CONSERVATIVE SIMULATIONS OF ATOMIZATION

APPLYING THE HEIGHT FUNCTION METHOD TO RUDMAN DUAL GRIDS

by

Kristopher Thomas Olshefski

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Mechanical Engineering

MONTANA STATE UNIVERSITY
Bozeman, Montana

November 2019

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

LIST OF FIGURES – CONTINUED

Figure Page

LIST OF FIGURES – CONTINUED

Figure Page

LIST OF FIGURES – CONTINUED

Figure Page

NOMENCLATURE

μ Dynamic viscosity
$\mathbf{n}$ Normal vector
$\mathbf{u}$ Velocity vector

ABSTRACT

Gas-liquid flows can be significantly influenced by the surface tension force, which controls the shape of the interface. The surface tension force is directly proportional to the interface curvature and an accurate calculation of curvature is essential for predictive simulations of the flow types. Furthermore, methods that consistently and conservatively transport momentum, which is discontinuous at the gas-liquid interface, are necessary for robust and accurate simulations. Using a Rudman dual mesh, which discretizes density on a twice as fine mesh, provides consistent and conservative discretizations of mass and momentum. The height function method is a common technique to compute an accurate curvature as it is straightforward to implement and provides a second-order calculation.

When a dual grid is used, the standard height function method fails to capture fine grid interface perturbations and these perturbations can grow. When these growing perturbations are left uncorrected, they can result in nonphysical dynamics and eventual simulation failure. This work extends the standard height function method to include information from the Rudman dual mesh. The proposed method leverages a fine-grid height function method to compute the fine-gird interface perturbations and uses a fine-grid velocity field to oppose the fine-grid perturbations. This approach maintains consistent mass and momentum transport while also providing accurate interface transport that avoids non-physical dynamics. The method is tested using an oscillating droplet test case and compared to a standard height function. Various iterations of the fine grid method are presented and strengths and shortcomings of each are discussed.

1

INTRODUCTION

The study of fluid dynamics covers a wide array of phenomena across wide spans of scales. Current research can be found exploring everything from biomedical simulations of capillary flows [38] to astrophysical approximations of star formations [1]. Researchers are studying bio-inspired dynamics of animals to gain insight into efficiency shortcomings of man made vessels [3]. Other research focuses on oceanic current modeling to gain a deeper understanding of the natural world [35]. The focus of this research however, is on multiphase flows. In the context of this research, interest is restricted to flows where both a liquid and gas are present. These flows are referred to as gas-liquid flows, multiphase flows, or free surface flows in literature [23, 33, 37]. Examples of gas-liquid flows can be seen throughout the natural world as well as industrial application and may be some of the most wide spread and common flow types on Earth. Bubbles rising in a carbonated drink, rain falling through the air, paint exiting a can of spray paint, a wave on the surface of the ocean; these are all examples of gas-liquid multiphase flows. These types of flows are of particular academic interest due to the ubiquity with which they emerge in industrial and engineering applications. For example, for efficient combustion to occur in a gasoline or diesel engine, the fuel must first be atomized. Atomization is the process by which a liquid is broken from a stream into a fine mist or distribution of smaller droplets. This process occurs not only in automobile engines but also, paint spray applicators, aerosols, and fire sprinkler systems.

For engineering applications there are two main pathways by which multiphase flows are investigated. Experimentation is a common method of study inside many

scientific disciplines; the field of fluid dynamics is no different. Many researchers design experimental studies to gain deeper insight into the behavior of multiphase flows. Some examples of experimental techniques include shadowgraphy, x-ray imaging, or Particle image velocimetry (PIV) [10, 29, 36]. Computational fluid dynamics (CFD) is the second discipline used to study multiphase flows and is the focus this research. CFD utilizes the computational power of modern computing to digitally investigate fluid flow problems.

Within the field of CFD, simulation models are often classified by the scale to which they resolve turbulent structures. However, these methods all share the commonality that in some form or another, they all solve the Navier-Stokes equations which govern fluid flow. While many approaches exist, there are three main schemes: Reynolds Averaged Navier-Stokes (RANS), Large Eddy Simulation (LES), and Direct Numerical Simulation (DNS). These methods scale in fidelity respectively. RANS methods solve the time averaged Navier-Stokes equations. This is the least computationally intense method of the three but requires a modeling scheme for all turbulent structures. Due to its low computational cost, RANS models are most commonly seen in industry application where a rough approximation of the turbulence scale is sufficient for design purposes. LES models are more computationally expensive than RANS models but are still fairly common in industrial engineering applications as well as academic research. LES methods solve filtered Navier-Stokes equations. This means that a filter is applied to the governing equations and then a closure model allows for the numerical simulation of the equations [24]. The filtering operation and subsequent closure models give LES methods the ability to resolve some larger turbulent structures within a flow. While DNS methods are the most expensive of the three, they have the advantage of providing the most complete solution of the mathematical model. This is because direct simulations resolve the entire range of

turbulent scales with no modeling necessary. Due to the computational expense however, DNS methods are usually reserved for academic interests in which a critical understanding of the realistic flow characteristics is required. An example of a simulated atomizing jet can be seen in Figure 1.1. The image in this figure was produced from a DNS simulation. DNS methodology is the focus of this research.

Figure 1.1: DNS Simulation of Atomizing Jet

Based on their primary research focus, those involved in CFD research are often categorized into two groups; simulation focused or numerically focused. Simulation based research utilizes in-house or commercial software packages to answer specific questions about fluid flow. An example of this might be a researcher trying to optimize an airfoil shape for specific flight conditions. With any number of software packages, various shapes and flow conditions can be simulated to determine an optimal geometry. Numerical methods for fluid dynamics focus on the accurate and efficient implementation of the Navier-Stokes equations, the governing equations of fluid mechanics, into numerical algorithms. An example of this might be the development of a new, more computationally efficient, method for solving partial

differential equations. The focus of the research presented in this paper falls into this category.

Numerous challenges exist concerning numerical methods of multiphase flows. One challenge of particular interest is the accurate representation of the interface of the two fluids which exists for all gas-liquid multiphase flows. This intersection presents notable challenge because the transition from one phase to another is partially represented as a mathematical discontinuity. That is, the change in density of the two fluids at the interface, is decidedly difficult to approximate mathematically. Surface tension is the governing force controlling the shape and behavior of the interface. The surface force ($\mathbf{f}_\sigma$) is expressed as $\mathbf{f}_\sigma = \sigma\kappa\mathbf{n}$ [7]. Here, σ is given as the surface tension coefficient, a value empirically determined for fluids. In this research, the surface tension coefficient is assumed to remain constant. The vector perpendicular to the interface is $\mathbf{n}$. The curvature of the interface is given by κ. Various curvature estimation schemes exist. Review of the mechanics which are required for this estimation along with a novel curvature scheme are the focus of the research to be discussed in the remainder of this work.

THEORY

The following section is an attempt to summarize several fundamental concepts used in simulating fluid flows and lay the necessary framework for understanding the subsequent fine grid method, which is the primary focus of this work. Each of the methods discussed are present within NGA, the computational platform used in this research. Additionally, while these methods are prevalent in the CFD community, they are not the only options available. The interested reader is directed to [33] for a more comprehensive assessment.

Computational Platform

The proposed curvature estimation method exists as a module within a larger computational framework. While this scheme can be incorporated into any number of solvers, the platform used for this work was NGA [6, 8]. NGA solves low-Mach number, variable density formulations of mass and momentum conservation laws

$$\frac{\partial \rho_\phi}{\partial t} + \nabla \cdot (\rho_\phi \boldsymbol{u}_\phi) = 0 \quad \text{and} \tag{2.1}$$

$$\frac{\partial \rho_\phi \boldsymbol{u}_\phi}{\partial t} + \nabla \cdot (\rho_\phi \boldsymbol{u}_\phi \otimes \boldsymbol{u}_\phi) = -\nabla p_\phi + \nabla \cdot \left(\mu_\phi \left[\nabla \boldsymbol{u}_\phi + \nabla \boldsymbol{u}_\phi^{\mathsf{T}}\right]\right) + \rho_\phi \boldsymbol{g} \tag{2.2}$$

where ρ_ϕ is the density, $\boldsymbol{u}_\phi = [u, v, w]_\phi$ is the velocity field vector, t is time, p_ϕ is the hydrodynamic pressure, μ_ϕ is the dynamic viscosity, and $\boldsymbol{g}$ is the gravitational acceleration. The subscript ϕ indicates the phase and takes values of $\phi = g$ or $\phi = l$ in the gas or liquid phase, respectively.

These equations have been written in both the gas and liquid phases and are connected through jump conditions at the phase interface. For example, the jumps

in density and viscosity at the interface Γ are written as

$$[\rho]_\Gamma = \rho_l - \rho_g \quad \text{and} \tag{2.3}$$

$$[\mu]_\Gamma = \mu_l - \mu_g. \tag{2.4}$$

In the absence of a phase change, the velocity field is continuous, i.e.,

$$[\boldsymbol{u}]_\Gamma = 0. \tag{2.5}$$

The pressure is discontinuous due to contributions from surface tension and the normal component of the viscous stress, i.e.,

$$[p]_\Gamma = \sigma\kappa + 2\,[\mu]_\Gamma\, \boldsymbol{n}^\mathsf{T} \cdot \nabla \boldsymbol{u} \cdot \boldsymbol{n}, \tag{2.6}$$

where σ is the surface tension coefficient and κ is the interface curvature. This is the curvature that is computed with the height function method.

These equations are discretized using a Cartesian mesh with pressure and other scalars located at cell centers and velocity components located at cell faces. Time is discretized using an iterative second order Crank-Nicolson formulation with a semi-implicit correction on each subiteration [4]. The interface is represented with a geometric volume-of-fluid (VoF) method [19,22]. The NGA code is highly parallelized, allowing for scalable simulations and fast run times and has been applied to many atomization applications [7, 20, 30].

Streaktube Introduction

NGA uses structures known as streaktubes to advect scalar quantities through time. For this research, we need to move flux values of velocity through cell faces over

a time step. Streaktubes work by approximating an amount of a scalar value needing to be advected with time to a volumetric quantity. Integrating this volume over time ensures that values are conservative. Streaktube development relies on conservation laws applied to a fixed control volume. The following derivation presents advection of a scalar partial differential equation (PDE) into a scalar quantity as it progresses with time due to geometric flux. While the interested reader can be directed to **??** for similar derivations of equation 2.14, this derivation is valuable in that it shows exact relations between liquid volume fraction (α) transport, and a generally advected function, $f(x,t)$ [22]. This derivation shows that fluxes used within NGA to advect a function through time can be calculated by formulating streaktubes at each cell face of a control volume and evaluating that function at the current time step. The derivation provides rigorous validation that any conserved scalar quantity that fluxes through a control volume face at a given time step can be calculated in the previous time step as a volumetric quantity and, represented geometrically as such, as displayed by Figure 2.1.

Streaktube Derivation

Assume we have a conserved scalar $f(\boldsymbol{x},t)$ that evolves in a solenoidal velocity field as

$$\frac{\partial f}{\partial t} + \nabla \cdot (\boldsymbol{u} f) = 0 \tag{2.7}$$

where $\boldsymbol{x}$ is the spatial coordinate value, t is time, and $\boldsymbol{u}$ is the velocity field which is known [19]. Integrating this equation over a time step ($t^n \rightarrow t^{n+1}$) and the area of the control volume while using Gauss' theorem on the advection term, we find

$$\int_{CV} (f(\boldsymbol{x},t^{n+1}) - f(\boldsymbol{x},t^n))dV + \int_{t^n}^{t^{n+1}} \oint_{CS} f\boldsymbol{u} \cdot \boldsymbol{n}_{CS} dS dt = 0. \tag{2.8}$$

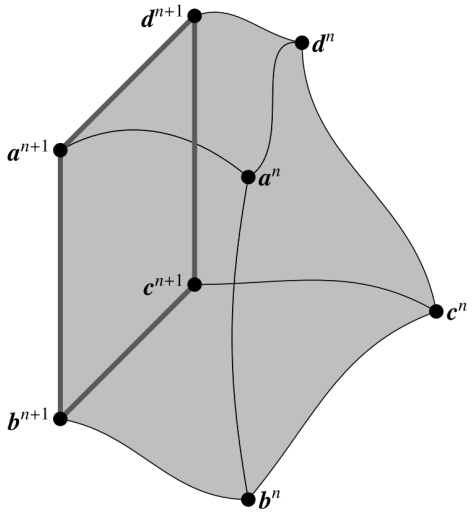


Figure 2.1: The scalar quantity to be advected is approximated and projected out of the control volume for integration over time [19].

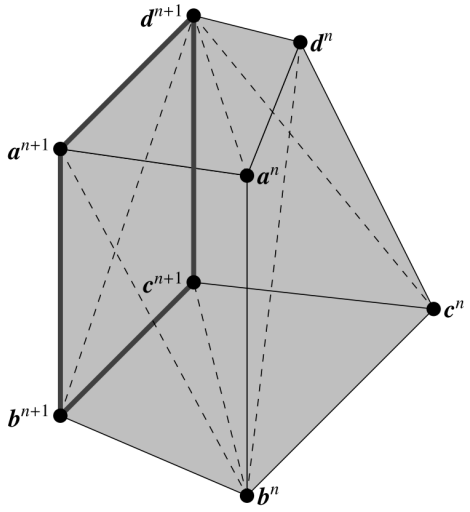


Figure 2.2: The representative volume is partitioned into simplicies (tetrahedra in 3D) until they are sufficiently small such that the simplicies contain only one phase of fluid [19].

The right term is the surface flux through the control volume and is dependent on $f(\boldsymbol{x}, t)$ for $t^n \leq t \leq t^{n+1}$, which is not normally a known quantity. However, a discrete representation of $f(\boldsymbol{x}, t^n)$ is typically known and and update equation can be used to determine $f(\boldsymbol{x}, t^{n+1})$. So, the flux is reformulated so that it is solely dependent on $f(\boldsymbol{x}, t^n)$. This is achieved by subdividing the surface of the control volume CS into sub-surfaces ∂CS_i. On each sub-surface ∂CS_i we can link the flux volume $\Omega_i(t)$ to the bounding surface $\omega_i(t)$. $\Omega_i(t)$ is just the signed volume which flows through the sub-surface ∂CS_i over a given time step Δt. The sign of the volume is dependent on the orientation into or out of the control volume and is positive if flowing out of, and negative if flowing into, the control volume [22].

Integrating Equation 2.7 over the flux volume Ω_i and again using Gauss' theorem

on the second term we find

$$\int_{\Omega_i(t)} \frac{\partial f}{\partial t} dV + \oint_{\omega_i(t)} f\boldsymbol{u} \cdot \boldsymbol{n}_{\Omega_i} dS = 0. \tag{2.9}$$

Here, $\boldsymbol{n}_{\Omega_i}$ is the outward facing normal to the flux volume $\Omega_i(t)$. The bounding surface ω_i is partitioned as $\omega_{i,F} = \omega_i \cap \partial CS_i = \partial CS_i$, which is fixed and $\omega_{i,M} = \omega_i \backslash \partial CS_i$ which is a material surface [22]. This is valid by defining part of ω_i as having zero flux of f and therefore must move with the flow. The other portion of ω_i is defined such that it coincides with ∂CS_i, which is fixed in time. Integrating Eq. 2.9 over time and using the previous partition we find that

$$\int_{t^n}^{t^{n+1}} \int_{\Omega_i(t)} \frac{\partial f}{\partial t} dV dt + \int_{t^n}^{t^{n+1}} \int_{\omega_{i,M}(t)} f\boldsymbol{u} \cdot \boldsymbol{n}_{\Omega_i} dS dt + \int_{t^n}^{t^{n+1}} \int_{\partial CS_i} f\boldsymbol{u} \cdot \boldsymbol{n}_{\Omega_i} dS dt = 0. \tag{2.10}$$

Because of its similarity to the flux term in Eq. 2.8, the right term of the above equation will provide a bridge between the two equations. The remaining terms can be made more clear using Leibniz's method which states

$$\frac{d}{dt} \int_{\Omega_i(t)} f dV = \int_{\Omega_i(t)} \frac{\partial f}{\partial t} dV + \int_{\omega_{i,M}(t)} f\boldsymbol{u} \cdot \boldsymbol{n}_{\Omega_i} dS. \tag{2.11}$$

With this, we can see that a $\Omega_i(t)$ is a streaktube projected backward in time from the surface ∂CS_i over the time step Δt. Integrating Eq. 2.11 with time results in

$$\int_{\Omega_i(t^{n+1})} f(\boldsymbol{x}, t^{n+1}) dV - \int_{\Omega_i(t^n)} f(\boldsymbol{x}, t^n) dV = \int_{t^n}^{t^{n+1}} \int_{\Omega_i(t)} \frac{\partial f}{\partial t} dV dt + \int_{t^n}^{t^{n+1}} \int_{\omega_{i,M}(t)} f\boldsymbol{u} \cdot \boldsymbol{n}_{\Omega_i} dS dt. \tag{2.12}$$

By definition, $\Omega_i(t^{n+1})$ is zero, making the first term equal to zero. We can adopt the notation that $\Omega_i = \Omega_i(t^n)$ and call this term the flux volume. Subtracting Eq. 2.12

from Eq. 2.10 we find that

$$\int_{t^n}^{t^{n+1}} \int_{\partial CS_i} f\boldsymbol{u} \cdot \boldsymbol{n}_{\Omega_i} dS dt = \int_{\Omega_i} f(\boldsymbol{x}, t^n) dV, \tag{2.13}$$

which gives us a simple relation between the flux through the sub-surface ∂CS_i and the volume integral over Ω_i.

To obtain a useful time advancement equation we would like to combine Eqn. 2.8 and Eqn. 2.13. However, doing so requires a relation between the normal terms, $\boldsymbol{n}_{CV}$ and $\boldsymbol{n}_{\Omega}$ as these normals are both defined using the same surface. This means that either, the two normals are identical or face opposite directions. We adopt a signed volume convention where the volume is positive if $\boldsymbol{n}_{CV} = \boldsymbol{n}_{\Omega}$ and negative otherwise. This convention results in positive volumetric fluxes out of the control volume and negative fluxes into the control volume. With this relationship we can finally obtain

$$\int_{CV} (f(\boldsymbol{x}, t^{n+1}) - f(\boldsymbol{x}, t^n)) dV + \sum_{i=1}^{N_S} \int_{\Omega_S} f(\boldsymbol{x}, t^n)) dV = 0 \tag{2.14}$$

where Ω_S is now the signed streaktube. The final form provides that the change in the scalar f within the control volume CV over the timestep Δt must be equal to the volume integral of the signed flux volumes.

Mesh Generation

In numerical analysis, the grid or mesh, often refers to the manner in which a domain being simulated is subdivided into smaller sections and points. Traditionally these points are denoted using an i,j scheme and neighboring points will be referenced from an initial point as seen in Figure 2.3. For example, if a value is being calculated at the (i, j) cell and requires information from the neighbors to the left and right, the neighbors are referenced as $(i-1, j)$ and $(i+1, j)$ respectively. If the calculation

requires information from neighbors on the top and bottom however, the cells are referenced as $(i, j+1)$ and $(i, j-1)$. This method is extended similarly in the z-direction for 3D applications [5].

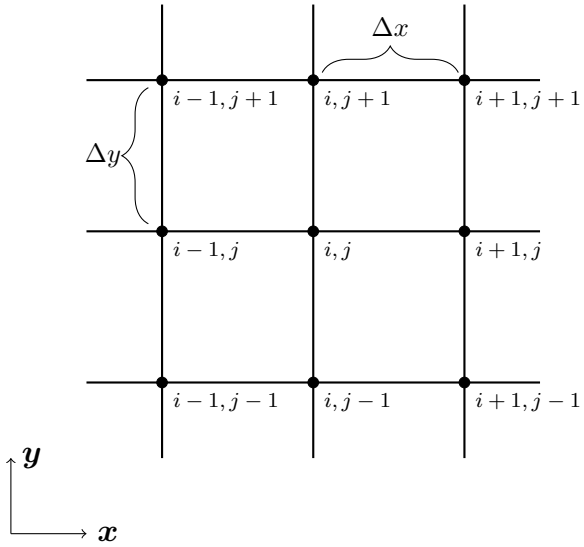


Figure 2.3: Typical (i, j) notation of structured grid cells. Grid cells span Δx and Δy in the horizontal and vertical directions respectively.

The type of grid used can be classified as being in one of two categories, structured or unstructured [2]. The choice of whether to use a structured or unstructured grid should be considered on a case by case basis. However, there are some defining characteristics of each that are important to recognize prior to implementation. A structured grid in 2D can be thought of a series of quadrilateral elements (bricks) placed side by side in a uniform fashion [5]. Here, neighboring elements are referenced by adding or subtracting from the base cell indices as indicated above [2]. Figure 2.3 is an example of a structured grid. An unstructured grid does not retain uniformity and is often comprised of triangular, rectangular, or hexagonal elements in 2D applications [34]. To reference neighboring cells in an unstructured grid, storage of cell-to-cell pointers are required [5]. Unstructured grids normally

require a greater amount of memory storage and can result in slower computation times than that of a structured grid [13]. A simplified example of an unstructured grid can be referenced in Figure 2.4. The NGA code, which is this focus of this research, uses a structured grid approach for simplicity and computational efficiency. However, it should be noted that unstructured grids are increasing in popularity as the accuracy obtained from modeling complex flow geometry may be higher than that of a structured grid [12].

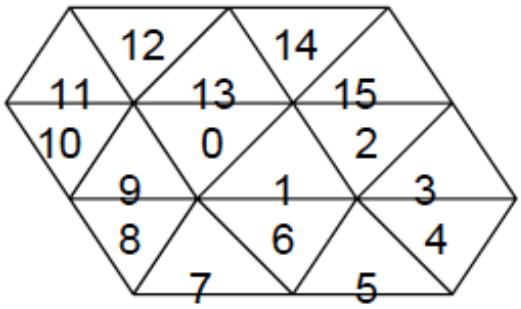


Figure 2.4: Simple Unstructured Grid [5]

Rudman Dual Grid Formulation

Solution of the incompressible Navier-Stokes equations traditionally occurs on whats known as a staggered grid. On a staggered grid, pressure is typically stored at cell centers and velocity components are stored at cell faces [33]. Building from the structured grid example given in Figure 2.3, this implementation is illustrated in Figure 2.5. This approach was first introduced by Harlow and Welch (1965) and is now considered the standard approach in structured mesh CFD applications [9]. For incompressible flows, staggered grids offer the advantage of tightly coupling fluid property variables as well as eliminating pressure-velocity checkerboarding [27]. The Rudman dual grid approach, first presented by Rudman (1998), is a technique

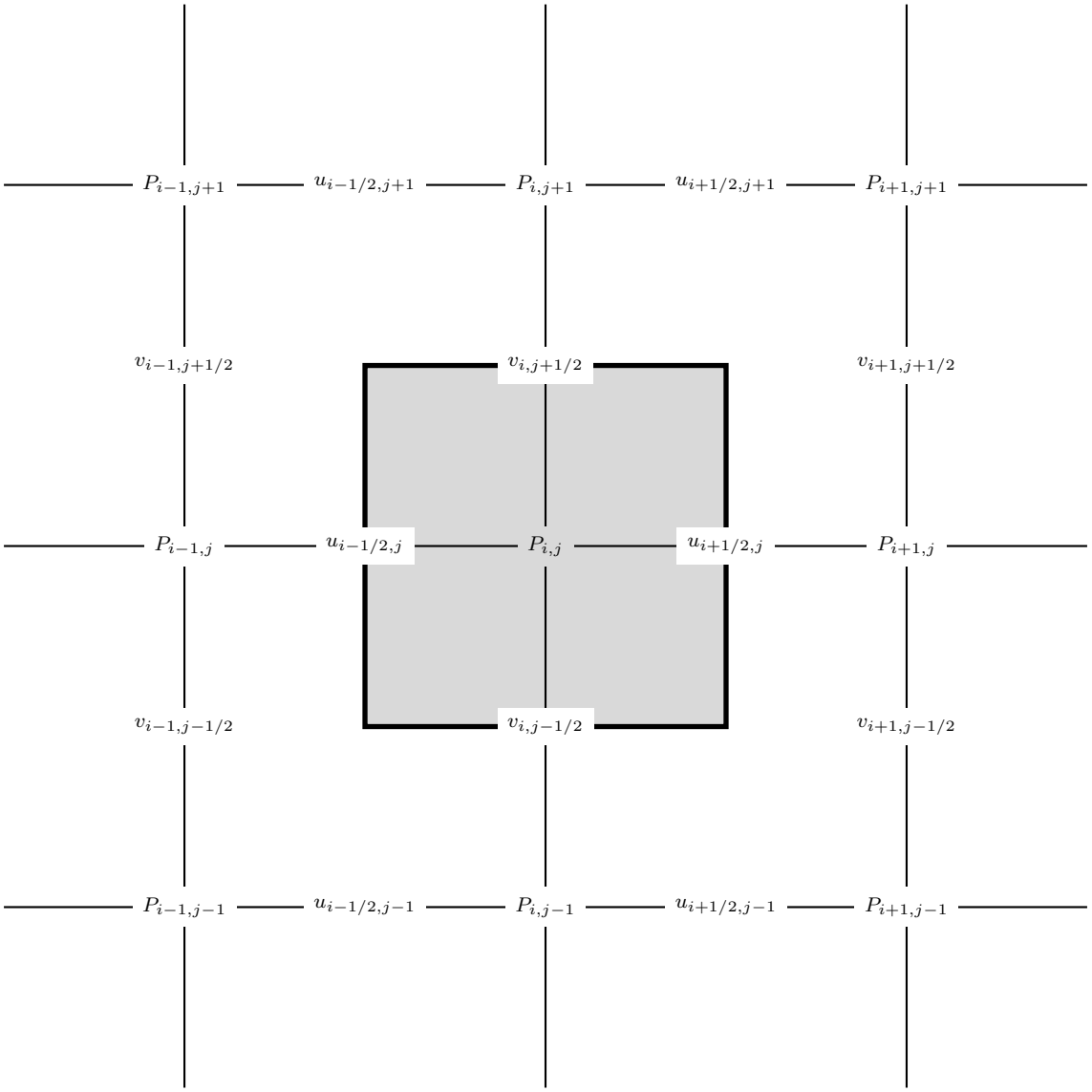


Figure 2.5: Typical notation of structured grid cells. For each cell, pressure values are stored at the center of the control volume, shaded in gray here. Velocity components are stored at the center of the cell faces.

developed for high density ratio, multiphase flows, which accurately conserves both mass and momentum. This is achieved by calculating momentum-flux values on a twice as fine mesh near the fluid interface as illustrated in Figure 2.5 [27]. The Rudman dual mesh is incorporated into NGA and is essential to the formulation of the method presented in this research.

Volume of Fluid Formulation

In order to accurately predict the behavior of a multiphase system, the location of the interface of the two fluids must be determined. One method for completing this is known as the Volume of Fluid (VOF) method . This method was first introduced by Hirt & Nichols (1981) and has been expanded upon by several others [?,?,?,?,12]. The defining contribution of the VOF method proposed by Hirt & Nichols is the introduction of a scalar marker function assigned to each mesh cell which is indicative of the volume of a given fluid within that cell [12]. This allows for interface to be tracked by the value of the marker cell in surrounding cells. For example, if $VOF = 1.0$ indicates a region of liquid and $VOF = 0.0$ indicates a region of gas, then a cell with a value $0.0 \leq VOF \leq 1.0$ indicates a cell which contains interface. A one-dimensional example of this advection can be seen in Figure 2.6. It is important to recognize that for 1D flow, volume of a fluid may only be advected into a new cell once the current cell is full [33]. However, this is not true for cases of increased dimensionality and advection criteria can become tedious for 2D and 3D flow fields.

The advection procedure for a VOF method is completed in two primary steps. First, the interface needs to be geometrically constructed. Second, the constructed interface is advected with the current velocity field [33]. Figure 2.6 depicts one-dimensional interface advection. In this case, geometric reconstruction is accomplished with a single vertical line. Adoption of this method is straightforward.

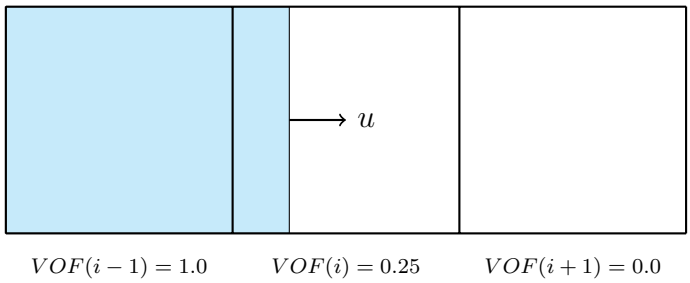


Figure 2.6: Advection of a one-dimensional fluid interface

As simulations increase to two or three dimensional analysis however, this problem quickly becomes non-trivial. Early attempts to resolve this difficulty include the Simple Line Interface Calculation (SLIC) method of Noh and Woodward (1976) [15]. Here, the reconstructed interface is made up of lines which align parallel to the mesh in both x and y directions. Improvement to the SLIC method was presented by

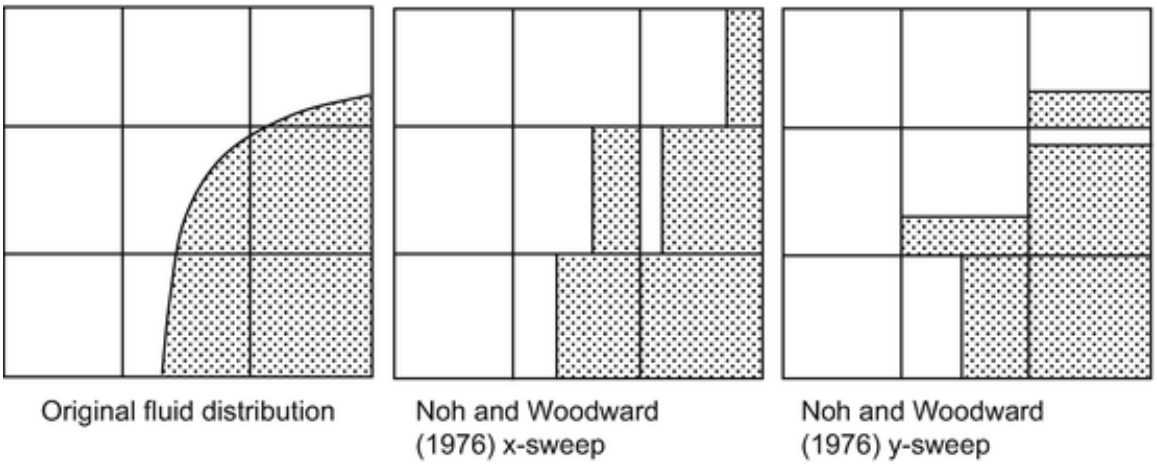


Figure 2.7: Simple Line Interface Calculation of Noh and Woodward adapted from [**?**]

Youngs (1982) known as the Piecewise Linear Interface Calculation (PLIC) [**?**]. In the PLIC method, instead of aligning interfacial reconstruction lines with the mesh, a line (2D) or plane (3D) is oriented with a normal vector which is evaluated from the volume fraction gradient [**?**]. An example of PLIC can be seen in Figure 2.8. This

method is a popular geometric reconstruction scheme still today and is used in NGA for this research.

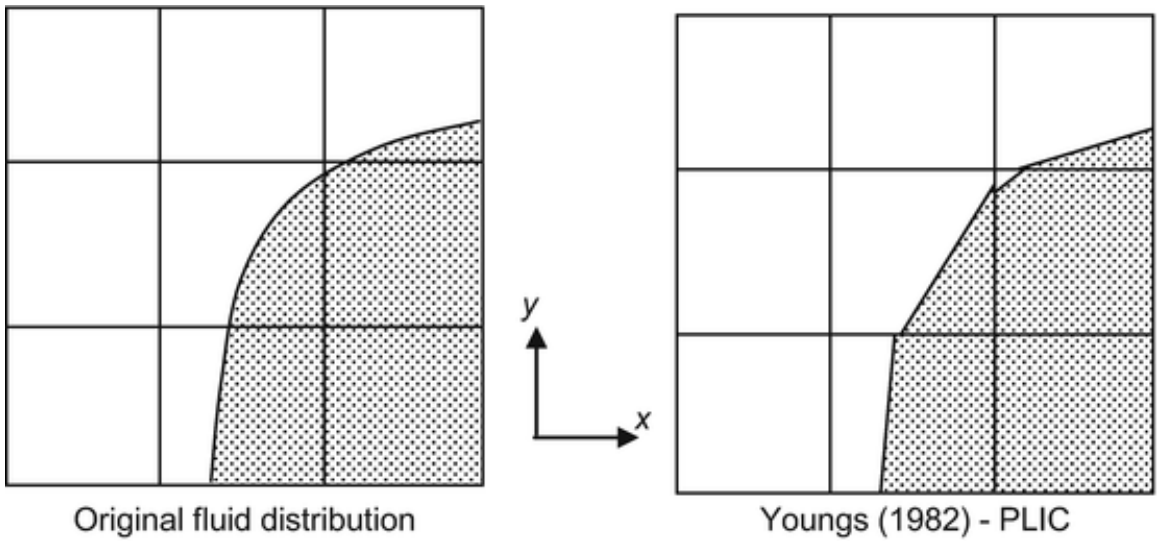


Figure 2.8: Piecewise Linear Interface Calculation of Youngs adapted from [?]

Curvature Estimation - Height Function Methods

Curvature can be discretely described as the reciprocal of radius of a given surface as illustrated in Figure 2.9 [17]. Accurate simulations of multiphase flows require an

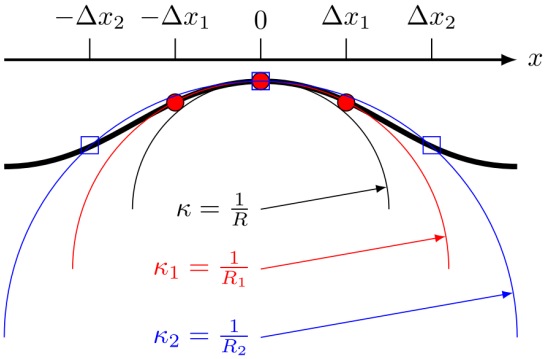
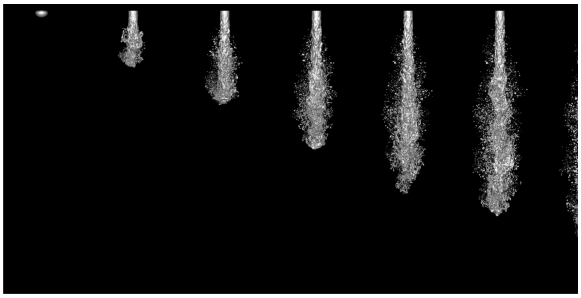
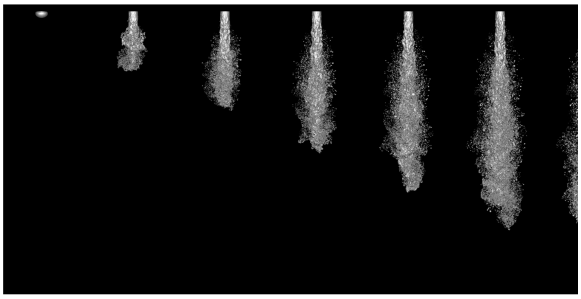


Figure 2.9: Curvature calculations for varying radii. Curvature can be calculated as the reciprocal of the radius of a curve [17].

accurate estimation of interface curvature. Figure 2.10 illustrates this importance by presenting two atomizing jet simulations. These simulations vary only by their Weber number, which is directly proportional to the surface tension term, and thereby, the curvature at the interface. It is clear that the jet seen in Figure 2.10(b) is experiencing far greater breakup than that seen in Figure 2.10(a). The correct representation of reality is highly dependent on the estimation of curvature made by the numerical model. The focus of this work is on modified height function methods which are one



(a) Re = 5000, We = 2000



(b) Re = 5000, We = 5000

Figure 2.10: Atomization simulations with varying Weber number, adopted from [?].

method of curvature evaluation. Height functions work by summing volume fractions (α) within columns of cells to form heights as

$$h_i = \sum_{j-N}^{j+N} \alpha_{i,j} \Delta y \tag{2.15}$$

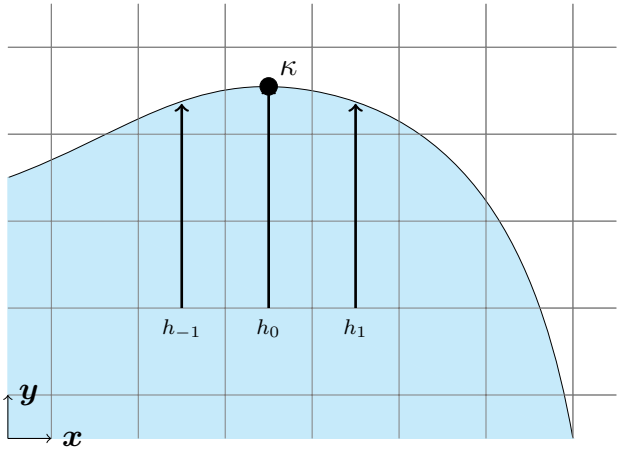


Figure 2.11: Traditional height function with a three cell stencil.

where N represents the number of cells above and below the cell of interest to construct the column. A similar approach is taken using widths where an interface is more vertical than horizontal. Figure 2.11 gives an example of heights at a liquid-gas interface. In the simplest execution of a height function, approximation of the curvature (κ) at the point on the grid is achieved by using a simple finite difference of the heights to approximate first and second derivatives as

$$H_x = \frac{h_{i-1} - h_{i+1}}{2\Delta x} \tag{2.16}$$

$$H_{xx} = \frac{h_{i+1} - 2h_i + h_{i-1}}{\Delta x^2}. \tag{2.17}$$

Finally, curvature is calculated using

$$\kappa = \frac{-H_{xx}}{(1 + H_x^2)^{\frac{3}{2}}}. \tag{2.18}$$

Clearly the above explanation is relevant for two dimensional flows. Extension to three dimensional flows requires two dimensional approximations of derivatives and

results in a curvature calculation of

$$\kappa = \frac{-H_{xx} - H_{yy} - H_{xx}H_y^2 - H_{yy}H_x^2 + 2H_{xy}H_xH_y}{(1 + H_x^2 + H_y^2)^{\frac{3}{2}}}. \tag{2.19}$$

Height functions remain a prevalent method for estimating curvature because of their relative ease of implementation and their second order convergence. However, several adjustments to the standard model have been proposed. Adjustments include changing the stencil size over which the heights are gathered [31, 32], separating the columns from the computational mesh [21], combining heights and widths for approximation [25], and applying the approach to level sets [18].

METHODOLOGY

Throughout the course of this research various schemes were attempted, often as a reaction to the problems arising in simulation dynamics. Because the code has undergone so many modifications, it is not reasonable to provide results from every attempt. Instead, this chapter aims to provide the reader with a generalized overview of the various methods that were attempted and provide insight into the logic behind the decisions that were made. It is be helpful to be aware that there are three main components to the method presented in this work: fine grid height functions, fine grid velocity establishment, and fine grid flux integration. Each of these constituent pieces are complex and affect the overall effectiveness of the method. While this reporting addresses these as separate pieces, it is important to recognize that development occurred concurrently and modification to these different approaches may have occurred out of the order that is presented. For ease of explanation, separate sections will be devoted to the three portions and will attempt to provide a thorough discussion of the strengths and shortcomings of each.

Problem Statement

When a dual grid is used, the standard height function method fails to capture the dynamics occurring on the fine grid. Left unmitigated, these dynamics can lead to fine grid interfacial perturbations, small discontinuities in interface structures. These perturbations can grow uncontrollably and result in non-physical dynamics materializing in simulations. An example of this uncontrolled growth resulting in non-physical dynamics can be seen in Figure 3.1. The focus of this research is to develop an extension of the standard height function to include information from the Rudman dual mesh. The proposed fine grid method retains the benefit of consistent mass and momentum transport while also providing accurate interface transport that

avoids non-physical dynamics.

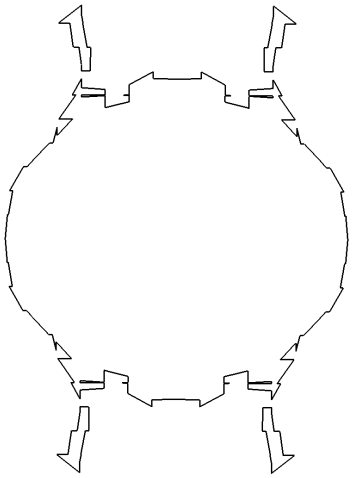


Figure 3.1: Standard height function method interface breakup.

Problem Approach

When considering how to inform the coarse grid from the finer mesh, a natural direction is to adopt a height function method directly onto the fine grid. Implementing a height function is straightforward as previously described and, the same curvature stencil should result in a more accurate curvature estimation as there is more information available. Figure 3.2 gives an example of what this could look like. Notice that while the stencil that the curvature is computed over remains the same as in Figure 2.11, there are now twice as many heights since information is pulled from columns built from fine grid cells.

Method Evaluation - Test Cases

To further quantify the problem that is occurring, baseline test cases which highlight the shortcomings of current methods are necessary. To this end, the height function method functionality is evaluated by calculating the curvature of an exactly defined VOF field. The curvature of a circle for example, is exactly equal to the

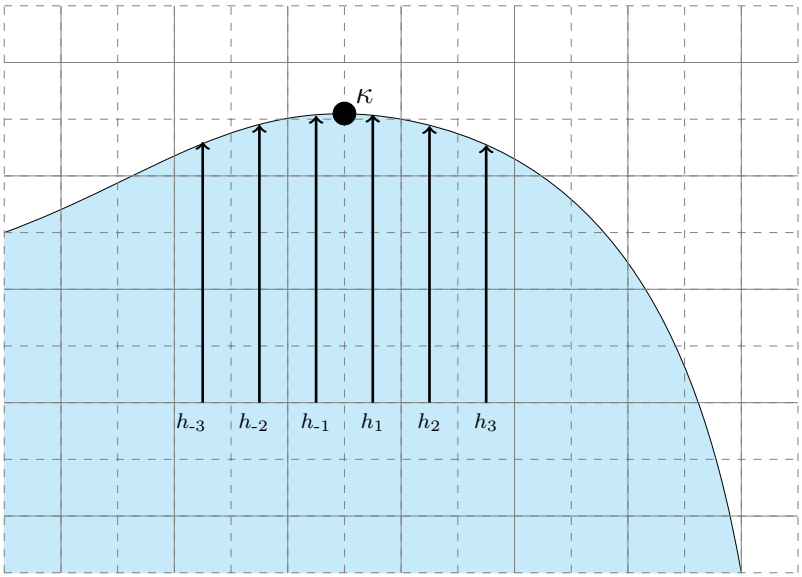


Figure 3.2: Fine grid height function with the same three cell stencil as in Fig. 2.11.

reciprocal of the radius as seen in Figure 2.9. A static circle is placed in a VOF field and the method computes the curvature. Results are compared against an analytic solution. Additionally, an oscillating two-dimensional droplet is used to assess the height function method and the proposed solution methods in a dynamic environment. This test case was chosen as it is considered a standard benchmark problem for testing the accurate prediction of multiphase flow behavior [28]. Additionally, for the height function method, the oscillating droplet offers an extensive testing of interface orientations which is important for measuring the robustness of the method. The interface is initialized with an ellipse. Surface tension drives the droplet's semi-major axis to fluctuate between alignment with the x and y axes as seen in Figures 3.3 - 3.5. The period of oscillation T_e, is a function of surface tension coefficient (σ), density (ρ_l and ρ_g), and equivalent circular radius(R), and can be computed analytically as [26]

$$T_e = 2\pi\sqrt{\frac{(\rho_l + \rho_g)R^3}{6\sigma}}. \tag{3.1}$$

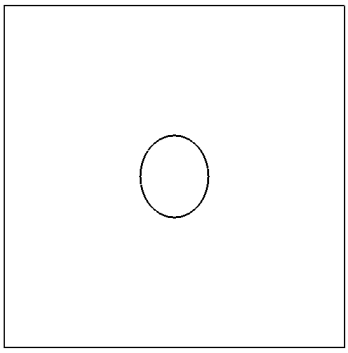


Figure 3.3: Initially, an ellipse is initialized in the domain.

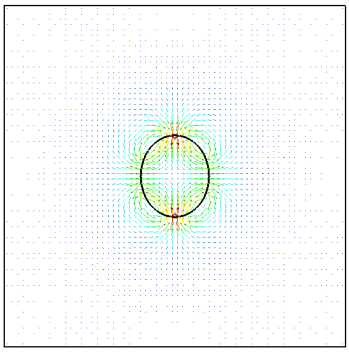


Figure 3.4: Surface tension begins driving motion of the ellipse.

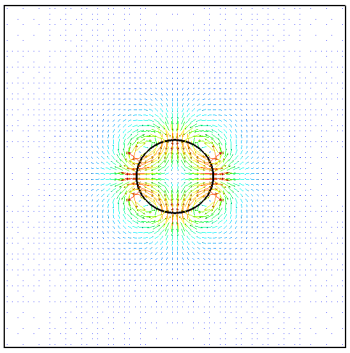


Figure 3.5: Droplet extends and oscillation continues.

To establish successful algorithm performance of NGA, an alternate curvature scheme was selected and an oscillating droplet test case ran. Simulation parameters include a density ratio of 1000, viscosity in both the liquid and gas phases are zero, and no walls are present in the computational domain. An ellipse with a center at the origin and a major axis parallel to the x-axis can be described as

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1. \tag{3.2}$$

Within the test case, a and b are explicitly defined. A semi-major radius of $a = 0.24$ and a semi-minor axes radius of $b = 0.20$ define the initial displacement of the droplet. The total domain length is set to 1.0. These parameters are chosen as they align with the analytic solution assumptions made. For the initial baseline, a mesh of 64x64 grid cells make up the domain. The mesh size was chosen specifically to address a concern for having a sufficient number of grid cells across the diameter of the droplet. A sufficient number of grid cells across a diameter can be considered as any minimum number of cells which provide well defined heights to the height function. Well defined heights require that cells containing only one phase of fluid are present

above and below the cell of interest. Figure 3.30 gives a good depiction of ill-defined heights. Figure 3.6 shows kinetic energy conservation through the progression of the simulation. Simulation success is quantified by normal periods of oscillation and diminishing kinetic energy with time. In an ideal case, no diminishment of kinetic energy would occur as the case is set up such that no energy can be lost to viscosity or boundary conditions. However, due to numerical dissipation, this loss is often observed. Close adherence to this model in Figure 3.6 can be used as a measure of success with methods from here forward. Alternatively, Figure 3.7 shows the result

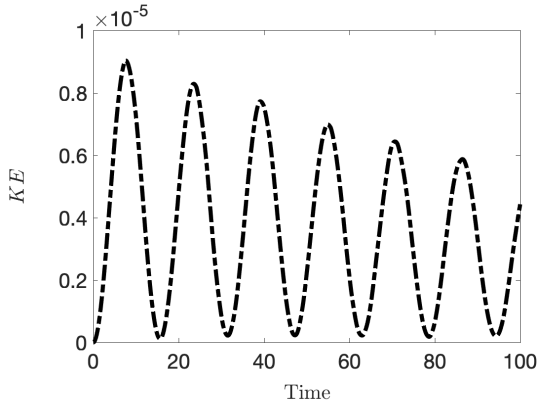


Figure 3.6: For the oscillating droplet test case, capturing kinetic energy over time is a way to discern whether the model is progressing as desired. Regular periods and linearly decreasing kinetic energy represent a favorable result. Simulation using ACES Method [17]

of the same test case ran with a standard, coarse grid, curvature estimation scheme. The uncontrolled growth in kinetic energy seen in the standard method is a result of the aforementioned interfacial perturbations. The curvature estimation error and resulting non-physical dynamics are the motivation for this research.

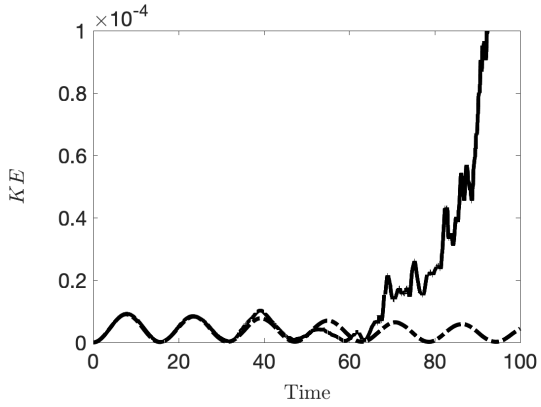


Figure 3.7: Kinetic Energy with Time - Standard Height Function Method(-) vs ACES method(.-)

Fine Grid Height Function Methods

Fifth Order Method

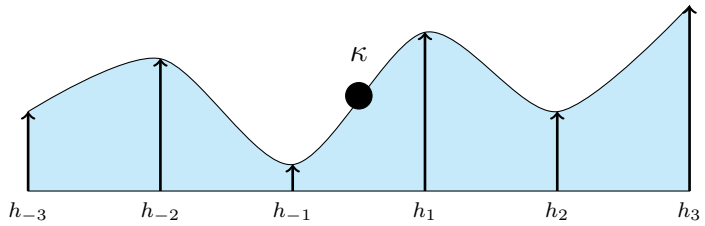


Figure 3.8: *Proposed*5^{th} order fit of heights using columns built from fine grid cells.

As seen in Figure 3.2, there are now six columns over which information is being provided. With this information, it is possible to derive fifth order approximations for the first and second derivatives needed for the curvature calculation. Figure 3.8 gives an approximate representation of what a fit might look like for a given point. A general formula for deriving a finite difference approximation given a set of points is given as

$$f'_j + \sum_{k=0}^{2} a_k f_{j+k} = O(?) \tag{3.3}$$

$$\frac{dh}{dx} = a_{-3}h_{-3} + a_{-2}h_{-2} + a_{-1}h_{-1} + a_{+1}h_{+1} + a_{+2}h_{+2} + a_{+3}h_{+3}. \tag{3.4}$$

Here, a_k are the coefficients associated with the linear Taylor series which need to be solved for [14]. For the six heights given by our fine grid stencil, Table 3.1 shows how the linear equations can be formed to find the first derivative. Traditionally, from the

Table 3.1: Taylor Series Table

	f	f'	f''	f'''	f^{iv}	f^{v}
f'	0	1	0	0	0	0
$a_{-3}f_{-3}$	$\frac{a_{-3}}{0!}$	$a_{-3}\frac{(-3h)}{1!}$	$a_{-3}\frac{(-3h)^2}{2!}$	$a_{-3}\frac{(-3h)^3}{3!}$	$a_{-3}\frac{(-3h)^4}{4!}$	$a_{-3}\frac{(-3h)^5}{5!}$
$a_{-2}f_{-2}$	$\frac{a_{-2}}{0!}$	$a_{-2}\frac{(-2h)}{1!}$	$a_{-2}\frac{(-2h)^2}{2!}$	$a_{-2}\frac{(-2h)^3}{3!}$	$a_{-2}\frac{(-2h)^4}{4!}$	$a_{-2}\frac{(-2h)^5}{5!}$
$a_{-1}f_{-1}$	$\frac{a_{-1}}{0!}$	$a_{-1}\frac{(-h)}{1!}$	$a_{-1}\frac{(-h)^2}{2!}$	$a_{-1}\frac{(-h)^3}{3!}$	$a_{-1}\frac{(-h)^4}{4!}$	$a_{-1}\frac{(-h)^5}{5!}$
$a_{+1}f_{+1}$	$\frac{a_{+1}}{0!}$	$a_{+1}\frac{(h)}{1!}$	$a_{+1}\frac{(h)^2}{2!}$	$a_{+1}\frac{(\frac{h}{2})^3}{3!}$	$a_{+1}\frac{(\frac{h}{2})^4}{4!}$	$a_{+1}\frac{(\frac{h}{2})^5}{5!}$
$a_{+2}f_{+2}$	$\frac{a_{+2}}{0!}$	$a_{+2}\frac{(2h)}{1!}$	$a_{+2}\frac{(2h)^2}{2!}$	$a_{+2}\frac{(\frac{2h}{2})^3}{3!}$	$a_{+2}\frac{(\frac{2h}{2})^4}{4!}$	$a_{+2}\frac{(\frac{2h}{2})^5}{5!}$
$a_{+3}f_{+3}$	$\frac{a_{+3}}{0!}$	$a_{+3}\frac{(3h)}{1!}$	$a_{+3}\frac{(3h)^2}{2!}$	$a_{+3}\frac{(\frac{3h}{2})^3}{3!}$	$a_{+3}\frac{(\frac{3h}{2})^4}{4!}$	$a_{+3}\frac{(\frac{3h}{2})^5}{5!}$

columns in Table 3.1, six equations can be used to solve for the six unknown variables. With these coefficients, approximations can be derived for a fitting function. The first iteration of the 5^{th} order height function used the above method to determine a polynomial with which to pass our heights through. Figure 3.9 shows the solution of the resulting fifth order finite difference first derivative used to approximate a solution of e^x plotted against an analytic solution. Figure 3.10 provide quantifying evidence that the method holds fifth order accuracy to machine precision. Figures 3.11 and 3.12 show similar behavior and similar stability for the solution of the second derivative.

The simplest test case for testing a curvature scheme is to compute the curvature of an exact VOF field. We establish a simple case where a circle is given a radius of

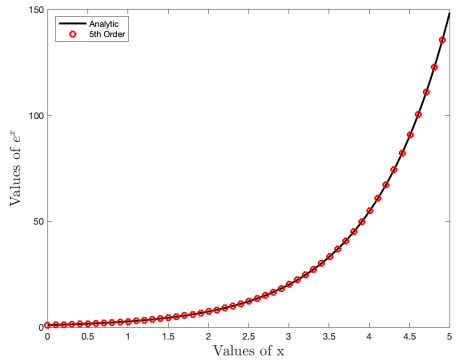


Figure 3.9: Proposed 5^{th} order method calculating values of e^x plotted against analytic solution.

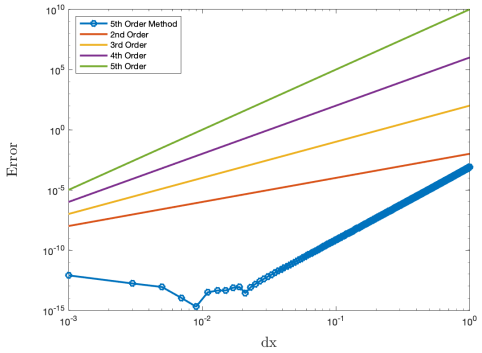


Figure 3.10: Proposed 5^{th} order method approximating first derivative. Method holds 5^{th} order accuracy to machine precision.

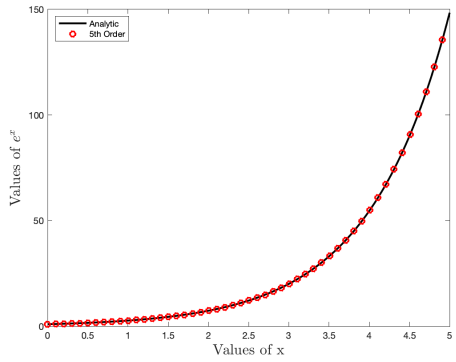


Figure 3.11: Proposed 5^{th} order method calculating values of e^x plotted against analytic solution.

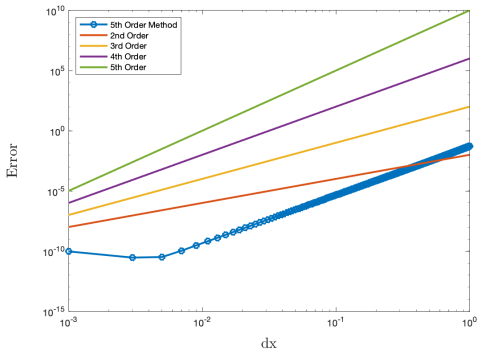


Figure 3.12: Proposed 5^{th} order method approximating second derivative. Method holds 5^{th} order accuracy to machine precision.

0.2. Initial testing of this method provided accurate representation of the exact VOF field but struggled to effectively run the oscillating droplet simulation. Subsequent investigation determined that this approximation of the 5^{th} order method was not sufficient for application of the height function. While the above method provides the Taylor series approximation needed for a point-wise fitting function, column heights used in the height function method require integral quantities of the Taylor series. This is because VOF values are also integral quantities, not linear quantities. For this adjustment, the same basic principal applies. A Taylor series expansion is performed to find function values over which to integrate (h_x). Heights (H_i) are determined by integrating values of h_x over the area of the subcell as seen in Eqn. 3.5. With heights determined, coefficients needed for derivative approximation are found using a constrained least squares approach which enforces second order accuracy. Derivatives are approximated using the heights and coefficients. Figure 3.13 provides an approximate visualization of the scheme as it differs from the representation in Figures 3.2 & 3.8.

$$H_i = \int_{x_i}^{x_{i+1}} h(x)dx \tag{3.5}$$

Figures 3.14 & 3.15 again show the method solution for a first and second derivative evaluated against a fifth order error and provide evidence that the method provides fifth order accuracy. Results from the exact VOF feild approximation indicated that the method performed as anticipated and accuracy increased with mesh refinement. With the successful implementation and initial calculation, the method was again tested using the oscillating droplet test case. Here, instead of the dissipation of kinetic energy we saw behavior similar to that of the standard height function method. The scheme was stable initially and then kinetic energy grew uncontrollably until eventual simulation failure.

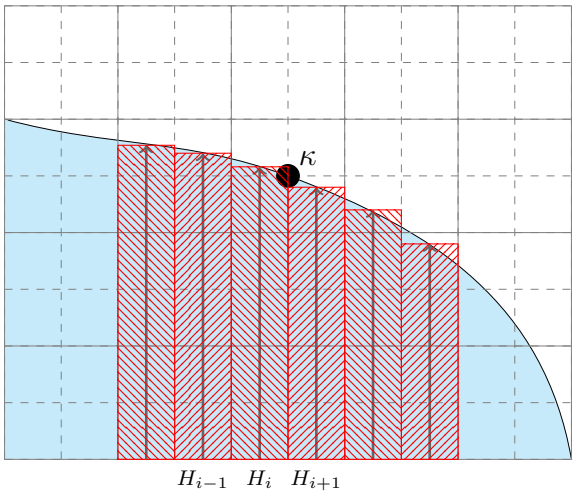


Figure 3.13: An appropriate fine grid height function requires integral values of heights as opposed to point-wise values.

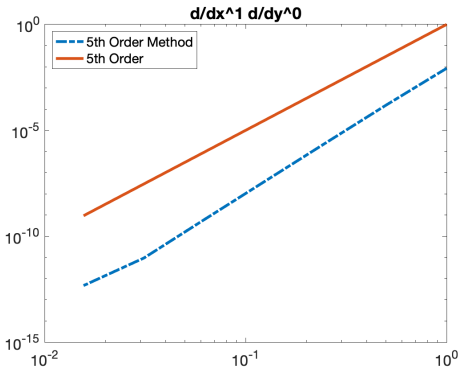


Figure 3.14: Improved 5^{th} order method approximating a first derivative. The method still holds approximately 5^{th} order accuracy.

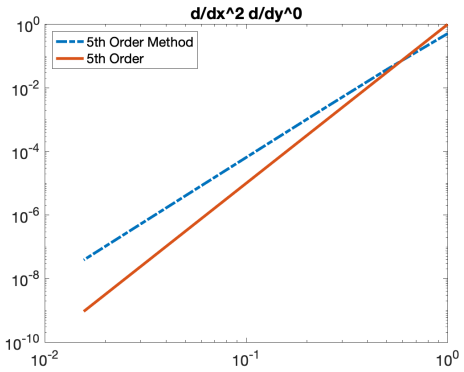


Figure 3.15: Improved 5^{th} order method approximating a second derivative. The method still holds approximately 5^{th} order accuracy.

Importance of Scale

Fifth order polynomials as used here, can have large oscillations in the curvature calculation leading to the errors which were observed in the droplet test case.This error is likely due to the scheme being influenced by areas of both highly positive and highly negative curvature within the fit. In a 2018 article, Owkes et al. found that the scale with which curvature is computed on heavily influences the growth of interface perturbations [17].This article also suggested that at certain scales, a second order method was more agreeable with accurate interface dynamics than a fourth order method on the same scale [17]. With this, we considered the possibility that a fifth order function on the fine grid scale may be over-fitting the points and reconstructing an interface with more perturbations than actually exist, essentially exacerbating our problem as opposed to alleviating it.

Second Order Fine Grid Height Function Method

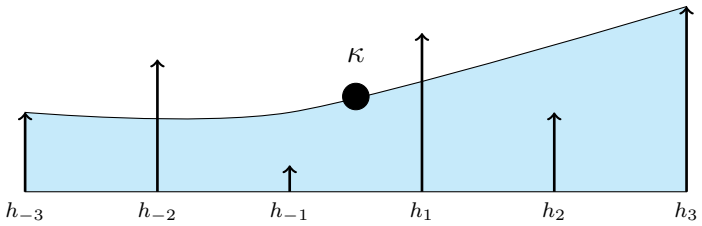


Figure 3.16: The proposed 2^{nd} order fit still utilizes information from six fine grid heights but hopes to reduce the potential for over-fitting.

Because the error order of a method comes second to accurately representing physical dynamics, we decided to use the same stencil but shift to a 2^{nd} order polynomial. The general idea is represented in Figure 3.16. Calculating a 2^{nd} order method was done using the same general finite difference technique as described previously but to 2^{nd} order accuracy. To approximate the same Taylor series to 2^{nd}

order accuracy, a symmetry assumption was made about the coefficients. Namely that

$$\begin{aligned} a_{-1} &= -a_1 \\ a_{-2} &= -a_2 \\ a_{-3} &= -a_3 \end{aligned} \tag{3.6}$$

which lead to undetermined operators that could be used as tuning parameters. The aim of this method was to use the same fine grid information to inform the coarse model but to smear the function to reduce unwanted perturbations. Again with mesh refinement, calculation of an exact curvature field produced encouraging results. Parameterization of the free variables allowed for an optimized solution which produced more favorable results than the fifth order method. However, the method still allowed for the uncontrolled growth of fine grid fluctuations which resulted in eventual simulation failure.

Gaussian Filtering Second Order Method

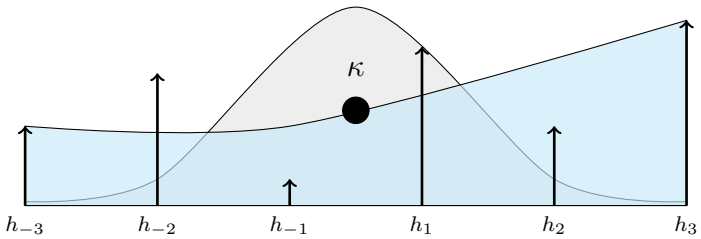


Figure 3.17: Applying a Gaussian filtering kernal to the previous 2^{nd} order method allows for an adjustment of scale of the fit.

In the 5^{th} order method, curvature is evaluated at the center of the stencil. However, a more reasonable curvature can be obtained by computing an average curvature over the stencil. This is done by filtering Eqn. 3.4 with a Gaussian

distribution as in Eqn. 3.7. This smooths the polynomial and allows for a more realistic estimation of the curvature. Averaging is achieved using a convolution with a weighting kernel. Transition to ζ space simplifies this scheme and is defined as $\zeta = \frac{2}{\Delta x}(x - x_i)$. This scheme requires only one floating variable (σ) as seen in Eqn. 3.8. Varying this parameter modifies the scale and filtering kernel of the scheme, allowing optimal operating parameters to be determined. Here, scale refers to the size of the computational stencil with which the curvature is computed as previously described by Owkes et al. [17]. This scheme provided results which were better than previous iterations and significantly more accurate than the standard height function method. However, this model also provided undesirable results at later time steps when simulating an oscillating droplet.

$$\kappa = \int_{-3}^{3} G(\zeta)\kappa(\zeta)d\zeta \tag{3.7}$$

$$G(\zeta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-\zeta^2}{2\sigma^2}} \tag{3.8}$$

Fine Grid Velocities

With all implementations of the fine grid height function we found the remaining existence of interfacial perturbations. While the fine grid height function method does provide a more rigorous representation of curvature, it does not reduce or remove these perturbations which are nonphysical and lead to the eventual failure of the simulation. It became clear that actively reducing these discontinuities could provide the interface smoothness necessary for a successful fine grid height function incorporation. To reduce the influence of these perturbations we chose to implement an additional velocity which existed only on the fine grid.

Implementation of Fine Grid Velocities

Initial attempts to include a fine grid velocity started by considering a coarse grid cell. Because of the use of a Rudman dual grid, the VOF calculation occurs on the fine grid. The PLIC which constructs the interface is also calculated on the fine grid. The discontinuities between PLIC's are what create the parasitic interfacial perturbations which have been discussed. An example of this can be seen in Figure 3.18. Using a correction velocity which is based on the area of this

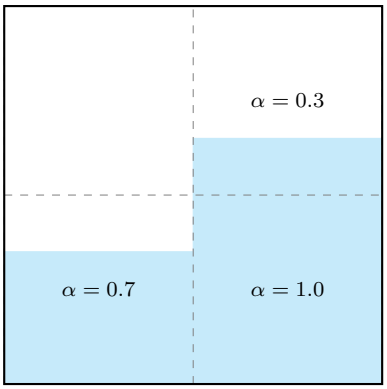


Figure 3.18: Discontinuities at regions where PLICS join are the source of interface perturbations.

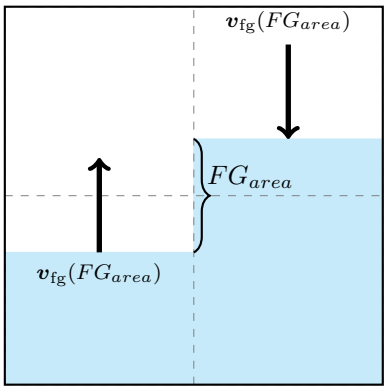


Figure 3.19: Discontinuities provide an area with which to scale a fine grid velocity.

discontinuity as in Figure 3.19 allows for a proportional correction that is only present where discontinuities exist. This method seemed promising initially by collapsing very simple geometric discontinuities such as those seen in Figure 3.19 but, could not provide adequate solutions for more complex fields such as an oscillating droplet simulation.

In trying to develop a more robust approach to the area based velocity method, an approach developed by Herrmann was discovered and seemed as though it may provide a solution [11]. This method utilizes a spring-damper analogy to approximate interface motion and is derived originally from the Taylor analogy breakup model

(TAB) of O'Rouke and Amsden [16]. The update equation proposed by Herrmann is

$$\frac{\partial \boldsymbol{u}_{\text{fg}}}{\partial t} + (\bar{\boldsymbol{u}} + \boldsymbol{u}_{\text{fg}}) \cdot \nabla \boldsymbol{u}_{\text{fg}} = c_\sigma \frac{\sigma}{\rho} \bar{\kappa}(\kappa_{\text{fg}} - \bar{\kappa}) - c_\mu \frac{\mu}{\rho L^2} \boldsymbol{u}_{\text{fg}} \tag{3.9}$$

where, $\bar{\boldsymbol{u}}$ is the resolved velocity, $\boldsymbol{u}_{\text{fg}}$ is the fine grid velocity, σ is the surface tension coefficient, ρ is density, $\bar{\kappa}$ is the resolved curvature, κ_{fg} is the fine grid curvature, μ is the dynamic viscosity, L is a modeling length scale and C_σ and C_μ are surface tension and viscous scaling coefficients, respectively [11]. The left side of the equation includes a velocity time rate of change as well as a convective term, while the right side includes the surface tension term which acts as a spring force and the viscous term which is analogous to a damping force. By this analogy, system behavior can be moderated by scaling the value on the surface tension or viscous coefficients. One important feature of this method is that it incorporates a difference of coarse grid curvature ($\bar{\kappa}$) and fine grid curvature (κ_{fg}). This relation draws a link between the two scales and allows for a coupling of the fine grid and coarse grid methods. While there are several ways to approximate the difference in curvature, we chose to begin with a simple approximation of the coarse grid curvature and used a standard height function method. The fine grid curvature was computed using a second order height function approximation. The generalized update algorithm for Herrmann's equation can be summarized as

1. Solve for coarse grid curvature, κ, using Eqn. 2.18.

2. Solve for fine grid curvature , κ_{fg}, again using Eqn. 2.18 but with the 2^{nd} order method as outlined previously.

3. Interpolate coarse grid velocity components to subcell center using a linear averaging scheme.

4. Compute the gradient of velocity as in Eqn. 3.12.

5. Compute normal vector

6. Compute viscous term as a linear average, $\mu = \frac{\mu_{\text{liq}}+\mu_{\text{gas}}}{2}$.

7. Compute density term as a linear average, $\rho = \frac{\rho_{\text{liq}}+\rho_{\text{gas}}}{2}$.

8. Interpolate accompanying fine grid velocity terms (for $\boldsymbol{x}$ direction, $\boldsymbol{v}$ & $\boldsymbol{w}$ terms are needed) to the fine grid face to use in the convective term of the update equation.

9. Update fine grid velocity value at cell face using Eqn. 3.10

10. Iterate through time as $\boldsymbol{u}_{\text{fg}}^{\text{n+1}} = \boldsymbol{u}_{\text{fg}}^{\text{n}} + \Delta t \boldsymbol{u}_{\text{fg, update}}$

The update equation using equation 3.9, simplified to one-dimension can be written as

$$\boldsymbol{u}_{\text{fg,update}} = -(\bar{\boldsymbol{u}} + \boldsymbol{u}_{\text{fg}}) \cdot \nabla \boldsymbol{u}_{\text{fg}} - c_\sigma \frac{\sigma}{\rho} \bar{\kappa}(\kappa_{\text{fg}} - \bar{\kappa}) - c_\mu \frac{\mu}{\rho L^2} \boldsymbol{u}_{\text{fg}} \tag{3.10}$$

which discretizes as

$$\begin{aligned} \boldsymbol{u}_{\text{fg,update}} = -(\bar{\boldsymbol{u}} + \boldsymbol{u}_{\text{fg}}(\text{s,i,j,k})) \frac{d\boldsymbol{u}_{\text{fg}}}{dx} - (\bar{\boldsymbol{v}} + \boldsymbol{v}_{\text{fg}}) \frac{d\boldsymbol{u}_{\text{fg}}}{dy} - (\bar{\boldsymbol{w}} + \boldsymbol{w}_{\text{fg}}) \frac{d\boldsymbol{u}_{\text{fg}}}{dz} \\ - C_\sigma \frac{\sigma}{\rho} \kappa(\kappa_{\text{fg}} - \kappa) \boldsymbol{n}_{\text{x}} - C_\mu \frac{\mu \boldsymbol{u}_{\text{fg}}}{\rho L^2} \boldsymbol{u}_{\text{fg}}(\text{s,i,j,k}) \end{aligned} \tag{3.11}$$

and the gradient is discretized using a central finite difference as

$$\nabla \boldsymbol{u}_{\text{fg,x}} = \frac{\boldsymbol{u}_{\text{fg}}(\text{s}^+,\text{i}^+,\text{j,k}) - \boldsymbol{u}_{\text{fg}}(\text{s}^-,\text{i}^-,\text{j,k})}{\Delta x_{\text{fg}}}. \tag{3.12}$$

Density and viscous terms are linear averages of the respective phase components, and $\boldsymbol{u}_{\text{cg}}$ are interpolated coarse grid velocity components from each respective direction.

Note that the s index in Eqn. 3.12 refers to the fine grid cell currently being accessed and the operator s^+ refers to the direction of the adjoining subcell. The same notation is true for the i index. Illustration of subcell placement can be seen in Figure 3.20. Because movement within the fine grid can become tedious, NGA incorporates lookup tables which help to minimize confusion. These lookup tables take advantage of the repetitive sequencing of subcell identifiers to ensure correct referencing of neighboring cells and subcells. If subcell index 1 is being accessed, the left and right neighbor are both subcell index 2, what changes is the coarse grid index. These quick reference routines make finite differencing operations less confusing by allowing for directional referencing.

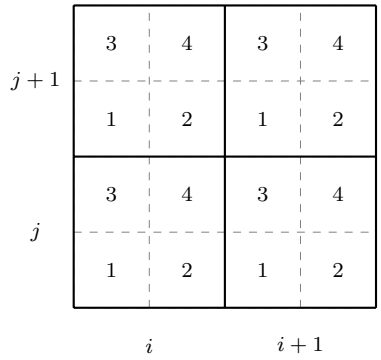


Figure 3.20: Fine grid orientation of subcell indicies.

Modification of Fine Grid Velocity

Initial results of the fine grid velocity method coupled with height function calculation of curvature suggested a fundamental problem. While explicit curvature calculation remained consistent with previous methods, the oscillating droplet test case again proved difficult. Close inspection of Eq. 3.9 reveals that the equation is not well-poised when the coarse-grid curvature, $\bar{\kappa}$, is zero as the entire spring force goes to zero even if fine-grid interface perturbations exist and $\kappa_{\text{fg}} \neq 0$. An alternative

is to base the source term on the difference between coarse and fine-grid curvatures and add a delta function that restricts the source term to only be non-zero at the interface. The approximation of the Dirac delta function is calculated as the absolute difference in liquid volume fractions between cells divided by the mesh size as

$$\delta = \frac{|\alpha(\mathrm{s,i,j,k}) - \alpha(\mathrm{s^{-}\text{,}i^{-}\text{,}j,k})|}{0.5\Delta x}. \tag{3.13}$$

Additionally, a pressure term is added to ensure fine grid velocity remains divergence free, further enforcing momentum conservation. With these modifications, the proposed equation to create the fine-grid velocity can be written as

$$\frac{\partial \boldsymbol{u}_{\mathrm{fg}}}{\partial t} + (\bar{\boldsymbol{u}} + \boldsymbol{u}_{\mathrm{fg}}) \cdot \nabla \boldsymbol{u}_{\mathrm{fg}} = c_\sigma \frac{\sigma}{\rho} \delta(\kappa_{\mathrm{fg}} - \bar{\kappa}) - c_\mu \frac{\mu}{\rho L^2} \boldsymbol{u}_{\mathrm{fg}} - \nabla P_{\mathrm{fg}} \tag{3.14}$$

along with the continuity equation

$$\nabla \cdot \boldsymbol{u}_{\mathrm{fg}} = 0. \tag{3.15}$$

The modified update algorithm is now

1. Solve for coarse grid curvature, κ, using Eqn. 2.18.
2. Solve for fine grid curvature , κ_{fg}, again using Eqn. 2.18 but with the 2^{nd} order method as outlined previously.
3. Interpolate coarse grid velocity components to subcell center using a linear averaging scheme.
4. Compute the gradient of velocity as in Eqn. 3.12.
5. Compute normal vector

6. Compute viscous term as a linear average, $\mu = \frac{\mu_{\text{liq}}+\mu_{\text{gas}}}{2}$.

7. Compute density term as a linear average, $\rho = \frac{\rho_{\text{liq}}+\rho_{\text{gas}}}{2}$.

8. Interpolate accompanying fine grid velocity terms (for $\boldsymbol{x}$ direction, $\boldsymbol{v}$ & $\boldsymbol{w}$ terms are needed) to the fine grid face to use in the convective term of the update equation.

9. Calculate delta function using equation 3.13

10. Predict fine grid velocity value at cell face using Eqn. 3.10

11. Solve pressure Poisson equation

12. Correct velocity value using updated pressure field

13. Iterate through time as $\boldsymbol{u}_{\text{fg}}^{\text{n+1}} = \boldsymbol{u}_{\text{fg}}^{\text{n}} + \Delta t \boldsymbol{u}_{\text{fg, update}}$

Test case results from this iteration again did not represent successful implementation. We began considering the relationship of the difference in curvatures. It seemed that there were various ways to approximate the coarse grid curvature $\bar{\kappa}$ but, that our goal was to influence this value with information from the fine grid. One approximation for $\bar{\kappa}$ is that it is made up as an average of the fine grid curvatures. That is, for a given coarse grid cell, the average of the fine grid curvatures should produce a value close to the coarse grid curvature. We changed the calculation of the coarse grid curvature such that it was an average value of the fine grid curvatures using a simple linear averaging technique. An advantage of this method was that it tied the fine grid velocity to one curvature, the fine grid curvature. This provided us with more freedom over how to calculate the fine grid curvature value and removed a second dependency (the value of $\bar{\kappa}$) for calculating fine grid velocity in our update

equation. This method also allowed for a variation in the scale of the smearing acting over the coarse grid curvature.

Initially, the curvature was calculated as the average of the fine grid curvatures within the cell. An advantageous extension of this method allows for further smearing to include information of surrounding cells. While this method is convenient in that it's fairly straightforward and provides the previously mentioned advantages, it relies on the assumption that the coarse grid and fine grid curvatures are correlated. This may not be true as the scale at which each is calculated can greatly affect the overall calculation. However, results seem the most promising thus far using this method. Equation 3.14 is representative of the scheme implemented at the time of this publication. Additional efforts were taken to find an optimal parameterization strategy with little success. Figure 3.21 shows a parameterization study where C_σ and C_μ varied from $1e^{-6}$ to $1e^{6}$ and an oscillating droplet test case was ran. While a portion of the runs seem initially successful, tuned parameter values were not found to be consistent across various mesh sizes and no conclusive optimal configuration was found.

Fine Grid Velocity Flux Incorporation

Away from the phase interface, NGA uses high-order finite difference operators that conservatively transport mass, momentum, and any other scalars [8]. Near the phase interface however, the finite difference operators are inappropriate due to the discontinuous interface region. To accurately and conservatively transport near the interface, an unsplit geometric semi-Lagrangian flux method is used [19, 22]. This method relies on signed streaktubes, which contain the region of the domain that moves through a computational cell face during the time step [22]. The streaktube is represented with a collection of tetrahedra as shown in Figures 2.1 & 2.2 and

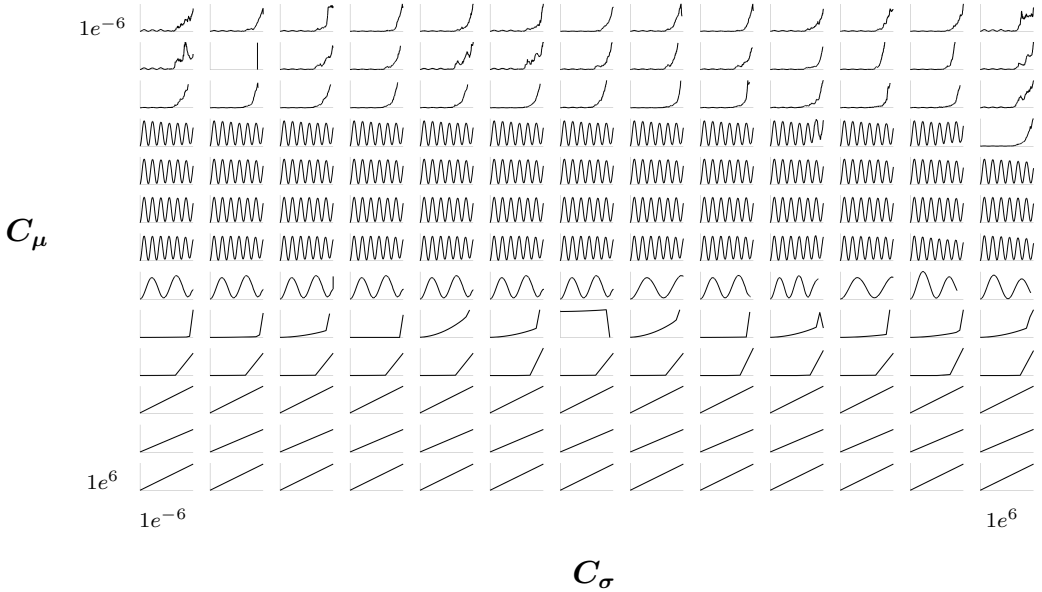


Figure 3.21: Parameterization study of an oscillating droplet test case. Plots show kinetic energy over time as seen in previous studies. The varying parameters are C_σ and C_μ which vary from $1e^{-6}$ to $1e^{6}$.

computational geometry is used to compute the flux of liquid, mass, momentum, and any scalars [22]. A correction is incorporated which forces a solenoidal condition for the cell. The scheme contains two primary steps. First, a project function approximates an amount of a scalar quantity to project back in time as a volumetric quantity as seen in Figure 2.1. Then, the volume that was projected is broken up into simplicies until all simplicies contain only one phase as described in the previous section and shown in Figure 2.2. These simplicies (tetrahedra in 3D) ensure consistent mass and momentum transport.

The addition of a fine grid velocity requires that the additional velocity be incorporated into the flux, this is achieved by adjusting the streaktube scheme. However, several different options exist for doing so. Initially, we incorporated the fine grid velocity directly into the project function. The project function approximates

the amount of a scalar value which will need to be advected and projects the quantity back in time. This is done so that the quantity can then be integrated over the time step as described previously. While this method works well across coarse cell faces, it does not always work between subcell faces. As illustrated by Figure 3.22, when the net velocity being fluxed over subcells within a coarse cell is zero, the project function would assume there was no additional flux to be accounted for. This does not provide a consistent scheme as the fine grid divergence would not be equal to zero globally as required. Seeing this error, we moved to manually adjusting the streaktubes. So,

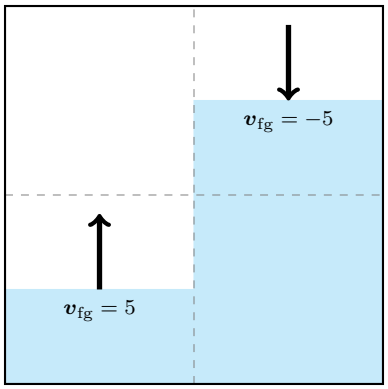


Figure 3.22: Where net velocity fluxing within a coarse cell is zero, the project function does not incorporate the necessary flux values.

after the project function projects values back in time, the volume is broken into simplicies. We adjusted the simplicies such that the total volume of the streaktube included the additional velocity values. Originally the total streaktube volume could be calculated as

$$V_{st} = \boldsymbol{u} A_{CS} \Delta t. \tag{3.16}$$

Adding the fine-grid velocity correction is done by modifying the additional flux due to the fine-grid velocities so that the volume becomes

$$V_{st} = (\boldsymbol{u}_{\text{cg}} + \boldsymbol{u}_{\text{fg}}) A_{CS} \Delta t. \tag{3.17}$$

For the finite difference scheme this entails adding the fine-grid velocities associated with a cell face onto the convection velocity at that face. This addition provides an explicit numerical representation of the fine grid velocity correction and maintains conservation laws.

As previously stated, the incorporation of the fine grid height function, fine grid velocity, and flux incorporation all happened simultaneously. However, a description of all methodologies have now been provided. With the most recent iteration of all three schemes integrated into the fine grid method, the baseline test cases were once again evaluated. Results are shown in Figures 3.23 & 3.24 and show considerable improvement from the standard height function method. However, test cases still resulted in nonphysical dynamics when moving to more complex geometries such as 3D flow fields. All test cases examined which include three dimensional geometry resulted in errors which eventually caused simulation failure. Because of this continued failure, it seemed appropriate to reevaluate the validity of the original methodology and take a deeper look into the mechanisms which were the cause of these deficiencies.

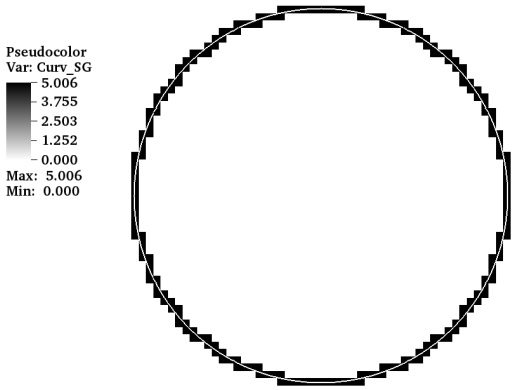


Figure 3.23: Calculation of an exact VOF field by the current implementation of the fine grid curvature scheme.

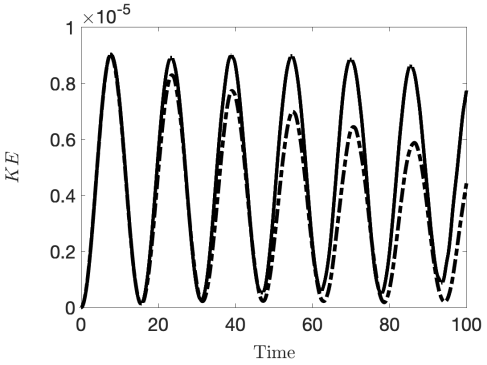


Figure 3.24: Kinetic energy plotted over time for an oscillating droplet test case. (-) Current implementation (.-) ACES method

Simplified Test Cases

With each of the schemes previously discussed, the method of evaluation consisted of an explicit curvature calculation of an exact VOF field followed by an oscillating droplet test case. Consistently, the exact curvature estimation was successful and the oscillating droplet problematic. The transition between the test cases began to seem like too great of a jump in complexity to have an intimate understanding of the progression of errors occurring. To gain a deeper understanding of the source of failure, it seemed necessary to have more simplified test cases with which to study the method. To this end, four simplified test cases were developed which we hoped would expose the limitations of the schemes.

Sine Wave Test Case

A sinusoidal interface was constructed as seen in Figure 3.25. The test case was created specifically to test several key aspects of the method. First, by modifying the geometry of the sinusoid its possible to test internal and external face fluxes independently or jointly. Secondly, because the interface is highly symmetric, variation in velocity and curvature values across the domain helps to highlight deficiencies in boundary conditions as well as inter-cell transport schemes. The goal of this test case is for the fine grid scheme to force a collapse of the interface so that a final curvature of zero is achieved.

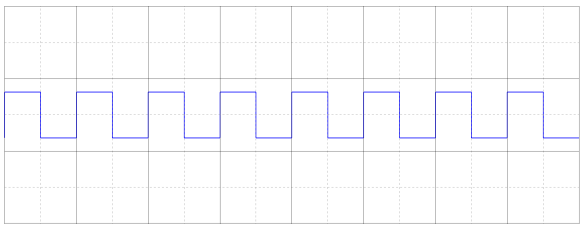


Figure 3.25: A simplified geometry test case is presented with a sinusoidal interface.

Interfacial Protrusion Test Case

The test cases shown in Figures 3.26, 3.27, & 3.28 are all of a simple linear interface at various orientations. At the center of each domain, a protrusion was placed on the interface. The aim of this test case was to recreate specific, isolated, scenarios to see if the fine grid method could accomplish collapsing the interface to a line and obtain a zero curvature across the domain. The protrusion on each interface was built to be approximately one cell width in both height and width.

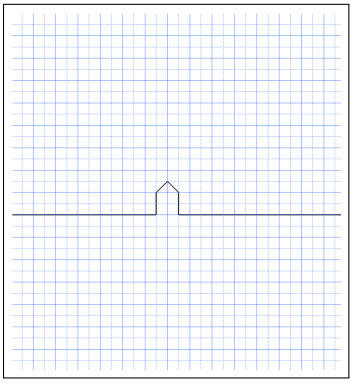


Figure 3.26: A simplified geometry test case is presented with a horizontal interface that includes a protrusion of approximately one cell width.

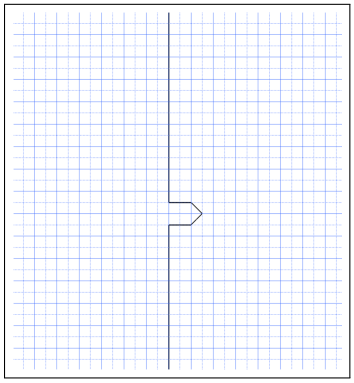


Figure 3.27: A simplified geometry test case is presented with a vertical interface that includes a protrusion of approximately one cell width.

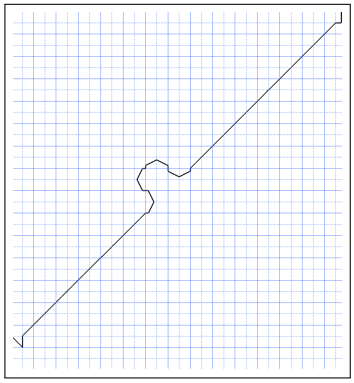


Figure 3.28: A simplified geometry test case is presented with a diagonal interface that includes a protrusion of approximately one cell width.

Results

For each of the above test cases, with the exception of the 45° test case, the method was successful in collapsing the interface to a line. This was an interesting discovery and seemed to coincide with results seen in failing oscillating droplet test cases. During the oscillating droplet test, often nonphysical geometries would begin to appear at each of the quadrant regions of the droplet as shown in Figure 3.29. This finding exposed a key limitation of the model. Well defined heights are not

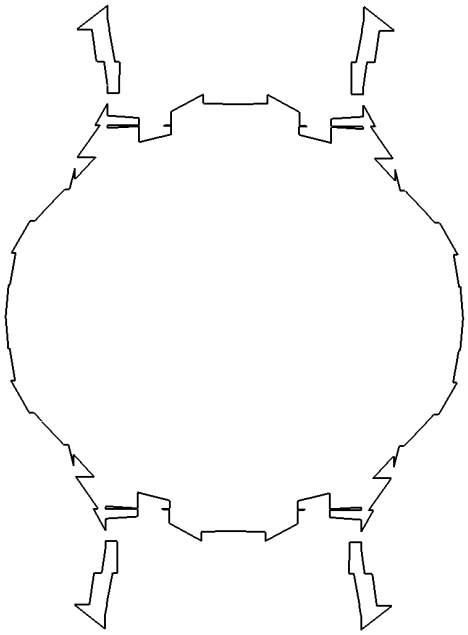


Figure 3.29: Result displaying nonphysical structures typically found in failing oscillating droplet simulations. Often, structures would grow from the four quadrant regions where well defined heights are hardest to obtain.

present in this case and cause problems for the height function method. When heights are not well defined, the approximation of derivatives is not accurate and results in nonphysical geometries. The problematic region is highlighted in Figure 3.30. Here, we can see that using heights or widths can result in a region of ill-defined heights. The only way to avoid this complication is with mesh refinement. However, instead of indicating an ill-defined test case, this suggests a general shortcoming of height functions.

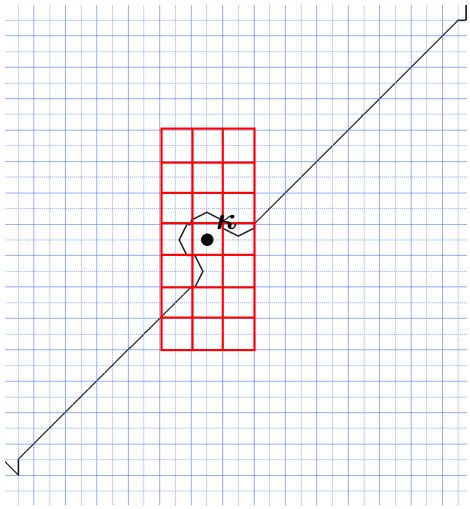


Figure 3.30: For the diagonal protrusion test case it is evident that the height function is unable to create the necessary well defined heights that are required for an accurate curvature estimation when a protrusion is on the order of a coarse grid cell.

DISCUSSION

At the current state, the fine grid method can successfully predict curvature for a select set of geometries. No successful execution of three dimensional flows have been performed or analyzed. Rigorous investigation of the methods outlined above have left the researchers with several fundamental concerns and drawbacks of this method.

Summary

1. It is clear that this method, in all iterations, requires well defined heights for successful execution. However, achieving well defined heights may require mesh sizes which are unrealistic for direct numerical simulation. While there may possibly be ways to work around this issue, research thus far has focused on performing successful simulations with geometries that provide ideal mesh size ratios. Investigation into more complex schemes should only be attempted after this has been achieved.

2. Incorporation of fine grid velocities seemed to have a positive impact on success of the system but required several modifications from the originally proposed method. The addition of a second pressure equation makes this method computationally expensive. So much so that simulation run time can increase by 40%. This increased expense contradicts original motivation of the investigation. Modification of the height function method was chosen as implementing the method is straightforward and the cost is low compared to more complex methods. To that point, while the fine grid height function is straightforward to implement, fine grid velocity incorporation and the subsequent incorporation of semi-Lagrangian fluxes are not.

3. While little investigation has been done with three dimensional flow fields, accurate and realistic curvature becomes significantly harder to achieve as fitting is now done using planes instead of lines and well defined heights have the ability to drop off in multiple dimensions.

4. At present, efforts have been considering a simulation which doesn't "blow up" to be a success. However, no efforts have gone into investigating the accuracy of this model. This is vital to understanding the merit of the method.

Future Work

Continuation of the presented work has the potential to yield a method which could benefit the multiphase community and could be implemented into any in-house solver which utilizes a Rudman dual mesh. Moving forward, it would be best practice to take time to set up a series of checkpoints which could provide a pass/ fail analysis of the current iteration of the fine grid method. A test suite of several different test cases which increase in complexity with each successful completion of a previous case could be an excellent way to gauge whether a method is more or less successful than the previous attempt. Throughout the presented work, we focused on the oscillating droplet test case as our benchmark and if the method "blew up", we simply moved on to the next modification. During the final months of this work, we chose to take steps back to more ideal test cases and learned a great deal about the shortcomings of the method as discussed previously. Moving forward, we would recommend the implementation of a unit test suite to apply a more methodical approach to troubleshooting.

There have been many implementations presented in this work. It is possible that some combination of methods which have been described could result in an overall successful scheme. With the previously described test suite, it would be

possible to assign metrics to the amount of success of a given combination of methods as the program moves through various complexities of test cases. From here, all combinations could be analyzed and the best combination used to go forward. A truly successful implementation would need to provide realistic curvature calculations for a wide range of flow types and geometries in both 2D and 3D. Developing test cases which range in flow type and dimension could expose further shortcomings of the method with which to investigate.

Conclusion

The focus of this work was to develop a curvature estimation method which uses information from a mesh that is twice as fine as the prescribed computational grid. This is motivated standard height function methodology which fails to capture perturbations that exist on this fine grid. Left unmitigated, these perturbations can grow unbounded and result in nonphysical dynamics as well as simulation failure. The described method applies a height function onto the fine grid. Additionally, fine grid velocities are introduced to actively reduce the influence of the fine grid perturbations. The proposed method maintains consistent mass and momentum transport and provides accurate interface transport. The method in its current state can only successfully predict curvatures for select geometries. Future work may provide insight into the shortcomings of the method which could be resolved. However, this work also exposes certain geometric scenarios in which the height function methodology is insufficient. While it can be useful to explore the full breadth of available methods and the possibilities that each provide, it is also important to recognize the fundamental shortcomings that may come with a method. In this work, we believe a fundamental shortcoming of the height function method has been uncovered.

REFERENCES CITED

[1] M. Alexander. Simulation of binary-single star and binary-binary scattering. *Journal of Computational Physics*, 64(1):195 – 219, 1986.

[2] Anderson, John David. *Computational fluid dynamics: the basics with applications*. McGraw-Hill, 2010.

[3] M. Bergmann and A. Iollo. Modeling and simulation of fish-like swimming. *Journal of Computational Physics*, 230(2):329 – 348, 2011.

[4] H. Choi and P. Moin. Effects of the computational time step on numerical solutions of turbulent flow. *Journal of Computational Physics*, 113(1):1 – 4, 1994.

[5] D. Darmofal. Structured vs. Unstructured Grids.

[6] O. Desjardins, G. Blanquart, G. Balarac, and H. Pitsch. High order conservative finite difference scheme for variable density low mach number turbulent flows. 227:7125–7159, 2008.

[7] O. Desjardins, J. O. Mccaslin, and M. Owkes. Direct numerical and large-eddy simulation of primary atomization in complex geometries . (December 2016), 2013.

[8] O. Desjardins, V. Moureau, and H. Pitsch. An accurate conservative level set/ghost fluid method for simulating turbulent atomization. 227(18):8395–8416, 2008.

[9] F. H. Harlow and J. E. Welch. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids*, 8:2182–2189, Dec. 1965.

[10] T. J. Heindel. A Review of X-Ray Flow Visualization With Applications to Multiphase Flows. *Journal of Fluids Engineering*, 133(7), 2011.

[11] M. Herrmann. A sub-grid surface dynamics model for sub-filter surface tension induced interface dynamics. *Computers and Fluids*, 87:92–101, 2013.

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

[13] F. Magoules. *Computational fluid dynamics*. Chapman & Hall CRC, 2011.

[14] P. Moin. *Fundamentals of Engineering Numerical Analysis*. Cambridge University Press, 2 edition, 2010.

[15] W. F. Noh and P. Woodward. SLIC /simple line interface calculation/. In A. I. van de Vooren and P. J. Zandbergen, editors, *Some Methods of Resolution of Free Surface Problems*, volume 59 of *Lecture Notes in Physics, Berlin Springer Verlag*, pages 330–340, 1976.

[16] P. J. O'Rourke and A. A. Amsden. The tab method for numerical calculation of spray droplet breakup. In *SAE Technical Paper*. SAE International, 11 1987.

[17] M. Owkes, E. Cauble, J. Senecal, and R. A. Currie. Importance of curvature evaluation scale for predictive simulations of dynamic gasliquid interfaces. *Journal of Computational Physics*, 365:37–55, 2018.

[18] M. Owkes and O. Desjardins. A discontinuous galerkin conservative level set scheme for interface capturing in multiphase flows. *Journal of Computational Physics*, 249:275 – 302, 2013.

[19] M. Owkes and 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*, 270:587–612, 2014.

[20] M. Owkes and O. Desjardins. Large-eddy simulation study of injector geometry on liquid jet in cross-flow and validation with experiments. (December), 2014.

[21] M. Owkes and O. Desjardins. A mesh-decoupled height function method for computing interface curvature. *Journal of Computational Physics*, 281:285 – 300, 2015.

[22] M. Owkes and O. Desjardins. A mass and momentum conserving unsplit semi-Lagrangian framework for simulating multiphase flows. *Journal of Computational Physics*, 332:21–46, 2017.

[23] J. Palmore and O. Desjardins. A volume of fluid framework for interface-resolved simulations of vaporizing liquid-gas flows. *Journal of Computational Physics*, 399:108954, 2019.

[24] S. B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[25] S. Popinet. An accurate adaptive solver for surface-tension-driven interfacial flows. *Journal of Computational Physics*, 228(16):5838 – 5866, 2009.

[26] L. Rayleigh. On the capillary phenomena of jets. *Proc. R. Soc. London*, 29:71–97, 1879.

[27] M. Rudman. A volume-tracking method for incompressible multifluid flows with large density variations. 1998.

[28] A. Salih and S. Ghosh Moulic. Oscillation of a liquid drop in a zero-gravity environment - a benchmark problem for two-phase flow computations. 12 2002.

[29] G. S. Settles and M. Hargather. A review of recent developments in schlieren and shadowgraph techniques. 2017.

[30] P. Sheehy and M. Owkes. Numerical Study of Electric Reynolds Number on Electrohydrodynamic (Ehd) Assisted ... *Atomization and Sprays*, (December), 2016.

[31] 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):110–136, 2003.

[32] M. Sussman, K. Smith, M. Hussaini, M. Ohta, and R. Zhi-Wei. A sharp interface method for incompressible two-phase flows. *Journal of Computational Physics*, 221(2):469 – 505, 2007.

[33] G. Tryggvason, R. Scardovelli, and S. Zaleski. *Direct Numerical Simulations of Gas-Liquid Multiphase Flow*. 01 2011.

[34] Tu, Jiyuan and Yeoh, Guan Heng and Liu, Chaoqun. *Computational fluid dynamics: a practical approach*. Butterworth-Heinemann, 2013.

[35] J. Weare. Particle filtering with path sampling and an application to a bimodal ocean current model. *Journal of Computational Physics*, 228(12):4312 – 4331, 2009.

[36] J. Westerweel, G. E. Elsinga, and R. J. Adrian. Particle image velocimetry for complex and turbulent flows. *Annual Review of Fluid Mechanics*, 45(1):409–436, 2013.

[37] A. Xuan and L. Shen. A conservative scheme for simulation of free-surface turbulent and wave flows. *Journal of Computational Physics*, 378:18 – 43, 2019.

[38] T. Yamaguchi, T. Ishikawa, K.-i. Tsubota, Y. Imai, M. Nakamura, and T. Fukui. Computational blood flow analysis new trends and methods. *Journal of Biomechanical Science and Engineering*, 1:29–50, 01 2006.

APPENDIX: POINT-WISE HEIGHT FUNCTION COEFFICIENT CODE

The following code was written in Matlab to find finite difference coefficients for the point-wise 5^{th} order and 2^{nd} order height function methods.

```
clear all; close all; clc


%%
%---------------------------%
%     dh/dx [6 Eqns]        %
%---------------------------%

syms h a3 a_3 a2 a_2 a0 a1 a_1
eqn1= a_3                 +a_2               +a_1             +a1            +a2
eqn2= a_3*(-5*h/2)        +a_2*(-3*h/2)      +a_1*(-h/2)      +a1*(h/2)      +a2*(3
eqn3= a_3*(-5*h/2)^2/2    +a_2*(-3*h/2)^2/2  +a_1*(-h/2)^2/2  +a1*(h/2)^2/2  +a2*(3
eqn4= a_3*(-5*h/2)^3/6    +a_2*(-3*h/2)^3/6  +a_1*(-h/2)^3/6  +a1*(h/2)^3/6  +a2*(3
eqn5= a_3*(-5*h/2)^4/24   +a_2*(-3*h/2)^4/24 +a_1*(-h/2)^4/24 +a1*(h/2)^4/24 +a2*(3
eqn6= a_3*(-5*h/2)^5/120 +a_2*(-3*h/2)^5/120+a_1*(-h/2)^5/120+a1*(h/2)^5/120+a2*(3


sol = solve( [eqn1, eqn2, eqn3, eqn4, eqn5, eqn6] , [a1, a2,a3,a_1,a_2,a_3]);
h  = 1;
format long
a_3x = eval(sol.a_3);
a_2x = eval(sol.a_2);
a_1x = eval(sol.a_1);
```

```
a1x  = eval(sol.a1);
a2x  = eval(sol.a2);
a3x  = eval(sol.a3);
hx5   = -[a_3x; a_2x; a_1x; a1x; a2x; a3x];

%%
%---------------------------%
%    d2h/dx2 [6 Eqns]       %
%---------------------------%

syms a3 a_3 a2 a_2 a0 a1 a_1 h
eqn1= a_3                +a_2               +a_1             +a1            +a2
eqn2= a_3*(-5*h/2)       +a_2*(-3*h/2)      +a_1*(-h/2)      +a1*(h/2)      +a2*(3
eqn3= a_3*(+5*h/2)^2/2   +a_2*(+3*h/2)^2/2  +a_1*(+h/2)^2/2  +a1*(h/2)^2/2  +a2*(3
eqn4= a_3*(-5*h/2)^3/6   +a_2*(-3*h/2)^3/6  +a_1*(-h/2)^3/6  +a1*(h/2)^3/6  +a2*(3
eqn5= a_3*(+5*h/2)^4/24  +a_2*(+3*h/2)^4/24 +a_1*(+h/2)^4/24 +a1*(h/2)^4/24 +a2*(3
eqn6= a_3*(-5*h/2)^5/120 +a_2*(-3*h/2)^5/120+a_1*(-h/2)^5/120+a1*(h/2)^5/120+a2*(3

sol = solve([eqn1, eqn2, eqn3,eqn4, eqn5, eqn6], [a1, a2,a3,a_1,a_2,a_3]);
h  = 1;
format long
a_3xx = eval(sol.a_3);
a_2xx = eval(sol.a_2);
a_1xx = eval(sol.a_1);
a1xx  = eval(sol.a1);
a2xx  = eval(sol.a2);
```

```
a3xx  = eval(sol.a3);
hxx5   = -[a_3xx; a_2xx; a_1xx; a1xx; a2xx; a3xx];


x=linspace(0,5,500);
dx=linspace(1,1e-3, length(x));
ana = exp(x);
f   = [ exp(x-5.*dx/2);
exp(x-3.*dx/2);
exp(x-   dx/2);
exp(x+   dx/2);
exp(x+3.*dx/2);
exp(x+5.*dx/2) ];
hxxw = [-1/8;7/8;-3/4;-3/4;7/8;-1/8];

Hxx5 =  sum(hxx5.*f)./dx.^2;
Hxxw =  sum(hxxw.*f)./dx.^2;

er5xx = sqrt((Hxx5-ana).^2) ./ sqrt(ana.^2);



%%
% ===============
%     PLOTS
% ===============
```

```
h=figure(1); clf(1)
plot (x,ana,'k-','LineWidth',2)
hold on
plot (x(1:10:end),Hxx5(1:10:end),'ro','LineWidth',2)
xlabel('Values of x','interpreter','latex','fontsize',16)
ylabel('Values of $e^x$','interpreter','latex','fontsize',16)
legend('Analytic','5th Order','Location','NorthWest')
%title('Calculation of $\frac{d^2h}{dx^2}$ vs Analytic Solution','interpreter','la
basefile = 'figs';
saveas(h,fullfile(basefile,'2ndDer.png'));

Hx5 =  sum(hx5.*f)./dx;
er5 = sqrt((Hx5-ana).^2) ./ sqrt(ana.^2);

h=figure(2); clf(2)
plot (x,ana,'k-','LineWidth',2)
hold on
plot (x(1:10:end),Hx5(1:10:end),'ro','LineWidth',2)
xlabel('Values of x','interpreter','latex','fontsize',16)
ylabel('Values of $e^x$','interpreter','latex','fontsize',16)
legend('Analytic','5th Order','Location','NorthWest')
%title('Calculation of $\frac{dh}{dx}$ vs Analytic Solution','interpreter','latex'
basefile = 'figs';
saveas(h,fullfile(basefile,'1stDer.png'));

h=figure(3); clf(3)
```

```
loglog(dx,er5,'-o','LineWidth',2);
hold on ;
loglog(dx,dx.^(2)*1e-2,'LineWidth',2);
hold on ;
loglog(dx,dx.^(3)*1e2,'LineWidth',2);
hold on ;
loglog(dx,dx.^(4)*1e6,'LineWidth',2);
hold on ;
loglog(dx,dx.^(5)*1e10,'LineWidth',2);
xlabel_h=xlabel({'','dx',''},'interpreter','latex','fontsize',16);
ylabel_h=ylabel({'','Error',''},'interpreter','latex','fontsize',16);
%title('Convergence of $\frac{dh}{dx}$ Error With Mesh Refinement','interpreter','
legend('5th Order Method','2nd Order','3rd Order','4th Order','5th Order', 'Locati
set(gca, 'Color', 'none'); % Sets axes background
basefile = 'figs';
saveas(h,fullfile(basefile,'1stErr.png'));

h=figure(4); clf(4)
loglog(dx,er5xx,'-o','LineWidth',2);
hold on ;
loglog(dx,dx.^(2)*1e-2,'LineWidth',2);
hold on ;
loglog(dx,dx.^(3)*1e2,'LineWidth',2);
hold on ;
loglog(dx,dx.^(4)*1e6,'LineWidth',2);
hold on ;
```

```
loglog(dx,dx.^(5)*1e10,'LineWidth',2);
xlabel_h=xlabel({'','dx',''},'interpreter','latex','fontsize',16);
ylabel_h=ylabel({'','Error',''},'interpreter','latex','fontsize',16);
%title('Convergence of $\frac{d^2h}{dx^2}$ Error With Mesh Refinement','interprete
legend('5th Order Method','2nd Order','3rd Order','4th Order','5th Order', 'Locati
set(gca, 'Color', 'none'); % Sets axes background
basefile = 'figs';
saveas(h,fullfile(basefile,'2ndErr.png'));
```

APPENDIX: INTEGRAL HEIGHT FUNCTION COEFFICIENT CODE

The following code was written in Matlab to find finite difference coefficients for the integral 5^{th} order height function method.

```
%function [ hx,hxx ] = weights
% Compute weights for 5th order height function method
clear; clc

syms x z xik zik dx_ dz

% Order of expansion
M=5; N=0;

% Taylor series incluing upto max(M,N) order terms
h=sym(zeros(M+1,N+1));
for n=0:N
for m=0:M
h(m+1,n+1)=(x-xik)^m*(z-zik)^n/(factorial(m)*factorial(n));
end
end

% Allocate Matrices
Z=sym(zeros((M+1)*(N+1),(M+1)*(N+1)));

B=sym(eye  ((M+1)*(N+1),(M+1)*(N+1)));
```

```
% Loop over cells in stencil
c=0;
for kp=0;%-3:2
for ip=-3:2
c=c+1; % Column number
%fprintf('%i/6 \n',c)
% Integrate h over this cell to form height
H=int(h,x,xik+(ip)*dx_,xik+(ip+1)*dx_)/(dx_);
% Get coefficents from F
Z(:,c)=H(:);
end
end

% Constrained least squares that enforces 2nd-order
% finite difference opperators and pushes magnitude
%%Z(6,

% Solve for A's (in a vector)
Av=Z\B;

%%   Weights for needed derivatives    %
% ---------------------------------- %
dx_=1;
c=0;
for n=0;%:N
```

```
for m=0:M
c=c+1;
if     m==1 && n==0;  hx =eval(reshape(Av(:,c),6,1)); % h_x
elseif m==0 && n==1;  hz =eval(reshape(Av(:,c),5,5)); % h_z
elseif m==2 && n==0;  hxx=eval(reshape(Av(:,c),6,1)); % h_xx
elseif m==0 && n==2;  hzz=eval(reshape(Av(:,c),5,5)); % h_zz
elseif m==1 && n==1;  hxz=eval(reshape(Av(:,c),5,5)); % h_xz
end
end
end
% Test accuracy
figure(1); clf(1)
syms x z
% Function
f=x^5+2*x*cos(x);
% Location to compute derivative
xo=1;
zo=1;

% Compute integral of f for construction of heights
syms z1 z2 x1 x2
intf=matlabFunction(int(int(f,z,z1,z2),x,x1,x2),'Vars',[x1,x2,z1,z2]);

% Loop over derivatives to test
c=0;
for n=0:N % d/dy^n
```

```
for m=0:M % d/dx^m
c=c+1;
% Get correct A's for this derivative
A=reshape(Av(:,c),6,1);

% Grid size to compute heights on
dxs=[1,1/2,1/4,1/8,1/16,1/32,1/64];
error=zeros(1,length(dxs));
for ndx=1:length(dxs);
dx=dxs(ndx);
dz=1; %dxs(ndx);
exact=subs(diff(diff(f,z,n),x,m),[x,z],[xo,zo]);
% Compute heights
ht=zeros(6,1);
for j=0;%-2:2
for i=-3:2
ht(i+4,j+1)=intf(xo+(i)*dx,xo+(i+1)*dx,zo+(j-1/2)*dz,zo+(j+1/2)*dz)/(dx*dz);
end
end

% Perform sum(A*heights) (Evaluates A with this dx/dy)
dx_=dx;
computed=sum(sum(eval(A).*ht));

% Analysis
error(ndx)=abs(exact-computed);
```

```
%fprintf('Exact=%5.5f, Computed=%5.5f, Error=%5.5e\n',exact,computed,error(ndx))
end
%         subplot(N+1,M+1,c)
%         loglog(dxs,error,'-o')
%         hold on
%         loglog(dxs,dxs.^5)
%         title(['d/dx^',num2str(m),' d/dy^',num2str(n)])
%         drawnow
if c==2 || c ==3
h=figure(c); clf(c)
plot(N+1,M+1)
loglog(dxs,error,'-.','LineWidth',3)
hold on
loglog(dxs,dxs.^5,'LineWidth',3)
title(['d/dx^',num2str(m),' d/dy^',num2str(n)],'interpreter','latex','fontsize',16
legend('5th Order Method','5th Order', 'Location', 'NorthWest')
set(gca, 'Color', 'none','Fontsize',16);
drawnow
basefile = 'figs';
saveas(h,fullfile(basefile,['WeightsDer',num2str(c),'.png']));
end
end
end

%% Convert to Fortran notation (Removing
% hx = [-1/15; -1/9; -1/3;  1/3; 1/9; 1/15];
```

```
% hx = [-1/8; -1/8;    0;    0; 1/8; 1/8];
% hxx= [ 1/8;  1/8; -1/4; -1/4; 1/8; 1/8];
dhs={'hx','hxx'};
dx_=1;
dz_=1;
for n=1:length(dhs)
dh=eval(dhs{n});
for kp=0;%-2:2
%         for ip=-3:2
%            strs{ip+4}=eval(dh(ip+4,kp+1));
%         end
fprintf('data dhd%s(:,%+i) / %+5.15e_WP, %+5.15e_WP, %+5.15e_WP, %+5.15e_WP, %+5.1
end
end
%end
```