3.23}$$

Simplices created from the same ordering of vertices are used to represent all of the different flux volume shapes, which greatly simplifies the implementation. For example, Fig. 3.8 shows three two-dimensional flux volumes that represent the three main categories of shapes that are possible in two dimensions. In the figure, the cell face is the line $\boldsymbol{ab}$, $\boldsymbol{d}$ is the location of $\boldsymbol{a}$ at t^n found by solving Eq. 3.20, and $\boldsymbol{c}$ is the location of $\boldsymbol{b}$ at t^n. In all of the cases the flux volumes are represented using the simplices $\Delta\boldsymbol{abc}$ and $\Delta\boldsymbol{acd}$. The sign of each simplex can be calculated using Eq. 3.21, and with these definitions it is found to be consistent with the sign of the flux that the simplex is representing.

Note that simplices may extend outside of $\widetilde{\Omega}_{p,i}$, i.e.,

$$\left\{\widetilde{\Omega}_{p,i} \cup \Delta_{i,j}\right\} \setminus \left\{\widetilde{\Omega}_{p,i} \cap \Delta_{i,j}\right\} \neq \emptyset, \tag{3.24}$$

as shown by case C in Fig. 3.8. However, whenever a simplex extends outside of $\widetilde{\Omega}_{p,i}$, another simplex of opposite sign also extends outside and the net contribution is zero.

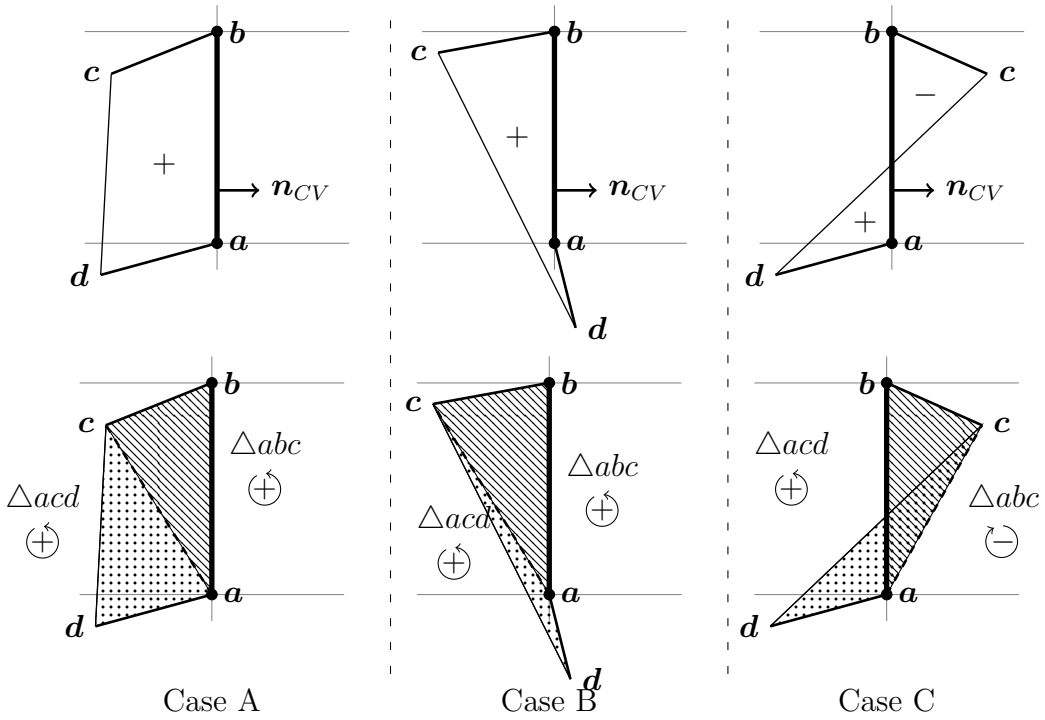


Figure 3.8: Example of the discrete representation of two-dimensional flux volumes using signed simplices.

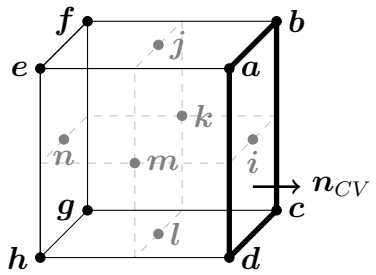
This systematic approach to constructing the simplices is readily extendable to three dimensions. Similarly to the two-dimensional implementation, the vertices of a computational cell face are transported back in time using Eq. 3.20, the simplices are created using a predefined ordering of the vertices, and the sign of the simplex determines the direction of the flux. Figure 3.9 provides details on the predefined ordering of the vertices used to make the simplices. Details of discretizations that use 6 and 20 simplices per flux volume are included.

The ordering of the vertices in three dimensions is important to ensure that the faces that are shared between neighboring flux volumes are discretized in a consistent way and that no gaps or overlaps are formed in the geometry. Figure 3.10 illustrates two ways in which the face of a flux volume can be shared with a neighboring flux volume. In the figure, the shaded faces are shared between neighboring flux volumes and should be discretized using the same representation by all flux volumes that share the face. The ordering of the vertices provided in Fig. 3.9 has been constructed such that flux volume faces are consistently discretized by neighboring flux volumes. This is trivial for the 20 simplex discretization because all faces are discretized the same way using the four corners and the face barycenter. Contrarily, the 6 simplex discretization approximates each face with two triangles that depend on which diagonal is used. Therefore, an equivalent to the 20 simplex discretization that uses face barycenters could be used to discretize flux volumes that are produced in the context of an unstructured code.

Using this framework, the calculation of the volume and liquid volume within the flux volume becomes systematic and straightforward. First, the cell face vertices are transported back in time to the beginning of the time-step, t^n. Then, the flux volume is approximated with a collection of predefined simplices, provided in Fig. 3.9 for a Cartesian mesh. These simplices are defined to be non-overlapping and their sign is consistent with the definitions from Section 3.2. Finally, the volume and the amount of liquid within a simplex remain to be calculated.

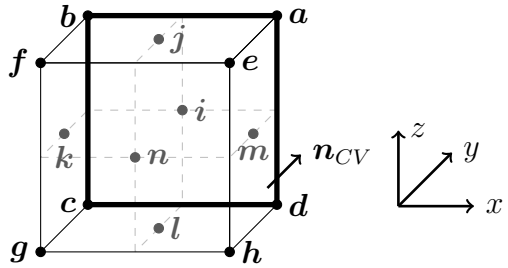
Flux calculation

In order to solve Eq. 3.18, the quantities $\alpha_{p,i}$ and $\mathcal{U}_{p,i}$ defined using Eqs. 3.17 and 3.14 need to be calculated. The problem is simplified thanks to the discrete



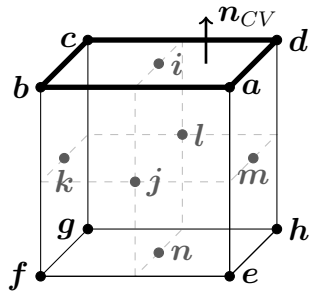
x face

6 Simplices				
1	g	d	c	f
2	f	c	b	d
3	f	b	a	d
4	g	h	d	f
5	f	e	h	d
6	f	e	d	a



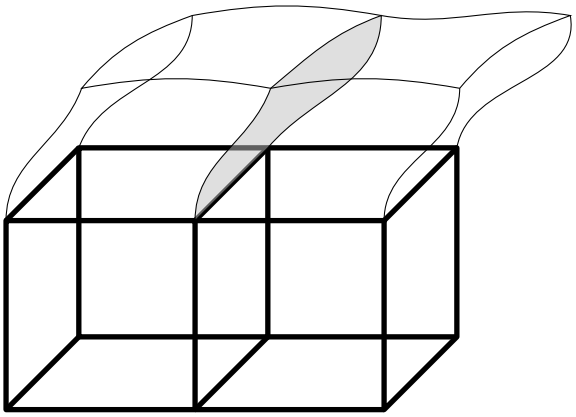
y face

20 Simplices				
1	a	e	j	i
2	e	f	j	i
3	f	b	j	i
4	b	a	j	i
5	b	f	k	i
6	f	g	k	i
7	g	c	k	i
8	c	b	k	i
9	c	g	l	i
10	g	h	l	i
11	h	d	l	i
12	d	c	l	i
13	d	h	m	i
14	h	e	m	i
15	e	a	m	i
16	a	d	m	i
17	e	h	n	i
18	h	g	n	i
19	g	f	n	i
20	f	e	n	i

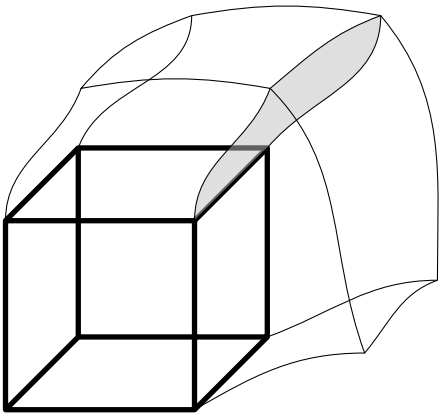


z face

Figure 3.9: Ordering of vertices used in the construction of simplices. The figures on the left show the ordering of the vertices and the tables on the right provide a list of the four vertices used to construct each simplex. The vertices $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, and $\boldsymbol{d}$ are located on the corners of the cell face. The vertices $\boldsymbol{e}$, $\boldsymbol{f}$, $\boldsymbol{g}$, and $\boldsymbol{h}$ are the locations of the vertices $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, and $\boldsymbol{d}$ at time t^n, respectively, found by solving Eq. 3.20. Vertex $\boldsymbol{i}$ is located at the barycenter of the cell face $\boldsymbol{abcd}$. Vertex $\boldsymbol{n}$ is the location of vertex $\boldsymbol{i}$ at time t^n, found by solving Eq. 3.20. Vertices $\boldsymbol{j}$, $\boldsymbol{k}$, $\boldsymbol{l}$, and $\boldsymbol{m}$ are found by solving Eq. 3.20 for $\frac{1}{2}\Delta t$ from points $\frac{1}{2}(\boldsymbol{a}+\boldsymbol{b})$, $\frac{1}{2}(\boldsymbol{b}+\boldsymbol{c})$, $\frac{1}{2}(\boldsymbol{c}+\boldsymbol{d})$, and $\frac{1}{2}(\boldsymbol{d}+\boldsymbol{a})$, respectively.



(a) Face shared between flux volumes associated with neighboring computational cells.



(b) Face shared between flux volumes associated with the same computational cell.

Figure 3.10: Illustration of flux volume faces that are shared by neighboring flux volumes. The shaded faces are shared between the two flux volumes shown in each figure. The discrete representation of the non-flat face needs to be consistent between all flux volumes that share the face.

representation of the flux volume as a collection of signed simplices, thus we only need to calculate the liquid volume fraction and volume of a simplex, denoted $\widetilde{\alpha}_{\Delta_{p,i,j}}$ and $\widetilde{\mathcal{V}}_{\Delta_{p,i,j}}$, respectively. The discrete quantities $\widetilde{\alpha}_{p,i}$ and $\widetilde{\mathcal{U}}_{p,i}$ can then be

generated using

$$\widetilde{\alpha}_{p,i} = \frac{\sum_{j=1}^{N_{\text{sims}}} \widetilde{\alpha}_{\Delta_{p,i,j}} \widetilde{\mathcal{V}}_{\Delta_{p,i,j}}}{\sum_{j=1}^{N_{\text{sims}}} \widetilde{\mathcal{V}}_{\Delta_{p,i,j}}} \text{ and} \tag{3.25}$$

$$\widetilde{\mathcal{V}}_{p,i} = \sum_{j=1}^{N_{\text{sims}}} \widetilde{\mathcal{V}}_{\Delta_{p,i,j}}, \tag{3.26}$$

along with the discrete equivalent of Eq. 3.14.

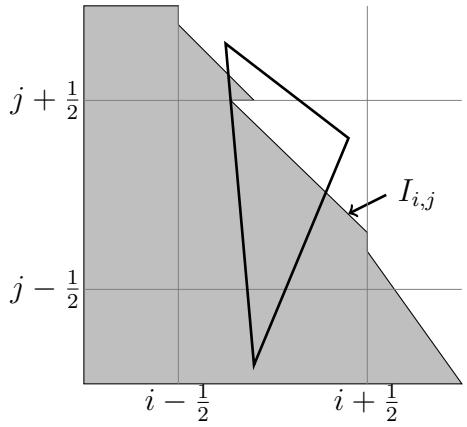
The signed volume of a simplex can be calculated easily by combining the sign convention, Eq. 3.22, and the Cayley-Menger determinant [72], which in three dimensions can be written as

$$\widetilde{\mathcal{V}}_{\Delta_{p,i,j}} = -\frac{(\boldsymbol{a}-\boldsymbol{d}) \cdot ((\boldsymbol{b}-\boldsymbol{d}) \times (\boldsymbol{c}-\boldsymbol{d}))}{6} \tag{3.27}$$

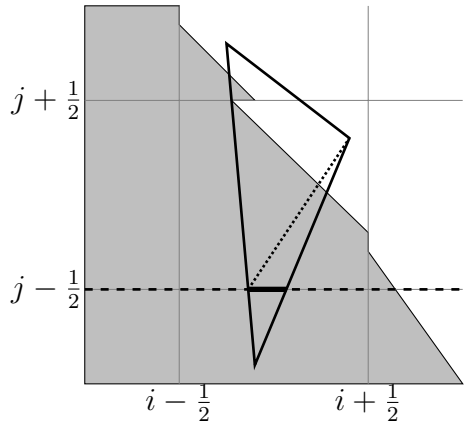
where $\Delta_{p,i,j}$ has vertices $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, and $\boldsymbol{d}$.

The liquid volume fraction in a simplex is more complicated to compute and depends on the location of the gas-liquid interface. In our method, the gas-liquid interface is represented using the PLIC scheme described in Section 3.3.1, which uses a piecewise planar representation of the interface that is local to each computational cell. To calculate $\widetilde{\alpha}_{p,i}$, a given simplex is cut by cell faces and divided into regions that are local to each computational cell, and then cut by the gas-liquid interface, resulting in regions that are exclusively within the liquid phase or exclusively within the gas phase. Then, the liquid volume is easily calculated as the sum of volumes within the liquid phase. To make the algorithm tractable, the shapes that are created from cutting a simplex by a plane are partitioned into a new collection of simplices that can easily be cut again. The process continues until each simplex lies within only one phase.

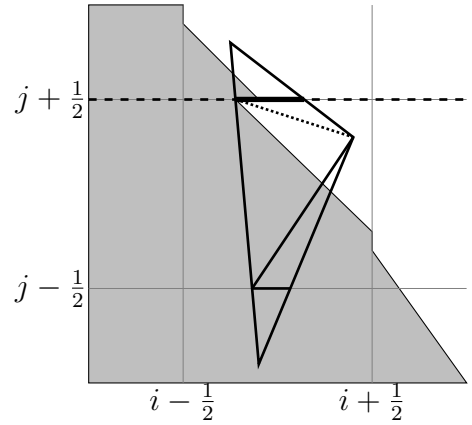
For example, Fig. 3.11 shows the process used to cut a simplex by two planes that are coincident with the computational grid, followed by a cut by the PLIC



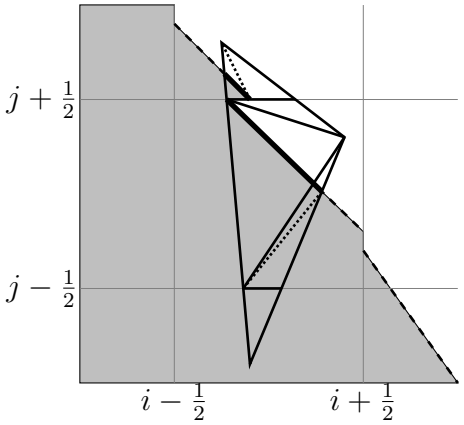
(a) Original simplex and PLIC reconstruction of interface indicated with $I_{i,j}$ in cell i, j.



(b) Simplex is cut by plane at $j - \frac{1}{2}$ (thick solid line). Resulting shapes are partitioned into simplices (dotted line).



(c) Simplices in (b) are cut by plane at $j + \frac{1}{2}$ (thick solid line). Resulting shapes are partitioned into simplices (dotted line) that are unique to one computational cell.



(d) Simplices from (c) are cut by liquid-gas interface reconstruction I (thick solid line). Resulting shapes are partitioned into simplices (dotted lines) that are within one computational cell and one phase.

Figure 3.11: Steps used to calculate the liquid volume fraction within a simplex that crosses multiple planes.

representation of the interface. The simplex shown in Fig. 3.11(a) is first cut by the plane indicated by the $j - \frac{1}{2}$ face of the cell considered, which results in two shapes. The bottom shape is a triangle and the top shape is a quadrilateral. The triangle is already a simplex, but the quadrilateral needs to be partitioned into two simplices as shown by Fig. 3.11(b). Next, the three simplices are cut by the plane at the $j + \frac{1}{2}$ index, which again leads to the partition of a simplex into a triangle and a quadrilateral. The quadrilateral is divided into two simplices as shown by Fig. 3.11(c). Finally, the simplices are cut by the reconstructed gas-liquid interface and partitioned into more simplices as shown by Fig. 3.11(d). Now it is straightforward to compute the liquid volume within the original simplex from the volumes of the new simplices that contain liquid. For example, the liquid volume within the simplex in Fig. 3.11(a) is equal to the sum of the volumes of the shaded simplices in Fig. 3.11(d).

The number of planes by which the simplex is cut depends on the location of the simplex vertices. In our implementation, we identify which planes the simplex needs to be cut by using a initialization routine that is based on the location of the vertices.

3.3.3 Construction of conservative fluxes

As described in Section 3.2.2, the flux velocity must be solenoidal to ensure discrete conservation. In our implementation a staggered velocity field is used, meaning that the face-normal component of the velocity is available at the center of each computational cell face. There is no guarantee that interpolating this velocity to cell face vertices and projecting the vertices back in time to create a flux volume will produce a flux velocity $\mathcal{U}_{p,i}$ that is also solenoidal. Therefore, the flux volume

is modified to ensure discrete conservation. The modification is typically small and does not alter the second-order accuracy of the scheme as shown in the verification tests.

Several approaches have been used previously to modify the fluxes to improve conservation. Liovic et al. [73] scaled the multi-dimensional fluxes with conservative one-dimensional fluxes. Mencinger and Žun [74] used a parametric correction approach to modify the time-step used to project each vertex in the construction of the flux volume, but the extension to three-dimensions is unclear. López et al. [2] and Hernández et al. [3] used analytical relations to modify the size of the flux volume so that a conservative flux is constructed. In their approaches, the flux volume is constrained by the volume of a one-dimensional conservative flux built using a solenoidal face velocity, e.g., $\widehat{\mathcal{U}}_{p,i} = (\boldsymbol{u} \cdot \boldsymbol{n}_{CV})_{p,i}$, where $\widehat{\mathcal{U}}_{p,i}$ is the modified flux velocity and $(\boldsymbol{u} \cdot \boldsymbol{n}_{CV})_{p,i}$ is the normal component of the solenoidal velocity on the i^{th} face of the p^{th} computational cell. The resulting modification is equivalent to adjusting the time-step used in Eq. 3.20 to project the vertices along streak-lines. A similar approach is used in the proposed scheme. However, modifying the time-step can create overlapping regions between neighboring flux volumes when the faces are non-planar as shown in Fig. 3.12. Therefore, the modification is performed in a way that avoids creating overlapping regions that would result in a conservation error.

The proposed method consists of adding additional simplices to the flux volume such that $\widehat{\mathcal{U}}_{p,i} = (\boldsymbol{u} \cdot \boldsymbol{n}_{CV})_{p,i}$. The additional simplices must not modify the discretization of faces of the flux volume that are shared with neighboring flux volumes in order to ensure that no overlapping regions are created. Therefore, the simplices are added onto the projected face, e.g., the face of the flux volume

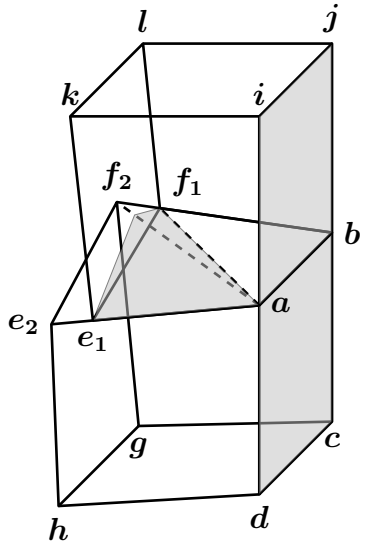


Figure 3.12: [
Overlapping region between rescaled fluxes]Overlapping region formed between neighboring flux volumes $\boldsymbol{abcde_2f_2gh}$ and $\boldsymbol{baijf_1e_1kl}$ when simple rescaling (i.e., time-step adjustment) is used to create conservative fluxes. The face with vertices $\boldsymbol{abf_1e_1}$ is triangulated with diagonal $\boldsymbol{af_1}$. Rescaling the bottom flux volume moves the projected vertices creating the shaded overlapping region.

opposite from the cell face $CS_{p,i}$. The number of additional simplices is equal to the number of simplices used to discretize the projected face. The volume of the additional simplices $\mathcal{V}_{\text{cor}}$ can be calculated from the difference between the pre-modified flux velocity, the solenoidal flux velocity, and Eq. 3.14, leading to

$$\mathcal{V}_{\text{cor}} = \left((\boldsymbol{u}\cdot\boldsymbol{n}_{CV})_{p,i} - \mathcal{U}_{p,i}\right)\Delta t\mathcal{A}_{p,i}. \tag{3.28}$$

Figure 3.13 shows a two-dimensional example for the flux through an x-face $\boldsymbol{ab}$ with flux volume $\boldsymbol{abcd}$. The flux volume is modified by adding the simplex $\boldsymbol{dco}$. A closed form analytic expression for the coordinates of vertex $\boldsymbol{o}$ can be formed by enforcing two constraints. We choose to constrain the y coordinate of vertex $\boldsymbol{o}$ by the y coordinate of $\boldsymbol{n}$, which is the barycenter of the face $\boldsymbol{cd}$. The x coordinate is constrained by the volume $\mathcal{V}_{\text{cor}}$. Similarly, the additional simplex on a y-face flux volume is constrained by the x component of the barycenter and the correction

volume.

This approach to modify the flux volume is easily extended to three dimensions, as shown for example in Figure 3.14. The flux volume $\boldsymbol{abcdefgh}$, associated with the x-face $\boldsymbol{abcd}$ is modified by adding the two simplices $\boldsymbol{efho}$ and $\boldsymbol{fgho}$. The location of vertex $\boldsymbol{o}$ is determined by enforcing three constraints. Two constraints come from enforcing that the y and z coordinates of vertex $\boldsymbol{o}$ are equal to the y and z coordinates of $\boldsymbol{n}$, the barycenter of face $\boldsymbol{efgh}$. The x coordinate is constrained by the correction volume, i.e.,

$$\begin{aligned} \mathcal{V}_{\text{cor}} &= \mathcal{V}_{\Delta(efho)} + \mathcal{V}_{\Delta(fgho)} \\ &= -\frac{(\boldsymbol{e}-\boldsymbol{o})\cdot((\boldsymbol{f}-\boldsymbol{o})\times(\boldsymbol{h}-\boldsymbol{o}))}{6} - \frac{(\boldsymbol{f}-\boldsymbol{o})\cdot((\boldsymbol{g}-\boldsymbol{o})\times(\boldsymbol{h}-\boldsymbol{o}))}{6}. \end{aligned} \tag{3.29}$$

Similar constraints are enforced for flux volumes associated with y and z computational cell faces. The three constraints provide an analytic relation that relates the location of vertices $\boldsymbol{e}$, $\boldsymbol{f}$, $\boldsymbol{g}$, and $\boldsymbol{h}$ and the volume $\mathcal{V}_{\text{cor}}$ to the location of vertex $\boldsymbol{o}$, which can be evaluated to quickly compute the modification to the flux volume. The analytic relations for computing $\boldsymbol{o}$ are provided in Algorithm 1 for all the faces of a computational cell.

If the flux volume $\boldsymbol{abcdefgh}$ in the three-dimensional example is discretized with 20 simplices, four additional simplices are added because the projected face is discretized with four simplices. The additional simplices are $\boldsymbol{heno}$, $\boldsymbol{efno}$, $\boldsymbol{fgno}$, and $\boldsymbol{ghno}$. The y and z coordinates of $\boldsymbol{o}$ are constrained by the barycenter $\boldsymbol{n}$ and the x coordinate is constrained by the volume $\mathcal{V}_{\text{cor}}$. An analytical relation can be found to solve for the x coordinate that is similar to the relation in Eq. 3.29.

Algorithm 1 SolenoidalFlux: routine to compute two additional simplices to construct conservative flux. Algorithm assumes the original flux is discretized with six simplices as shown in Fig. 3.9.

function SolenoidalFlux($D, \boldsymbol{V}, \mathcal{V}_{\text{cor}}$)
  input D ▷ Direction of face
  input $\boldsymbol{V}$ ▷ Vertices on flux volume as shown in Fig. 3.14
  input $\mathcal{V}_{\text{cor}}$ ▷ Volume of additional simplices
  $\boldsymbol{e} \leftarrow \boldsymbol{V}_5$
  $\boldsymbol{f} \leftarrow \boldsymbol{V}_6$
  $\boldsymbol{g} \leftarrow \boldsymbol{V}_7$
  $\boldsymbol{h} \leftarrow \boldsymbol{V}_8$
  $\boldsymbol{n} \leftarrow \frac{1}{4}(\boldsymbol{e} + \boldsymbol{f} + \boldsymbol{g} + \boldsymbol{h})$
  switch D **do**
    case 1
      $o_1 \leftarrow (6\mathcal{V}_{\text{cor}} + e_1f_2h_3 - e_1f_3h_2 - e_2f_1h_3 + e_2f_3h_1 + e_3f_1h_2 - e_3f_2h_1 - e_1f_2e_3$
        $+ e_1f_3e_2 + e_2f_1e_3 + e_3h_1e_2 - e_3f_1e_2 + f_1g_2h_3 - f_1g_3h_2 - f_2g_1h_3$
        $+ f_2g_3h_1 + f_3g_1h_2 - f_3g_2h_1 + e_1h_2e_3 - e_1h_3e_2 - e_2h_1e_3 - f_1g_2e_3$
        $+ f_1g_3e_2 + f_2g_1e_3 - f_3g_1e_2 - g_1h_2e_3 + g_1h_3e_2 + g_2h_1e_3 - g_3h_1e_2)$
        $/(e_2f_3 - e_3f_2 - e_2h_3 + e_3h_2 + f_2g_3 - f_3g_2 + g_2h_3 - g_3h_2)$
      $o_2 \leftarrow n_2$
      $o_3 \leftarrow n_3$
    case 2
      $o_1 \leftarrow n_1$
      $o_2 \leftarrow -(6\mathcal{V}_{\text{cor}} + e_1f_2h_3 - e_1f_3h_2 - e_2f_1h_3 + e_2f_3h_1 + e_3f_1h_2 - e_3f_2h_1 - e_1f_2e_3$
        $+ e_2f_1e_3 - e_2f_3e_1 + e_3f_2e_1 + f_1g_2h_3 - f_1g_3h_2 - f_2g_1h_3 + f_2g_3h_1$
        $+ f_3g_1h_2 - f_3g_2h_1 + e_1h_2e_3 - e_2h_1e_3 + e_2h_3e_1 - e_3h_2e_1 - f_1g_2e_3$
        $+ f_2g_1e_3 - f_2g_3e_1 + f_3g_2e_1 - g_1h_2e_3 + g_2h_1e_3 - g_2h_3e_1 + g_3h_2e_1)$
        $/(e_1f_3 - e_3f_1 - e_1h_3 + e_3h_1 + f_1g_3 - f_3g_1 + g_1h_3 - g_3h_1)$
      $o_3 \leftarrow n_3$
    case 3
      $o_1 \leftarrow n_1$
      $o_2 \leftarrow n_2$
      $o_3 \leftarrow (6\mathcal{V}_{\text{cor}} + e_1f_2h_3 - e_1f_3h_2 - e_2f_1h_3 + e_2f_3h_1 + e_3f_1h_2 - e_3f_2h_1 + e_1f_3e_2$
        $- e_2f_3e_1 - e_3f_1e_2 + e_3f_2e_1 + f_1g_2h_3 - f_1g_3h_2 - f_2g_1h_3 + f_2g_3h_1$
        $+ f_3g_1h_2 - f_3g_2h_1 - e_1h_3e_2 + e_2h_3e_1 + e_3h_1e_2 - e_3h_2e_1 + f_1g_3e_2$
        $- f_2g_3e_1 - f_3g_1e_2 + f_3g_2e_1 + g_1h_3e_2 - g_2h_3e_1 - g_3h_1e_2 + g_3h_2e_1)$
        $/(e_1f_2 - e_2f_1 - e_1h_2 + e_2h_1 + f_1g_2 - f_2g_1 + g_1h_2 - g_2h_1)$
  return $N_{\text{Add}} \leftarrow 2$ ▷ Number of additional simplices
  return $\boldsymbol{S_1} = [\boldsymbol{e}, \boldsymbol{f}, \boldsymbol{h}, \boldsymbol{o}]^{\mathsf{T}}$ ▷ First additional simplex
  return $\boldsymbol{S_2} = [\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{h}, \boldsymbol{o}]^{\mathsf{T}}$ ▷ Second additional simplex
end function

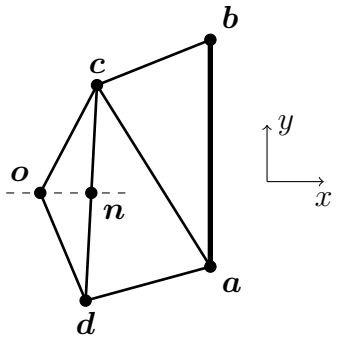


Figure 3.13: Modification of two-dimensional flux volume to create solenoidal flux. Original flux volume associated with face $\boldsymbol{ab}$ has vertices $\boldsymbol{abcd}$. The additional simplex with vertices $\boldsymbol{cdo}$ is added.

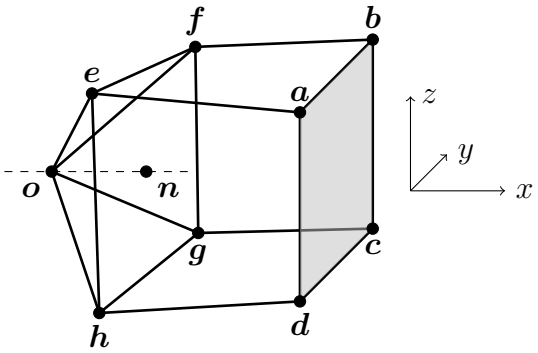


Figure 3.14: Modification of three-dimensional flux volume to create solenoidal flux. Original flux volume associated with face $\boldsymbol{abcd}$ has vertices $\boldsymbol{abcdefgh}$. Two additional simplices with vertices $\boldsymbol{efho}$ and $\boldsymbol{fgho}$ are added if six simplices are used to discretize the original volume.

3.3.4 Parallelization

The proposed scheme has been implemented using a domain decomposition parallelization strategy within the NGA computational platform [56]. The geometric transport routines require the geometry of neighboring cells and the PLIC reconstruction in those cells. Using standard operations within NGA to update ghost cells on domain boundaries, the interface normal vector and the liquid volume fraction are communicated. With the communicated information, the PLIC reconstruction is computed on each processor and the geometric algorithm is used

to advect the liquid volume fraction. The resulting scheme has minimal communication requirements and is expected to have excellent scale-up properties.

3.3.5 Extension to unstructured meshes

The proposed algorithm can be implemented within the context of an unstructured mesh with only minor modifications. The discretization of the flux volumes should be consistent with the cell face geometry, and faces of the flux volumes that are shared with neighboring flux volumes should be discretized such that there are no overlaps or gaps. The sign of the simplices should be constructed so that positive fluxes are represented by positively orientated simplices and negative fluxes are represented by negative simplices. The modification of the flux volume to make conservative fluxes can be performed using the proposed method. Once the flux volume is formed as a collection of simplices, the simplices are cut into regions that are unique to one computational cell using, for example, a polyhedron clipping and capping algorithm [64, 66]. Finally, the simplices are cut by the PLIC reconstruction within each cell and the fluxes are computed. In summary, the main ideas of the proposed method can all be extended to unstructured grids, namely partitioning the flux volume into a collection of simplices, assigning a sign to each simplex to determine the flux contribution, and correcting the flux volume with additional simplices. Furthermore, Algorithm 5 and the look-up tables that provide an efficient method to cut a simplex by a plane are not mesh dependent and could be used in an unstructured code. Additional details will depend on the detailed topology of the mesh and are beyond the scope of this paper.

3.3.6 Implementation

In this section, the process used to calculate the liquid and volume fluxes is detailed using pseudo-code. Algorithm 2 shows the general framework for updating the liquid volume fraction. First, the interface normal and PLIC reconstruction are computed using the methodology described in Section 3.3.1. Next, the fluxes are calculated, then the liquid volume fraction is updated.

Algorithm 3 provides the methodology to compute the fluxes. In the algorithm, the flux volume is created on every face on the computational mesh. Then, the flux volume is divided into simplices using PartitionFlux, which, due to the order of the vertices used to create the simplices, have the same sign as the flux volume they represent. Next, the flux volume is modified by adding additional simplices constructed using SolenoidalFlux (Algorithm 1). Finally, the signed volume and signed liquid volume within each simplex are calculated and added to running sums. The sign follows from the orientation of the simplex that is evaluated using SimplexSign, which should be based on Eq. 3.22. Note that the computational cost can be reduced by computing each flux once and using the value to update both computational cells that share the face. Care must be taken to ensure the sign of the flux is correct when updating each cell.

The liquid volume within a simplex is calculated with SimplexLiquidVolume (Algorithm 4). The algorithm takes a simplex, cuts the simplex by a plane and divides the resulting shapes into new simplices using CutSimplex. The new simplices are cut by another plane and divided into more new simplices. The process continues until each of the simplices is contained within a single computational cell and on one side of the reconstructed gas-liquid interface.

The operation to cut a simplex by a plane is performed by CutSimplex (Algorithm 5). This algorithm computes the distance between each vertex of the simplex and the plane that the simplex is being cut with. Based on the sign of the four distances, the number of intersections between the plane and the simplex edges is calculated and the intersection points are saved. Finally, the simplex is partitioned into a collection of new simplices using the original vertices and the intersection points. Note that the orientation of the simplices used in the partition of the original simplex is not important, only the orientation of the original simplex is used to determine the sign of the flux contribution, i.e., SimplexSign only appears in Algorithm 3 and depends on the original simplex.

To improve the efficiency of the CutSimplex algorithm we have introduced look-up tables. A case number, $Case \in \{1, \ldots, 16\}$, is created that classifies the simplex based on the sign of the distances between the simplex vertices and the cut-plane. The case, in conjunction with the look-up tables, provides

- NumberIntersect($Case$): the number of intersections between the edges of the simplex and the cut-plane,
- IndexEndPoint($v, n, Case$): the index of the v^{th} end point on the end of the edge involved in the n^{th} intersection between the simplex and the cut-plane,
- NumberSimsPart($Case$): the number of simplices in the partition of the original simplex,
- IndexSimVert($v, n, Case$): the index of the v^{th} vertex on the n^{th} simplex used to partition the original simplex.

For example, $Case = 2$ corresponds to a simplex with the 1^{st} vertex on the pos-

itive side of the plane and the 2nd, 3rd, and 4th vertices on the negative side, as shown in Fig. 3.15. This simplex has three edges that intersect the plane and therefore NumberIntersect(2) = 3. The first intersection is between the plane and the edge with vertices 1 and 2, thus IndexEndPoint$(:, 1, 2) = [1, 2]$. The second intersection is with the edge with vertices 1 and 3, and the third intersection is with the edge with vertices 1 and 4, thus IndexEndPoint$(:, 2, 2) = [1, 3]$ and IndexEndPoint$(:, 3, 2) = [1, 4]$. The number of simplices used to partition the cut simplex is four, which is provided by NumberSimsPart(2) = 4. Finally, the indices of the vertices used to construct the new simplices are provided by IndexSimVert$(v, n, Case)$, where v is the vertex number and n is the simplex number. For our example, [IndexSimVert$(:, :, 2) = \big[[1, 5, 6, 7], [4, 2, 3, 6], [4, 2, 5, 6], [4, 5, 6, 7], [0, 0, 0, 0], [0, 0, 0, 0]\big]$, which provides the information needed to create the four simplices in our partition. The first simplex has vertices that correspond to the points with the index 1, 5, 6, and 7. The second simplex has vertices 4, 2, 3, and 6. The third and fourth simplices have vertices 4, 2, 5, 6 and 4, 5, 6 , 7, respectively. Note that the zeros in IndexEndPoint and IndexSimVert indicate null values and are used to pad the arrays.

Algorithm 2 UpdateVOF: framework to update the liquid volume fraction

function UpdateVOF($\boldsymbol{\alpha}_p^n, \Delta t$)
  input $\boldsymbol{\alpha}_p^n$ ▷ Liquid volume fraction at t^n
  input Δt ▷ Time-step
  $[\boldsymbol{n}] \leftarrow$ InterfaceNormal ▷ Compute interface normal vectors
  $[\boldsymbol{I}] \leftarrow$ InterfaceReconstruction$(\boldsymbol{n}, \boldsymbol{\alpha}_p^n)$ ▷ (Section 3.3.1)
  $[\boldsymbol{\alpha}_{p,i}, \boldsymbol{\mathcal{U}}_{p,i}] \leftarrow$ CalcFlux$(\boldsymbol{I}, \Delta t)$ ▷ Compute fluxes (Algorithm 3)
  for $p = 1 \rightarrow N_{CV}$ **do** ▷ Loop over control volumes in mesh
    $\alpha_p^{n+1} \leftarrow \alpha_p^n - \dfrac{\Delta t}{\mathcal{V}_p} \sum_{i=1}^{N_S} (\alpha_{p,i} \mathcal{U}_{p,i} \mathcal{A}_{p,i})$ ▷ Update using Eq. 3.18
  end for
  return $\boldsymbol{\alpha}_p^{n+1}$
end function

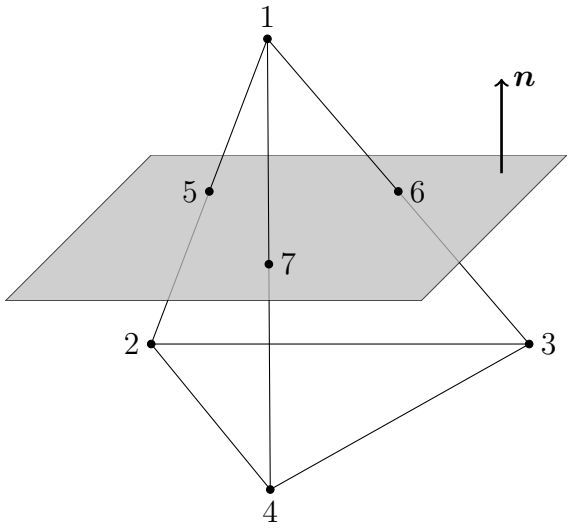


Figure 3.15: Example simplex and cut-plane classified as $Case = 2$. Vertices indicated with their index number.

3.4 Verification tests

The proposed scheme is studied using a variety of test cases. All of the tests use a second-order Runge-Kutta method to solve Eq. 3.20. The flux volumes are discretized using eight simplices, six from the initial discretization plus two additional simplices to construct conservative fluxes. All tests use a specified velocity field and do not require the velocity field to be obtained from a Navier-Stokes solver.

3.4.1 Zalesak's disk

The first verification test case is known as Zalesak's disk [57] and tests the ability of the proposed VOF scheme to transport a two-dimensional shape with sharp corners. The velocity field is specified to produce solid body rotation using

$$\begin{aligned} u &= -2\pi y, \\ v &= +2\pi x. \end{aligned} \tag{3.30}$$

Algorithm 3 CalcFlux : returns $\boldsymbol{\alpha}_{p,i}$ and $\boldsymbol{\mathcal{U}}_{p,i}$, i.e., arrays of the liquid volume fraction fluxes and the mean flux velocity associated with the flux volume $\Omega_{p,i}$

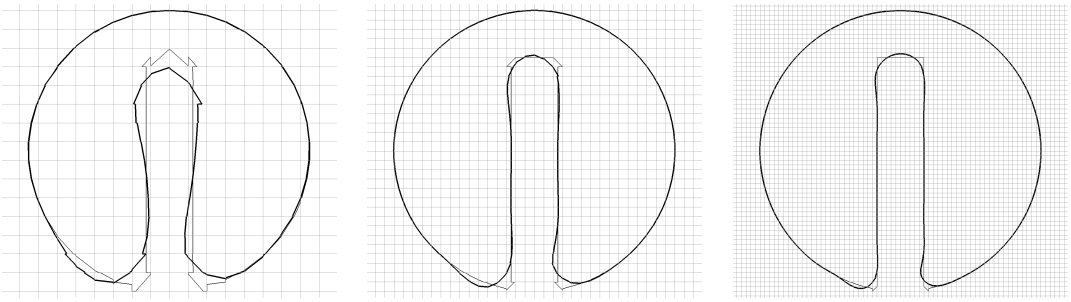
function CalcFlux($\boldsymbol{I}, \Delta t$)
  input $\boldsymbol{I}$ ▷ Interface reconstruction
  input Δt ▷ Time-step
  $\boldsymbol{\mathcal{L}}_p \leftarrow 0$ ▷ Zero arrays
  $\boldsymbol{\mathcal{V}}_p \leftarrow 0$
  for $p = 1 \rightarrow N_{CV}$ **do** ▷ Loop over control volumes in mesh
    for $i = 1 \rightarrow N_S$ **do** ▷ Loop over faces of p^{th} control volume
      $[N_{\text{planes}}, \boldsymbol{P}] \leftarrow$ CutPlanes(p, i) ▷ Construct cut-planes
      $\boldsymbol{V} \leftarrow$ FaceVertices(p, i) ▷ Create vertices on face (Fig. 3.9)
      $\boldsymbol{V} \leftarrow$ ProjectVertices$(\boldsymbol{V}, \Delta t)$ ▷ Project vertices using Eq. 3.20
      $[N_{\text{Sim}}, \boldsymbol{S}] \leftarrow$ PartitionFlux$(\boldsymbol{V})$
        ▷ Partition flux volume into simplices according to Fig. 3.9
      $\mathcal{V}_{\text{cor}} \leftarrow$ CorrecttionVolume$(D_i, N_{\text{Sim}}, \boldsymbol{S})$
        ▷ Additional volume needed to correct flux, Eq. 3.28
      $[N_{\text{Add}}, \boldsymbol{S}] \leftarrow$ SolenoidalFlux$(D_i, \boldsymbol{V}, \mathcal{V}_{\text{cor}})$
        ▷ Create additional simplices to construct conservative flux
      for $n = 1 \rightarrow N_{\text{Sim}} + N_{\text{Add}}$ **do**
        ▷ Loop over simplices and update volume and liquid volume
        $\mathcal{V}_{p,i} \leftarrow \mathcal{V}_{p,i} +$ SimplexVolume$(\boldsymbol{S}(n)) \cdot$ SimplexSign$(\boldsymbol{S}(n))$
        $\mathcal{L}_{p,i} \leftarrow \mathcal{L}_{p,i} +$ SimplexLiquidVolume$(\boldsymbol{S}(n), \boldsymbol{P}, N_{\text{planes}}, \boldsymbol{I})$
          $\cdot$SimplexSign$(\boldsymbol{S}(n))$
      end for
      $\alpha_{p,i} \leftarrow \mathcal{L}_{p,i}/\mathcal{V}_{p,i}$ ▷ Compute liquid volume fraction flux
      $\mathcal{U}_{p,i} \leftarrow \dfrac{\mathcal{V}_{p,i}}{\Delta t \mathcal{A}_{p,i}}$
    end for
  end for
  return $\boldsymbol{\alpha}_{p,i}$
  return $\boldsymbol{\mathcal{U}}_{p,i}$
end function

The shape is a notched disk with diameter 0.3, notch width of 0.05, initially centered at $(x, y) = (0, 0.25)$ within a square domain $[-0.5, 0.5]^2$. The disk shape should not change given the specified velocity field and should simply rotate about the origin. The disk is rotated for one revolution using various meshes. Figure 3.16 shows the shape of Zalesak's disk after it has been rotated. The images in the figure, and subsequent figures, show the PLIC representation of the gas-liquid

Algorithm 4 SimplexLiquidVolume : returns the amount of liquid within the simplex $\boldsymbol{S}$

function SimplexLiquidVolume($\boldsymbol{S}, \boldsymbol{P}, N_{\text{planes}}, \boldsymbol{I}$)
  input $\boldsymbol{S}$ ▷ Array of simplex vertices
  input $\boldsymbol{P}$ ▷ Array of cut-planes
  input N_{planes} ▷ Number of planes in the array $\boldsymbol{P}$
  input $\boldsymbol{I}$ ▷ Array of planes that represent the gas-liquid interface
  $L_{\text{vol}} \leftarrow 0$
  $[N_1, \boldsymbol{S}_1] \leftarrow$ CutSimplex($\boldsymbol{S}(:), \boldsymbol{P}(1)$) ▷ Cut by first plane
  for $i = 1 \rightarrow N_1$ **do**
    $[N_2, \boldsymbol{S}_2] \leftarrow$ CutSimplex($\boldsymbol{S}_1(i,:), \boldsymbol{P}(2)$) ▷ Cut by second plane
    ⋮
    for $j = 1 \rightarrow N_{N_{\text{planes}}-2}$ **do**
      $[N_{N_{\text{planes}}-1}, \boldsymbol{S}_{N_{\text{planes}}-1}] \leftarrow$ CutSimplex$\big(\boldsymbol{S}_{N_{\text{planes}}-2}(j,:), \boldsymbol{P}(N_{\text{planes}}-1)\big)$
      for $k = 1 \rightarrow N_{N_{\text{planes}}-1}$ **do**
        $[N_{N_{\text{planes}}}, \boldsymbol{S}_{N_{\text{planes}}}] \leftarrow$ CutSimplex$\big(\boldsymbol{S}_{N_{\text{planes}}-1}(k,:), \boldsymbol{P}(N_{\text{planes}})\big)$
        for $m = 1 \rightarrow N_{N_{\text{planes}}}$ **do**
          $p \leftarrow$ SimplexIndex$\big(\boldsymbol{S}_{N_{\text{planes}}}(m,:)\big)$
            ▷ Get index of cell in which this simplex is
          $[N_I, \boldsymbol{S}_I] \leftarrow$ CutSimplex$\big(\boldsymbol{S}_{N_{\text{planes}}}(m,:), \boldsymbol{I}_p\big)$
            ▷ Cut by gas-liquid interface within this cell
          for $n = 1 \rightarrow N_I$ **do**
            if Distance$\big(\frac{1}{4}\sum_{v=1}^{4} \boldsymbol{S}_I(n,v), \boldsymbol{I}_p\big) < 0$ **then**
              ▷ Simplex is on liquid side
              $\mathcal{L} \leftarrow \mathcal{L} +$ SimplexVolume($\boldsymbol{S}_I(n,:)$)
                ▷ Add to liquid volume
            end if
          end for
        end for
      end for
      ⋮
    end for
  end for
  return $\mathcal{L}$ ▷ Volume of liquid within $\boldsymbol{S}$
end function

interface. Even on the coarsest 50^2 mesh, the method is able to maintain the notch and the shape of the rotation disk closely resembles the reference solution.



(a) 50^2 mesh (b) 100^2 mesh (c) 200^2 mesh

Figure 3.16: Zalesak's disk after one rotation of various meshes. The thick line is the computed solution and the thin line indicates the initial condition on each mesh.

To test the proposed VOF scheme quantitatively we use the error norms

$$E_{\text{mass}}(t) = \sum_{p=1}^{N_{\text{CV}}} \mathcal{V}_p \alpha_p(t) - \sum_{p=1}^{N_{\text{CV}}} \mathcal{V}_p \alpha_p^e(t), \tag{3.31}$$

$$E_{\text{bound}}(t) = \max\Big(- \min_{p=1\ldots N_{\text{CV}}} \mathcal{V}_p \alpha_p(t), \max_{p=1\ldots,N_{\text{CV}}} \mathcal{V}_p (\alpha_p(t) - 1) \Big) \tag{3.32}$$

and

$$E_{\text{shape}}(t) = \sum_{p=1}^{N_{\text{CV}}} \mathcal{V}_p \Big| \alpha_p(t) - \alpha_p^e(t) \Big|, \tag{3.33}$$

where $\alpha_p(t)$ and $\alpha_p^e(t)$ are the computed and exact liquid volume fraction within the p^{th} computational cell at time t, respectively. E_{mass} provides a measure of how the amount of liquid mass within the domain compares to the reference solution. E_{bound} is an error norm that measures overshoots or undershoots of α. $E_{\text{shape}}(t)$ depends on the distribution of the liquid within the domain and provides an error for the liquid shape at time t [2,3]. The errors are expected to increase throughout the simulation, hence the errors are computed at the end of the simulation and are indicated with the shorthand notation E_{mass}, E_{bound}, and E_{shape}, respectively.

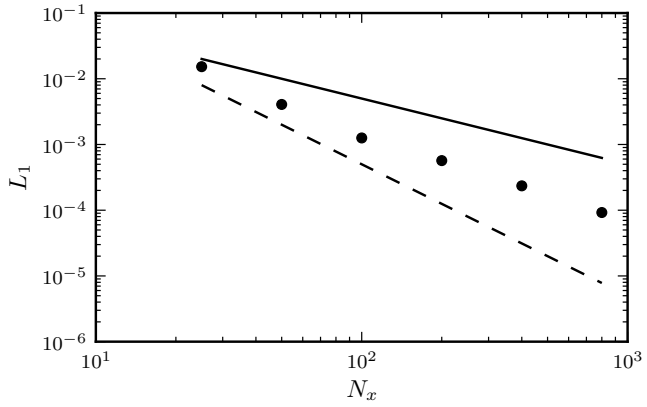


Figure 3.17: Convergence of the E_{shape} error at the end of the simulation for the transport of Zalesak's disk. Solid and dashed lines shows first- and second-order convergence, respectively.

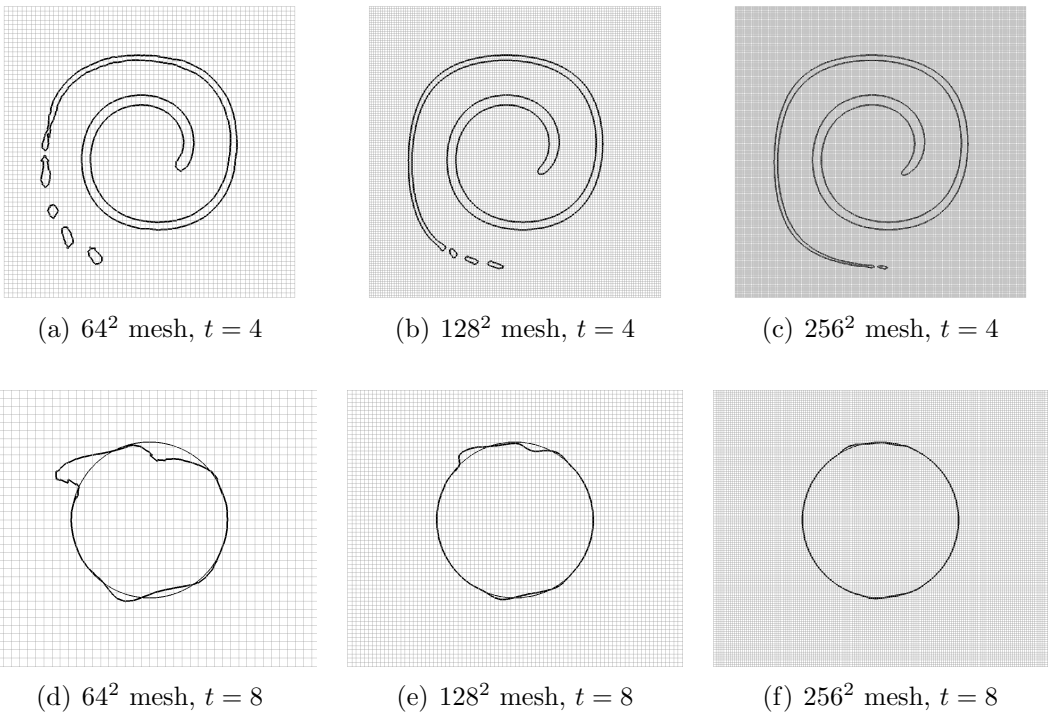
Figure 3.17 shows how the E_{shape} error converges under mesh refinement. A convergence rate between first- and second-order is observed. It is likely that the sharp corners in the solution reduce the convergence rate from the expected second-order, which is the rate observed in all other verification tests below. Additionally, the mass and boundedness errors, shown in Table 3.1, remain at machine precision for all of the meshes.

N_x	E_{shape}	E_{mass}	E_{bound}
25	1.526e-02	4.629e-18	3.526e-17
50	4.066e-03	3.011e-17	6.389e-18
100	1.257e-03	4.409e-18	9.588e-18
200	5.684e-04	3.705e-18	1.082e-17
400	2.348e-04	2.317e-18	1.227e-17
800	9.221e-05	1.937e-17	1.407e-17

Table 3.1: Error norms for the transport of Zalesak's disk simulations.

3.4.2 Two-dimensional deformation

This test case, proposed by Leveque [75], consists of stretching and un-stretching a disk in a vortex. The simulation is initialized with a two-dimensional disk of



(a) 64^2 mesh, $t = 4$ (b) 128^2 mesh, $t = 4$ (c) 256^2 mesh, $t = 4$

(d) 64^2 mesh, $t = 8$ (e) 128^2 mesh, $t = 8$ (f) 256^2 mesh, $t = 8$

Figure 3.18: Effect of mesh size on two-dimensional deformation test case. The top images show the disk at maximum deformation. The bottom images show the result at the end of the simulation with the thick line and the thin line indicates the initial condition.

diameter 0.3 centered at $(x, y) = (0, 0.25)$ within a unit square domain $[-0.5, 0.5]^2$. The disk is stretched using

$$\begin{aligned} u &= -2\sin^2(\pi x)\sin(\pi y)\cos(\pi y)\cos(\pi t/8), \\ v &= +2\sin^2(\pi y)\sin(\pi x)\cos(\pi x)\cos(\pi t/8). \end{aligned} \tag{3.34}$$

The velocity field stretches the disk until time is $t = 4$; then, the velocity field is reversed for another four time units and the disk returns to its initial state.

Figure 3.18 shows snapshots of the disk in the fully stretched state ($t = 4$) and at the end of the simulation ($t = 8$) on various meshes. On the coarsest mesh, the liquid in the tail region reaches the resolution limit of the mesh and the tail breaks into a series of droplets. This causes the liquid to move away from the reference solution, and the final shape does not match the expected solution. Note that this

behavior is expected for methods with good mass conservation properties [28]. On the finest mesh, very little of the liquid in the tail is moved into droplets and the final shape matches the exact solution very well.

Figure 3.19 provides quantitative results and shows the E_{shape} error. Second-order convergence and small values are obtained for the shape error at the end of the simulation. Table 3.2 provides the mass and boundedness errors which remain at machine precision for all of the meshes.

Table 3.3 provides a comparison with results reported by López et al. [2] obtained using EMFPA. EMFPA is very similar to the proposed method, but is limited to two dimensions. Table 3.3 shows similar shape errors at times $t = 0.5$ and $t = 2$ for both approaches. Note that the reference solution used for computing the shape errors is obtained on a $N_x = 1024$ mesh.

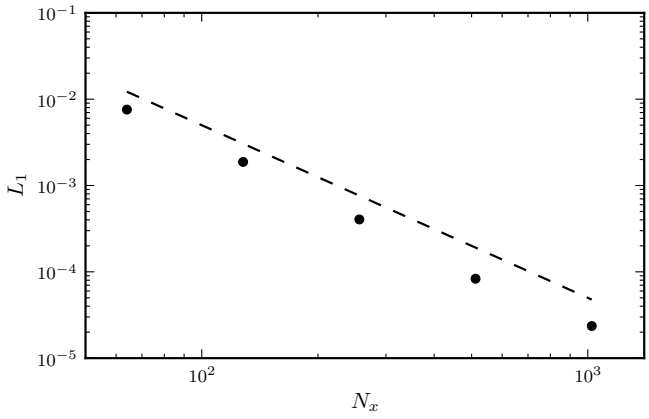


Figure 3.19: Convergence of the E_{shape} error at the end of the two-dimensional deformation test. Dashed line shows second-order convergence.

N_x	E_{shape}	E_{mass}	E_{bound}
64	7.576e-03	9.755e-15	6.517e-17
128	1.876e-03	1.290e-14	6.328e-17
256	4.045e-04	1.392e-14	8.741e-17
512	8.320e-05	1.736e-14	9.325e-17
1024	2.356e-05	1.678e-14	1.043e-16

Table 3.2: Error norms for the two-dimensional deformation test.

	$E_{\text{shape}}(t = 0.5)$		$E_{\text{shape}}(t = 2)$	
N_x	EMFPA [2]	Proposed	EMFPA [2]	Proposed
32	2.93e-03	1.58e-03	1.22e-02	2.00e-02
64	7.58e-04	4.43e-04	3.35e-03	3.33e-03
128	1.75e-04	1.19e-04	7.95e-04	8.90e-04

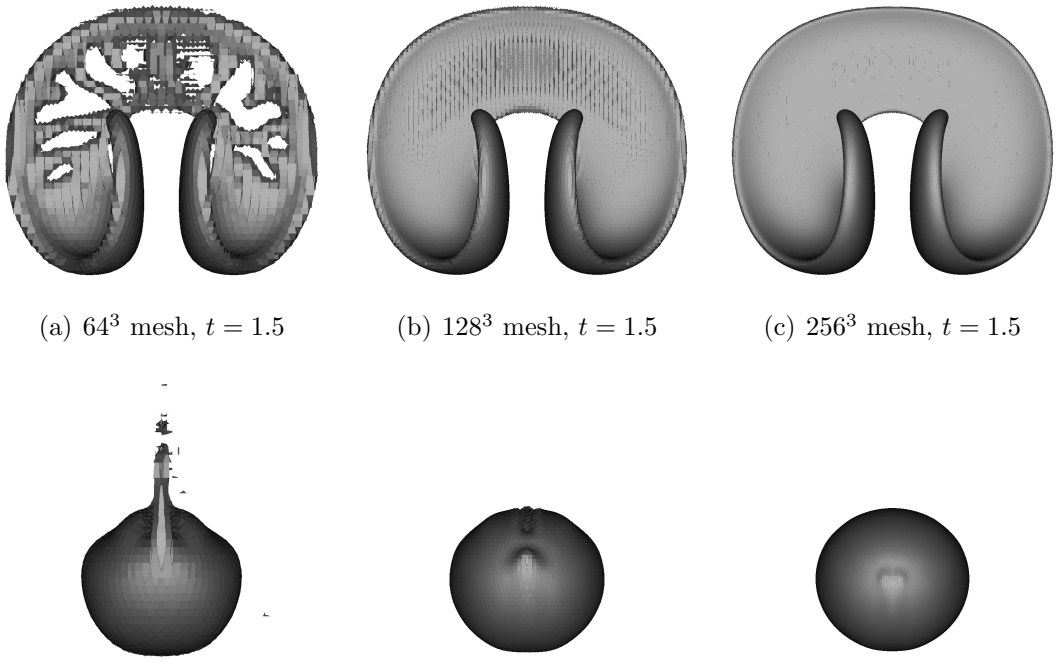
Table 3.3: Shape error of proposed scheme compared with EMFPA of López et al. [2] for the two-dimensional deformation test. The error norm is evaluated at $t = 0.5$ and $t = 2$.

3.4.3 Three-dimensional deformation

This test case is similar to the two-dimensional deformation test case and was also proposed by Leveque [75]. It focuses on the behavior of the VOF scheme when a liquid sheet becomes under-resolved. The simulation is initialized with a droplet of diameter 0.3 centered at $(x, y, z) = (0.35, 0.35, 0.35)$ within a cube domain $[0, 1]^3$. The droplet is stretched until $t = 1.5$, then the velocity field is reversed and the liquid is un-stretched until $t = 3$. The velocity field used to stretch and un-stretch the droplet is

$$\begin{aligned} u &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos(\pi t/3), \\ v &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos(\pi t/3), \\ w &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos(\pi t/3). \end{aligned} \tag{3.35}$$

Figure 3.20 shows snapshots of the gas-liquid interface at maximum stretching ($t = 1.5$) and at the end of the simulation ($t = 3$), when the droplet shape should match the initial condition. At maximum stretching, a thin sheet is formed that



(a) 64^3 mesh, $t = 1.5$ (b) 128^3 mesh, $t = 1.5$ (c) 256^3 mesh, $t = 1.5$

(d) 64^3 mesh, $t = 3.0$ (e) 128^3 mesh, $t = 3.0$ (f) 256^3 mesh, $t = 3.0$

Figure 3.20: PLIC interface for the droplet in three-dimensional deformation flow on various meshes. Snapshots on the top show the droplet at maximum deformation ($t = 1.5$). The droplets at the end of the simulation ($t = 3$) are shown on the bottom.

falls below the resolution of the mesh when a 64^3 mesh is used. The method moves the under-resolved liquid from the sheet into resolvable structures. As the mesh is refined, less of the sheet becomes under-resolved, and on the 256^3 mesh, the liquid sheet is maintained. Hence, discrepancies in the final shape are reduced with mesh refinement.

As shown in Fig. 3.21, second-order convergence is obtained for the E_{shape} error. In Table 3.4 the E_{shape} error is compared with results provided by Hernández et al. [3] using the FMFPA-3D scheme. The proposed method and FMFPA-3D produce very similar results with slightly lower errors using the proposed method. This is expected since both methods are un-split geometric formulations. However, in addition to the lower errors, the proposed scheme provides discrete conservation,

as indicated by E_{mass} in Table 3.4. The table also provides timing data for the simulations performed using the proposed method. The simulations were performed using a hyperthreaded dual 6-core X5670 3 GHz CPU with 48 GB of RAM. The result shows the average time per time-step throughout the simulation. Finally, E_{bound} is found to remain at machine zero on all meshes.

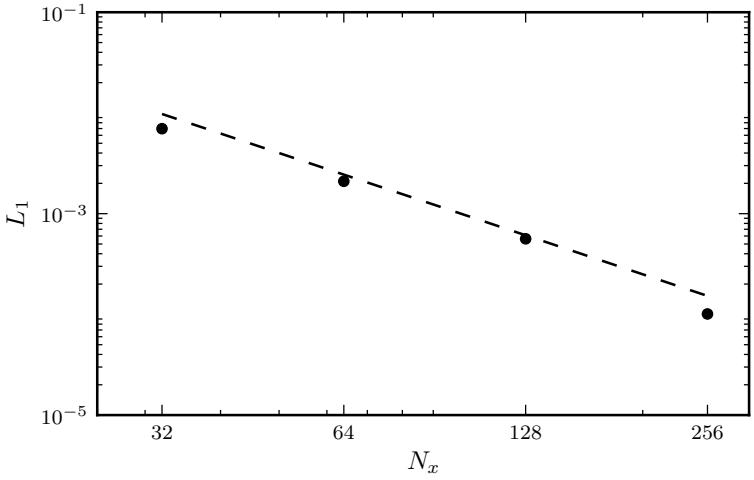


Figure 3.21: Convergence of the E_{shape} error for the three-dimensional deformation simulations. The dashed line shows second-order convergence.

	E_{shape}				
N_x	FMFPA-3D [3]	Proposed	E_{mass}	E_{bound}	Time/time-step (s)
32	7.440e-03	6.978e-03	1.194e-15	1.202e-17	0.78
64	2.790e-03	2.096e-03	2.479e-15	2.341e-17	2.85
128	7.140e-04	5.625e-04	1.675e-14	2.752e-17	12.2
256	-	1.010e-04	3.870e-14	4.690e-17	45.5

Table 3.4: Comparison of E_{shape} errors at end of three-dimensional deformation test using proposed method and those reported in Hernández et al. [3]. A E_{shape} error on the 256^3 mesh was not provided by Hernández et al. The table also provides the average time per time-step for the simulations performed using the proposed code. Mass and boundedness errors are also provided.

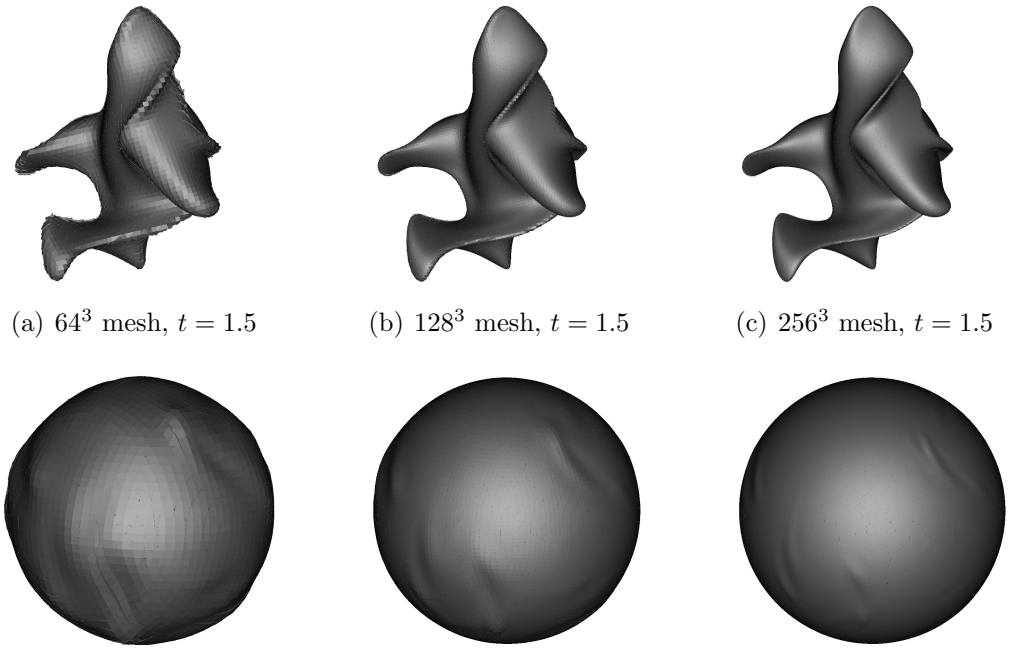
3.4.4 Droplet in homogeneous isotropic turbulence

This numerical experiment was designed to test the performance of the proposed scheme in a more realistic flow situation. The test consists of the deformation of a three-dimensional droplet in a complex velocity field. The velocity field was created from an instantaneous snapshot of synthetic homogeneous isotropic turbulence, denoted by $\boldsymbol{u}_0$. This solenoidal velocity field was created in spectral space from a Passot-Pouquet model spectrum. The same velocity field is used for all of the test cases presented below. Using that velocity field, the droplet is deformed for 1.5 time units; then, the velocity is reversed for another 1.5 time units. This is achieved using a temporally varying cosine function, i.e.,

$$\boldsymbol{u} = \boldsymbol{u}_0 \cos\Big(\frac{\pi t}{3}\Big). \tag{3.36}$$

The domain used for the simulation is $[0, 2\pi]^3$, and the droplet of diameter π is initialized at $(x, y, z) = (\pi, \pi, \pi)$.

Figure 3.22 shows the shape of the droplet after it has been deformed by the turbulence ($t = 1.5$), and at the end of the simulation, when the initial shape of the droplet should be recovered. The results are presented on three different meshes, namely 64^3, 128^3, and 256^3. The overall qualitative shape agrees very well between the various cases at the middle and end of the simulations. Figure 3.23 shows the convergence of the E_{shape} error under mesh refinement, showing second-order accuracy. The mass and boundedness errors, shown in Table 3.5, remain at machine precision for all mesh levels.



(a) 64^3 mesh, $t = 1.5$ (b) 128^3 mesh, $t = 1.5$ (c) 256^3 mesh, $t = 1.5$

(d) 64^3 mesh, $t = 3.0$ (e) 128^3 mesh, $t = 3.0$ (f) 256^3 mesh, $t = 3.0$

Figure 3.22: Gas-liquid interface for the droplet in homogeneous isotropic turbulence test on various meshes. Snapshots on the top show the droplet at maximum deformation ($t = 1.5$). The droplets at the end of the simulation ($t = 3$) are shown on the bottom.

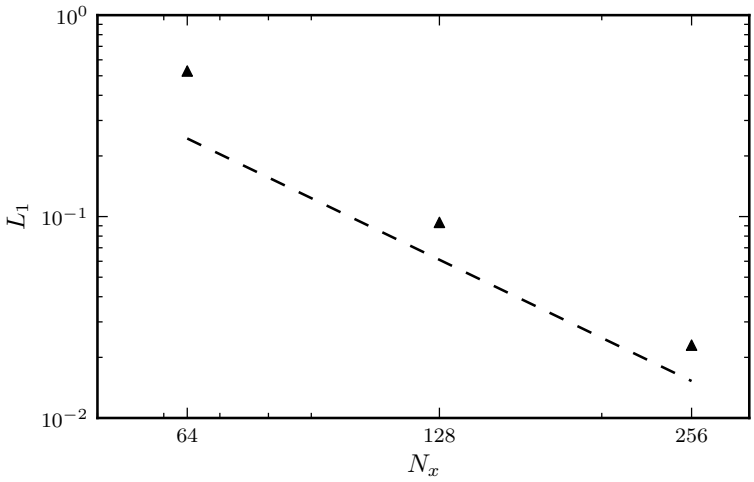


Figure 3.23: Convergence of the E_{shape} error (triangles) for the simulation of the droplet in homogeneous isotropic turbulence. Dashed line shows second-order convergence.

N_x	E_{shape}	E_{mass}	E_{bound}
64	5.281e-01	5.472e-16	1.124e-17
128	9.357e-02	1.198e-15	1.198e-17
256	2.300e-02	1.290e-14	7.598e-17

Table 3.5: Error norms for the droplet in homogeneous isotropic turbulence test case.

3.5 Conclusions

In this paper, we have developed and tested a bounded, conservative, un-split, three-dimensional geometric transport scheme that was applied to the piecewise linear interface calculation (PLIC) volume-of-fluid (VOF) method. The scheme leverages two key ideas that make it straightforward to implement. The first is the use of simplices to represent semi-Lagrangian flux volumes. The simplices are created using the same vertices for all flux volume geometries, which greatly simplifies the process of discretizing the complex shapes. The second idea is a simple sign convention for identifying if a simplex contributes positively or negatively to the flux. The scheme was verified using a collection of canonical test cases including Zalesak's disk, two- and three-dimensional deformation tests, and the deformation of a droplet in three-dimensional homogeneous isotropic turbulence. In all of the test cases, the method produced excellent results even on coarse meshes. Second-order convergence, discrete conservation, and boundedness were demonstrated.

3.6 Additional algorithms

Algorithm 5 CutSimplex : cuts a simplex by a plane and partitions resulting shapes into new simplices using look-up tables

function CutSimplex($\boldsymbol{S}, \boldsymbol{P}$)
  input $\boldsymbol{S}$ ▷ Array of simplex vertices, i.e. $\boldsymbol{S} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4]$
  input $\boldsymbol{P}$ ▷ Plane equation coefficients, i.e. $\boldsymbol{P} = \{[a, b, c, d] \mid ax + by + cz = d\}$
  $\boldsymbol{Pt}(1:4, :) \leftarrow \boldsymbol{S}$ ▷ Copy simplex vertices into point array
  for $i = 1 \rightarrow 4$ **do**
    $\boldsymbol{d}(i) \leftarrow$ Distance($\boldsymbol{Pt}(i), \boldsymbol{P}$) ▷ Calculate distance between point and plane
  end for
  $Case \leftarrow 1 + 1\left(\frac{1}{2} + \frac{1}{2}\,\mathrm{sign}(\boldsymbol{d}(1))\right) + 2\left(\frac{1}{2} + \frac{1}{2}\,\mathrm{sign}(\boldsymbol{d}(2))\right) + 4\left(\frac{1}{2} + \frac{1}{2}\,\mathrm{sign}(\boldsymbol{d}(3))\right) + 8\left(\frac{1}{2} + \frac{1}{2}\,\mathrm{sign}(\boldsymbol{d}(4))\right)$ ▷ Create case number (1-16)
  for $n = 1 \rightarrow$ NumberIntersect($Case$) **do**
    ▷ Loop over intersections between simplex edges and plane
    $I_1 \leftarrow$ IndexEndPoint($1, n, Case$) ▷ Get index of points on edge
    $I_2 \leftarrow$ IndexEndPoint($2, n, Case$)
    $\boldsymbol{Pt}(4+n, :) \leftarrow \boldsymbol{Pt}(I_1, :) - \frac{d(I_1)}{d(I_2) - d(I_1)}(\boldsymbol{Pt}(I_2, :) - \boldsymbol{Pt}(I_1, :))$
    ▷ Calculate intersection and append to points array
  end for
  $N_{\mathrm{Out}} \leftarrow$ NumberSimsPart($Case$) ▷ Number of simplices in partition
  for $n = 1 \rightarrow N_{\mathrm{Out}}$ **do**
    for $v = 1 \rightarrow 4$ **do**
      $\boldsymbol{S}_{\mathrm{Out}}(n, :) \leftarrow \boldsymbol{Pt}($IndexSimVert$(v, n, Case), :)$
      ▷ Create simplices in partition
    end for
  end for
  return N_{Out} ▷ Number of simplices returned
  return $\boldsymbol{S}_{\mathrm{Out}}$ ▷ Vertices of simplices returned
end function

Algorithm 6 Look-up Tables : provide useful quantities based on the case number of the simplex

▷ Number of intersections between simplex and plane

1: $\textsc{NumberIntersect} \leftarrow [0, 3, 3, 4, 3, 4, 4, 3, 3, 4, 4, 3, 4, 3, 3, 0]$

▷ Indices of endpoints on line that intersects plane

2: $\textsc{IndexEndPoint}(:, :, \ 1) \leftarrow \big[[0, 0], [0, 0], [0, 0], [0, 0]\big]$
3: $\textsc{IndexEndPoint}(:, :, \ 2) \leftarrow \big[[1, 2], [1, 3], [1, 4], [0, 0]\big]$
4: $\textsc{IndexEndPoint}(:, :, \ 3) \leftarrow \big[[2, 3], [2, 4], [2, 1], [0, 0]\big]$
5: $\textsc{IndexEndPoint}(:, :, \ 4) \leftarrow \big[[1, 4], [2, 4], [1, 3], [2, 3]\big]$
6: $\textsc{IndexEndPoint}(:, :, \ 5) \leftarrow \big[[3, 4], [3, 1], [3, 2], [0, 0]\big]$
7: $\textsc{IndexEndPoint}(:, :, \ 6) \leftarrow \big[[1, 4], [3, 4], [1, 2], [3, 2]\big]$
8: $\textsc{IndexEndPoint}(:, :, \ 7) \leftarrow \big[[2, 4], [3, 4], [2, 1], [3, 1]\big]$
9: $\textsc{IndexEndPoint}(:, :, \ 8) \leftarrow \big[[4, 1], [4, 2], [4, 3], [0, 0]\big]$
10: $\textsc{IndexEndPoint}(:, :, \ 9) \leftarrow \big[[4, 1], [4, 2], [4, 3], [0, 0]\big]$
11: $\textsc{IndexEndPoint}(:, :, 10) \leftarrow \big[[1, 3], [4, 3], [1, 2], [4, 2]\big]$
12: $\textsc{IndexEndPoint}(:, :, 11) \leftarrow \big[[2, 3], [4, 3], [2, 1], [4, 1]\big]$
13: $\textsc{IndexEndPoint}(:, :, 12) \leftarrow \big[[3, 4], [3, 1], [3, 2], [0, 0]\big]$
14: $\textsc{IndexEndPoint}(:, :, 13) \leftarrow \big[[3, 2], [4, 2], [3, 1], [4, 1]\big]$
15: $\textsc{IndexEndPoint}(:, :, 14) \leftarrow \big[[2, 3], [2, 4], [2, 1], [0, 0]\big]$
16: $\textsc{IndexEndPoint}(:, :, 15) \leftarrow \big[[1, 2], [1, 3], [1, 4], [0, 0]\big]$
17: $\textsc{IndexEndPoint}(:, :, 16) \leftarrow \big[[0, 0], [0, 0], [0, 0], [0, 0]\big]$

▷ Number of simplices in partition of original simplex

18: $\textsc{NumberSimsPart} \leftarrow \big[1, 4, 4, 6, 4, 6, 6, 4, 4, 6, 6, 4, 6, 4, 4, 1\big]$

▷ Indices of vertices used to partition simplex

19: $\textsc{IndexSimVert}(:, :, \ 1) \leftarrow \big[[1, 2, 3, 4], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]\big]$
20: $\textsc{IndexSimVert}(:, :, \ 2) \leftarrow \big[[1, 5, 6, 7], [4, 2, 3, 6], [4, 2, 5, 6], [4, 5, 6, 7], [0, 0, 0, 0], [0, 0, 0, 0]\big]$
21: $\textsc{IndexSimVert}(:, :, \ 3) \leftarrow \big[[2, 5, 6, 7], [1, 3, 4, 6], [1, 3, 5, 6], [1, 5, 6, 7], [0, 0, 0, 0], [0, 0, 0, 0]\big]$
22: $\textsc{IndexSimVert}(:, :, \ 4) \leftarrow \big[[5, 6, 8, 2], [5, 7, 8, 1], [5, 8, 1, 2], [5, 6, 8, 4], [5, 7, 8, 3], [5, 8, 4, 3]\big]$
23: $\textsc{IndexSimVert}(:, :, \ 5) \leftarrow \big[[3, 5, 6, 7], [2, 4, 1, 6], [2, 4, 5, 6], [2, 5, 6, 7], [0, 0, 0, 0], [0, 0, 0, 0]\big]$
24: $\textsc{IndexSimVert}(:, :, \ 6) \leftarrow \big[[5, 6, 8, 3], [5, 7, 8, 1], [5, 8, 1, 3], [5, 6, 8, 4], [5, 7, 8, 2], [5, 8, 4, 2]\big]$
25: $\textsc{IndexSimVert}(:, :, \ 7) \leftarrow \big[[5, 6, 8, 3], [5, 7, 8, 2], [5, 8, 2, 3], [5, 6, 8, 4], [5, 7, 8, 1], [5, 8, 4, 1]\big]$
26: $\textsc{IndexSimVert}(:, :, \ 8) \leftarrow \big[[1, 2, 3, 7], [1, 2, 6, 7], [1, 5, 6, 7], [4, 5, 6, 7], [0, 0, 0, 0], [0, 0, 0, 0]\big]$
27: $\textsc{IndexSimVert}(:, :, \ 9) \leftarrow \big[[4, 5, 6, 7], [3, 1, 2, 6], [3, 1, 5, 6], [3, 5, 6, 7], [0, 0, 0, 0], [0, 0, 0, 0]\big]$
28: $\textsc{IndexSimVert}(:, :, 10) \leftarrow \big[[5, 6, 8, 4], [5, 7, 8, 1], [5, 8, 1, 4], [5, 6, 8, 3], [5, 7, 8, 2], [5, 8, 3, 2]\big]$
29: $\textsc{IndexSimVert}(:, :, 11) \leftarrow \big[[5, 6, 8, 4], [5, 7, 8, 2], [5, 8, 2, 4], [5, 6, 8, 3], [5, 7, 8, 1], [5, 8, 3, 1]\big]$
30: $\textsc{IndexSimVert}(:, :, 12) \leftarrow \big[[4, 1, 2, 7], [4, 1, 6, 7], [4, 5, 6, 7], [3, 5, 6, 7], [0, 0, 0, 0], [0, 0, 0, 0]\big]$
31: $\textsc{IndexSimVert}(:, :, 13) \leftarrow \big[[5, 6, 8, 4], [5, 7, 8, 3], [5, 8, 3, 4], [5, 6, 8, 2], [5, 7, 8, 1], [5, 8, 2, 1]\big]$
32: $\textsc{IndexSimVert}(:, :, 14) \leftarrow \big[[3, 4, 1, 7], [3, 4, 6, 7], [3, 5, 6, 7], [2, 5, 6, 7], [0, 0, 0, 0], [0, 0, 0, 0]\big]$
33: $\textsc{IndexSimVert}(:, :, 15) \leftarrow \big[[2, 3, 4, 7], [2, 3, 6, 7], [2, 5, 6, 7], [1, 5, 6, 7], [0, 0, 0, 0], [0, 0, 0, 0]\big]$
34: $\textsc{IndexSimVert}(:, :, 16) \leftarrow \big[[1, 2, 3, 4], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]\big]$

CHAPTER 4

TRANSPORT OF QUANTITIES WITH DISCONTINUITIES

Transporting scalar or vector quantities with discontinuities is a situation often found in multiphase flow simulations. For example, species concentrations are likely to be defined within a single phase. Another example is electric charges in electrohydrodynamic (EHD) systems wherein the electric charges are only present within the liquid phase. In this chapter, we present a conservative and consistent discretization that describes the motion of such quantities. This work is a direct extension of the methodology presented in Chapter 3, where similar ideas are used to transport the phase interface. Using the same methodology to transport the phase interface and these quantities ensures discrete consistency which is needed to avoid spurious over/undershoots and to achieve discrete conservation.

4.1 Mathematical formulation

In this section the conservation law for a quantity is recast into a form that is discretizable in the presence of a phase interface with the associated discontinuities. The formulation closely follows the derivation in Chapter 3, but is extended with additional source and flux terms.

The evolution of a scalar $\psi(\boldsymbol{x}, t)$ in a solenoidal velocity field is described by

$$\frac{\partial \psi}{\partial t} + \nabla \cdot (\boldsymbol{u}\psi) = S + \nabla \cdot \boldsymbol{F}, \tag{4.1}$$

where $\boldsymbol{x}$ is the spatial coordinate, t is time, $\boldsymbol{u}$ is the velocity field that is assumed to be known, S is a source term, and $\boldsymbol{F}$ is an additional conservative flux (e.g., diffusion). Integrating this equation over a discrete time-step $\Delta t = t^{n+1} - t^n$ and

fixed control volume CV (e.g., a computational cell) with bounding surface CS and using Gauss' theorem allows us to write

$$\int_{CV}\left(\psi^{n+1}-\psi^{n}\right)\mathrm{d}V+\int_{t^n}^{t^{n+1}}\oint_{CS}(\psi\boldsymbol{u}-\boldsymbol{F})\cdot\boldsymbol{n}_{CV}\,\mathrm{d}S\,\mathrm{d}t=\int_{t^n}^{t^{n+1}}\int_{CV}S\,\mathrm{d}V\,\mathrm{d}t. \quad (4.2)$$

Where we have introduced the shorthand notation $\psi^n=\psi(\boldsymbol{x},t^n)$ and $\boldsymbol{n}_{CV}$ is the outward pointing normal to the control volume CV.

Following the procedure in Chapter 3 the convective flux is reformulated to depend on quantities at t^n which are typically known leading to

$$\begin{aligned}\int_{CV}(\psi^{n+1}-\psi^{n})\,\mathrm{d}V=&\sum_{i=1}^{N_S}\left(\int_{\Omega_i^{n+}}\psi^n\,\mathrm{d}V-\int_{\Omega_i^{n-}}\psi^n\,\mathrm{d}V\right)\\&+\sum_{i=1}^{N_S}\left(\int_{t^n}^{t^{n+1}}\int_{\omega_{i,M}^{+}(t)}\boldsymbol{F}\cdot\boldsymbol{n}_{\Omega_i}\,\mathrm{d}S\mathrm{d}t-\int_{t^n}^{t^{n+1}}\int_{\omega_{i,M}^{-}(t)}\boldsymbol{F}\cdot\boldsymbol{n}_{\Omega_i}\,\mathrm{d}S\mathrm{d}t\right)\\&+\sum_{i=1}^{N_S}\left(\int_{t^n}^{t^{n+1}}\int_{\Omega_i^{+}(t)}S\,\mathrm{d}V\,\mathrm{d}t-\int_{t^n}^{t^{n+1}}\int_{\Omega_i^{-}(t)}S\,\mathrm{d}V\,\mathrm{d}t\right),\end{aligned} \quad (4.3)$$

which is an *exact* relation. The previous equation depends on evaluating integrals of ψ^n over Ω_i^n, which are the flux volumes at $t=t^n$. These integrals are well defined since ψ^n is typically known and Eq. 4.3 is solved to find ψ^{n+1}. The terms that describe the flux due to $\boldsymbol{F}$ and source S depend on quantities throughout the time-step and need to be computed on the moving and deforming flux volume $\Omega_i(t)$ and flux volume surface $\omega_{i,M}(t)$, respectively.

4.2 Numerical methods

Equation 4.3 describes the transport of ψ within a control volume by convective fluxes, the source S, and the generic flux F. In this section, the equation is discretized. To simplify the discussion we will focus on the specific example where ψ is the electric charge density in electrohydrodynamic atomization systems. However,

the numerical approach can be generalized to other quantities that only exist in a single phase or are discontinuous at the phase interface.

Electrohydrodynamic (EHD) enhanced atomization is a process where liquid fuel is charged within a grounded combustion chamber. The EHD effects can significantly improve the atomization process [76]. In these flows, the motion of the conserved electric charges is described by [77]

$$\frac{\partial q}{\partial t} + \nabla \cdot \boldsymbol{J} = S_q, \tag{4.4}$$

where $\boldsymbol{J}$ is the current density and S_q is a source.

$$\boldsymbol{J} = q\boldsymbol{u} + q\kappa\boldsymbol{E} - D\nabla q, \tag{4.5}$$

where κ_i is the ionic mobility coefficient and D is the molecular diffusion coefficient. The three terms that contribute to the current density can be described as convection due to the velocity field, convection due to the electrical velocity $\kappa\boldsymbol{E}$, and diffusion. The electric charges only exist in the liquid phase and a no-flux condition exists at the phase interface. This equation can be described in the framework of the proposed method with $\psi = q$, $S = S_q$, and $\boldsymbol{F} = D\nabla q - q\kappa_i\boldsymbol{E}$. Details of the discretization for the convective fluxes, the flux of F, and the source term are provided in the following sections.

4.2.1 Convective fluxes

The convective fluxes,

$$\int_{\Omega_i^{n+}} \psi^n \, \mathrm{d}V - \int_{\Omega_i^{n-}} \psi^n \, \mathrm{d}V, \tag{4.6}$$

are evaluated such that consistency is maintained with the VOF interface transport. Evaluating the convective fluxes involves integrating f over the flux volumes

Ω_i^n. In realistic velocity fields these streak-tubes can have complicated geometries. In order to evaluate the convective flux integrals, the streak-tube is approximated as a collection of simplices (triangles in two dimensions or tetrahedra in three dimensions). The difference between the discrete and physical streak-tube will introduce a conservation error, however a correction can be added to the discrete streak-tube to ensure discrete conservation [42]. With the simplices, the convective flux integrals are reduce to an integral over a simplex.

Evaluating the integral of ψ over a simplex is performed using the following systematic approach:

1. Ω_i is partitioned into a collection of simplices S.
2. The discrete representation of Ω_i^n is corrected by appending simplices to S.
3. Each simplex $s \in S$ is cut by the computational mesh and partitioned into new simplices M_s local to one computational cell.
4. Each simplex $m \in M_s$ is cut by the gas-liquid interface and partition into new simplices $P_{s,m}$ local to one computational cell and phase.
5. The integral over each simplex $p \in P_{s,m}$, i.e., $\int_p \psi^n \mathrm{d}V$, is evaluated using data local to the phase and computational cell that p is within.
6. The integral over Ω_i^n is computed using

$$\int_{\Omega_i^{n+}} \psi^n \,\mathrm{d}V - \int_{\Omega_i^{n-}} \psi^n \,\mathrm{d}V = \sum_{s \in S} \mathrm{sign}(s) \sum_{m \in M_s} \sum_{p \in P_{s,m}} \int_p \psi^n \,\mathrm{d}V, \tag{4.7}$$

   where $\mathrm{sign}(s)$ is the sign of the orientation of the simplex s as described in Chapter 3.

Step 3 requires cutting a simplex by the planes that comprise the computational mesh which can be performed using a computational geometry toolbox as

described in Chapter 3. Step 4 requires cutting a simplex by the gas-liquid interface. To simplify this step, while maintaining the second-order accuracy of the method, the interface is approximated using the piecewise linear interface calculation (PLIC) [31]. PLIC approximates the interface using a linear function, e.g., a line in two dimensions or a plane in three dimensions. This approximation facilitates using the same computational geometry routines to perform Steps 3 and 4. Additional details of the methodology including algorithms are provided in Chapter 3.

In step 5 the integral of ψ is computed over the simplex p that is local to one computational cell. For VOF transport $\psi = f$ where f is the liquid distribution function, i.e.,

$$f(\boldsymbol{x}, t) = \begin{cases} 1, & \text{if } \boldsymbol{x} \text{ is in the liquid at time } t, \\ 0, & \text{if } \boldsymbol{x} \text{ is in the gas at time } t. \end{cases} \tag{4.8}$$

The integral $\int_p f^n \mathrm{d}V$ reduces to

$$\int_p f \, \mathrm{d}V = \begin{cases} \mathcal{V}_p, & \text{if } p \text{ is in the liquid phase,} \\ 0 & \text{if } p \text{ is in the gas phase,} \end{cases} \tag{4.9}$$

where $\mathcal{V}_p$ is the volume of the simplex p.

Transport of the electric charge density q is more complicated due to spatial variations away from the phase interface. These variations must be accounted for in order to construct a second-order accurate transport scheme. Within each control volume (e.g., computational cell) the electric charge density is approximated using a local second-order Taylor series expansion $\hat{q}$, i.e.,

$$\hat{q}(\boldsymbol{x}, t^n) = q(\boldsymbol{B}, t^n) + \nabla q(\boldsymbol{B}, t^n) \cdot (\boldsymbol{x} - \boldsymbol{B}), \tag{4.10}$$

where $\boldsymbol{B}$ is the barycenter of computational cell weighted by f, i.e.,

$$\boldsymbol{B} = \int_{CV} f\boldsymbol{x}\,\mathrm{d}V. \tag{4.11}$$

Using this definition for the barycenter $\boldsymbol{B}$ ensures that the Taylor series is conservative and $\overline{q}_{CV} = q(\boldsymbol{B},t) = \int_{CV} q(\boldsymbol{x},t)\,\mathrm{d}V = \int_{CV} \hat{q}(\boldsymbol{x},t)\,\mathrm{d}V$, where we have introduced the cell average value $\overline{q}_{CV}$ in the control volume which is the value stored in the numerical code.

The gradient operator in Eq. 4.10 needs to be constructed carefully since q is only defined in the liquid phase. The gradient is computed using a least squares fit of $q(\boldsymbol{B},t)$ in the control volume and the nearest neighbors that contain liquid. With this definition of $\hat{q}$, the integral in Step 5 is computed with the second-order approximation

$$\int_p q(\boldsymbol{x},t^n)\,\mathrm{d}V \approx \begin{cases} \int_p \hat{q}(\boldsymbol{x},t^n)\,\mathrm{d}V = \mathcal{V}_p\hat{q}(\boldsymbol{B}_p,t^n), & \text{if } p \text{ is in the liquid phase,} \\ 0 & \text{if } p \text{ is in the gas phase,} \end{cases} \tag{4.12}$$

where $\boldsymbol{B}_p$ is the barycenter of the simplex p.

Away from the gas-liquid interface q varies spatially but does not contain discontinuities. Therefore, the convective fluxes in Eq. 4.1 can be discretized using many different methods. In this work, we use third-order WENO fluxes [78, 79] away from the interface and the geometric fluxes near the phase interface.

4.2.2 Additional fluxes

In Eq. 4.3, the generic $\boldsymbol{F}$ fluxes have been recast into integrals over $\omega_{i,M}$, the material surface of the flux volume Ω_i, i.e.

$$\int_{t^n}^{t^{n+1}} \int_{\omega_{i,M}^{+}(t)} \boldsymbol{F} \cdot \boldsymbol{n}_{\Omega_i} \, \mathrm{d}S\mathrm{d}t - \int_{t^n}^{t^{n+1}} \int_{\omega_{i,M}^{-}(t)} \boldsymbol{F} \cdot \boldsymbol{n}_{\Omega_i} \, \mathrm{d}S\mathrm{d}t. \tag{4.13}$$

Exactly evaluating this integral would require computing the flux $\boldsymbol{F}$ over this time-dependent, deforming surface. As a result, the choice is made to approximate this integral. Two approximations are natural and are

$$\int_{t^n}^{t^{n+1}} \int_{\omega_{i,M}^{\pm}(t)} \boldsymbol{F} \cdot \boldsymbol{n}_{\Omega_i} \, \mathrm{d}S\mathrm{d}t \approx \Delta t \int_{\omega_{i,M}^{\pm}(t^n)} \boldsymbol{F^n} \cdot \boldsymbol{n}_{\Omega_i} \, \mathrm{d}S \text{ or} \tag{4.14}$$

$$\int_{t^n}^{t^{n+1}} \int_{\omega_{i,M}^{\pm}(t)} \boldsymbol{F} \cdot \boldsymbol{n}_{\Omega_i} \, \mathrm{d}S\mathrm{d}t \approx \Delta t \int_{\omega_{i,M}^{\pm}(t^{n+1})} \boldsymbol{F^{n+1}} \cdot \boldsymbol{n}_{\Omega_i} \, \mathrm{d}S, \tag{4.15}$$

which are the fluxes evaluated at t^n and t^{n+1}.

Note that when evaluating the integral that describes the flux of $\boldsymbol{F}$ across $\omega_{i,M}(t)$, part of $\omega_{i,M}(t)$ is shared between neighboring flux volumes. This part will be positive for one flux volume and negative for the other since either the sign of the flux volume or the normal vectors $\boldsymbol{n}_{\Omega_i}$ are opposite. Therefore, the integral only needs to be evaluated on the portion of $\omega_{i,M}(t)$ that is not shared with another flux volume.

As will be described in Section 4.2.4, an implicit formulation is needed due to the small liquid cells that are present. Writing Eq. 4.14 in an implicit form is challenging since the surface $\omega_{i,M}(t^n)$ is complex, but could be formed using a least squares operator that depends on neighboring cells. However, writing Eq. 4.15 is significantly more straightforward since $\omega_{i,M}(t^{n+1}) = \partial CS_i$ is aligned with the computational mesh.

Therefore, we choose to construct the flux using Eq. 4.15. For the example of electric charge density, q is only defined in the liquid phase and the integral can be discretely approximated by

$$\int_{t^n}^{t^{n+1}} \int_{\omega_{i,M}^+(t)} \boldsymbol{F} \cdot \boldsymbol{n}_{\Omega_i} \, \mathrm{d}S\mathrm{d}t - \int_{t^n}^{t^{n+1}} \int_{\omega_{i,M}^-(t)} \boldsymbol{F} \cdot \boldsymbol{n}_{\Omega_i} \, \mathrm{d}S\mathrm{d}t \approx -\Delta t \mathcal{A}_{\partial CS_i}^w \boldsymbol{F}|_{\partial CS_i} \cdot \boldsymbol{n}_{CV}, \tag{4.16}$$

where $\mathcal{A}_{\partial CS_i}^w$ is the wetted area of the surface ∂CS_i evaluated at $t = t^{n+1}$.

The additional fluxes for electric charge density are $\boldsymbol{F} = D\nabla q - q\kappa_i \boldsymbol{E}$. In this work the diffusive flux, $D\nabla q$, is computed using central differences. The electric convective flux, $q\kappa_i \boldsymbol{E}$, needs to be upwinded for stability reasons. We use a third-order QUICK scheme [80] when all the points in the stencil are in the liquid phase and the upwind scheme for any cells near the interface where the QUICK stencil is not well defined.

4.2.3 Source term

The source terms in Eq. 4.3 is written as

$$\sum_{i=1}^{N_S} \left(\int_{t^n}^{t^{n+1}} \int_{\Omega_i^+(t)} S \, \mathrm{d}V \, \mathrm{d}t - \int_{t^n}^{t^{n+1}} \int_{\Omega_i^-(t)} S \, \mathrm{d}V \, \mathrm{d}t \right). \tag{4.17}$$

Similarly to the flux of F terms, exactly computing this integral is difficult since Ω_i is a time-varying complex shape. Therefore, the integral is approximated. There are a variety of approximations, but an approximation that can easily be written implicitly is

$$\sum_{i=1}^{N_S} \left(\int_{t^n}^{t^{n+1}} \int_{\Omega_i^+(t)} S \, \mathrm{d}V \, \mathrm{d}t - \int_{t^n}^{t^{n+1}} \int_{\Omega_i^-(t)} S \, \mathrm{d}V \, \mathrm{d}t \right) \approx \Delta t \mathcal{V}_{CV}^w S|_{CV}, \tag{4.18}$$

where $\mathcal{V}_{CV}^w$ is the wetted volume of CV evaluated at $t = t^{n+1}$.

4.2.4 Implicit formulation

In many situations ψ is only defined in one phase, like it is for the electric charge density example. Therefore the computational cells on which ψ is computed can be arbitrarily small and require an unrealistically small time-step to respect the Courant-Friedrichs-Lewy (CFL) condition. As a result, an implicit formulation is necessary. The convective fluxes are built using semi-Lagrangian ideas and are unconditionally stable. The additional fluxes F and the source term S need to be written implicitly. In this work we use a modified approximate factorization technique known as the diagonally dominate alternating direction implicit (DDADI) procedure [81]. This approach has successfully been used in other cut-cell formulations [82].

4.3 Verification tests

4.3.1 Discontinuous scalar transport test

This test case assesses the ability of the methodology to transport a complex electric charge density. The test consists of transporting a two-dimensional liquid cylinder of diameter R within a unit-square domain with periodic boundary conditions. The electric charge density is initialized with a Gaussian distribution within the liquid phase, i.e.,

$$q_l(r, t = 0) = \frac{1}{\xi\sqrt{2\pi}} e^{-r^2/(2\xi^2)}, \tag{4.19}$$

where r is the radial coordinate and ξ is the standard deviation. In the gas phase, $q_g = 0$, creating a discontinuity at the gas-liquid interface. The parameters for this test are $R = 0.25$ and $\xi = 0.2$, which produces the electric charge density

distribution shown in Fig. 4.1. The charge density is transported with a uniform velocity $\boldsymbol{u} = [1, 0]^{\mathsf{T}}$.

Figure 4.2 shows a profile of the liquid electric charge density q_l after the cylinder has been transported for one flow-through time on a 50^2 mesh. The two results were computed using a first-order approximation of q and the second-order Taylor series approximations of q shown with Eq. 4.10. When the first-order approximation is used, errors appear near $r = 0.25$ where q is discontinuous and the geometric routines are used. The second-order reconstruction provides a significantly more accurate transported electric charge density.

Figure 4.3 shows an L_2 error for the electric charge density. This error is defined in general for a variable Θ as

$$L_2(\Theta) = \frac{\sqrt{\sum_{j=1}^{N} \left(\Theta_j - \Theta_j^e\right)^2}}{\sqrt{\sum_{j=1}^{N} \left(\Theta_j^e\right)^2}}, \tag{4.20}$$

where Θ_j^e is the exact value of Θ within the j^{th} computational cell and N is the number of cells in the domain. The error for the transport test shows first-order and second-order convergence when first- or second-order Taylor series approximation of q are used, respectively. These results highlight the importance of using a second-order reconstruction of q. Furthermore, the test demonstrates the ability of the proposed methodology to transport a discontinuous scalar with second-order accuracy. By construction, the method is expected to be conservative. Conservation of q was computed for this test case and found to be at machine precision, i.e., $\mathcal{O}(10^{-16})$.

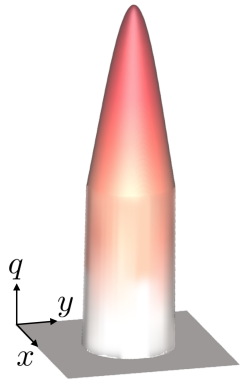


Figure 4.1: Initial electric charge density used in discontinuous scalar transport test.

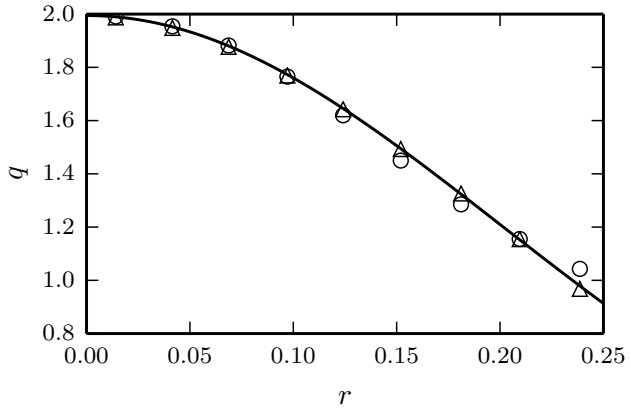


Figure 4.2: Transported electric charge density within liquid phase for discontinuous scalar transport test case using a 50^2 mesh. The reconstruction of q in Eq. 4.10 is varied from first order ($\circ$) to second order ($\triangle$). The exact solution is shown with the solid line.

4.3.2 Discontinuous scalar diffusion test

This test case assesses the implementation of the diffusion term in the electric charge conservation equation. The test problem consists of diffusing electric charge density within an liquid cylinder with a no-flux boundary condition at the phase interface. The two-dimensional test uses a unit-square domain. A liquid cylinder of radius R is placed at the center of the domain with an initial electric charge density given by a Gaussian distribution, i.e., Eq. 4.19.

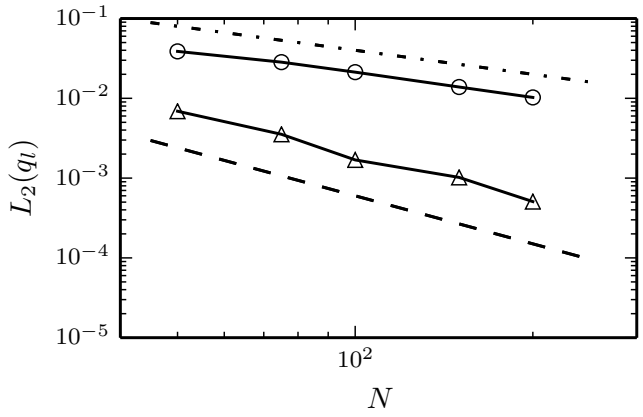


Figure 4.3: $L_2(q_l)$ error for discontinuous scalar transport test case. The reconstruction of q in Eq. 4.10 is varied from first order ($\circ$) to second order ($\triangle$). First- and second-order convergence shown with dash-dotted and dashed lines, respectively.

The transient problem in the liquid is described by Eq. 4.4 with $\boldsymbol{u}_l = 0$, $\boldsymbol{E}_l = 0$, and $S = 0$ leading to

$$\frac{\partial q_l}{\partial t} = \nabla \cdot (D_l \nabla q_l). \tag{4.21}$$

The previous equation, with the no-flux boundary condition at $r = R$ and a boundedness condition at $r = 0$, has solution (see Section 4.A)

$$q_l(r,t) = \frac{2}{R^2} \sum_{n=1}^{\infty} \left(\frac{J_0(\lambda_n r)}{J_0^2(\lambda_n R)} e^{-D_l \lambda_n^2 t} \int_0^R r' J_0(\lambda_n r') q_l(r', t=0) \mathrm{d}r' \right), \tag{4.22}$$

where J_η is the Bessel function of the first kind of order η and λ_n is the n^{th} root of J_1. Note that this equation is evaluated numerically using 200 terms of the infinite series and the integral is evaluated using the midpoint rule with 2000 intervals. These numbers where chosen to be large enough to not affect the results.

The parameters for this test are $\xi = 0.05$, $R = 0.25$, and $D_l = 0.01$. Figure 4.4 shows the temporal evolution of the electric charge density computed on a 50^2 mesh. Excellent agreement with the analytic solution is observed even at late times. Figure 4.5 shows the convergence of the L_2 error, which shows the expected

second-order rate. The conservation of q was verified and remained at machine precision for all the simulations.

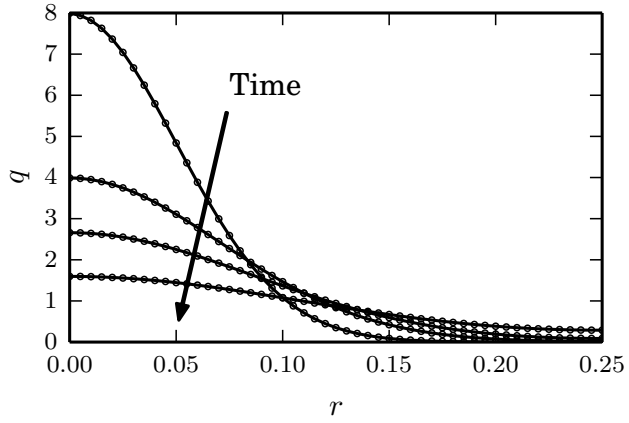


Figure 4.4: Solution for diffusion test case at $t = [0, 0.125, 0.25, 0.5]$. Analytic solution shown with solid line. Computed electric charge density with a 50^2 mesh shown with circles.

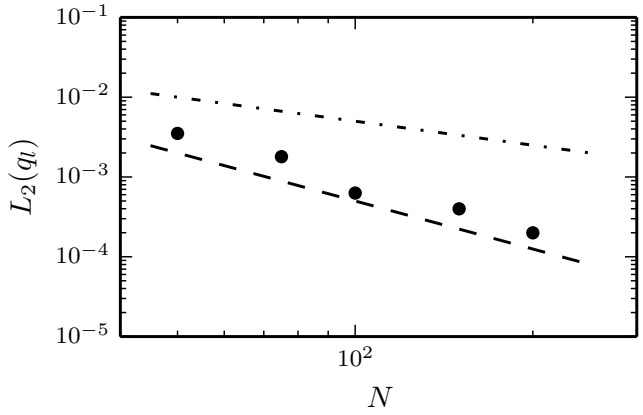


Figure 4.5: $L_2(q_l)$ error for diffusion test case. First- and second-order convergence shown with dash-dotted and dashed lines, respectively.

4.4 Conclusions

In this chapter a numerical framework is provided to solve conservation laws for discontinuous quantities. Convective transport is performed using the same frame-

work that is used to transport the VOF representation of the phase interface. This ensures discrete consistency between the two transport steps and avoids spurious over/undershoots and conservation errors. The method is tested using canonical test cases and demonstrates the expected second-order accuracy and discrete conservation.

4.A Analytic solution to diffusion in a cylinder

The solution to the diffusion of electric charge density within a liquid cylinder of diameter R is described by

$$\frac{\partial q_l(r,t)}{\partial t} = \nabla \cdot (D_l \nabla q_l(r,t)), \tag{4.23}$$

with boundary conditions

$$\left.\frac{\partial q_l}{\partial r}\right|_{r=0} = 0 \quad \text{and} \quad \left.\frac{\partial q_l}{\partial r}\right|_{r=R} = 0, \tag{4.24}$$

and initial condition $q_l(r, t = 0) = Q(r)$.

Assuming D_l is a constant parameter, this equation can be solved using separation of variables by assuming the solution has the form $q_l = \psi(r)\Gamma(t)$. Plugging this ansatz into Eq. 4.23 and writing the equation in cylindrical coordinates leads to

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial \psi(r)}{\partial r}\right) = \frac{1}{D_l}\frac{\partial \Gamma(t)}{\partial t}. \tag{4.25}$$

Since the left side of this equation only depends on r and the right side only depends on t, both sides are equal to a constant $-\lambda^2$. The temporal part is

$$\frac{\partial \Gamma}{\partial t} + D_l \lambda^2 \Gamma = 0 \tag{4.26}$$

with solution

$$\Gamma(t) = A_1 e^{-D_l \lambda^2 t}, \tag{4.27}$$

where A_1 is a constant.

The spatial part is

$$r^2 \frac{\partial^2 \psi}{\partial r^2} + r \frac{\partial \psi}{\partial r} + r^2 \lambda^2 \psi = 0, \tag{4.28}$$

which has solution

$$\psi(r) = A_2 J_0(\lambda r), \tag{4.29}$$

where A_2 is a constant and J_n is the Bessel function of the first kind of order n. Therefore we can write

$$q_l = A J_0(\lambda r) e^{-D_l \lambda^2 t}, \tag{4.30}$$

where $A = A_1 A_2$. Applying the no-flux boundary condition at $r = R$ is equivalent to enforcing

$$\left. \frac{\partial J_0(\lambda r)}{\partial r} \right|_{r=R} = J_1(\lambda R) = 0. \tag{4.31}$$

Which is satisfied if $\lambda_n R$ for $n = 1, \ldots, \infty$ are the roots of J_1. Therefore, Eq. 4.30 becomes

$$q_l = \sum_{n=1}^{\infty} A_n J_0(\lambda_n r) e^{-D_l \lambda_n^2 t}. \tag{4.32}$$

The initial condition is used to find the A_n constants. To begin we write

$$q_l(r, t = 0) = Q(r) = \sum_{n=1}^{\infty} A_n J_0(\lambda_n r). \tag{4.33}$$

Next, the previous equation is multiplied by $r J_0(\lambda_m r)$ and integrated leading to

$$\begin{aligned} &\int_0^R r' J_0(\lambda_m r') Q(r') \mathrm{d}r' \\ &\quad = \sum_{n=1}^{\infty} A_n \int_0^b r' J_0(\lambda_m r') J_0(\lambda_n r') \mathrm{d}r' \\ &\quad = \sum_{n=1}^{\infty} A_n \int_0^b r' J_0^2(\lambda_n r') \mathrm{d}r' \\ &\quad = \frac{b^2 A_n}{2} \big(J_0^2(\lambda_n R) - J_1^2(\lambda_n R) \big), \end{aligned} \tag{4.34}$$

where the orthogonality of the Bessel functions was employed. Noting that $J_1(\lambda_n R) = 0$ due to the no-flux boundary condition the constants are

$$A_n = \frac{2}{b^2 J_0^2(\lambda_n R)} \int_0^R r' J_0(\lambda_n r') Q(r') \mathrm{d}r'. \tag{4.35}$$

This leads to the analytic solution

$$q_l(r,t) = \frac{2}{b^2} \sum_{n=1}^{\infty} \frac{J_0(\lambda_n r)}{J_0^2(\lambda_n R)} e^{-D_l \lambda_n^2 t} \int_0^R r' J_0(\lambda_n r') Q(r') \mathrm{d}r'. \tag{4.36}$$

CHAPTER 5

HEIGHT FUNCTION INTERFACE CURVATURE CALCULATION

5.1 Introduction

Simulations of gas-liquid flows are often significantly influenced by the dynamics at the phase interface. For predictive simulations of relevant engineering flows, an accurate surface tension force is needed to avoid spurious curvature induced flows near the interface. The surface tension force is directly proportional to the interface curvature, and therefore the problem is reduced to computing an accurate interface curvature.

The height function method [38–40] is an approach for computing the interface curvature from an approximate representation of the phase interface and is commonly used in the context of volume-of-fluid (VOF) schemes. It has also been used successfully in the context of the accurate conservative level set (ACLS) [41], although this paper will assume that a VOF representation of the interface is available.

The VOF method is a popular technique to capture the location of the phase interface and has been used since the early 1970's when variants of the approach were introduced by DeBar [31], Hirt and Nichols [25], and Noh and Woodward [33]. VOF schemes store the ratio of liquid volume to cell volume, known as the liquid volume fraction α, within each computational cell, i.e.,

$$\alpha = \frac{1}{\mathcal{V}_{\mathrm{CV}}} \int_{\mathrm{CV}} f(\boldsymbol{x}, t) \, \mathrm{d}V, \tag{5.1}$$

where CV is a control volume (i.e., a computational cell) with volume $\mathcal{V}_{\mathrm{CV}}$ and f

is the indicator function, defined as

$$f(\boldsymbol{x},t) = \begin{cases} 1, & \text{if } \boldsymbol{x} \text{ is in the liquid at time } t, \\ 0, & \text{if } \boldsymbol{x} \text{ is in the gas at time } t. \end{cases} \tag{5.2}$$

VOF schemes differ in how the liquid volume fraction is transported. Early methods used flux splitting wherein one-dimensional transport steps are used successively [25]. Since then un-split schemes have been developed, such as the two-dimensional schemes of Pilliod and Puckett [26] and López et al. [2]. The extension of such methods to three-dimensions is not straight forward and the development of three-dimensional un-split schemes has only occurred recently by Hernández et al. [3], Le Chenadec and Pitsch [36], and the fully conservative formulation by Owkes and Desjardins [42].

All VOF methods require calculation of the interface curvature from the liquid volume fraction field. The interface curvature can be computed directly from the α-field by calculating the interface normal as $\boldsymbol{n} = -\nabla\alpha/|\nabla\alpha|$ and the interface curvature as $\kappa = -\nabla\cdot\boldsymbol{n}$. Because the α-field is based on the discontinuous indicator function f, the calculation can be improved by using a smoothed α-field [16]. Note that both approaches often do not converge with mesh refinement [83]. Alternatively, the height function approach has been shown to provide a converging interface curvature.

In its simplest form, the height function method consists of integrating the liquid volume fraction in the pseudo-normal direction in the cell of interest and neighboring cells, creating a collection of heights. The curvature is then calculated using finite difference operators on those heights. The pseudo-normal direction is the Cartesian direction x, y, or z with the largest component of the interface normal vector. Assuming the pseudo-normal direction is x for a computational cell

with Cartesian index i, j, k, the heights $\boldsymbol{h}$ are computed using

$$h_{j'k'} = \sum_{i'=i-(N_H-1)/2}^{i+(N_H-1)/2} \alpha_{i'j'k'} \Delta x \quad \text{for} \quad \begin{cases} j' = j-(N_N-1)/2, \ldots, j+(N_N-1)/2 \\ k' = k-(N_N-1)/2, \ldots, k+(N_N-1)/2 \end{cases}, \tag{5.3}$$

where $\alpha_{i'j'k'}$ is the liquid volume fraction within the i', j', k' cell. N_H controls the number of the cells in each height and values of $N_H = 3$, 5, and 7 have been considered in the literature [84–87]. N_N sets the number of neighboring heights that are computed. For second- and fourth-order methods, $N_N = 1$ and 2, respectively [87]. The curvature is calculated from the heights using finite difference operators such as the second order operator in two dimensions

$$\kappa = 2\frac{H_{yy}}{\left(1+H_y^2\right)^{3/2}}\left(\frac{\partial\alpha_{ijk}/\partial x}{|\partial\alpha_{ijk}/\partial x|}\right) \tag{5.4}$$

and three dimensions

$$\kappa = 2\frac{H_{yy}+H_{zz}+H_{yy}H_z^2+H_{zz}H_y^2-2H_{yz}H_yH_z}{\left(1+H_y^2+H_z^2\right)^{3/2}}\left(\frac{\partial\alpha_{ijk}/\partial x}{|\partial\alpha_{ijk}/\partial x|}\right), \tag{5.5}$$

with

$$H_y = \frac{h_{j+1,k}-h_{j-1,k}}{2\Delta y}, \tag{5.6a}$$

$$H_z = \frac{h_{j,k+1}-h_{j,k-1}}{2\Delta z}, \tag{5.6b}$$

$$H_{yy} = \frac{h_{j+1,k}-2h_{j,k}+h_{j-1,k}}{\Delta y^2}, \tag{5.6c}$$

$$H_{zz} = \frac{h_{j,k+1}-2h_{j,k}+h_{j,k-1}}{\Delta z^2}, \text{ and} \tag{5.6d}$$

$$H_{yz} = \frac{h_{j+1,k+1}-h_{j+1,k-1}-h_{j-1,k+1}+h_{j-1,k-1}}{2\Delta y\ 2\Delta z}. \tag{5.6e}$$

Figure 5.1 shows a two-dimensional example of the application of the height function method to compute κ_1 with $N_H = 5$. The three heights used to compute κ_1 are shown with solid lines. All of the heights are well defined since each height

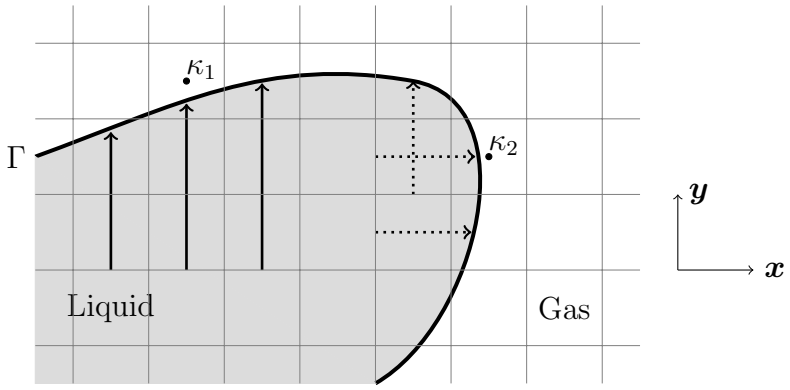


Figure 5.1: Example of heights (solid arrows) used to compute the interface curvature κ_1 and heights and widths (dotted arrows) used to compute κ_2.

is computed in a column that contains in a cell that is entirely liquid (i.e., $\alpha = 1$) and cell that is entirely gas (i.e., $\alpha = 0$). With the three heights, Eq. (5.4) can be used to compute the curvature.

When large interface curvature and under-resolved interface features exist, the number of well-defined heights are available to compute the curvature can be insufficient. For example, only two well-defined heights are available in the pseudo-normal direction to compute the curvature κ_2 in Fig. 5.1. A generalization of the height function method was introduced by Popinet [88] and combines heights computed in multiple directions to compute the curvature. For example, κ_2 in Fig. 5.1 can be computed by combining the two horizontal heights ("widths") and one vertical height. Using the heights from multiple directions, a parabola

$$p(t_1) = a_1 t_1^2 + a_2 t_1 + a_3 \tag{5.7}$$

in two dimensions or a paraboloid

$$p(t_1, t_2) = a_1 t_1^2 + a_2 t_2^2 + a_3 t_1 t_2 + a_4 t_1 + a_5 t_2 + a_6 \tag{5.8}$$

in three dimensions can be fit through the heights. In the previous equation t_1 and t_2 are the tangential components of an orthonormal coordinate system with

origin located at the interface barycenter of the cell where the curvature is being computed. In Popinet's formulation [88], the mean curvature is computed from the the parabola or paraboloid using

$$\kappa = 2\frac{a_1}{(1+a_2^2)^{3/2}} \tag{5.9}$$

or

$$\kappa = 2\frac{a_1(1+a_5^2) - a_3a_4a_5 + a_2(1+a_4^2)}{(1+a_4^2+a_5^2)^{3/2}}, \tag{5.10}$$

for two and three dimensions, respectively. For a highly under-resolved interface an adequate number of well-defined heights and widths may not be available. In such situations, Popinet [88] proposed fitting the parabola or paraboloid with interface barycenter within a cell and its nearest neighbors. Popinet's approach is hierarchical and uses the standard height function method if it is well defined. If not, then a paraboloid is fit through heights and widths. Finally a paraboloid fit through interface barycenters is used if an inadequate number of heights and widths are available.

The proposed method provides an alternative approach by constructing heights in an orthonormal coordinate system and using standard finite difference operators to compute the curvature. The scheme can be viewed as the standard height function method applied in a mesh-decoupled direction instead of the pseudo-normal, mesh-aligned direction. The approach remains robust for under-resolved interfaces and avoids the need for an additional method to compute the curvature for such interfaces. Furthermore, the proposed method could be used in the context of an unstructured mesh, although this application is not considered in this work.

This paper is organized as follows: Section 5.2 describes the details of the proposed scheme, Sections 5.3 and 5.4 provides verification and validation results obtained with the approach, and finally conclusions are drawn in Section 5.5.

5.2 Methodology

The proposed method computes heights within columns not aligned with the computational mesh but rather aligned with the interface normal vector, which is assumed known as a prerequisite. Since the columns are not tied to the mesh, as in the traditional height function method, there is flexibility in the column's definition. We parameterize the column geometry using column length L, width W, and spacing S_W as shown in Fig. 5.2. In three dimensions, the column is also parameterized using the column depth D and spacing S_D. The number of columns could be varied, but in this work three and nine columns are used in two and three dimensions, respectively. This is the minimum number of columns for a second order method.

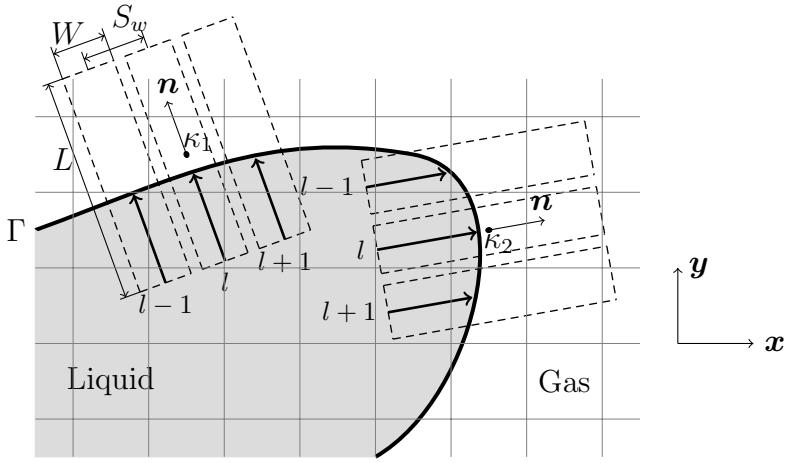


Figure 5.2: Example of mesh-decoupled columns and heights used to compute the interface curvatures κ_1 and κ_2.

Within the n^{th} column associated to the i^{th} computational cell, $\mathcal{C}_{i,n}$, the height is calculated using

$$h_{i,n} = \frac{1}{WD} \int_{\mathcal{C}_{i,n}} f(\boldsymbol{x}, t) \, \mathrm{d}V. \tag{5.11}$$

The integral in the previous equation depends on the distribution of liquid within each computational cell. Therefore, the interface location must be reconstructed

from the liquid volume fraction field, forming an explicit definition of the gas and liquid phases. In the proposed method, we use the piecewise linear interface calculation (PLIC) [31,34,35] representation of the interface within each computational cell. PLIC approximates the interface using a line in two dimensions and a plane in three dimensions. This linear reconstruction is constrained such that it is perpendicular to the interface normal vector and the amount of liquid under the line (plane) is consistent with the liquid volume fraction α [70].

The main difficulty in evaluating the integrals in Eq. (5.11) lies in the mismatch between the geometry of the columns and the geometry of the mesh. We use a computational geometry toolbox to evaluate the integrals. The geometry toolbox performs the following steps:

1. Each column is partitioned into a collection of simplices (triangles in two dimensions and tetrahedra in three dimensions). The minimum number of simplices needed to represent a column is two in two dimensions and five in three dimensions, as shown in Figs. 5.3 and 5.4, respectively.
2. Each simplex is then cut by the mesh and the remaining piece are partitioned into new simplices.
3. The simplices from the previous step are cut by the PLIC representation of the interface within each cell and the volume of liquid is computed.
4. The integral in Eq. (5.11) is evaluated by combining the liquid volumes from all the simplices.

The computational toolbox used to cut the simplices by the mesh and the PLIC interface consists of identifying how a plane cuts a simplex and partitioning the simplex into new simplices localized to one side of the plane. Recursive cutting

results in simplices that are within a single phase. Additional details of the cutting routine is provided by Owkes and Desjardins [42], where the cutting routines are used to transport the liquid volume fraction α in a VOF scheme. Similar routines may be available in a other geometric VOF methods allowing for a straightforward implementation of the mesh-decoupled height function method.

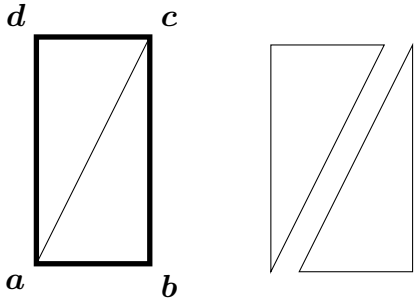


Figure 5.3: Partitioning of a two-dimensional column ***abcd*** using simplices ***abc*** and ***acd***.

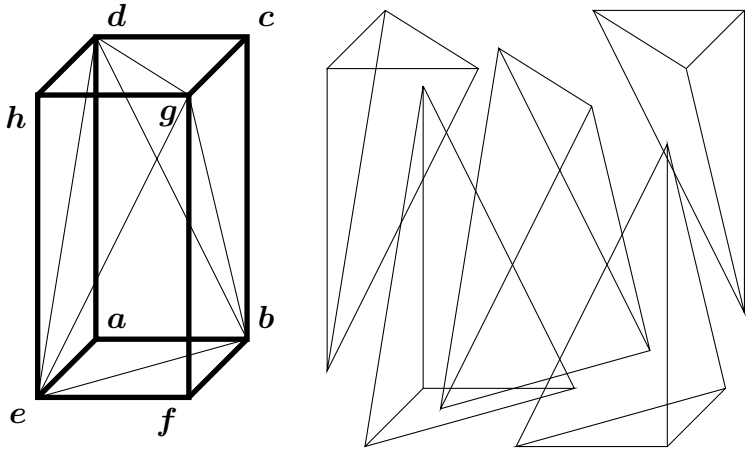


Figure 5.4: Partitioning of a three-dimensional column ***abcdefgh*** using five simplices ***befg***, ***abde***, ***bcdg***, ***degh***, and ***bdeg***.

Once the integrals in Eq. (5.11) are evaluated, the finite difference operators

Eq. (5.4) and (5.5) are used to compute the curvature with

$$H_y = \frac{h_{l+1,m} - h_{l-1,m}}{2S_W}, \tag{5.12a}$$

$$H_z = \frac{h_{l,m+1} - h_{l,m-1}}{2S_D}, \tag{5.12b}$$

$$H_{yy} = \frac{h_{l+1,m} - 2h_{l,m} + h_{l-1,m}}{S_W^2}, \tag{5.12c}$$

$$H_{zz} = \frac{h_{l,m+1} - 2h_{l,m} + h_{l,m-1}}{S_D^2}, \text{ and} \tag{5.12d}$$

$$H_{yz} = \frac{h_{l+1,m+1} - h_{l+1,m-1} - h_{l-1,m+1} + h_{l-1,m-1}}{4S_W S_D}. \tag{5.12e}$$

Figure 5.5 shows how the nine columns that are used to computed the curvature in three dimensions are defined.

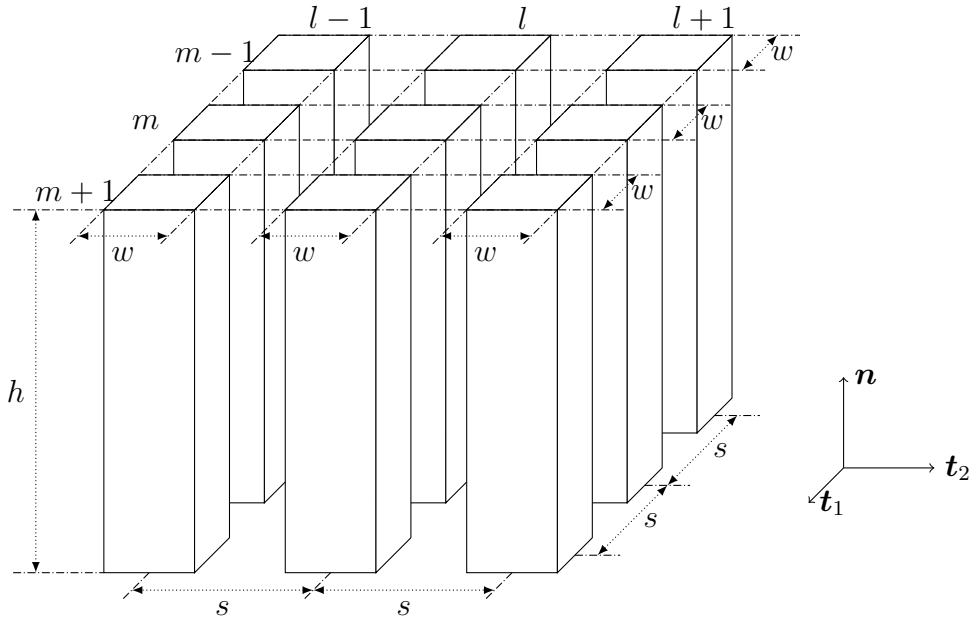


Figure 5.5: The nine columns used to compute curvature in three-dimensions. Column width, height, and spacing are described by w, h, and s, respectively. The columns are defined in a ortho-normal coordinate system based on the interface normal vector $\boldsymbol{n}$ and two tangential vectors t_1 and t_2.

5.3 Verification tests

In this section, results obtained with the proposed method are provided and discussed. Two different geometries are considered, namely a two-dimensional circle test case and a three-dimensional sphere test case.

5.3.1 Circle test case

This test consists of computing the curvature of a circle and comparing to the expected analytical value. The simple geometry of the circle allows for an exact α-field and normal vectors to be specified, therefore eliminating these sources of error from the analysis. Note that care is required when defining the interface normal vector, which is defined within the i^{th} computational cell using

$$\boldsymbol{n}_i = \frac{\int_{I \in \text{CV}_i} \boldsymbol{n}(\boldsymbol{x}, t)\, \mathrm{d}S}{\left| \int_{I \in \text{CV}_i} \boldsymbol{n}(\boldsymbol{x}, t)\, \mathrm{d}S \right|}, \tag{5.13}$$

where $I \in \text{CV}_i$ is the interface within the i^{th} computational cell and $\boldsymbol{n}$ is the interface normal vector. Simulations are performed with a circle of radius 0.2 within a unit square domain centered on the origin and discretized with a uniform Cartesian mesh with cell size Δx. The circle is positioned randomly near the center of the domain to avoid alignment with the mesh, thereby more closely mimicking realistic situations. Each random position is created by computing the center of the circle $\boldsymbol{C}_o$ using

$$\boldsymbol{C}_o = [R_1 \Delta x, R_2 \Delta y]^{\mathsf{T}}, \tag{5.14}$$

where R_1 and R_2 are uniform random variables on the interval $[-1, 1]$. For each mesh level, 50 simulations are performed with different random circle positions.

Two errors are computed, namely the L_2 and L_∞ errors defined using

$$L_2 = \frac{\sqrt{\sum_{i=1}^{N_{\text{cells}}} \left(\kappa_i - \kappa_{E,i}\right)^2}}{\sqrt{\sum_{i=1}^{N_{\text{cells}}} \kappa_{E,i}^2}}, \text{ and} \tag{5.15}$$

$$L_\infty = \max_{i=1\ldots N_{\text{cells}}} |\kappa_i - \kappa_{E,i}|, \tag{5.16}$$

where κ_i and $\kappa_{E,i}$ are the computed and exact curvatures within the i^{th} computational cell. Note that only cells that contain an interface are included in the summation and maximum operators in Eqs. (5.15) and (5.16).

Two variants of the proposed scheme are tested and results are presented below. The first uses the mesh-decoupled height functions to compute the curvature in all computational cells that contain the interface. The second uses only the mesh-decoupled height functions in cells where the standard height function method is ill-posed.

Mesh-decoupled height function method

The proposed scheme is tested by computing the curvature in all computational cells that contain the interface and comparing to the exact circle curvature. Figure 5.6 provides the L_2 and L_∞ errors on different meshes and allows the convergence properties to be analyzed. The errors are plotted as a function of N/D, the number of grid points across the diameter of the circle. In the figure, three different stencil sizes are studied including a small stencil $W = S_W = \Delta x$, a medium stencil $W = S_W = 2\Delta x$, and a large stencil $W = S_W = 3\Delta x$. A column length of $L = 2.5\Delta x$ is used in all of the variants and is found to be large enough to not affect the results. When the small stencil is used, little convergence is observed. However, for both the medium and large stencils, second-order convergence is observed in the L_2 error and convergence between first- and second-order is observed

for L_∞. However, both errors eventually plateau and fail to converge on the finest meshes.

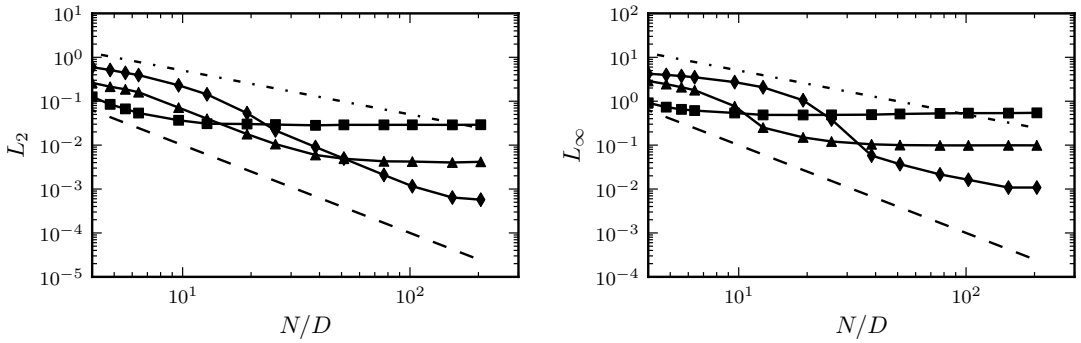


Figure 5.6: Convergence of L_2 and L_∞ curvature error with mesh refinement for the circle test case. Squares, triangles, and diamonds indicate stencils with $W = S_W = \Delta x$, $2\Delta x$, and $3\Delta x$, respectively. The dash dotted and dashed lines show first- and second-order convergence for comparison.

Instead of using the computational geometry toolbox described earlier to evaluate the integrals in Eq. (5.11), analytic relations can be used for a circular interface. This approach provides an opportunity to analyze the effect of the PLIC reconstruction on the results. Figure 5.7 shows the convergence of the L_2 and L_∞ errors when analytic expressions are used instead of the geometric routines that rely on the PLIC reconstruction of the interface. Both errors show second order convergence for all of the stencils. The small stencil produces the smallest error. This result is not surprising since the error is most likely dominated by the second-order finite difference operator which has an error that scales with $\mathcal{O}((S_W)^2)$.

The significant dependence of the curvature errors on the PLIC reconstruction is likely due to the discontinuous nature of the gas-liquid interface at cell faces. These results indicate that there is significant potential to generalize the proposed scheme beyond a PLIC reconstruction and use a higher order reconstruction of the interface [89], although this is beyond the scope of this paper.

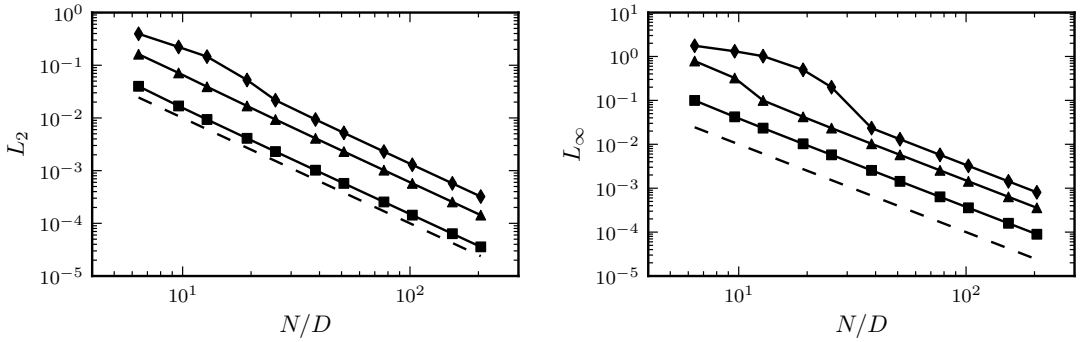
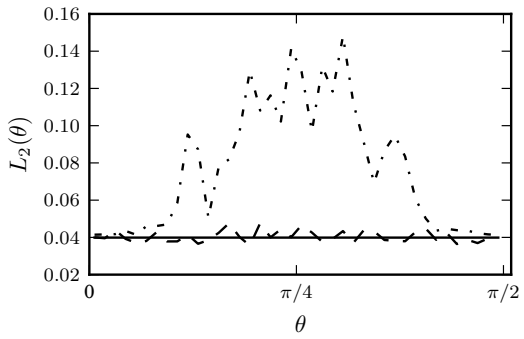
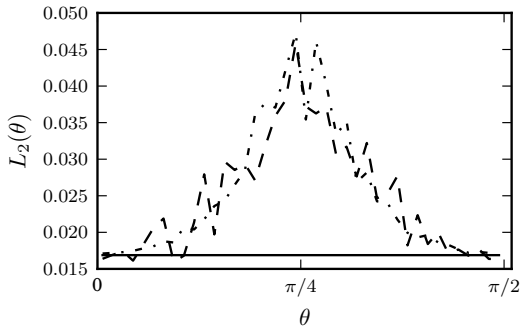


Figure 5.7: Convergence of L_2 and L_∞ curvature errors with mesh refinement for the circle test case with analytic evaluation of the integrals in Eq. (5.11). Squares, triangles, and diamonds indicate stencils with $W = S_W = \Delta x$, $2\Delta x$, and $3\Delta x$, respectively. The dashed line shows second order convergence for comparison.

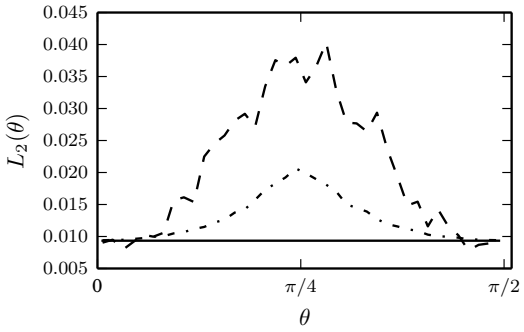
The angular error distribution is provided in Fig. 5.8, which shows the curvature errors using 16^2, 24^2, and 32^2 meshes leading to resolutions of $N/D = 6.4$, 9.6, and 12.8 respectively. To produce the data for each curve, 50 simulations are again conducted with random circle positions. The curvature in each computational cell that contain an interface is added to one of 40 bins based on the associated angle θ with respect to the horizontal axis. In each bin, an $L_2(\theta)$ error is computed using Eq. (5.15). The mesh-decoupled simulations are performed with $W = S_W = \Delta x$ and $L = 2.5\Delta x$. For the poorly-resolved circle with $N/D = 6.4$, the mesh-decoupled height function method produces lower errors than the standard height function approach. For the moderately-resolved droplet with $N/D = 9.6$, the mesh-decoupled method and the standard height function method compute curvatures with similar errors. On the fine mesh with $N/D = 12.8$, the standard height function method performs better than the mesh-decoupled height function method. The maximum errors occur at 45° from the horizontal and vertical axes as this is the location where the standard mesh-aligned height function method is unable to find well-defined heights and where the PLIC reconstruction exhibits the largest cell-to-cell discontinuities.



(a) $N/D = 6.4$



(b) $N/D = 9.6$



(c) $N/D = 12.8$

Figure 5.8: Dependency of the L_2 curvature error on angular position. Results computed with mesh-decoupled heights (dashed), standard mesh-aligned heights (dash dotted), and analytical expressions (solid).

Combined mesh-decoupled and mesh-aligned method

The mesh-decoupled height function method only converges when large stencils ($W = S_W = 2\Delta x$ or $3\Delta x$) are used and even fails to converge on fine meshes for these stencils (see Fig. 5.6). Such large stencils make the method computationally expensive. Interestingly, for under-resolved interface features the curvature has a relatively low error, especially when a small stencil ($W = S_W = \Delta x$) is used. Furthermore, under-resolved features are often problematic for the standard height function method that uses heights aligned with one of the coordinate axis. This is shown in Fig. 5.9 where the mesh-decoupled and standard height function methods are used to compute to compute the curvature of a circle. Therefore, there is potential to combine the two methods by using the standard height function method when all the heights are well defined and the mesh-decoupled method where the interface is under resolved.

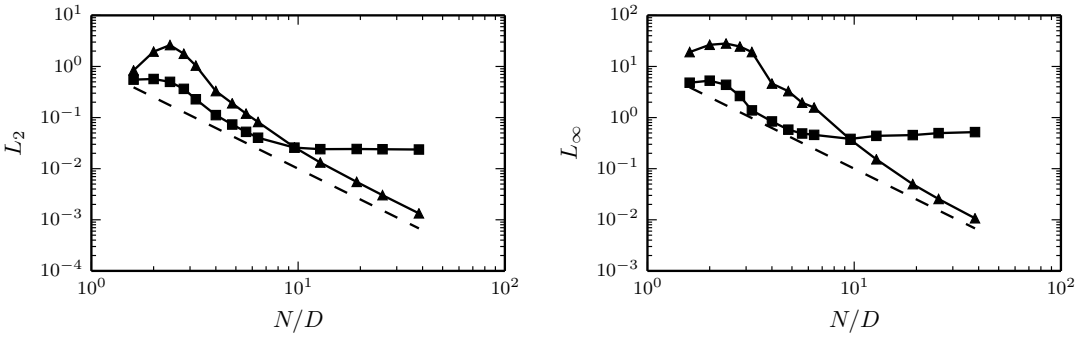


Figure 5.9: Convergence of L_2 and L_∞ curvature errors with mesh refinement for the circle test case. Squares and triangles indicate the mesh-decoupled and standard height function methods, respectively. The dashed line shows second order convergence for comparison.

The proposed combined method is based on the following procedure:

1. compute the pseudo-normal direction x, y, or z,

2. determine if the heights are well defined by checking if the maximum and minimum values of α within each column is 1 and 0, respectively, e.g., if $\max_{i'=i-(N_H-1)/2}^{i+(N_H-1)/2}(\alpha_{i'j'k'}) = 1$ and $\min_{i'=i-(N_H-1)/2}^{i+(N_H-1)/2}(\alpha_{i'j'k'}) = 0$ then $H_{j'k'}$ is well defined,

3. if all the heights are well defined then the standard height function approach is used, if not the mesh-decoupled height function method is used with $W = S_W = \Delta x$ and $L = 2.5\Delta x$.

Note that the formulas in step 2 are simple and more detailed methods to determine if a column is well defined exist, see for example [87,88]. These methods use a search algorithm that progresses outward from the cell of interest until an interface is found or the maximum column length is reached. However, the simple formulas allow for the column to be efficiently classified as well defined or not. If the column is not well defined, then the curvature is computed with the mesh-decoupled method.

This methodology is used to compute the curvature of the circle test case with $N_H = 7$. Figure 5.10 shows the L_2 and L_∞ errors for both the standard mesh-aligned height function method and the combined method. On the coarsest meshes the combined method produces a more accurate curvature. On finer meshes, the number of cell without well-defined heights decreases and the two approaches become indistinguishable with the expected second order convergence. This results highlights the usefulness of the proposed combined method for computing curvatures of under-resolved and well-resolved interfaces.

Figure 5.10 also compares the proposed scheme to our implementation of the method proposed by Popinet [88] wherein heights from multiple directions are combined to compute the curvature of under-resolved interfaces. We find that our

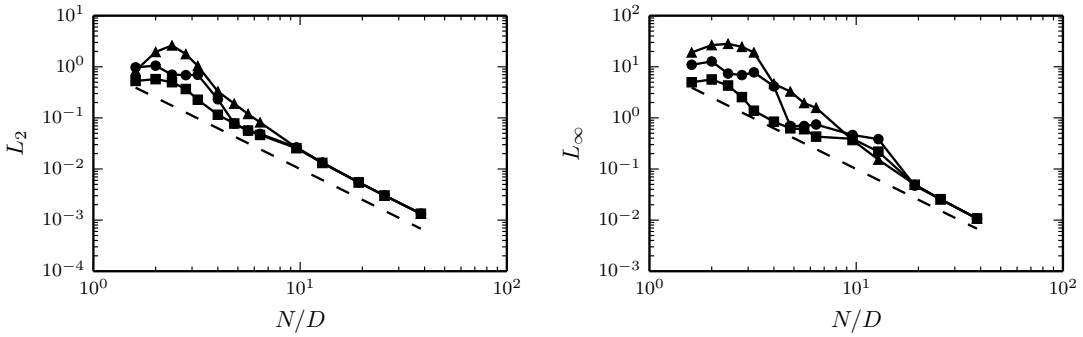


Figure 5.10: Convergence of L_2 and L_∞ curvature error with mesh refinement for the circle test case. Triangles, squares, and circles correspond to the standard height function method, the combined method, and our implementation of the method of Popinet [88], respectively. The dashed line shows second order convergence for comparison.

proposed scheme computes curvatures with similar errors except for interfaces that are significantly under resolved. In these situations, the proposed scheme seems to provide curvatures with lower errors than the method proposed by Popinet [88].

For under-resolved interfaces, combining heights and widths does not guarantee an accurate curvature calculation. This is because when heights and widths are combined, the physical location of interface represented by the heights or widths is important since they must exist in the same coordinate system. Therefore a physically relevant origin must be found to define the heights and widths. Popinet [88] suggests finding a cell that is completely full of liquid to use as the physical origin of each column. In the proposed method, only the difference in heights is used to compute the curvature and the physical location is irrelevant. Therefore, the method does not require well-defined heights to compute a curvature, although an error is introduced if well-defined heights are not available. For example, Fig. 5.11 shows an under-resolved droplet where no well-defined heights or widths can be defined since none of the computational cells are completely full. For these under-resolved interfaces, Popinet [88] suggests using the barycenter of interface within

the cell of interest and neighboring cells instead of ill-defined heights. The proposed scheme is able to compute the curvature of these under-resolved interfaces as shown in Fig. 5.12. In the calculation on the mesh-decoupled columns, the center height is well defined. The other two heights are not well defined but provide a good estimate for the interface location within each column. Note that the size of the columns will affect the curvature calculation when well-defined heights are not available, but we found that the curvature remains accurate and robust with $W = S_W = \Delta x$ and $L = 2.5\Delta x$.

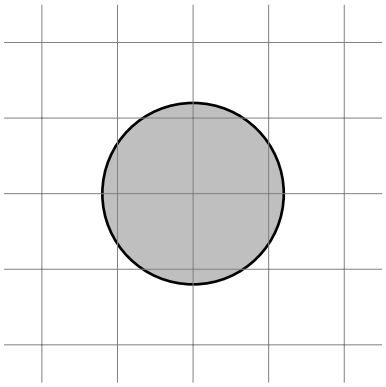


Figure 5.11: Example of a droplet where heights and widths are not well defined.

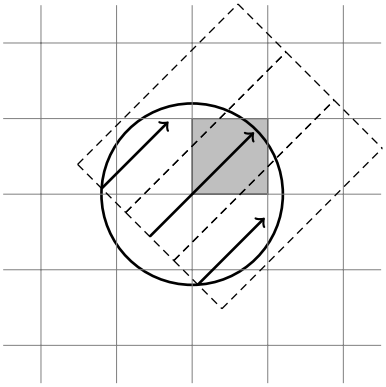


Figure 5.12: Example of how curvature within the shaded cell is computed using the proposed scheme.

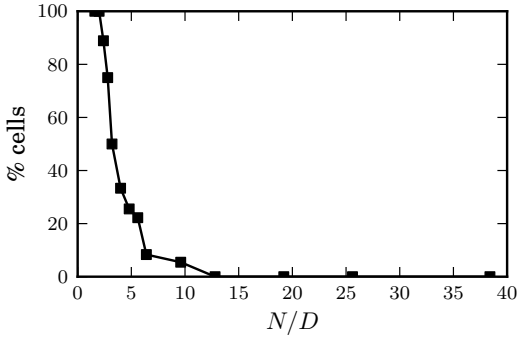


Figure 5.13: Percentage of cells in simulations of circle test case in which the standard height function method fails to have well-defined heights.

For a resolved interface, all the methods collapse onto one another since all ap-

proaches use the standard height function method in this situation. This is evident in Fig. 5.10 where the errors collapse when N/D is greater than approximately 10. Figure 5.13 shows the percentage of cells where the standard height function fails to have well-defined heights and either the mesh-decoupled method or the method proposed by Popinet [88] is used. When N/D is greater than 10, all of the cells have well-defined heights.

5.3.2 Sphere test case

This test case provides information on how the proposed scheme performs in three dimensions. The test consists of a sphere of radius 0.2 within a unit cubic domain centered on the origin and discretized using a uniform Cartesian mesh. The sphere is randomly positioned near the center of the domain with the same method we use for the randomly positioned circle with $\boldsymbol{C}_0 = [R_1 x, R_2 y, R_3 z]^\mathsf{T}$. Ten random sphere positions are used on each mesh level.

Figure 5.14 shows the L_2 and L_∞ errors for the sphere test. Results are calculated using the standard height function method, the combined method, and our implementation of Popinet's method [88]. Both the L_2 and L_∞ errors are significantly reduced when the combined method is used compared to the standard height function method. The proposed method also computes more accurate curvatures than the method of Popinet [88] on the coarsest meshes. Figure 5.15 shows the percentage of cells without well-defined heights, which are therefore treated with either the mesh-decoupled method or the method of Popinet [88]. On the finest mesh with 50 cells across the sphere diameter, the standard height function method is well defined and all the methods collapse onto one another.

The results shown in Fig. 5.14 was computed with $N_H = 9$. This larger stencil was needed to ensure the standard height function method is well defined for all computational cells with interface on the finer meshes. When $N_H = 7$, the standard height function method remains ill-posed for a few cells even on the finest meshes and the L_∞ error fails to converge as shown in Fig. 5.16.

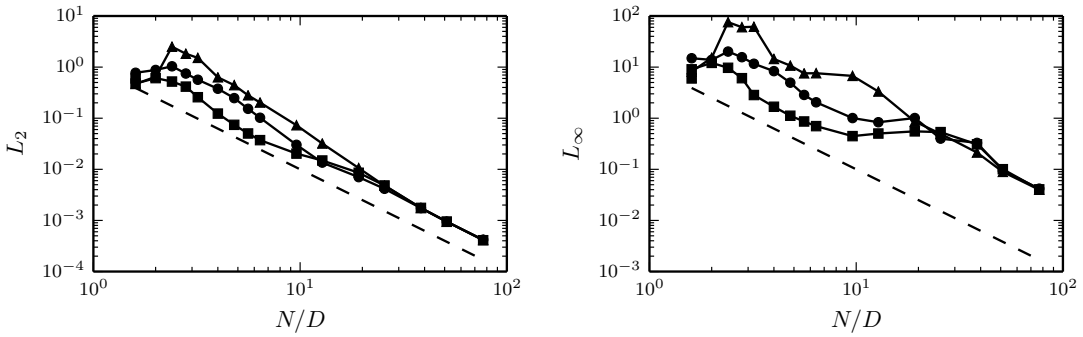


Figure 5.14: Convergence of L_2 and L_∞ curvature error with mesh refinement for the sphere test case with $N_H = 9$. Triangle, squares, and circles indicate results obtained with the standard height function method, the combined method, and our implementation of the method proposed by Popinet [88]. The dashed line shows second order convergence for comparison.

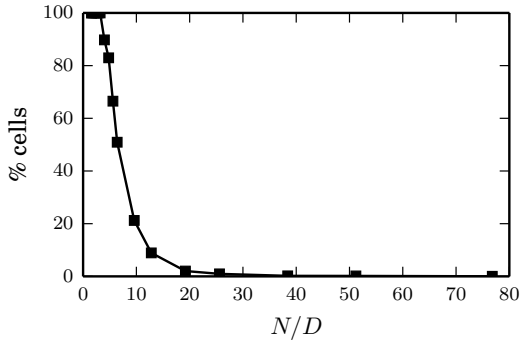


Figure 5.15: Percentage of cells in simulations of sphere test case in which the standard height function method fails to have well-defined heights.

Figure 5.17 shows the relative curvature error, $|\kappa_i - \kappa_{E,i}|/\kappa_{E,i}$, on the surface of a sphere represented with $N/D = 6.4$. The standard height function method has largest errors in cells where the interface is not aligned with one of the Cartesian

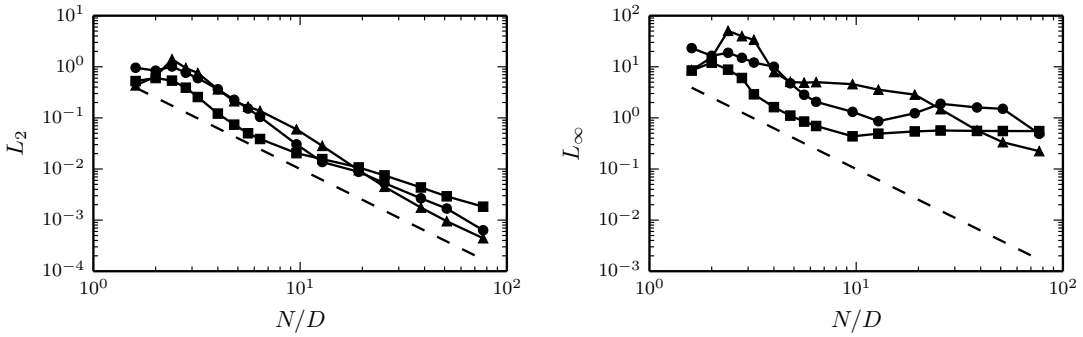
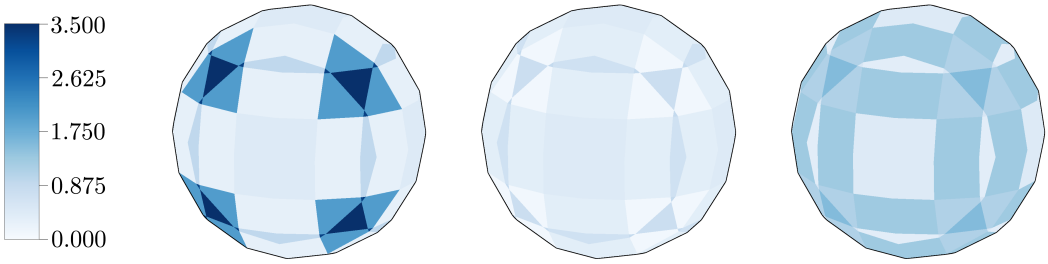


Figure 5.16: Convergence of L_2 and L_∞ curvature error with mesh refinement for the sphere test case with $N_H = 7$. Triangle, squares, and circles indicate results obtained with the standard height function method, the combined method, and our implementation of the method proposed by Popinet [88]. The dashed line shows second order convergence for comparison.

directions. The errors in these cells is reduced when either the combined method or the scheme of Popinet [88] is used.



(b) Standard (c) Combined (d) Popinet [88]

Figure 5.17: Curvature error on surface of sphere test case for different methods. Simulations are performed with $N/D = 6.4$.

5.4 Validation tests

Even though the curvature computed with the combined method has been shown to have a small error, the curvature errors could induce large spurious velocities. Therefore, spurious-currents [90] and standing-wave [58] test cases are used to assess curvatures in surface tension dominated flows. The test cases use the NGA

flow solver [28] with a conservative, un-split, geometric volume-of-fluid (VOF) method [42]. This section contains an overview of the numerical methods used to solve the Navier-Stokes equations followed by details of the test cases.

5.4.1 Solution of the Navier-Stokes equations

The incompressible form of the Navier-Stokes equations is used to describe the gas-liquid flow, which can be written as

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) = -\nabla p + \nabla \cdot \left(\mu \left[\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^{\mathsf{T}}\right]\right) + \rho \boldsymbol{g}, \tag{5.17}$$

where $\boldsymbol{u}$ is the velocity, t is time, ρ is the density, p is the pressure, μ is the dynamic viscosity, and $\boldsymbol{g}$ is the gravitational acceleration. The continuity equation can be written as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = \frac{\partial \rho}{\partial t} + \boldsymbol{u} \cdot \nabla \rho = 0, \tag{5.18}$$

where the incompressibility constraint $\nabla \cdot \boldsymbol{u} = 0$ is used.

The gas and liquid phases, indicated with the subscripts g and l, are separated by the interface denoted I. Discontinuities at the interface are written using the jump, i.e., $[\rho]_I = \rho_l - \rho_g$ and $[\mu]_I = \mu_l - \mu_g$ for the density and dynamic viscosity, respectively. The pressure is discontinuous and can be written as

$$[p]_I = \sigma \kappa + 2[\mu]_I \boldsymbol{n}^{\mathsf{T}} \cdot \nabla \boldsymbol{u} \cdot \boldsymbol{n}, \tag{5.19}$$

where σ is the surface tension coefficient. In the absence of phase change, the velocity is continuous across the interface, i.e., $[\boldsymbol{u}]_I = 0$.

NGA [28] is used to solve the variable density, low Mach number Navier-Stokes equations. NGA is formulated using high-order conservative finite difference methods staggered in both space and time, which have been shown to be well suited

for simulations of turbulent flows [56], a common occurrence in multiphase simulations. Note that second-order finite difference operators are used in this work since they greatly simplify the multiphase implementation.

The large discontinuity in density and the pressure jump due to surface tension that occur at the interface are handled using the ghost fluid method (GFM) [17]. The discontinuous viscosity is approximated using the height fraction method [85].

The interface is transported using an un-split, conservative, geometric volume-of-fluid scheme [42]. The approach leverages the computational geometry toolbox to systematically compute conservative liquid volume fraction fluxes. The resulting method is second order, discretely bounded, and conservative.

Consistency between interface and momentum transport has been shown to be important for simulations with large density ratios [91]. Therefore, we employ the un-split, conservative, geometric routines [42] to consistently convect the liquid volume fraction and momentum near the interface. The approach is similar to the semi-Lagrangian method of Le Chenadec and Pitsch [36], but uses fluxes that have been corrected to respect the solenoidal condition, forming a scheme that conserves both mass and momentum to machine precision.

5.4.2 Spurious-currents test case

This test consists of simulating a two-dimensional drop of diameter $D = 0.4$ centered within a computational domain with width and height equal to unity [90]. The surface tension coefficient $\sigma = 1$ and the viscosity ratio is unity with $\mu_l = \mu_g = 0.1$. The density ratio is set to unity implying $\rho_l = \rho_g = \rho$, where ρ is a free parameter that is used to set the Laplace number $\mathrm{La} = 1/(\mathrm{Oh})^2 = \rho\sigma D/\mu^2$,

where Oh is the Ohnesorge number. The spurious velocities due to errors in the curvature calculation that propagate through the surface tension force are measured with the capillary number $\mathrm{Ca} = |\boldsymbol{u}|_{\max}\mu/\sigma$. The capillary number is evaluated after a non-dimensional time of $t\sigma/(\mu D) = 250$.

Simulation with varying Laplace numbers are conducted on 32^2 and 64^2 meshes. Table 5.1 shows the capillary number, which remains small for all Laplace numbers and both meshes. The effect of the mesh on the spurious currents is also investigated by fixing the Laplace number at 12,000 and varying the mesh from 16^2 to 128^2. Table 5.2 and Fig. 5.18 provide the results, which show second-order convergence is obtained under mesh refinement and low capillary numbers are found for all meshes. Variability in the results in Tables 5.1 and 5.2 is likely due to the oscillations in the capillary number with time. Figure 5.19 shows the capillary number as a function of time for the simulation with Laplace number of 12,000 and a 32^2 mesh. Near a non-dimensional time of 250, the capillary number varies by roughly an order of magnitude due to temporal oscillations. Even with this variability all the capillary numbers are small indicating that curvature errors produce minimal spurious velocities.

Table 5.1 also provides a comparison of the combined method and the method proposed by Popinet [88]. The capillary numbers have similar magnitudes although the combined method has on average smaller capillary numbers. The combined method is roughly 10% more expensive. This is because evaluating the integrals in Eq. (5.11) using the computational geometry toolbox is computationally expensive. Since most cells in realistic simulations will likely have well-defined heights and the standard height function method will be used, the additional cost is not expected to impact simulation times significantly.

Table 5.1: Capillary number and time per time-step for various Laplace numbers. Simulations use 32^2 and 64^2 meshes and are performed with the combined method and our implementation of Popinet's method [88].

Laplace number	Capillary number				Time/step (s)	
	32^2 mesh		64^2 mesh		64^2 mesh	
	Combined	Popinet	Combined	Popinet	Combined	Popinet
12	1.65e-07	1.61e-05	9.27e-08	4.20e-05	0.159	0.131
120	4.59e-07	4.18e-05	2.34e-07	6.15e-05	0.148	0.126
1200	7.80e-07	3.57e-08	7.39e-08	2.14e-09	0.152	0.125
12000	2.18e-06	3.32e-06	1.52e-06	2.20e-06	0.162	0.137
120000	3.00e-06	2.03e-05	6.62e-06	1.09e-05	0.193	0.173
1200000	9.17e-07	9.93e-06	7.76e-06	3.13e-05	0.224	0.204

Table 5.2: Capillary number for Laplace number of 12,000 on various meshes using the combined method.

Laplace number	Capillary number			
	16^2 Mesh	32^2 Mesh	64^2 Mesh	128^2 Mesh
12,000	2.877e-06	2.898e-06	2.325e-07	5.905e-08

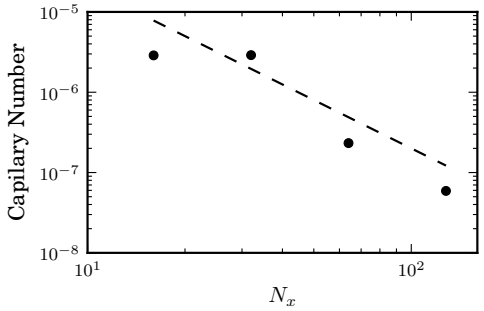


Figure 5.18: Convergence of capillary number for simulations with Laplace number of 12,000. Dashed line shows second-order convergence.

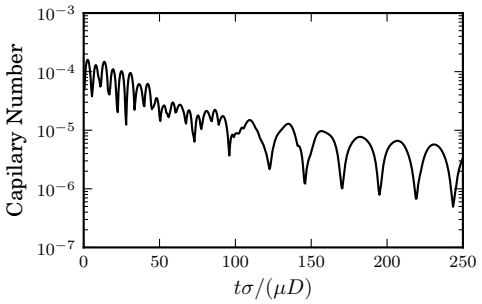


Figure 5.19: Time evolution of capillary number for simulation with Laplace number of 12,000 and 32^2 mesh.

5.4.3 Standing-wave test case

This test case consists of the viscous damping of a surface wave and depends on a significant interaction between surface tension and viscous forces. A two-dimensional domain of $[0, 2\pi]^2$ is used with periodic boundary conditions in the x-direction and slip conditions on the top and bottom. Two fluids are separated by a flat interface initially perturbed by a sinusoidal wave specified using $y = A_0 \cos(2\pi x/\lambda) + \pi$, where $\lambda = 2\pi$ is the perturbation wavelength and $A_0 = 0.01\lambda$ is the initial wave amplitude. Time is non-dimensionalized with the inviscid oscillation frequency $\omega_0 = \sqrt{\sigma/(\rho_l + \rho_g)}$.

Simulations are performed until a non-dimensional time of $\omega_0 t = 20$ following previous studies [28, 41, 59, 92]. Three meshes are considered, namely 16^2, 32^2, and 64^2 meshes, and the density ratio is set to either 1 with $\rho_l = \rho_g = 1$ or 1000 with $\rho_l = 1000$ and $\rho_g = 1$. The non-dimensional surface tension coefficient is $\sigma = 2$. The non-dimensional kinematic viscosity is set to $\nu = 0.064720863$ when the density ratio $\rho_l/\rho_g = 1$ and $\nu = 0.0064720863$ when $\rho_l/\rho_g = 1000$.

Results are shown in Figs. 5.20 and 5.21 for the two density ratios. The wave amplitude A normalized by the wavelength as a function of time and the error between the computation and theoretical solution A_T derived by Prosperetti [58] is plotted. For both density ratios the computed wave amplitude agrees well with the theoretical prediction. Furthermore, the amplitude error converges under mesh refinement, shown by Fig. 5.22, suggesting the wave physics are properly captured.

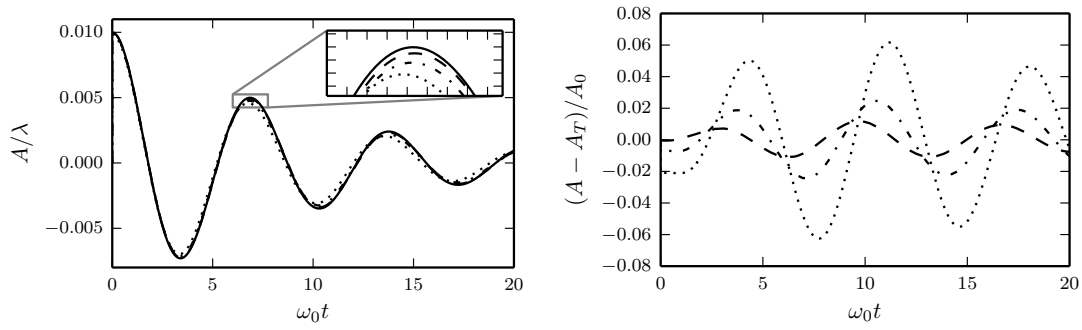


Figure 5.20: Standing wave test case with $\rho_l/\rho_g = 1$. Solution and error using mesh with 16^2, 32^2, and 64^2 cells shown with dotted, dash dotted, and dashed lines, respectively. Theoretical solution [58] shown with solid line.

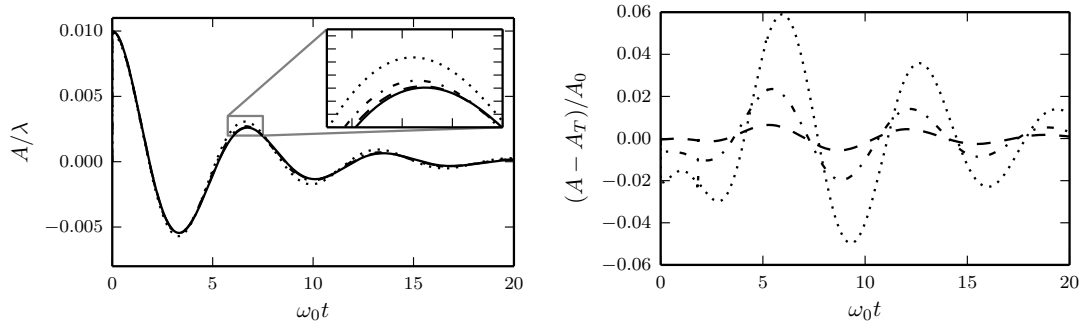


Figure 5.21: Standing wave test case with $\rho_l/\rho_g = 1000$. Solution and error using mesh with 16^2, 32^2, and 64^2 cells shown with dotted, dash dotted, and dashed lines, respectively. Theoretical solution [58] shown with solid line.

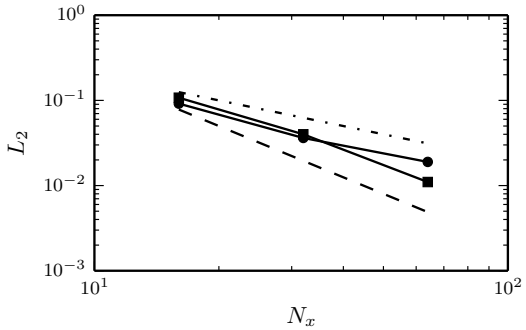


Figure 5.22: Convergence of L_2 amplitude error for standing wave test case. Circles and squares indicate density ratios of 1 and 1000, respectively. First- and second-order convergence shown with dash dotted and dashed lines.

5.5 Conclusions

In this paper, we propose and evaluate a mesh-decoupled height function method. The scheme leverages a computational geometry toolbox to evaluate height functions within columns that are not aligned with the computational mesh. The method is used to compute the curvature of a randomly positioned circle and sphere. An L_2 error shows second order convergence which eventually leveled off on the finest meshes due to effects of the discontinuous interface reconstruction. The proposed scheme is found to produce superior results to the standard height function method for under-resolved interface features. A combined method is presented that uses the standard height function method where it is well defined and the proposed mesh-decoupled height function method in under-resolved regions. The combined method is found to compute accurate curvatures even on very coarse meshes, and second order convergence is observed on fine meshes. The spurious-currents and standing-wave test cases are used to investigate the feasibility of using the combined method in realistic simulations. In the former, small capillary numbers (or spurious velocities) are found for all the mesh levels and Laplace numbers. In the latter, small errors in the wave amplitude that converge under mesh refinement are obtained. In summary, the proposed combined method is an improvement to the standard height function approach and an alternative to the scheme proposed by Popinet [88]. In geometric VOF schemes with computational geometry routines, the implementation may be trivial and forms a straightforward methodology to compute robust curvatures of under-resolved interfaces where the standard height function method often fails.

CHAPTER 6

SIMULATIONS OF PRIMARY ATOMIZATION

In this chapter the numerical methods described previously are applied to three engineering relevant atomizing flows. The flows are a liquid jet in cross-flow, an air-blast atomizer, and an electrically charged spray.

The first two cases are performed with the accurate conservative level set method described in Chapter 2 combined with a density-correction scheme that provides robustness at large density ratios [93]. This methods have been shown to be well suited for simulating turbulent gas-liquid flows [28]. The methods do exhibit a small but non-zero mass and momentum conservation error and tend to be overly diffusive at the interface due to first-order transport. Nonetheless, the methods are able to produce results that compare well with experiments.

The third test case uses the volume-of-fluid interface tracking scheme described in Chapter 3 combined with consistent transport of momentum and electric charges introduced in Chapter 4. This methodology alleviates the limitations of the level set scheme and conserves mass, momentum, and electric charge to machine precision and is second-order accurate. With this approach, excellent results for the simulations of the electrically charged jet are produced even on relatively coarse meshes.

6.1 Liquid jet in cross-flow

6.1.1 Introduction

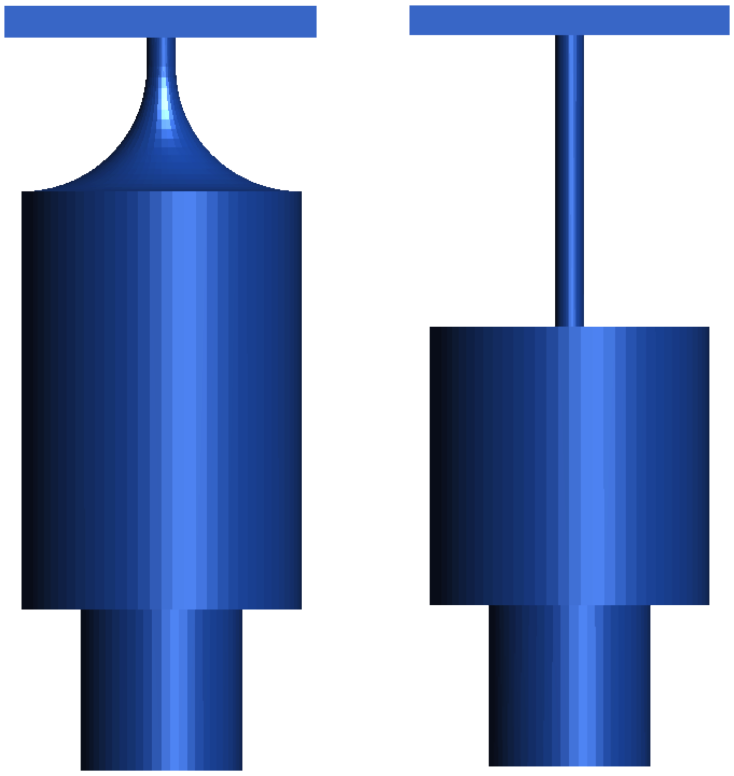
In this section, large-eddy simulation of the atomization of a liquid jet in cross-flow is performed. Two different injector geometries are investigated that result in significantly different liquid jets. One of the injectors, referred to as the round-edged injector, produces a laminar flow at the exit plane. The other injector, known as the sharp-edged injector, produces a turbulent flow that enhances the atomization of the liquid jet. The jet penetration, mean droplet size spatial distribution, and mean droplet velocity spatial distribution are compared to experimental results by Gopala [4], and good agreement is observed.

To perform the simulations, we employ a computational methodology that is accurate and robust even when large density ratios and turbulent flows are present. The accurate conservative level set is used to transport the gas-liquid interface. A density correction formulation is used to ensure consistency between the interface transport and momentum transport steps, making a robust scheme for simulating high density ratio flows. A conservative immersed boundary method is used to simulate the injector geometries, which avoids the complexity of generating a body-fitted mesh.

6.1.2 Simulation setup

A computational study of a liquid jet in cross-flow is presented. The study focuses on the comparison of the jets produced using two different injector geometries

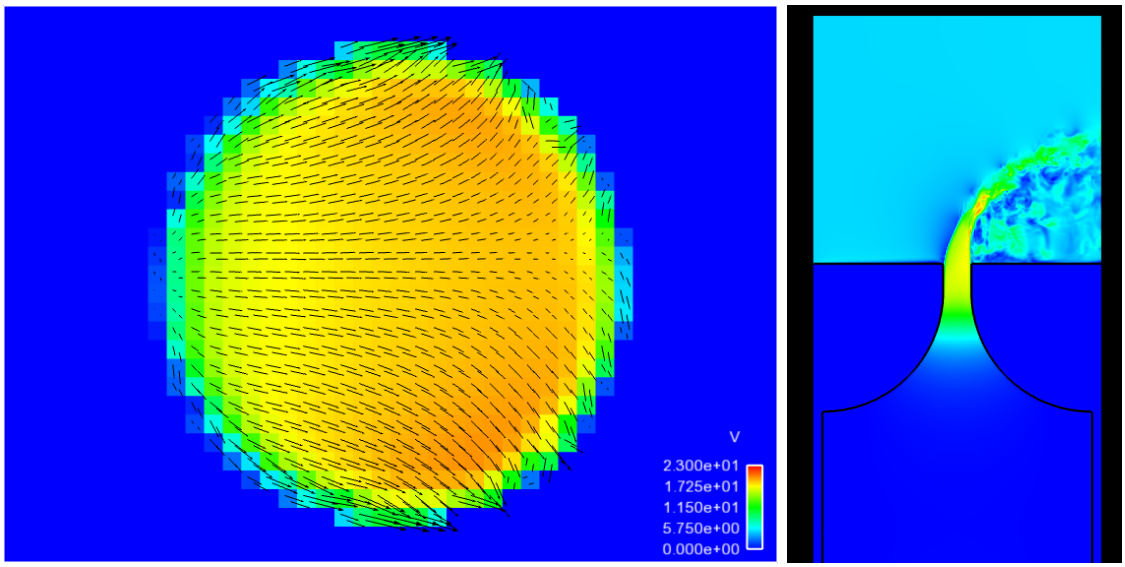
shown in Fig. 6.1 and referred to as round-edged and sharp-edged injectors. The round-edged injector features a smooth transition from the plenum to the injector exit plane, producing a laminar flow at the exit plane. The sharp-edged injector has sharp corners at the edge of the plenum followed by a long pipe that produces a turbulent flow at the exit of the injector. The flow through the injectors was computed in a preliminary simulation. The velocity field at the exit plane of the injector was saved for many flow through times and used as boundary conditions for the liquid jet in cross-flow simulations. Figure 6.2 shows snapshots of the flow in the round-edged and sharp-edged injectors.



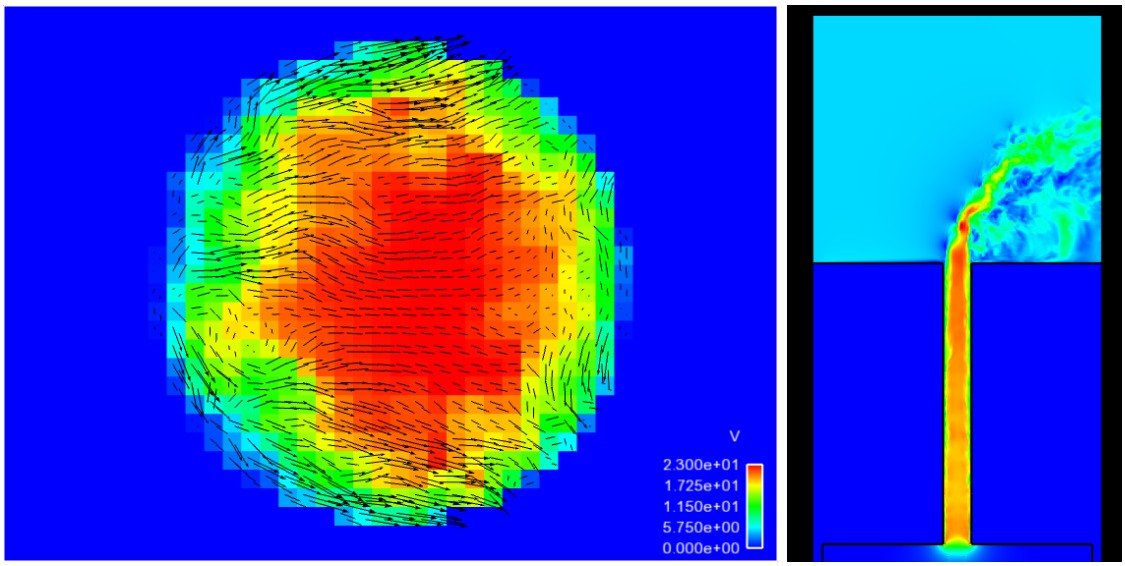
(a) Round-edged injector (b) Sharp-edged injector

Figure 6.1: Injector geometries used in liquid jet in cross-flow simulations. Liquid flows from bottom to top.

The simulations of the liquid jet in cross-flow were performed using a domain of $21D \times 21D \times 16D$, where $D = 0.00047$ m is the diameter of the liquid jet at the exit



(a) Round-edged injector



(b) Sharp-edged injector

Figure 6.2: Velocity field at the exit plane (left) and on a cut-plane through (right) the round-edged and sharp-edged injectors.

of the injector. The mesh is Cartesian with uniform computational cells arranged in a $1366 \times 1366 \times 1024$ mesh. The physical size of a computational cell is 7.3 μm^3. The simulations were performed using 12,240 cores on Lawrence Livermore National Laboratory's (LLNL) Sierra cluster. Non-dimensional properties of the flow are shown in Table 6.1. The large Reynolds and Weber numbers suggest this jet will undergo significant atomization.

6.1.3 Simulation results

Figure 6.3 shows an example snapshot of the gas-liquid interface for the simulation with the sharp-edged injector. Clearly, the jet undergoes dramatic atomization and many small droplets are produced. The velocity field on the exit plane and a plane through the center of the jet is shown in Fig. 6.4. Behind the liquid core a region of significantly smaller streamwise velocity is present. The vertical and spanwise velocities show there is turbulent flow in and around the liquid core, ligaments, droplets and other liquid structures within the flow.

Table 6.1: Non-dimensional properties for liquid jet in cross-flow

Property	Value
ρ_l/ρ_g	137
$\mathrm{Re}_l = \dfrac{\rho_l U_{\mathrm{jet}} D}{\mu_l}$	5430
$\mathrm{We}_l = \dfrac{\rho_l U_{\mathrm{jet}}^2 D}{\sigma}$	5000
$\mathrm{Re}_g = \dfrac{\rho_l U_{\mathrm{jet}} D}{\mu_g}$	9490
$\mathrm{We}_g = \dfrac{\rho_l U_{\mathrm{jet}}^2 D}{\sigma}$	500
$q = \dfrac{\rho_l U_{\mathrm{jet}}^2}{\rho_g U_g^2}$	10

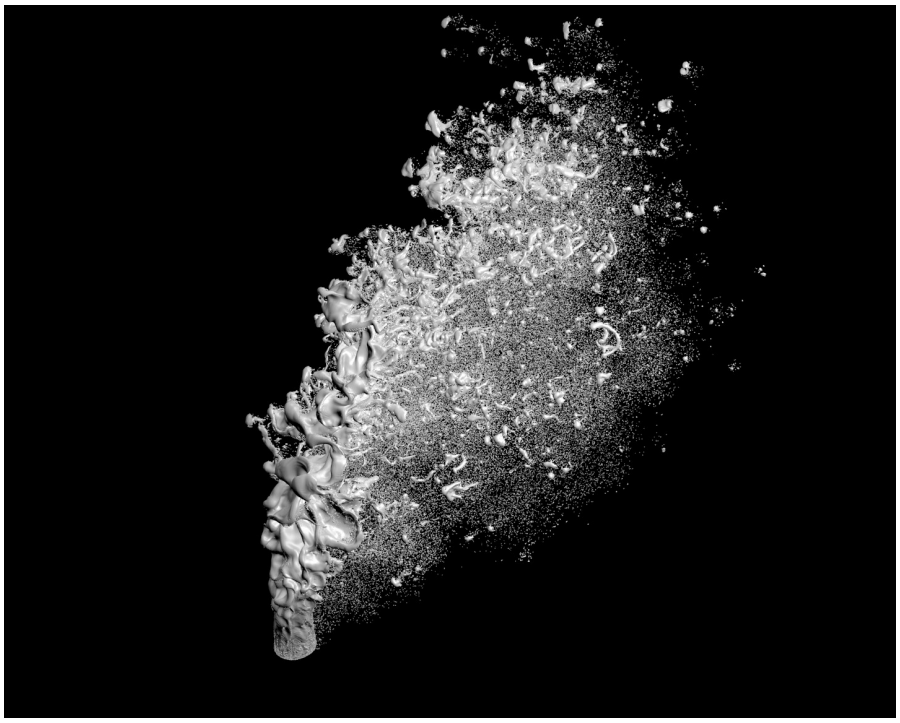


Figure 6.3: Rendering of liquid jet in cross-flow produced by the sharp-edged injector

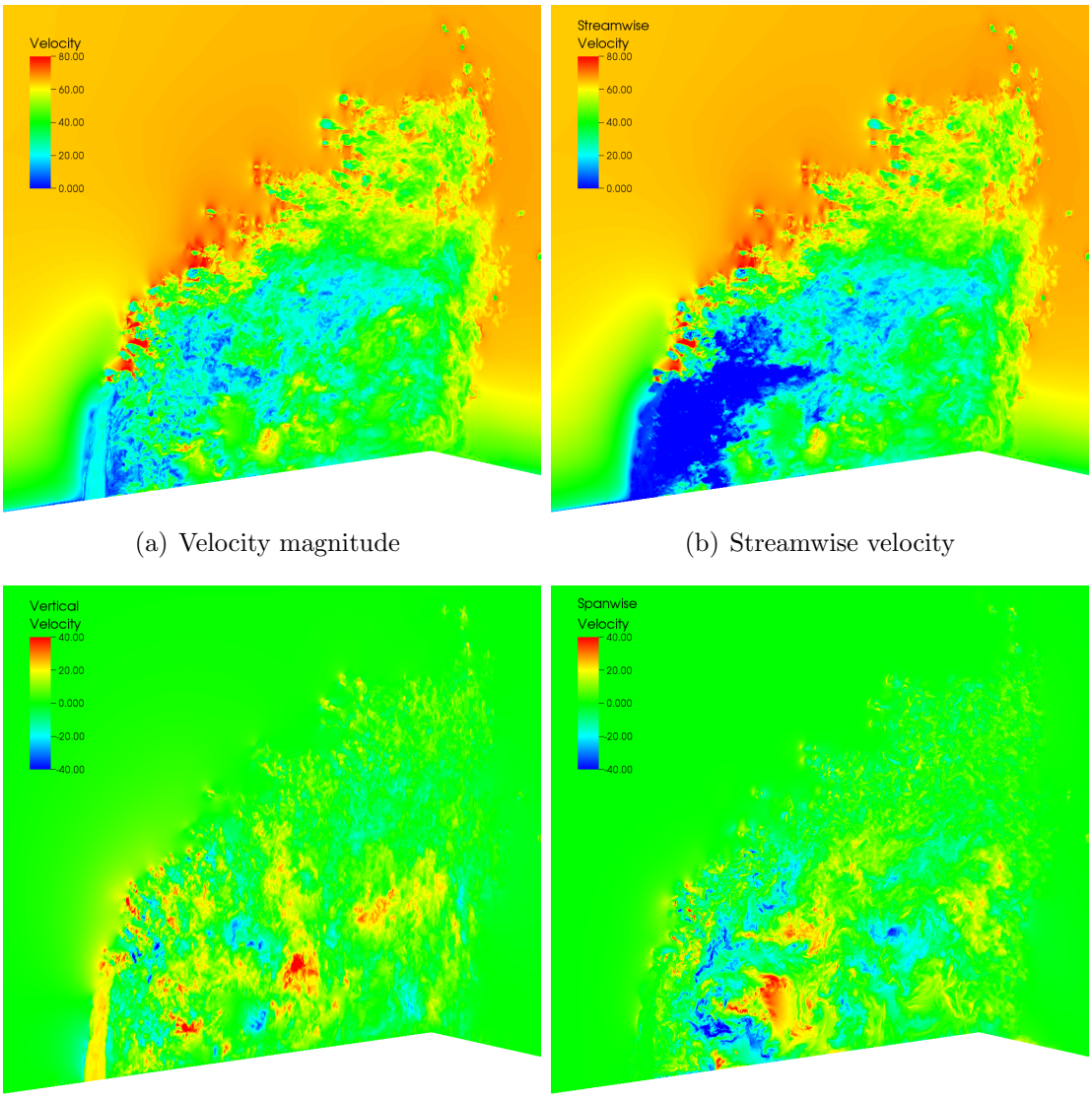
Snapshots of the gas-liquid interface are shown in Fig. 6.5 for the two injectors. Qualitatively, it is evident that the sharp-edged injector and associated turbulence inflow significantly enhance the atomization process through the formation of ligaments and other structures. The round-edged injector forms a coherent liquid sheet that eventually breaks into droplets through the effect of shear.

The penetration of the liquid jet is compared to experimental correlations and the result is shown in Fig. 6.6. In the figure, the gas-liquid interface from the simulations is shown along with a red line that indicates the experimental correlation for the outermost edge of the jet. The correlations are from Gopala [4] and are

$$\frac{x}{D} = 1.187 q^{0.437} \log\Big(1 + 1.123 \frac{z}{D}\Big), \text{ and} \tag{6.1}$$

$$\frac{x}{D} = 1.914 q^{0.415} \log\Big(1 + 2.238 \frac{z}{D}\Big), \tag{6.2}$$

for the round-edged and sharp-edged injectors, respectively. In these relations, q



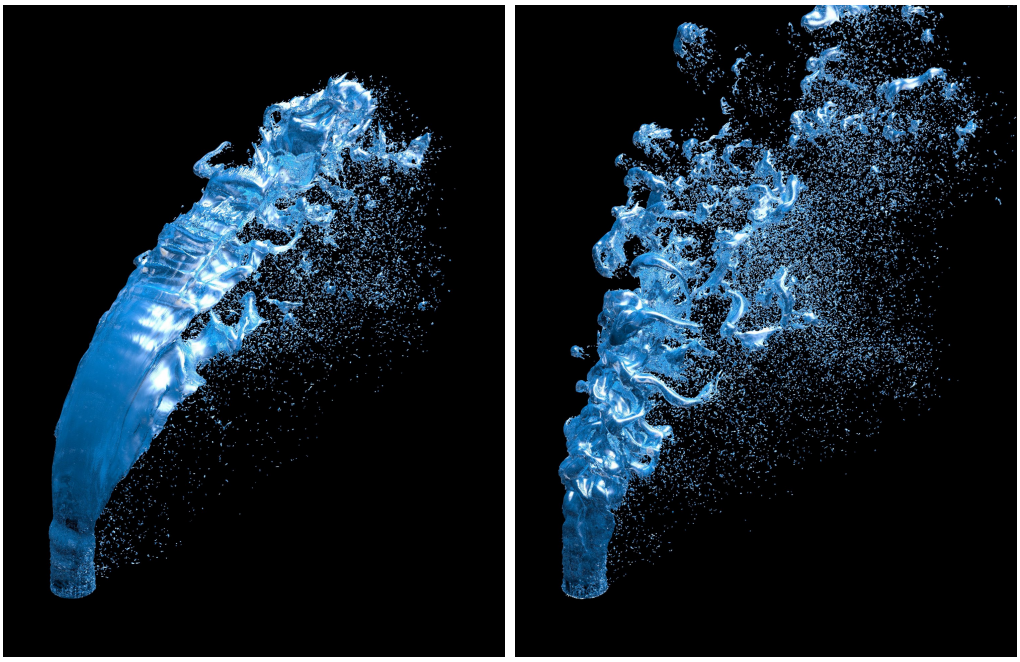
(a) Velocity magnitude

(b) Streamwise velocity

(c) Vertical velocity

(d) Spanwise velocity

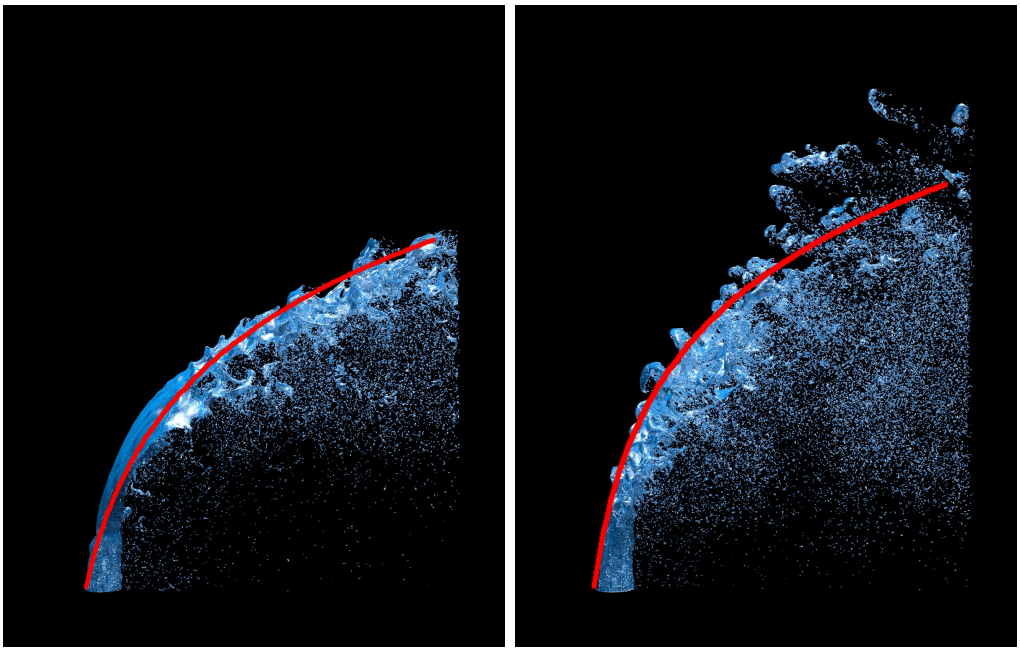
Figure 6.4: Velocity field within liquid jet in cross-flow produced by the sharp-edged injector



(a) Round-edged injector (b) Sharp-edged injector

Figure 6.5: Snapshot of the gas-liquid interface for the liquid jet in cross-flow from the two injector geometries.

is the momentum flux ratio, x is the vertical direction and z is the streamwise direction that is parallel with the bulk gas flow. For completeness, y is the spanwise direction. Very good agreement is observed in the penetration of the simulated atomizing jets with experimental results. The small difference between the results for the round-edged injector is likely due to the turbulence intensity within the high-speed gaseous cross-flow. In the simulations, a constant bulk velocity profile was used for a boundary condition for the cross-flow, whereas turbulent flow was present in the experiments. This difference is not expected to affect the sharp-edged injector as significantly since the liquid jet is turbulent and provides fluctuations that destabilizes the atomizing jet. In the jet produced by the round-edged injector, both the liquid jet and cross-flow boundary conditions do not have turbulent fluctuations and the flow takes longer to destabilize.



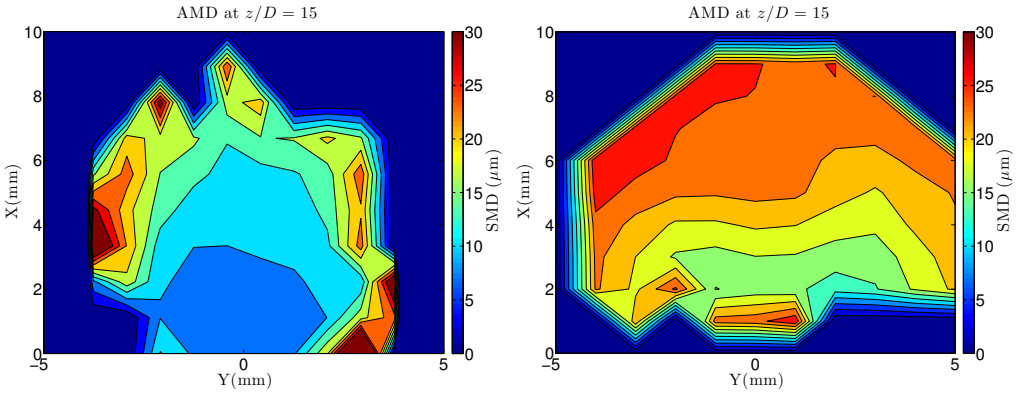
(a) Round-edged injector (b) Sharp-edged injector

Figure 6.6: Liquid jet in cross-flow penetration. Red line shows experimental correlation for outermost edge from Gopala [4].

A comparison of drop size between the simulation of the sharp-edged injector and the experiments is provided in Figs. 6.7 and 6.8. The figures compare the arithmetic and Sauter mean diameters denoted AMD and SMD, respectively. The comparison is performed at 15 diameters downstream of the injector. Droplet statistics were collected from the simulations for $15T$, where $T = D/U_{\text{jet}}$ is the characteristic flow time. The simulations produce results that are consistent with experimental measurements. The simulation is constrained in the spanwise direction (y) which is evident by the sharp edges on the sides of the simulation results. Larger structures are typically located near the top of the spray. Figure 6.3 shows that the larger structures are the remains of the liquid core that has broken into ligaments and other large structures. The smaller droplets near the bottom of the spray are formed as the liquid core breaks and from being stripped away from the sides of the liquid core.

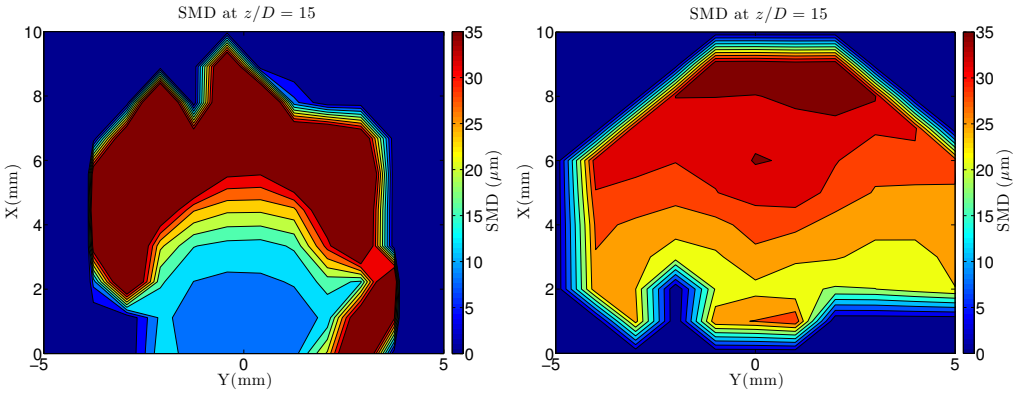
The velocity of the liquid structures was also calculated and compared with experimental results of Gopala [4]. Figures 6.9, 6.10, and 6.11 show the average vertical, spanwise, and streamwise components of droplet velocity at a streamwise position of $z/D = 15$. The vertical velocity shows that the large liquid structures near the top of the spray are continually moving upwards. This is consistent with the increasing penetration of the jet. The smaller droplets in the lower part of the spray are not moving significantly in the vertical direction. The spanwise velocity shows the spray is expanding in the spanwise direction. In the simulation results, the effect of the domain size is evident in the results and indicate the spray is constrained in the spanwise direction. The streamwise velocity shows the droplets are moving the fastest around the edges of the spray where the high-speed cross-flow has accelerated the liquid. A region of slower moving droplets exists in the center of the spray behind the central core of the jet.

While the simulation results generally match the experimental results, some differences are present. One possible cause of the discrepancy is the inability of the experimental diagnostic equipment to detect non-spherical droplets. A Phase Doppler Particle Analyzer (PDPA) was used in the experiments to measure droplet diameters and velocities [4]. PDPA devices have been shown to not correctly detect non-spherical particles [94]. Figure 6.12 shows a scatter plot of droplet eccentricity defined as the ratio of largest to smallest characteristic lengths of the droplet indicated with L_1 and L_2, respectively. Clearly, many non-spherical droplets with large eccentricities are present in the simulation results which may have been undetected in the experiments.



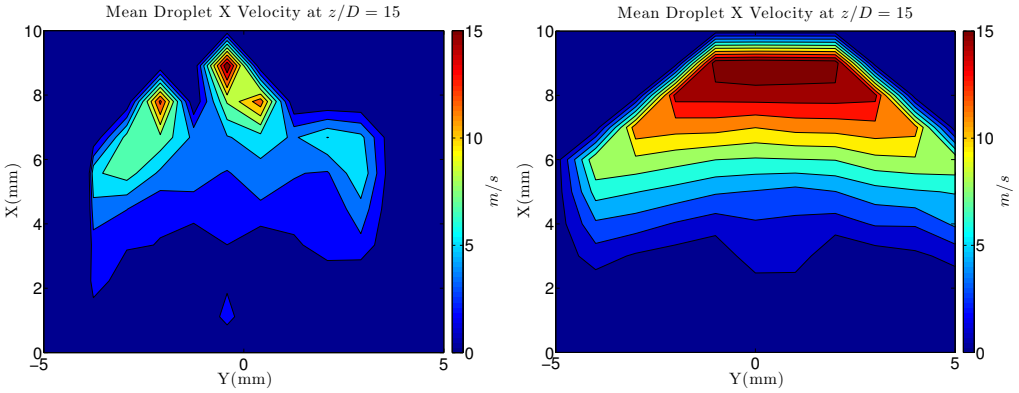
(a) Simulation result (b) Experimental result [4]

Figure 6.7: AMD



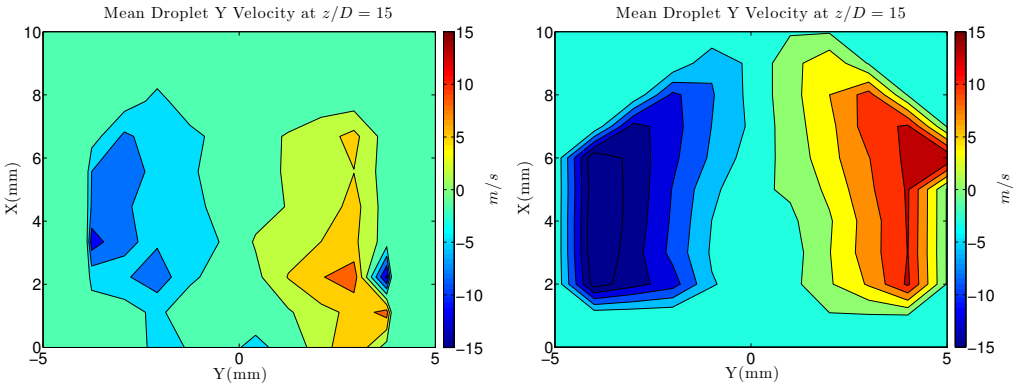
(a) Simulation result (b) Experimental result [4]

Figure 6.8: SMD



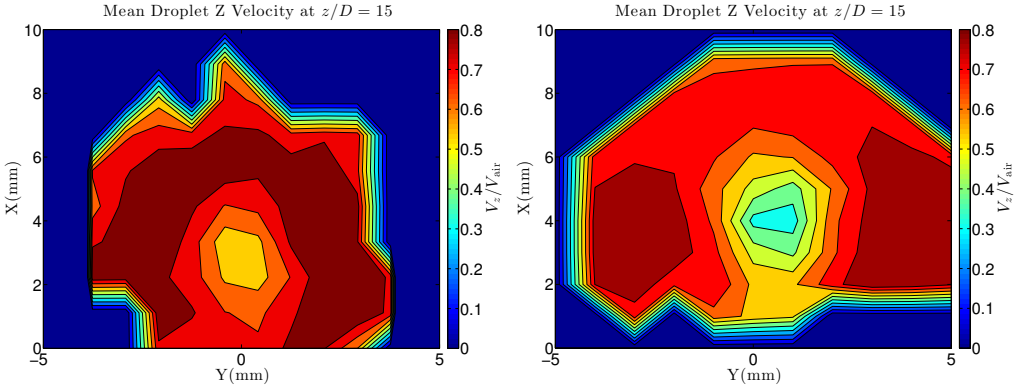
(a) Simulation result (b) Experimental result [4]

Figure 6.9: Vertical velocity



(a) Simulation result (b) Experimental result [4]

Figure 6.10: Spanwise velocity



(a) Simulation result (b) Experimental result [4]

Figure 6.11: Streamwise velocity

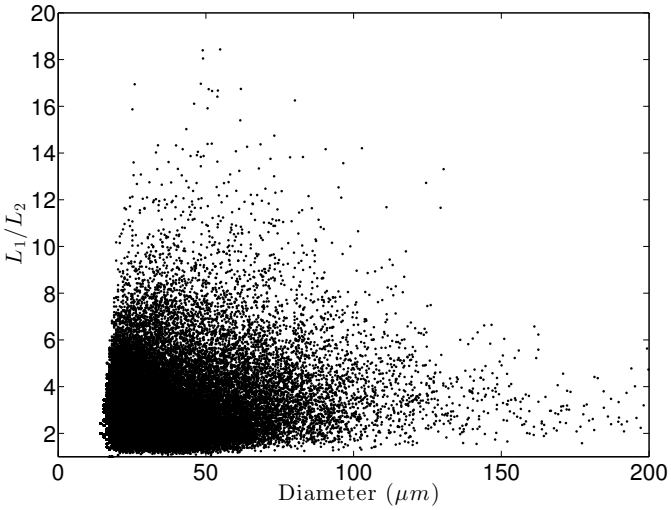


Figure 6.12: Scatter plot of eccentricity of droplets versus droplet size.

6.2 Air-blast n-dodecane atomization

Air-blast atomization of hydrocarbon fuels is of critical importance to the transportation sector, in particular for aircraft gas turbine engines. In this section, a co-annular air-blast n-dodecane injector is studied and compared with experimental data collected by TDA Research a collaborator on this project. The simulations utilize a discontinuous Galerkin discretization of the ACLS procedure [41]. Computational results are compared to experimental measurements, showing the satisfactory behavior of the simulation technique. In particular, the onset of breakup, most unstable wavelength, and drop size and velocity distributions are in good agreement, suggesting that the fundamental physics of air-blast atomization are well captured by the simulations.

6.2.1 Geometry and Numerical setup

The external mixing air-blast atomizer shown in Fig. 6.13 was designed after the one described by Marmottant and Villermaux [14]. The simple, externally mixed geometry is well-suited for numerical modeling and code validation. Since transitional or developing flows are much more difficult to simulate with accuracy, the tube lengths were chosen to ensure fully developed flows at the exit.

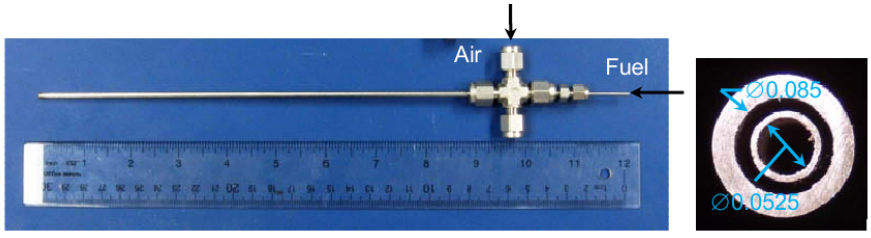


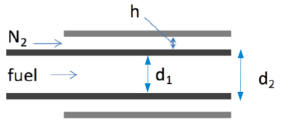
Figure 6.13: Geometry of air-blast atomizer.

A variety of quantities were gathered from both experiments [43] and simulations to facilitate a validation of numerical methods. Images of the experimental jet and renderings of the simulated jet were created and examined qualitatively. Probability density functions of drop size and drop velocity were calculated to show the probability of a droplet being created of a given diameter and with a certain velocity, respectively. n-Dodecane was injected with a co-flow of nitrogen, their respective properties are shown in Table 6.2 with the subscript l for liquid n-dodecane and g for nitrogen gas. The high density ratio of 597 is feasible due to the density-based flux correction scheme that ensures discrete consistency between fluxes of density and momentum.

Table 6.2: Properties of n-dodecane and nitrogen.

Density	ρ_l	746 kg/m^3
	ρ_g	1.25 kg/m^3
Surface Tension	σ	2.535×10^{-2} N/m
Dynamic Viscosity	μ_l	1.34×10^{-3} kg/m $\cdot$ s
	μ_g	1.718×10^{-5} kg/m $\cdot$ s

The injector geometry is detailed in Fig. 6.14 and consists of a straight jet of diameter d_1 surrounded by a co-flow of inner diameter d_2 and thickness h. The length of the injector is not shown in the sketch but is visible in Fig. 6.13 and has been designed so that the flow leaving the nozzle is fully developed.



$$d_1 = 1.3335 \times 10^{-3}\ m$$
$$d_2 = 1.5875 \times 10^{-3}\ m$$
$$h = 2.8575 \times 10^{-4}\ m$$

Figure 6.14: Air-blast injector dimensions.

The phase Reynolds and Weber numbers are based on the liquid velocity U_l

and gas velocity U_g, and are provided in Table 6.3. Reynolds numbers for the test case indicate that the flow of n-dodecane and the co-flow of N_2 is laminar. The laminar nature of the co-flow was confirmed by numerically simulating a periodic annular pipe under these conditions.

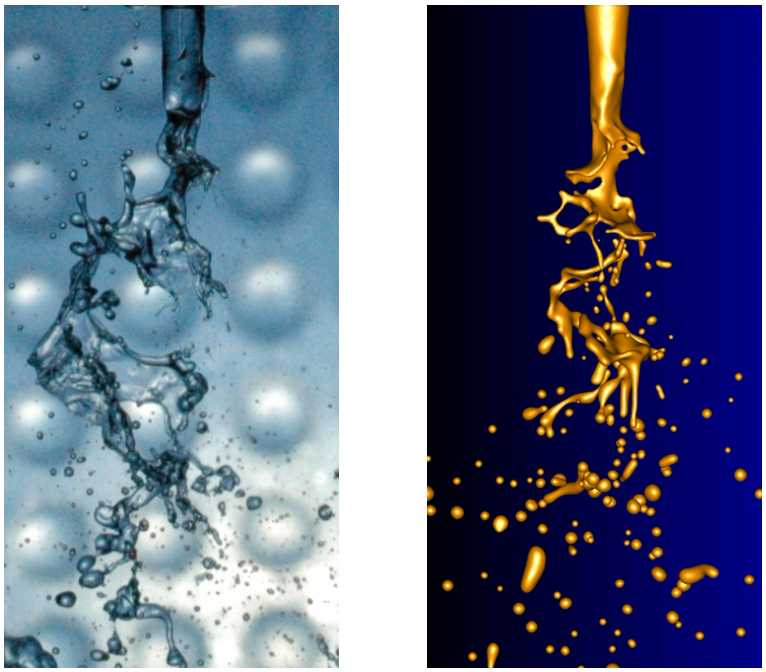
Table 6.3: Flow parameters for the test case.

U_l [m/s]	U_g [m/s]	Re_l	Re_g	We_l	We_g
1.8	69.89	1336	1453	127	321

The atomization simulation was performed on 1,024 processors using a mesh of $512 \times 256 \times 256$ grid cells. A CFL number below 0.9 was maintained throughout the simulation. Roughly 1.5 flow through times were used to allow the jet to reach a statistically steady state.

6.2.2 Shear instability results

Primary atomization under the flow parameters described above results in an initially smooth jet that rapidly breaks up into droplets. Figure 6.15 shows a side-by-side comparison of snapshots from the experiment and the simulation. Qualitatively, there is excellent agreement in the shape of the jet, the length-scales of instabilities, the onset of liquid break-up, and the distribution of large droplets. There appears to be smaller droplets in the experiment that are not found in the simulation result. It is postulated that such small droplets are formed when liquid accumulates on the outside of the nozzle and the jet interacts with this liquid. Figure 6.16 shows an instance of the n-dodecane jet interacting with the wetted nozzle.



(a) Photo of experiment.

(b) Rendering of simulation result.

Figure 6.15: Comparison of jet from (a) experiment and (b) simulation.

Figure 6.16: Example of nozzle wetting and the effect on break-up process.

At the exit plane of the nozzle there exists a shear layer between the fast-moving co-flow of gas and the slower moving liquid jet. Marmottant and Villermaux [14] showed that this flow destabilizes by means of a Kelvin-Helmholtz type of insta-

bility, and the most amplified wavenumber is given by

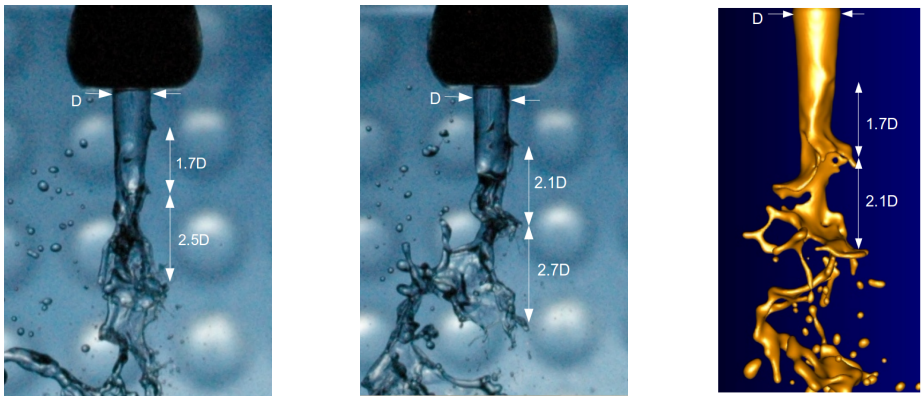
$$k_m \approx 1.5\left(\frac{\rho_g}{\rho_l}\right)^{1/2}\frac{1}{\delta}, \tag{6.3}$$

where δ is the vorticity thickness of the gas jet. An *a priori* estimate of δ is challenging to obtain due to the presence of the gap between the jet and co-flow. However, to build an estimate of δ, the boundary layer thickness is defined as the location where the velocity is 50% of its maximum. Using a Poiseuille velocity profile, consistent with the laminar inflow, this approximation results in $\delta = 2.3 \times 10^{-5}\,\mathrm{m}$ and $k_m = 2600\,\mathrm{m}^{-1}$. Converting from wavenumber to wavelength λ_{axi} leads to $\lambda_{\mathrm{axi}} = 2.4 \times 10^{-3}\,\mathrm{m}$ or 1.8 jet diameters ($1.8D$). Looking at Fig. 6.17(a) and 6.17(b), four different waves were identified and measured using a photo analysis program. The average wavelength was found to be $2.25D$. A similar analysis was performed for results obtained from the simulation and Fig. 6.17(c) shows that the average wavelength in the simulations is about $1.9D$.

In summary, the theoretical calculation predicts the wavelength to be near $1.8D$, the experiment showed $2.25D$, and $1.9D$ was found using the simulation. All of the values are of the same order of magnitude and agree reasonably well, indicating the leading break-up mechanism is an instability akin to Kelvin-Helmholtz. Furthermore, this analysis shows that the simulations are capable of capturing the shear layer and the effects it has on the flow.

6.2.3 Drop characteristics

Drops were identified and characterized in experiments and simulations since the size of droplets produced by primary atomization is an important result for combustion-related applications. For experiments, a TSI PIV system with the

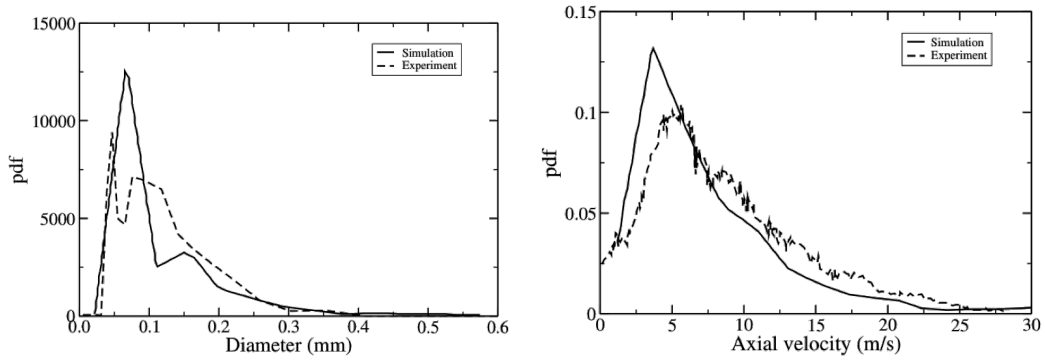


(a) Experiment: Time 1 (b) Experiment: Time 2 (c) Simulation

Figure 6.17: Measurement of shear instability using photos at two different times during the experiment (a,b) and a rendering of simulation data (c).

GSV option was used to measure droplet size and velocity. Simulations used a band-growth algorithm [95] to identify droplets and compute their size and velocity. Using the two methods, probability density functions of drop size were calculated. The results shown in Fig. 6.18(a) illustrate the excellent agreement in the size of droplets found in our simulations and experiments. The agreement between the probability density functions show that the simulations are capable of accurately predicting the break-up dynamics and could be used to predict drop sizes for design applications.

In addition to drop size distributions, droplet velocity distributions were also calculated. Figure 6.18(b) shows probability density functions of droplet axial velocity. Again, excellent agreement is found between the experiment and simulation, indicating that the droplets are forming with the correct velocity, which suggests in turn that the break-up mechanism is captured in the simulation.



(a) Drop sizes.

(b) Drop axial velocities.

Figure 6.18: Probability density function of (a) drop size and (b) drop velocity using experimental and simulation results.

6.3 Electrohydrodynamic assisted atomization

Electrohydrodynamics (EHD) is the field of science that describes systems with a significant interaction of fluid mechanics and electrostatics. EHD has successfully been used in a variety of engineering applications to control or produce fluid motion. Some examples include inkjet printing [96], mass spectrometry analysis of biomolecules [97], Taylor cones [98–103], microfluidic devices [104–109], agricultural sprays [110], and fuel atomization [76, 77, 98, 111–121], which is the focus of this work. EHD has the potential to enhance atomization and the corresponding increase in surface area improves the evaporation rate of fuels [114], as shown in Fig. 6.19. Additionally, EHD has been found to be a viable and useful method to control the droplet distribution within a combustion chamber under realistic direct injection spark ignition conditions [76].

Numerical studies related to EHD atomization have been conducted. Shrimpton and Kourmatzis explored the flow within the injector [113]. Many researchers have proposed methods to study EHD and applied the methods to simplified problems like droplet deformation [122, 123]. Van Poppel et al. [77] proposed a

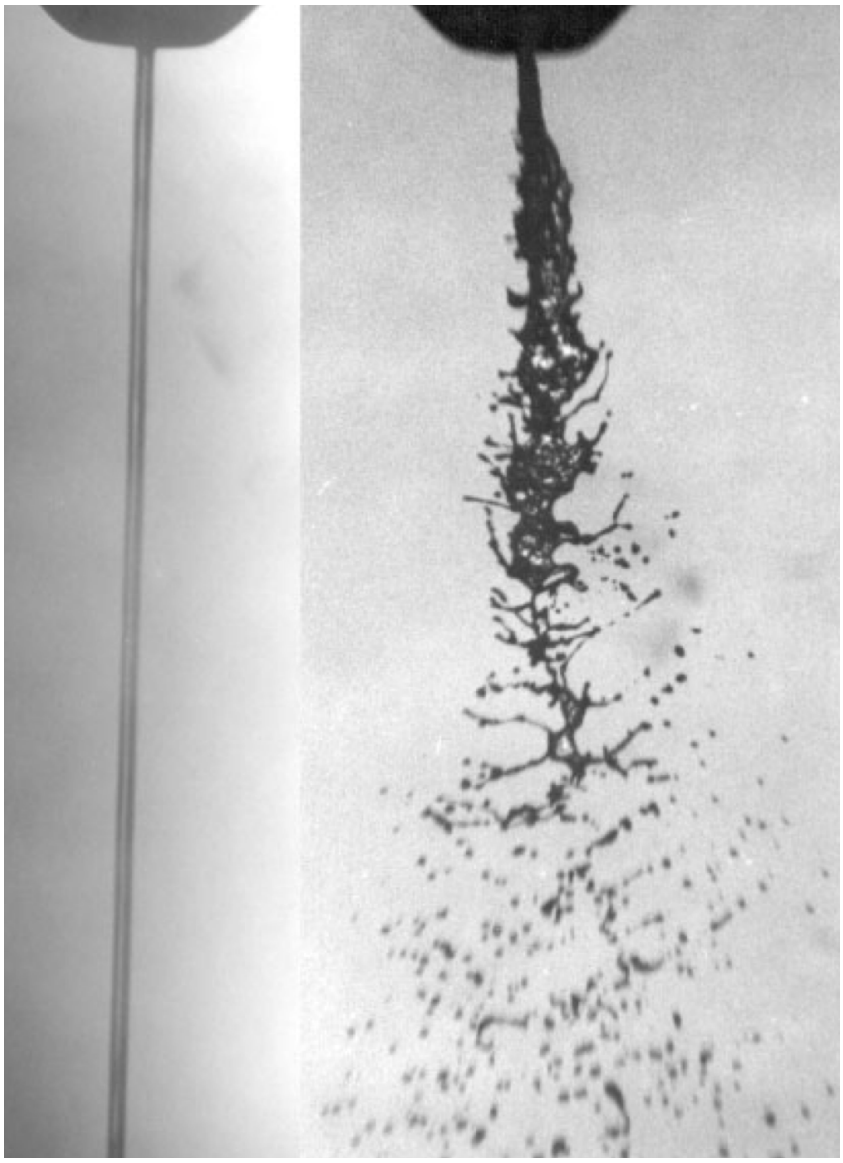


Figure 6.19: Effect of EHD on atomization. Uncharged (left) and charged (right) liquid jets. (used with permission [118])

methodology and successfully performed direct numerical simulations (DNS) of a charged atomizing jet. However, the electric charges were assumed to be uniformly distributed within the liquid. This assumption was shown, through a time-scale analysis, to be the better than the alternative commonly used assumption wherein charges instantly relax to the surface of the liquid. In this work, a conservation of charge equation is solved avoiding the necessity of making an assumption on the

charge distribution.

6.3.1 Mathematical Formulation

EHD atomizing flows are described by hydrodynamics and electrostatics. Assuming electrostatics instead of electrodynamics is equivalent to assuming magnetic effects can be ignored. This is appropriate since the EHD timescale is several orders of magnitude larger than the magnetic timescale as shown by Saville [124]. Therefore, the electric field is assumed to be continuously in equilibrium with the distribution of electric charges within the system. Maxwell's equations for electrostatics and conservation laws for mass and momentum describe electrostatic-hydrodynamic flows and are summarized in this section.

Conservation of mass and momentum for a low Mach number, variable density flow are given in both phases as

$$\frac{\partial \rho_i}{\partial t} + \nabla \cdot (\rho_i \boldsymbol{u}_i) = 0, \tag{6.4}$$

and

$$\frac{\partial \rho_i \boldsymbol{u}_i}{\partial t} + \nabla \cdot (\rho_i \boldsymbol{u}_i \otimes \boldsymbol{u}_i) = -\nabla p_i + \nabla \cdot \left(\boldsymbol{\sigma}_i^f + \boldsymbol{\sigma}_i^e\right) + \rho_i \boldsymbol{g}, \tag{6.5}$$

where ρ_i is the density, $\boldsymbol{u}_i$ is the velocity field vector, t is time, p_i is the hydrodynamic pressure, and $\boldsymbol{g}$ is the gravitational acceleration. The subscript $i = g$ or l and indicates variables in the gas or liquid phase, respectively. $\boldsymbol{\sigma}_i^f$ is the viscous stress tensor given by

$$\boldsymbol{\sigma}_i^f = \mu_i\left(\nabla \boldsymbol{u}_i + \nabla \boldsymbol{u}_i^{\mathsf{T}}\right) - \frac{2}{3}\mu_i (\nabla \cdot \boldsymbol{u}_i)\mathbb{I}, \tag{6.6}$$

where μ_i is the dynamic viscosity and $\mathbb{I}$ is the identity tensor. $\boldsymbol{\sigma}_i^e$ is Maxwell's

stress tensor

$$\boldsymbol{\sigma}_i^e = \varepsilon_i \boldsymbol{E}_i \otimes \boldsymbol{E}_i - \frac{\varepsilon_i}{2} \boldsymbol{E}_i \cdot \boldsymbol{E}_i \left(1 - \frac{\rho_i}{\varepsilon_i} \frac{\partial \varepsilon_i}{\partial \rho_i}\right) \mathbb{I}, \tag{6.7}$$

where ε_i is the electric permittivity and $\boldsymbol{E}_i$ is the electric field vector. Maxwell's stress tensor induces an electric body force that can be written as

$$\boldsymbol{f}_i^e = \nabla \cdot \boldsymbol{\sigma}_i^e = q \boldsymbol{E}_i - \frac{1}{2} \boldsymbol{E}_i^2 \nabla \varepsilon_i + \nabla \left(\frac{1}{2} \rho_i \frac{\partial \varepsilon_i}{\partial \rho_i} \boldsymbol{E}_i^2 \right), \tag{6.8}$$

where q_i is the volumetric electric charge density. The three terms in the electric body force are, from left to right, the Coulomb (or Lorentz) force, the dielectric force, and the electrostrictive force. The latter two are only important if a transient electric field exists or if the permittivity is spatially varying [125].

The electric field vector is irrotational due to the electrostatic assumption and can be expressed as the gradient of the scalar electric potential ϕ_i using

$$\boldsymbol{E}_i = -\nabla \phi_i. \tag{6.9}$$

The electric potential is related to the volumetric charge density by

$$-\nabla \cdot (\varepsilon_i \nabla \phi_i) = q_i, \tag{6.10}$$

which is referred to as the electric potential Poisson equation.

The dynamics of the electric charge density is described by the conservation equation

$$\frac{\partial q_i}{\partial t} + \nabla \cdot \boldsymbol{J}_i = 0, \tag{6.11}$$

where $\boldsymbol{J}_i$ is the current density, which can be written as

$$\boldsymbol{J}_i = q_i \boldsymbol{u}_i + q_i \kappa_i \boldsymbol{E}_i - D_i \nabla q_i, \tag{6.12}$$

where κ_i is the ionic mobility coefficient and D_i is the molecular diffusion coefficient. The three terms that contribute to the current density can be described as

convection due to the velocity field, convection due to the electrical velocity $\kappa_i \boldsymbol{E}_i$, and diffusion.

The equations above have been written in both the gas and liquid phases. They are connected through jump conditions at the phase interface. For example, the jumps in density, viscosity, and permittivity at the interface Γ are written as

$$[\rho]_\Gamma = \rho_l - \rho_g, \tag{6.13}$$

$$[\mu]_\Gamma = \mu_l - \mu_g, \tag{6.14}$$

$$[\varepsilon]_\Gamma = \varepsilon_l - \varepsilon_g. \tag{6.15}$$

In the absence of phase change the velocity field is continuous in the normal direction, i.e., $[\boldsymbol{u} \cdot \boldsymbol{n}]_\Gamma = 0$, where $\boldsymbol{n}$ is the interface normal vector. Analogously to the no-slip assumption, the tangential velocity at the interface is assumed to be continuous and can be written as $[\boldsymbol{u} \cdot \boldsymbol{t}_d]_\Gamma = 0$, for $d = 1, 2$. Combining the two jump conditions for the velocity field, it is clear that the velocity is continuous, i.e.,

$$[\boldsymbol{u}]_\Gamma = 0. \tag{6.16}$$

The pressure is discontinuous due to contributions from surface tension, viscous, and electric forces and can be written as

$$[p]_\Gamma = \gamma\kappa + \left[\boldsymbol{n}^\mathsf{T} \cdot \left(\boldsymbol{\sigma}^f + \boldsymbol{\sigma}^e\right) \cdot \boldsymbol{n}\right]_\Gamma, \tag{6.17}$$

where γ is the surface tension coefficient and κ is the interface curvature. The previous equation can be simplified to [77]

$$\begin{aligned} &[p]_\Gamma - 2[\mu]_\Gamma \boldsymbol{n}^\mathsf{T} \cdot \nabla \boldsymbol{u} \cdot \boldsymbol{n} - \gamma\kappa \\ &\quad = \frac{1}{2}\left[\varepsilon(\boldsymbol{E} \cdot \boldsymbol{n})^2 - \varepsilon(\boldsymbol{E} \cdot \boldsymbol{t}_1)^2 - \varepsilon(\boldsymbol{E} \cdot \boldsymbol{t}_2)^2\right]_\Gamma. \end{aligned} \tag{6.18}$$

The electric field is discontinuous if surface charges are present at the phase interface and can be written as

$$\boldsymbol{n} \cdot [\varepsilon \boldsymbol{E}]_\Gamma = q_s, \tag{6.19}$$

where q_s is the surface electric charge density. Due to the electrostatic assumption the electric field remains irrotational and $\boldsymbol{n} \times [\boldsymbol{E}]_\Gamma = 0$. A consequence of this relation is the tangential component of the electric field and the electric potential are continuous, i.e.,

$$[\boldsymbol{E} \cdot \boldsymbol{t}_d]_\Gamma = 0 \text{ for } d = 1, 2, \tag{6.20}$$

$$[\phi]_\Gamma = 0. \tag{6.21}$$

The balance of shear stress at the interface leads to

$$[\boldsymbol{n}^\mathsf{T} \cdot (\boldsymbol{\sigma}^f + \boldsymbol{\sigma}^e) \cdot \boldsymbol{t}_d]_\Gamma = 0 \text{ for } d = 1, 2. \tag{6.22}$$

Conservation of charge at the phase interface is described by

$$\begin{aligned} [\boldsymbol{J} \cdot \boldsymbol{n}]_\Gamma + \nabla_s \boldsymbol{J}_s = (\boldsymbol{n} \cdot \boldsymbol{u})[q]_\Gamma \\ - \frac{\partial q_s}{\partial t} - \boldsymbol{u}_s \cdot \nabla q_s + q_s \boldsymbol{n} \cdot (\boldsymbol{n} \cdot \nabla)\boldsymbol{u}, \end{aligned} \tag{6.23}$$

where ∇_s is the surface gradient operator, $\boldsymbol{J}_s$ is the surface charge current density, and $\boldsymbol{u}_s$ is the interface surface velocity.

6.3.2 Numerical methods

The equations described in the previous section are solved using the NGA computational platform [28, 56] using the VOF interface tracking methodology. The EHD governing equations are solved based on the work of Van Poppel et al. [77] which is modified in this work with the inclusion of the electric charge conservation

equation, Eq. 6.11. Additional details of the electric charge density transport are described below.

Electric charge density transport

For the application of EHD assisted atomization, electric charges are typically introduced into the liquid phase using a large potential within the injector [126]. The charged fuel is then injected into the combustion chamber. The electric charges remain in the liquid phase and are not present in the gas, i.e., $q_g = 0$. The electric charge density can vary spatially within the liquid and charges can accumulate near the phase interface due to charge repulsion. The charges form an electric boundary layer at the interface that can be represented as a surface charge density q_s or as a localized concentration of volumetric electric charge density q. In this work, the latter choice is made, which is valid provided there is sufficient resolution to capture the electric charge boundary layer. With $q_s = 0$, Eq. 6.23 simplifies to

$$[\boldsymbol{J} \cdot \boldsymbol{n}]_\Gamma = (\boldsymbol{n} \cdot \boldsymbol{u})[q]_\Gamma, \tag{6.24}$$

which is equivalent to zero flux of electric charge density through the phase interface.

Within the liquid phase, the temporal change in electric charge density is described by Eq. 6.11. The convection term in this equation is solved by computing third-order WENO-type fluxes [78, 79] away from the phase interface where it is smooth and using geometrically computed fluxes near discontinuities at the interface [127]. Diffusion fluxes are computed using second-order centered finite difference operators. At the interface the zero-flux constraint is enforced by scaling the diffusion and electric convection fluxes with the wetted area of each computational cell face.

6.3.3 Simulation results

Simulations of kerosene atomization are performed based on the experimental work by Yule and Shrimpton [116]. Figure 6.20 shows the geometry for the simulations. In the figure, d is the jet diameter and U is the mean jet velocity. The electric potential is set to zero on the four sides of the computational domain (x-y and x-z faces). Table 6.4 summaries the non-dimensional numbers for the simulations. Table 6.5 provides the physical parameters for the test case.

The simulations of the kerosene jet are performed by first computing an inflow velocity field. This velocity field is stored and used as a boundary condition for the jet simulations. The inflow was computed using a periodic pipe with a Reynolds number of 5000. Note that this does not match Reynolds number of the liquid jet. However, using Re = 4000 in a numerical simulation produces a laminar profile. It is likely the turbulent profile more closely matches the flow from the physical injector [116]. Ideally the flow through the injector should be simulated, however this flow will significantly depend on the motion of electric charges from the high potential needle to the liquid. Simulating this flow is beyond the scope of this paper and the simplified turbulent inflow is used.

Two computational domains are used to study the jet, namely a small and large domain of sizes $16d \times 8d \times 8d$ and $32d \times 16d \times 16d$, respectively. The same number of grid points is used to discretize both domains and consists of $512 \times 256 \times 256$ computational cells in the x, y, and z directions.

Figure 6.21 shows images of the gas-liquid interface computed in simulations of an uncharged and a charged kerosene jet on the smaller domain. Both jets have the same parameters except for the electric charge density. The presence of electrical

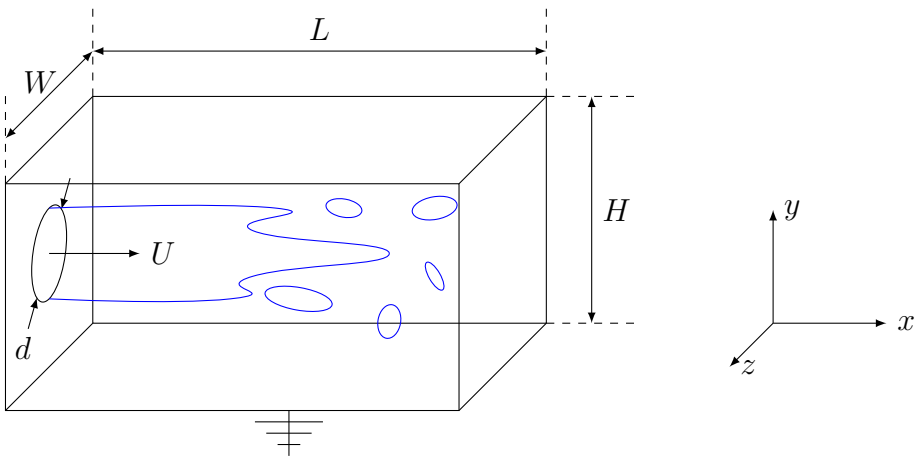


Figure 6.20: Geometry for simulations of EHD enhanced kerosene atomization. Note all four sides (xy-planes and xz-planes) are grounded, i.e., $\phi_i = 0$.

Table 6.4: Non-dimensional numbers used in the charged kerosene jet simulations.

Non-dimensional number		Value
Reynolds number	$\rho_l U d/\mu_l$	4000
Weber number	$\rho_l U^2 d/\gamma$	1700
Electro-inertial num.	$q_l^2 d^2/(\varepsilon_l \rho_l U^2)$	0.04
Electric Reynolds num.	$\varepsilon_l U/(d\kappa_l q_l)$	780
Electric Peclet number	$q_l \kappa_l d^2/(D\varepsilon)$	640
Density ratio	ρ_l/ρ_g	664
Viscosity ratio	μ_l/μ_g	51
Permittivity ratio	$\varepsilon_l/\varepsilon_g$	2.2
Relative length	L/d	16
Relative width	W/d	8
Relative height	H/d	8

charges clearly enhances the atomization process. The temporal evolution of the electrically charged jet is shown in Fig. 6.22. The deformation of the interface is caused by the Coulomb force that results in the creation of ligaments and droplets that are pushed away from the central core due to electric charge repulsion. This EHD effect creates unique features such as a relatively coherent central core and ligaments orientated in the radial direction. These features are consistent with those observed experimentally and shown in Fig. 6.19. Figure 6.23 shows the

Table 6.5: Physical parameters in charged kerosene jet simulations.

Parameter	Symbol	Units	Value
Mean velocity	U	m/s	10
Injector diameter	d	μm	500
Liq. density	ρ_l	kg/m^3	800
Gas density	ρ_g	kg/m^3	1.2
Liq. viscosity	μ_l	kg/m·s	1.0e-3
Gas viscosity	μ_g	kg/m·s	1.98e-5
Surface tension coef.	γ	N/m	0.0235
Liq. rel. permittivity	$\varepsilon_l/\varepsilon_o$	-	2.2
Gas rel. permittivity	$\varepsilon_g/\varepsilon_o$	-	1.0
Liq. ionic mobility	κ_l	m^2/V·s	1e-9
Liq. molecular diff.	D_l	m^2/s	1e-8

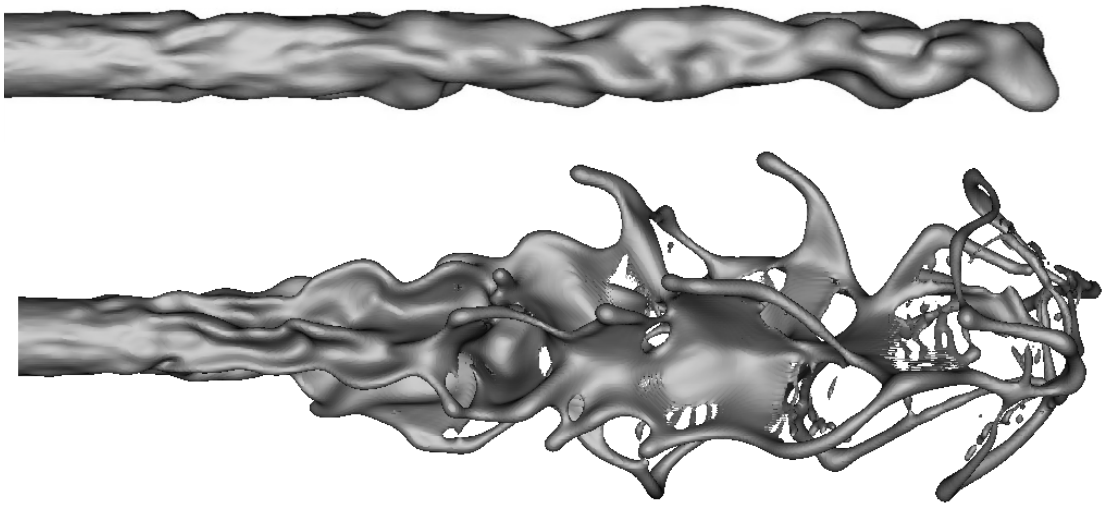


Figure 6.21: Snapshots of the uncharged kerosene jet (top) and the EHD enhanced atomizing jet (bottom) computed using the small computational domain.

electrically charged jet computed on the larger domain. The presence of the electric charges clearly continues to enhance the atomization process as the jet evolves further into the computational domain.

An interesting feature of the flow is highlighted in Fig. 6.24 where the electric charge density on the surface of the jet is shown. Clearly the electric charge density

is highest is droplets that are farthest from the center of the jet. This is because when the liquid is injected the mutual repulsion of electric charges creates electric charge boundary layers on the gas-liquid interface. This high concentration of electric charges is maintained when the contiguous liquid jet breaks into droplets. Furthermore, the highly charged droplets will experience an larger Coulomb force and the atomization process is enhanced due to the spatial variations in electric charge density. Note however that the variability in electric charge density is less than 5% of the injected charge density, so while the spatial variability exists it is small. Therefore, a reasonable approximation is to assume the electric charge density is constant within the liquid phase as was done by Van Poppel et al. [77].

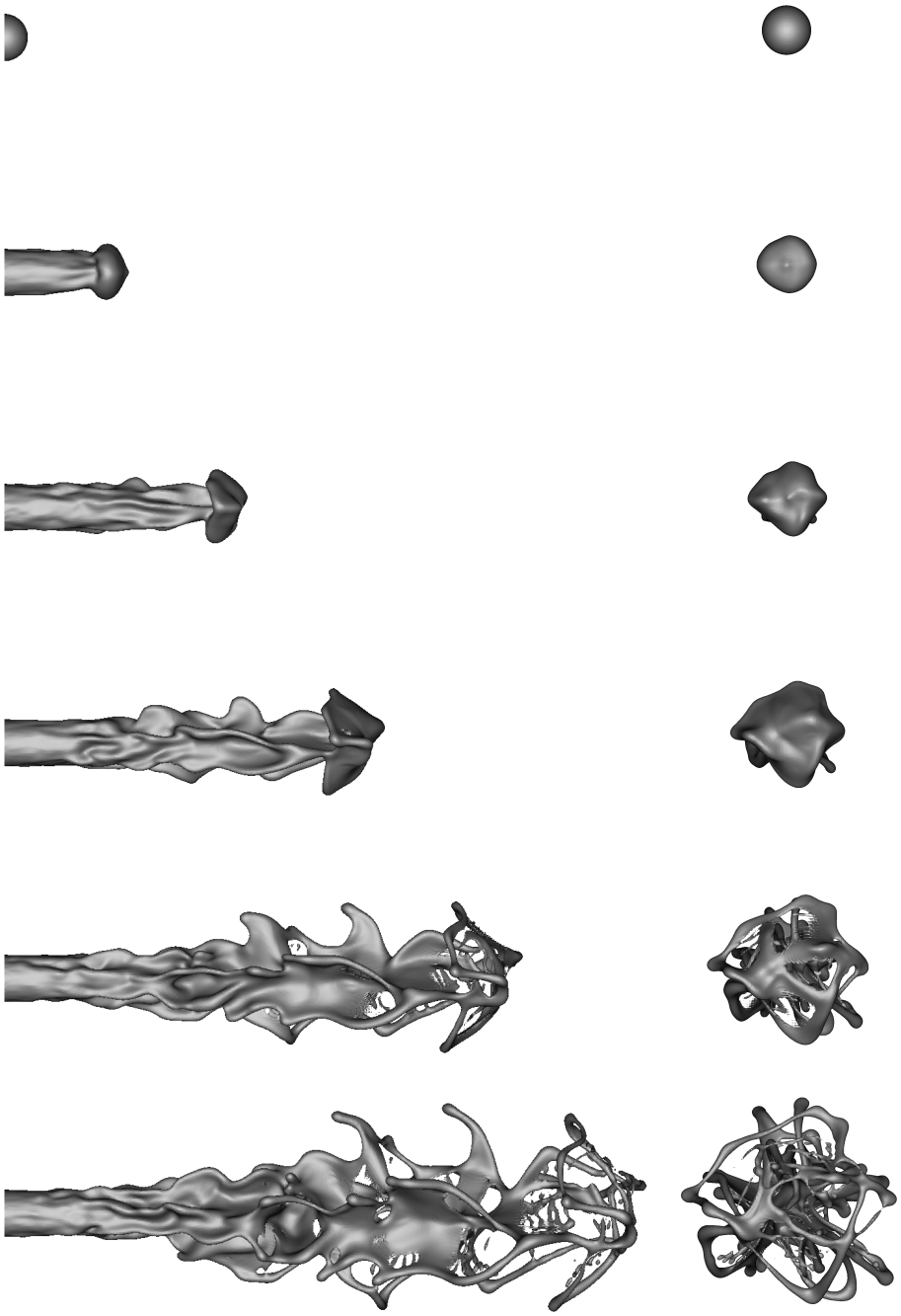


Figure 6.22: Snapshots of EHD enhanced kerosene atomization simulation. Time varies from $t = 0.0$ s (top) to $t = 0.0005$ s (bottom). side view (left), front view (right).

Figure 6.23: Snapshot of the charged EHD enhanced jet computed on the large domain.

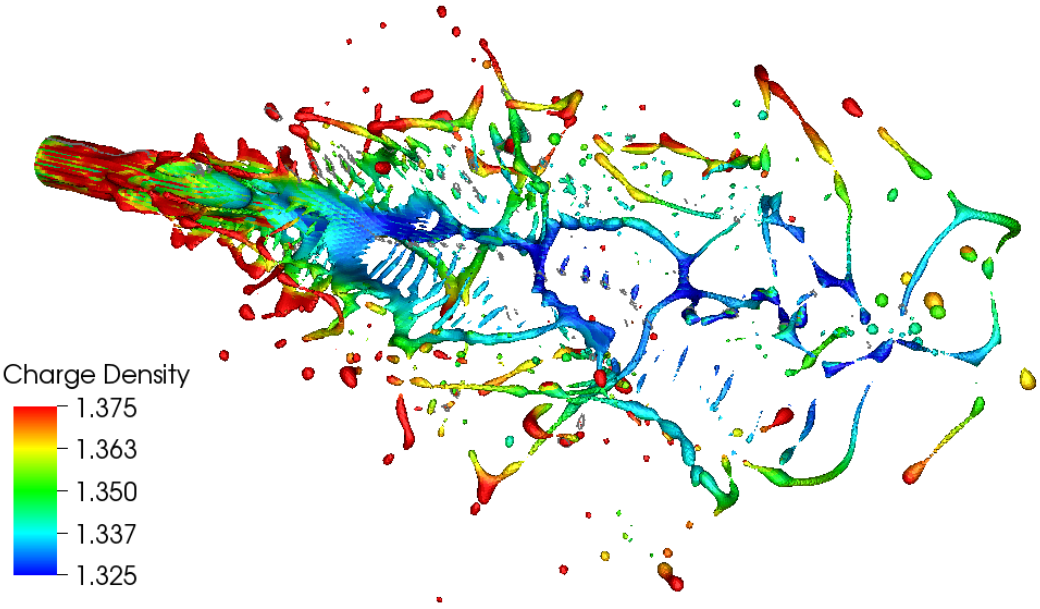


Figure 6.24: Electric charge density for EHD enhanced kerosene atomization simulation.

6.4 Conclusions

In this chapter, the numerical methods described throughout this dissertation have been used to simulate three realistic atomizing flows. The simulations highlight the usefulness in using numerical simulations to predict atomizing flows. Comparison of the simulation results to experiments demonstrates the numerical methods are capable of accurately predicting the multiphase dynamics. The numerical simulations provide a three-dimensional, time varying description of the flow field and interface dynamics. This wealth of data has the potential to advance our understanding of these flows.

CHAPTER 7

CONCLUSIONS

This dissertation describes numerical methods that allow for robust and accurate simulations of gas-liquid flows. The capability to accurately perform predictive simulations of these flows has the potential to significantly improve our understanding of the complex systems and aide in the engineering design of devices that exploit multiphase flows. However, the complexities of a deforming phase interface and the associated discontinuities have limited the availability of predictive simulations. The work in this dissertation serves to advance the state-of-the-art of numerical methods and allow for robust and accurate simulations of relevant engineering devices.

Two novel interface tracking schemes are proposed to improve the accuracy and conservation properties of such methods. The first is a discontinuous Galerkin discretization of the conservative level set. This discretization allows for a more accurate representation of the level set function without the need of a large computational stencil. The scheme is shown to have excellent scalability while improving the mass conservation properties of the conservative level set method. The second interface tracking scheme is based on the volume-of-fluid methodology and provides discrete conservation of mass. The scheme constructs un-split, three-dimensional, conservative fluxes using two key ideas that makes it straightforward to implement. The first is to represent the complex flux geometry using simplices, which are easier to manipulate computationally. The second is a novel sign convention that greatly simplifies how crossed fluxes are dealt with. Both interface tracking methods are verified using a variety of canonical test cases.

A numerical discretization for conservation laws with discontinuities at the phase interface is presented. The method is constructed such that discrete consistency with the volume-of-fluid interface transport is enforced. Consistency ensures the discontinuities are handled sharply and conservatively. The resulting scheme is tested and found to have discrete conservation and second-order accuracy.

In this work, the height function method, which is commonly used in the context of volume-of-fluid schemes to compute the interface curvature, is extended and applied to the conservative level set. Additionally, an advancement to the height function method is proposed that reduces curvature errors for under-resolved interfaces. Such interfaces are commonly found in simulations of atomizing flows where non-trivial topology changes occur on the same length-scale as the computational mesh.

These numerical methods have been used to simulate relevant atomizing flows. Namely, a liquid jet in cross-flow, an air-blast atomizing jet, and an electrically charged jet. The simulated results are compared to available experimental data and show excellent agreement suggesting the numerical methods are capable of predictive simulations of atomization.

Using simulations to study atomization has the potential to greatly improve our understanding of these strikingly dynamic and complex flows. Simulations will allow researchers to visualize and measure these flows in ways unattainable with current experimental techniques. With more knowledge of the important physical processes that govern multiphase flows, hopefully researchers can develop reduced order models that design engineers can use to improve the efficiency of engineering devices. Ultimately, this work may allow fuel injection systems to become more

efficient and mitigate the harmful effects of burning liquid fuels.

7.1 Future work

The numerical methods described in this dissertation provide a framework that is capable of simulating gas-liquid flows with second-order accuracy and discrete conservation while remaining robust in the presence of large density ratios. However, some aspects of the simulation methodology could benefit from improvements or extensions and are described below.

Phase change

Atomizing liquid jets typically create many small droplets. These droplets have a large surface area to volume ratio and evaporation will be important for their temporal evolution. Additionally, evaporation or condensation is important for many other application and developing a methodology for phase change will be an important aspect for predictive simulations of many flows. Phase change has been modeled previously, see for example [128], however, developing a conservative methodology that is consistent with the interface tracking scheme would be beneficial for predictive simulations of realistic turbulent atomizing flows with large density ratios, significant interface deformation, and phase change. This is a realistic combination that to the best of our knowledge has not been simulated.

High-order volume-of-fluid method

The volume-of-fluid scheme described in Chapter 3 is second order accurate. This method is constructed by transporting the liquid-volume fraction. There is potential to develop a scheme that transports higher-order moments. The moments could be used to construct higher-order volume-of-fluid schemes or to transport additional interface quantities such as the interface normal vector. A class of

schemes known as moment-of-fluid methods [129] have been developed but there are unanswered questions about what are the best moments to transport, how to transport the moments, how to use those moments to build an interface reconstruction. Viable candidates for moments include the barycenter of the liquid and the barycenter of the interface. Reconstructions can be a single linear function, multiple linear functions, or a higher order function.

Mesh-independent solutions

In simulations of gas-liquid flows, the interface dynamics are often dictated by the computational mesh. For example, when a ligament breaks into droplets the moment the ligament breaks is controlled by molecular dynamics and not described by the continuum equations we are solving. In a simulation, the breaking event is typically modeled using the limits of the computational mesh making the solution at some level dependent on the mesh. The development of a mesh-independent framework for interface topology changes would alleviate this constraint or a detailed study of the impacts the mesh dependency on the solution would be beneficial.

Large-eddy simulations of multiphase flows

Simulations of multiphase flows tend to be computational expensive due to the wide range of length and time scales. Therefore, the use of large-eddy simulations (LES) wherein only the large scales are resolved has the potential to reduce the cost and time needed to perform multiphase simulations. However, sub-grid phase interface dynamics must be modeled. Herrmann [130] provided a methodology wherein the phase interface is resolved on an auxiliary fine grid and the effects are filtered onto a coarser flow-solver LES mesh. However, the cost of solving the interface dynamics on a fine mesh will be substantial and may hinder the advantages of performing an LES. Alternatively, sub-grid interface scales could be

modeled using a more detailed understanding of interface dynamics on the smallest scales.

Uncertainty quantification

Quantifying uncertainty in computational fluid dynamics problems is an important and often neglected component of simulations. Effort is typically focused on determining the convergence order of numerical schemes used to discretize the governing equations; however, the fluid properties, boundary conditions, and initial conditions are usually taken to be exact even when large uncertainties are present in their definitions. Uncertainty quantification (UQ) is a rigorous methodology used to compute uncertainties in numerical models. While many UQ approaches are prohibitively expensive, a generalized polynomial chaos method has a reasonable computational cost [131]. Polynomial chaos has been used to study relevant canonical flows such as flow past a cylinder [132]. The extension of the methodology to multiphase flows has not been performed but has the potential to improve the credence of multiphase flow simulations by providing valuable information on the uncertainty in the simulation results.

BIBLIOGRAPHY

[1] Summary for policymakers. in: Climate change 2013: The physical science basis. contribution of working group I to the fifth assessment report of the Intergovernmental Panel on Climate Change, Tech. rep., IPCC, Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA (2013).

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

[3] J. Hernández, J. López, P. Gómez, C. Zanzi, F. Faura, A new volume of fluid method in three dimensions—Part i: Multidimensional advection method with face-matched flux polyhedra, International Journal of Numerical Methods in Fluids 58 (8) (2008) 897–921.

[4] Y. Gopala, Breakup characteristics of a liquid jet in subsonic crossflow, Ph.D. thesis, Georgia Institute of Technology (2012).

[5] Annual energy outlook 2013 with projections to 2040, Tech. Rep. DOE/EIA-0383, U.S. Energy Information Administration (2013).

[6] I. Grant, Particle image velocimetry: A review, Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 211 (1) (1997) 55–76.

[7] Y. Yeh, H. Z. Cummins, Localized fluid flow measurements with an HeNe laser spectrometer, Applied Physics Letters 4 (10) (1964) 176–178.

[8] D. N. White, Johann christian doppler and his effectA brief history, Ultrasound in Medicine & Biology 8 (6) (1982) 583–591.

[9] J. B. Blaisot, J. Yon, Droplet size and morphology characterization for dense sprays by image processing: application to the diesel spray, Experiments in Fluids 39 (6) (2005) 977–994.

[10] K.-C. Lin, M. Ryan, A. Sandy, S. Narayanan, J. Ilavsky, J. Wang, Investigation of droplet properties of supercritical ethylene jets using small angle X-ray scattering (SAXS) technique, Orlando, FL, 2008.

[11] S. Moon, Z. Liu, J. Gao, E. Dufresne, K. Fezzaa, J. Wang, X. Xie, M. C. Lai, Ultrafast X-ray phase-contrast imaging of high-speed fuel sprays from a two-hole diesel nozzle, in: ILASS Americas, 22nd annual conference on liquid atomization and spray systems, 2010.

[12] K.-C. Lin, C. Carter, K. Fezzaa, J. Wang, Z. Liu, X-ray study of pure- and aerated-liquid jets in a quiescent environment, in: 47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics, 2009.

[13] D. Moncton, The Advanced Photon Source a national sychrotrn radiation research facility, Argonne National Laboratory ANL/APS/TB-25-Rev. (1997).

[14] P. Marmottant, E. Villermaux, On spray formation, Journal of Fluid Mechanics 498 (2004) 73–111.

[15] G. M. Faeth, L. P. Hsiang, P. K. Wu, Structure and breakup properties of sprays, International Journal of Multiphase Flow 21 (Supplement 1) (1995) 99–127.

[16] J. U. Brackbill, D. B. Kothe, C. Zemach, A continuum method for modeling surface tension, Journal of Computational Physics 100 (2) (1992) 335–354.

[17] R. P. Fedkiw, T. Aslam, B. Merriman, S. Osher, A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method), Journal of Computational Physics 152 (2) (1999) 457–492.

[18] W. Dettmer, P. H. Saksono, D. Perić, On a finite element formulation for incompressible Newtonian fluid flows on moving domains in the presence of surface tension, Communications in Numerical Methods in Engineering 19 (9) (2003) 659–668.

[19] C. W. Hirt, A. A. Amsden, J. L. Cook, An arbitrary Lagrangian-Eulerian computing method for all flow speeds, Journal of Computational Physics 135 (2) (1997) 203–216.

[20] T. J. R. Hughes, W. K. Liu, T. K. Zimmermann, Lagrangian-Eulerian finite element formulation for incompressible viscous flows, Computer Methods in Applied Mechanics and Engineering 29 (3) (1981) 329–349.

[21] J. Sarrate, A. Huerta, J. Donea, Arbitrary Lagrangian-Eulerian formulation

for fluid-rigid body interaction, Computer Methods in Applied Mechanics and Engineering 190 (24-25) (2001) 3171–3188.

[22] M. Rudman, Volume-tracking methods for interfacial flow calculations, International Journal for Numerical Methods in Fluids 24 (7) (1997) 671–691.

[23] S. Osher, J. A. Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations, Journal of Computational Physics 79 (1) (1988) 12–49.

[24] J. Sethian, Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Cambridge University Press, 1999.

[25] C. Hirt, B. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, Journal of Computational Physics 39 (1) (1981) 201–225.

[26] J. E. Pilliod, E. G. Puckett, Second-order accurate volume-of-fluid algorithms for tracking material interfaces, Journal of Computational Physics 199 (2) (2004) 465–502.

[27] R. Scardovelli, S. Zaleski, Direct numerical simulation of free-surface and interfacial flow, Annual Review of Fluid Mechanics 31 (1) (1999) 567–603.

[28] O. Desjardins, V. Moureau, H. Pitsch, An accurate conservative level set/ghost fluid method for simulating turbulent atomization, Journal of Computational Physics 227 (18) (2008) 8395–8416.

[29] E. Olsson, G. Kreiss, A conservative level set method for two phase flow, Journal of Computational Physics 210 (1) (2005) 225–246.

[30] E. Olsson, G. Kreiss, S. Zahedi, A conservative level set method for two phase flow II, Journal of Computational Physics 225 (1) (2007) 785–807.

[31] R. DeBar, Fundamentals of the KRAKEN code, Tech. Rep. UCIR-760, LLNL (1974).

[32] B. Nichols, C. Hirt, Methods for calculating multi-dimensional, transient free surface flows past bodies, Tech. Rep. LA-UR-75-1932, Los Alamos National Laboratory (1975).

[33] W. F. Noh, P. Woodward, SLIC (simple line interface calculation), in: Proceedings of the Fifth International Conference on Numerical Methods in

Fluid Dynamics, no. 59 in Lect. Notes Phys., Springer Berlin Heidelberg, 1976, pp. 330–340.

[34] D. Youngs, Time-dependent multi-material flow with large fluid distortion, Numerical Methods for Fluid Dynamics (1982) 273–285.

[35] W. J. Rider, D. B. Kothe, Reconstructing volume tracking, Journal of Computational Physics 141 (2) (1998) 112–152.

[36] V. Le Chenadec, H. Pitsch, A 3D unsplit Forward/Backward volume-of-fluid approach and coupling to the level set method, Journal of Computational Physics 233 (0) (2013) 10–33.

[37] G. Tryggvason, R. Scardovelli, S. Zaleski, Direct Numerical Simulations of Gas-Liquid Multiphase Flows, Cambridge University Press, 2011.

[38] M. D. Torrey, L. D. Cloutman, R. C. Mjolsness, C. W. Hirt, NASA-VOF2D: a computer program for incompressible flows with free surfaces, NASA STI/Recon Technical Report N 86 (1985) 30116.

[39] B. D. Nichols, C. W. Hirt, R. S. Hotchkiss, SOLA-VOF: a solution algorithm for transient fluid flow with multiple free boundaries, NASA STI/Recon Technical Report N 81 (1980) 14281.

[40] J. Helmsen, P. Colella, E. G. Puckett, Non-convex profile evolution in two dimensions using volume of fluids, Tech. Rep. LBNL–40693, Lawrence Berkeley Lab., CA (United States) (Jun. 1997).

[41] M. Owkes, O. Desjardins, A discontinuous Galerkin conservative level set scheme for interface capturing in multiphase flows, Journal of Computational Physics 249 (15) (2013) 275–302.

[42] M. Owkes, O. Desjardins, A computational framework for conservative, three-dimensional, unsplit, geometric transport with application to the volume-of-fluid (VOF) method, Journal of Computational Physics 270 (1) (2014) 587–612.

[43] O. Desjardins, J. McCaslin, M. Owkes, P. Brady, Direct numerical and large-eddy simulation of primary atomization in complex geometries, Atomization and Sprays 23 (11) (2013) 1001–1048.

[44] P. Rasetarinera, M. Y. Hussaini, An efficient implicit discontinuous spectral Galerkin method, Journal of Computational Physics 172 (2) (2001) 718–738.

[45] J. Remacle, N. Chevaugeon, E. Marchandise, C. Geuzaine, Efficient visualization of high-order finite elements, International Journal for Numerical Methods in Engineering 69 (4) (2007) 750–771.

[46] B. Cockburn, C. Shu, Runge-Kutta discontinuous Galerkin methods for convection-dominated problems, Journal of Scientific Computing 16 (3) (2001) 173–261.

[47] E. Marchandise, J. Remacle, N. Chevaugeon, A quadrature-free discontinuous Galerkin method for the level set equation, Journal of Computational Physics 212 (1) (2006) 338–357.

[48] E. Marchandise, P. Geuzaine, N. Chevaugeon, J. Remacle, A stabilized finite element method using a discontinuous level set approach for the computation of bubble dynamics, Journal of Computational Physics 225 (1) (2007) 949 – 974.

[49] M. M. Francois, S. J. Cummins, E. D. Dendy, D. B. Kothe, J. M. Sicilian, M. W. Williams, A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework, Journal of Computational Physics 213 (1) (2006) 141–173.

[50] D. L. Chopp, Some improvements of the fast marching method, SIAM Journal of Scientific Computing 23 (1) (2001) 230–244.

[51] D. L. Chopp, Computing minimal surfaces via level set curvature flow, Journal of Computational Physics 106 (1) (1993) 77–91.

[52] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible Two-Phase flow, Journal of Computational Physics 114 (1) (1994) 146–159.

[53] B. Cockburn, C. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II. general framework, Mathematics of Computation 52 (186) (1989) 411–35.

[54] H. Luo, L. Luo, R. Nourgaliev, V. A. Mousseau, N. Dinh, A reconstructed discontinuous Galerkin method for the compressible Navier-Stokes equations

on arbitrary grids, Journal of Computational Physics 229 (19) (2010) 6961–6978.

[55] X. Zhang, C. Shu, Maximum-principle-satisfying and positivity-preserving high-order schemes for conservation laws: survey and new developments, Proceedings of the Royal Society of London. Series A: Mathematical and physical sciences 467 (2011) 2752–2776.

[56] O. Desjardins, G. Blanquart, G. Balarac, H. Pitsch, High order conservative finite difference scheme for variable density low mach number turbulent flows, Journal of Computational Physics 227 (15) (2008) 7125–7159.

[57] S. T. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids, Journal of Computational Physics 31 (3) (1979) 335–362.

[58] A. Prosperetti, Motion of two superposed viscous fluids, Physics of Fluids 24 (7) (1981) 1217–1223.

[59] M. Herrmann, A balanced force refined level set grid method for two-phase flows on unstructured flow solver grids, Journal of Computational Physics 227 (4) (2008) 2674–2706.

[60] S. Popinet, S. Zaleski, A front-tracking algorithm for accurate representation of surface tension, International Journal for Numerical Methods in Fluids 30 (6) (1999) 775–793.

[61] A. Bagué, D. Fuster, S. Popinet, R. Scardovelli, S. Zaleski, Instability growth rate of two-phase mixing layers from a linear eigenvalue problem and an initial-value problem, Physics of Fluids 22 (9) (2010) 92–104.

[62] P. A. M. Boomkamp, B. J. Boersma, R. H. M. Miesen, G. V. Beijnon, A Chebyshev collocation method for solving two-phase flow stability problems, Journal of Computational Physics 132 (2) (1997) 191–200.

[63] J. McCaslin, O. Desjardins, A localized re-initialization equation for the conservative level set method, Journal of Computational Physics 262 (2) (2014) 408–426.

[64] H. T. Ahn, M. Shashkov, Multi-material interface reconstruction on generalized polyhedral meshes, Journal of Computational Physics 226 (2) (2007) 2096–2132.

[65] V. Le Chenadec, H. Pitsch, A monotonicity preserving conservative sharp interface flow solver for high density ratio two-phase flows, Journal of Computational Physics 249 (2013) 185–203.

[66] T. Maric, H. Marschall, D. Bothe, voFoam - a geometrical volume of fluid algorithm on arbitrary unstructured meshes with local dynamic adaptive mesh refinement using OpenFOAM, arXiv e-print 1305.3417 (May 2013).

[67] C. B. Ivey, P. Moin, Conservative volume of fluid advection method on unstructured grids in three dimensions, Center for Turbulence Research Annual Research Briefs (2012) 179–192.

[68] J. López, J. Hernández, Analytical and geometrical tools for 3D volume of fluid methods in general grids, Journal of Computational Physics 227 (12) (2008) 5939–5948.

[69] D. B. Kothe, W. J. Rider, S. J. Mosso, J. S. Brock, J. I. Hochstein, Volume tracking of interfaces having surface tension in two and three dimensions, in: AIAA 34th Aerospace Sciences Meeting and Exhibit, 1996.

[70] R. Scardovelli, S. Zaleski, Analytical relations connecting linear interfaces and volume fractions in rectangular grids, Journal of Computational Physics 164 (1) (2000) 228–237.

[71] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes 3rd Edition: The Art of Scientific Computing, Cambridge University Press, 2007.

[72] D. M. Y. Sommerville, An introduction to the geometry of n dimensions, Dover Publications, 1958.

[73] P. Liovic, M. Rudman, J.-L. Liow, D. Lakehal, D. Kothe, A 3D unsplit-advection volume tracking algorithm with planarity-preserving interface reconstruction, Computers & Fluids 35 (10) (2006) 1011–1032.

[74] J. Mencinger, I. Žun, A PLIC–VOF method suited for adaptive moving grids, Journal of Computational Physics 230 (3) (2011) 644–663.

[75] R. J. Leveque, High-resolution conservative algorithms for advection in incompressible flow, SIAM Journal on Numerical Analysis 33 (2) (1996) 627–665.

[76] J. Shrimpton, Pulsed charged sprays: application to DISI engines during early injection, International Journal for Numerical Methods in Engineering 58 (3).

[77] B. Van Poppel, O. Desjardins, J. Daily, A ghost fluid, level set methodology for simulating multiphase electrohydrodynamic flows with application to liquid fuel injection, Journal of Computational Physics 229 (20) (2010) 7977–7996.

[78] X.-D. Liu, S. Osher, T. Chan, Weighted essentially non-oscillatory schemes, Journal of Computational Physics 115 (1) (1994) 200–212.

[79] G.-S. Jiang, C.-W. Shu, Efficient implementation of weighted ENO schemes, Journal of Computational Physics 126 (1) (1996) 202–228.

[80] B. P. Leonard, A stable and accurate convective modelling procedure based on quadratic upstream interpolation, Computer Methods in Applied Mechanics and Engineering 19 (1) (1979) 59–98.

[81] R. W. MacCormack, Iterative modified approximate factorization, Computers & Fluids 30 (78) (2001) 917–925.

[82] P. Brady, O. Desjardins, A sharp, robust, discretely conservative cut-cell immersed boundary technique for complex three dimensional geometries, Journal of Computational Physics, under review.

[83] S. J. Cummins, M. M. Francois, D. B. Kothe, Estimating curvature from volume fractions, Computers & Structures 83 (6–7) (2005) 425–434.

[84] S. Afkhami, M. Bussmann, Height functions for applying contact angles to 2D VOF simulations, International Journal for Numerical Methods in Fluids 57 (4) (2008) 453–472.

[85] M. Sussman, K. Smith, M. Hussaini, M. Ohta, R. Zhi-Wei, A sharp interface method for incompressible two-phase flows, Journal of Computational Physics 221 (2) (2007) 469–505.

[86] M. Sussman, A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles, Journal of Computational Physics 187 (1) (2003) 110–136.

[87] G. Bornia, A. Cervone, S. Manservisi, R. Scardovelli, S. Zaleski, On the

properties and limitations of the height function method in two-dimensional cartesian geometry, Journal of Computational Physics 230 (4) (2011) 851–862.

[88] S. Popinet, An accurate adaptive solver for surface-tension-driven interfacial flows, Journal of Computational Physics 228 (16) (2009) 5838–5866.

[89] Y. Renardy, M. Renardy, PROST: a parabolic reconstruction of surface tension for the volume-of-fluid method, Journal of Computational Physics 183 (2) (2002) 400–421.

[90] B. Lafaurie, C. Nardone, R. Scardovelli, S. Zaleski, G. Zanetti, Modelling merging and fragmentation in multiphase flows with SURFER, Journal of Computational Physics 113 (1) (1994) 134–147.

[91] M. Rudman, A volume-tracking method for incompressible multifluid flows with large density variations, International Journal for Numerical Methods in Fluids 28 (2) (1998) 357–378.

[92] J. O. McCaslin, O. Desjardins, A localized re-initialization equation for the conservative level set method, Journal of Computational Physics 262 (2014) 408–426.

[93] O. Desjardins, V. Moureau, Methods for multiphase flows with high density ratio, Center for Turbulence Research Proceedings of the Summer Program (2010) 313.

[94] M. F. Smith, T. J. O'Hern, J. E. Brockmann, A comparison of two laser-based diagnostics for analysis of particles in thermal spray streams, Tech. Rep. SAND–95-1442C; CONF-9509182–3, Sandia National Labs., Albuquerque, NM (United States) (Jul. 1995).

[95] M. Herrmann, A parallel Eulerian interface tracking/Lagrangian point particle multi-scale coupling procedure, Journal of Computational Physics 229 (3) (2010) 745–759.

[96] H. Yudistira, V. D. Nguyen, P. Dutta, D. Byun, Flight behavior of charged droplets in electrohydrodynamic inkjet printing, Applied Physics Letters 96 (2) (2010) 023503–023503–3.

[97] J. B. Fenn, M. Mann, C. K. Meng, S. F. Wong, C. M. Whitehouse, Elec-

trospray ionization for mass spectrometry of large biomolecules, Science 246 (4926) (1989) 64–71, PMID: 2675315.

[98] K. Kim, R. J. Turnbull, Generation of charged drops of insulating liquids by electrostatic spraying, Journal of Applied Physics 47 (1964).

[99] I. Hayati, A. I. Bailey, T. F. Tadros, Mechanism of stable jet formation in electrohydrodynamic atomization, Nature 319 (6048) (1986) 41–43.

[100] D. P. H. Smith, The electrohydrodynamic atomization of liquids, IEEE Transactions on Industry Applications IA-22 (3) (1986) 527–535.

[101] M. Cloupeau, B. Prunet-Foch, Electrohydrodynamic spraying functioning modes: a critical review, Journal of Aerosol Science 25 (6) (1994) 1021–1036.

[102] A. M. Gan-Calvo, J. Dvila, A. Barrero, Current and droplet size in the electrospraying of liquids. scaling laws, Journal of Aerosol Science 28 (2) (1997) 249–275.

[103] J. M. Lpez-Herrera, A. Barrero, A. Lpez, I. G. Loscertales, M. Mrquez, Coaxial jets generated from electrified taylor cones. scaling laws, Journal of Aerosol Science 34 (5) (2003) 535–552.

[104] S. K. Cho, H. Moon, C.-J. Kim, Creating, transporting, cutting, and merging liquid droplets by electrowetting-based actuation for digital microfluidic circuits, Journal of Microelectromechanical Systems 12 (1) (2003) 70–80.

[105] M. Felten, W. Staroske, M. S. Jaeger, P. Schwille, C. Duschl, Accumulation and filtering of nanoparticles in microchannels using electrohydrodynamically induced vortical flows, Electrophoresis 29 (14) (2008) 29872996.

[106] P. Kazemi, P. Selvaganapathy, C. Ching, Electrohydrodynamic micropumps with asymmetric electrode geometries for microscale electronics cooling, IEEE Transactions on Dielectrics and Electrical Insulation 16 (2) (2009) 483–488.

[107] D. J. Laser, J. G. Santiago, A review of micropumps, Journal of Micromechanics and Microengineering 14 (6) (2004) R35.

[108] O. D. Velev, B. G. Prevo, K. H. Bhatt, On-chip manipulation of free droplets, Nature 426 (6966) (2003) 515–516.

[109] J. Zeng, T. Korsmeyer, Principles of droplet electrohydrodynamics for lab-on-a-chip, Lab on a Chip 4 (4) (2004) 265–277.

[110] S. Edward Law, Agricultural electrostatic spray application: a review of significant research and development during the 20th century, Journal of Electrostatics 5152 (2001) 25–42.

[111] W. Lehr, W. Hiller, Electrostatic atomization of liquid hydrocarbons, Journal of Electrostatics 30 (1993) 433–440.

[112] H. Romat, A. Badri, Internal electrification of diesel oil injectors, Journal of Electrostatics 5152 (2001) 481–487.

[113] J. Shrimpton, A. Kourmatzis, Direct numerical simulation of forced flow dielectric EHD within charge injection atomizers, IEEE Transactions on Dielectrics and Electrical Insulation (2010) 18.

[114] J. S. Shrimpton, Y. Laoonual, Dynamics of electrically charged transient evaporating sprays, International Journal for Numerical Methods in Engineering 67 (8) (2006) 1063–1081.

[115] J. Shrimpton, A. Yule, Characterisation of charged hydrocarbon sprays for application in combustion systems, Experiments in Fluids 26 (5) (1999) 460469.

[116] J. Shrimpton, A. Yule, Atomization, combustion, and control of charged hydrocarbon sprays, Journal of Atomization and Sprays 11 (2001) 365396.

[117] J. Shrimpton, A. Yule, Electrohydrodynamics of charge injection atomization: Regimes and fundamental limits, Journal of Atomization and Sprays 13 (2003) 173190.

[118] J. Shrimpton, A. Yule, Design issues concerning charge injection atomizers, Journal of Atomization and Sprays 14 (2004) 127142.

[119] J. Shrimpton, Charge Injection Systems: Physical Principles, Experimental and Theoretical Work, Springer-Verlag, 2009.

[120] J. Shrimpton, Y. Laoonual, Dynamics of electrically charged transient evaporating sprays, International Journal for Numerical Methods in Engineering 67 (8).

[121] A. Yule, J. Shrimpton, A. Watkins, W. Balachandran, D. Hu, Electrostatically atomized hydrocarbon sprays, Fuel 74 (7) (1995) 1094–1103.

[122] J. Lpez-Herrera, S. Popinet, M. Herrada, A charge-conservative approach for simulating electrohydrodynamic two-phase flows using volume-of-fluid, Journal of Computational Physics 230 (5) (2011) 1939–1955.

[123] G. Tomar, D. Gerlach, G. Biswas, N. Alleborn, A. Sharma, F. Durst, S. W. J. Welch, A. Delgado, Two-phase electrohydrodynamic simulations using a volume-of-fluid approach, Journal of Computational Physics 227 (2) (2007) 1267–1285.

[124] D. A. Saville, Electrohydrodynamics: The Taylor-Melcher leaky dielectric model, Annual Review of Fluid Mechanics 29 (1) (1997) 27–64.

[125] A. Kourmatzis, J. S. Shrimpton, Electrohydrodynamics and charge injection atomizers: A review of the governing equations and turbulence, Journal of Atomization and Sprays 19 (2009) 10451063.

[126] A. J. Kelly, Electrostatic atomizing device, US patent 4255777 a (Mar. 1981).

[127] M. Owkes, O. Desjardins, Consistent and conservative computational framework for simulations of electrohydrodynamic atomization, in: ILASS Americas, 26nd annual conference on liquid atomization and spray systems, 2014.

[128] Y. Sato, B. Nieno, A sharp-interface phase change model for a mass-conservative interface tracking method, Journal of Computational Physics 249 (2013) 127–161.

[129] V. Dyadechko, M. Shashkov, Moment-of-fluid interface reconstruction, Tech. Rep. LA-UR-05-7571, Los Alamos National Laboratory.

[130] M. Herrmann, A surface tension sub-grid model for phase interface dynamics, Center for Turbulence Research Proceedings of the Summer Program (2010) 333.

[131] O. Knio, O. Le Matre, Uncertainty propagation in CFD using polynomial chaos decomposition, Fluid Dynamics Research 38 (9) (2006) 616–640.

[132] D. Venturi, X. Wan, G. E. Karniadakis, Stochastic low-dimensional modelling of a random laminar wake past a circular cylinder, Journal of Fluid Mechanics 606 (2008) 339–367.