2020

Quadratic Moment-of-Fluid Interface Reconstruction

Jacob Spainhour

THE FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

QUADRATIC MOMENT-OF-FLUID INTERFACE RECONSTRUCTION

By

JACOB SPAINHOUR

A Thesis submitted to the
Department of Mathematics
in partial fulfillment of the requirements for graduation with
Honors in the Major

Degree Awarded:
Spring, 2020

The members of the Defense Committee approve the thesis of Jacob Spainhour defended on April 9, 2020.

Dr. Mark Sussman
Thesis Director

Dr. Bryan Quaife
Outside Committee Member

Dr. M. Nicholas Moore
Committee Member

Abstract

The Moment-of-fluid method introduced by Dyadechko and Shashkov [4] is a technique for numerically representing deforming boundaries in a localized manner. Because of the "locality" property of the Moment-of-fluid representation, the symmetric difference error for the Moment-of-fluid piecewise linear reconstruction algorithm is smallest among volume tracking algorithms. The novelty of the Moment-of-fluid method, when compared to standard volume-of-fluid methods, is that both the zeroth and first moments are discretized.

A natural extension of the piecewise linear moment-of-fluid method is to increase the order of the incorporated moment data so that a higher order interface can be reconstructed. In this paper, we present a quadratic moment-of-fluid method that reconstructs a second-order level set approximation from zeroth, first, and second moments. The novel Quadratic Moment-of-fluid algorithm, herein described and developed, is expected to lead to significantly more accurate computational fluid dynamics algorithms for multiphase flows with prominent surface tension effects. The surface tension force is proportional to the interface curvature. The interface curvature, derived from the piecewise parabolic interface reconstruction, is expected to be approximated with much improved accuracy to that derived from the piecewise linear moment-of-fluid reconstruction.

Contents

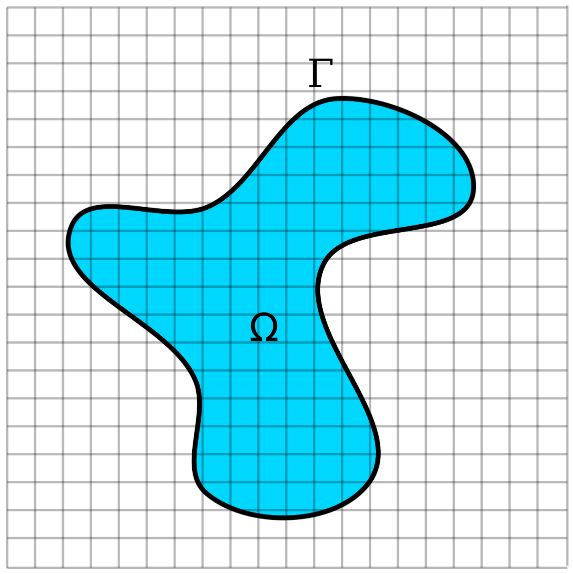
Chapter 1

Introduction

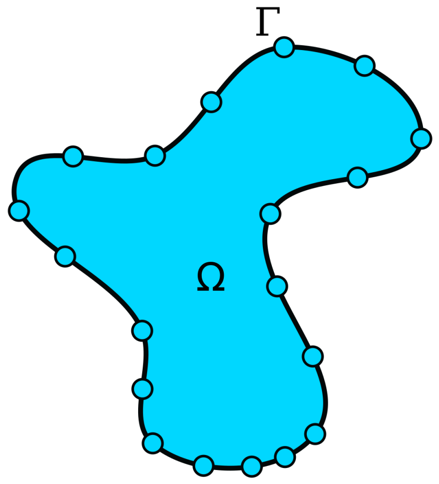
1.1 Interface Reconstruction Algorithms

Broadly speaking, the field of computational fluid dynamics concerns itself with the numerical simulation of fluid flows and computational solutions of their governing equations. These methods are necessarily restricted in their resolution and efficacy by the limits of modern computers. As a result, approaches to problems in CFD can typically be classified according to how they choose to represent the continuous fluid numerically, commonly referred to as the specification of the fluid. A *Lagragian* specification is one that analyses the motion and position of the fluid system by tracking discrete points identified within. By comparison, an *Eulerian* specification is one that tracks fixed points in space, often manifesting as computational cells that tessellate the domain. Methods developed according to this specification then observe how the fluid moves through these cells. In practice, most modern methods in CFD use some combination of the two specifications in order to combine their strengths for a given problem.

Methods of interface reconstruction can be further classified as interface *tracking* or interface *capturing*. For example, front tracking methods often approach the problem from an exclusively Lagrangian perspective. In these methods, markers are placed along the interface, being treated as particles under the influence of the velocity field applied to the fluid [5]. While this has benefits in being able to more directly construct the interface from its data points, and can accurately represent interface topology at scales impractical for equivalent Eulerian schemes, numerical implementations are frequently slow [4] and prone to complications when particle trackers become clustered along the interface [7]. The alternative is an interface capturing method along an Eulerian grid that instead records auxiliary data about the fluid for each cell, and has a separate step that reconstructs this data into a continuous representation of the fluid. A cartoon that demonstrates the general structure of these two specifications is presented in Figure 1.1. It is these kinds of interface capturing methods that receive the most focus within this paper, particularly those whose primary auxiliary component is the volume fractions of fluids in each cell.



(a) Eulerian

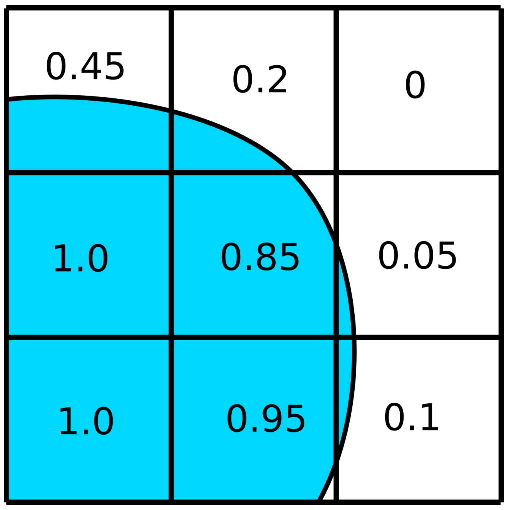


(b) Lagrangian

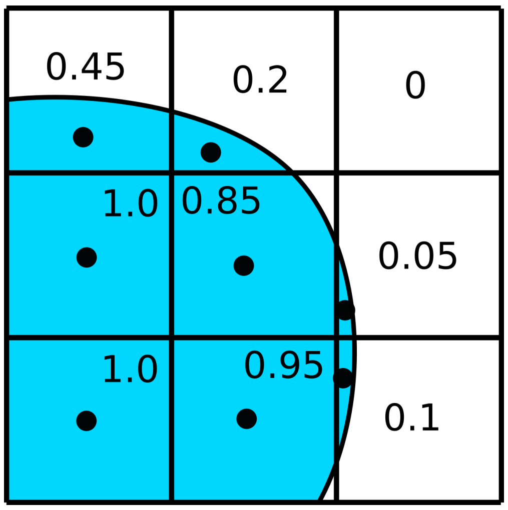
Figure 1.1: Different Models of Fluid Specification

Volume-of-fluid (VOF) methods are among the oldest and most practically used form of interface capturing. In line with the Eulerian perspective of a fluid, the only data maintained about cell topology is the fraction of the cell volume taken up by each of its component fluids. These volume fractions are then altered under the underlying velocity field, doing so by incorporating aspects of a Lagrangian interpretation of the grid itself. As with any interface capturing method, the interface is reconstructed directly from this volume information following each solution of the fluid simulation. Modern implementations of VOF methods involve Piecewise Linear Interface Calculation (PLIC) techniques, which construct an optimal linear interface for each cell in the domain [12]. Implementations of these methods are preferred for their simplicity and speed when added to a numerical fluid solver. However, VOF-PLIC methods are not without downsides. For example, the proper orientation of the linear interface within a single cell can only be computed by considering the orientation of neighboring cells. This makes the details of the implementation depend heavily on the geometry of the mesh, restrict the accuracy of the technique to the width of this cell-stencil. In spite of these limitations, volume-of-fluid methods are unique and highly desirable for their mass conserving properties, which often more than outweigh these drawbacks [4].

The Moment-of-fluid (MOF) method is an additional interface capturing method described on a fixed Eulerian grid. The details of this method will be introduced in greater detail later, as the Moment-of-fluid method as presented by Shashkov and Dyadechko [4] represents a theoretical framework within which we later present a novel expansion. The underlying motivation behind MOF is the development of a method that maintains more complex data within each cell beyond volume fractions to obtain a more accurate piecewise linear interface calculation. Where typical volume-of-fluid methods keep track only volume fractions within a cell, which itself is the zeroth moment, MOF expands on this by also considering the first moments of the fluid within a each cell. This comparison is demonstrated in Figure 1.2. This moment data can then be advanced in time through standard Lagrangian techniques, as it physically represents the centroid of the fluid, a single point in the cell. When used in an interface reconstruction algorithm, the volume fraction and centroid for each cell is sufficient to design a locally optimal PLIC [4].



(a) VOF: Volume Fractions



(b) MOF: Volume Fractions and Centroids

Figure 1.2: Data stored for each computational cell

Within this paper, we introduce a technique referred hereafter as the Quadratic Moment-of-fluid technique (QMOF), where the second moments are similarly maintained so that a Piecewise Parabolic Interface Calculation (PPIC) can be performed. By performing this QMOF-PPIC on each cell in the domain, we are left with an interface reconstruction technique that is more accurate than its PLIC counterparts, has advantages over second order VOF methods, and can directly and accurately capture curvature in a way inaccessible to a linear reconstruction. In demonstrating its merits, we restrict ourselves within this paper to a two-phase fluid in 2D. Under this construction, space not taken up by the designated reference fluid is assumed to be filled by its complement. A similar concession is made in restricting the computational cells to be strictly rectangular. Regardless, we acknowledge that the methods described are similarly applicable to arbitrary cells and in higher dimensions with minimal reformulation of the given theoretical framework.

1.2 Second Order Interface Capturing Methods

As volume-of-fluid methods have now been studied for decades [6], a number of second order, quadratically accurate Eulerian interface capturing methods have been introduced to combat the limitations of a linear reconstruction. An example that specifically uses a piecewise parabolic interface calculation is described by Price [7]. The PPIC technique derived therein operates on many of the same principles as the standard VOF method, tracking volume fractions across a uniform grid and using cells in a 3×3 grid to compute the optimal parabola for the center cell [7]. A similar approach is taken by Renardy and Renardy [9], which constructs a general second order level set function given an array of volume fractions with the explicit purpose of modeling surface tension effects. Another approach, a Quadratic Spline based Interface reconstruction method, is described by Diwakar, Das, and Sundararajan [3]. This technique begins with a discontinuous PLIC-derived interface approximation, and performs an interpolation that produces a second order, continuous interface [3].

Other methods have been explored that approximate the curvature on a fluid defined through volume factions without an explicit reconstruction of the interface. One such method introduced by Chorin [2] finds the best fit for an osculating circle in a volume-of-fluid given an array of partially filled fluid cells, from which curvature can be directly computed. More recently, a method introduced by Qi et al. uses neural networks to approximate curvature in a Volume-of-fluid regime [8], in which the network is trained according to a 3×3 collection of partially filled fluid cells. While each of these techniques has a variety of benefits over a standard PLIC reconstruction through the use of second order curves, it remains that each uses only volume data to perform their respective reconstruction. By contrast, the QMOF algorithm presented herein incorporates time-advanced moment data in addition to volume fractions. In doing so, a method is generated that can be applied to cells locally, where the benefits of this locality manifest in a smaller, more accurate stencil size, and certain technological advantages that well suit the algorithm to practical application.

1.3 Linear Moment-of-fluid Interface Reconstruction

The standard Moment-of-fluid method introduced by Dyadechko and Shashkov [4] described an optimal PLIC that considers the first moment data, equivalent to the centroid, of the fluid in a cell in addition to its volume fraction. As with all VOF-PLIC methods, we enforce a strict volume conservation property for each computational cell. Notably, this method represents linear interfaces exactly, and finds optimal non-linear interfaces with greater accuracy than Volume-of-fluid PLIC methods. In addition to increased accuracy, Moment-of-fluid methods in general have several aspects that allow it to be implemented efficiently in software. For example, as MOF algorithms can be applied to each cell individually without respect to its neighbors, then be collected into a black box that can be applied to cells individually without consideration of the non-local grid structure. In addition to providing highly desirable numerical results, this method can be physically justified on the basis that centroid data can be advanced in time along with the rest of the fluid, and tracked uniquely within each cell [4].

Formally, the Moment-of-fluid linear interface reconstruction method is defined in n-dimensions as follows. Let $U \in \mathbb{R}^n$ be a convex domain that represents a single computational cell. Within our implementation, we take U to be a hyperrectangle, but this is not necessary for a generalized MOF-PLIC method. Let Ω be a non-trivial subset of U such that $\emptyset \subsetneqq \Omega \subsetneqq U$, representing the intersection of a chosen reference fluid with a given cell. Because Ω is nonempty, it has nontrivial moment data. That is to say, the integrals we later define are evaluated over nonempty domains, and are not trivially zero. Define Γ to be the boundary of Ω interior to U, representing the fluid interface. The layout of one such computational cell is presented in Figure 1.3.

For the linear MOF, we consider the first two orders of moment data. The zeroth moment is the volume of the fluid, and is given by

$$M = \int_\Omega d\vec{x}.$$

While this value M is not strictly speaking equal to the volume fraction, M and the true volume fraction are equivalent in their use for the MOF-PLIC because the total volume in each cell is fixed. Similarly, we scale the first moments by the volume M so that they correspond to the geometric centroid of the reference fluid. This centroid has n independent components, and is computed according to the following integrals:

$$M_{x_i} = \frac{\int_\Omega x_i \, d\vec{x}}{M} \quad \text{for } i = 1, \ldots, n$$

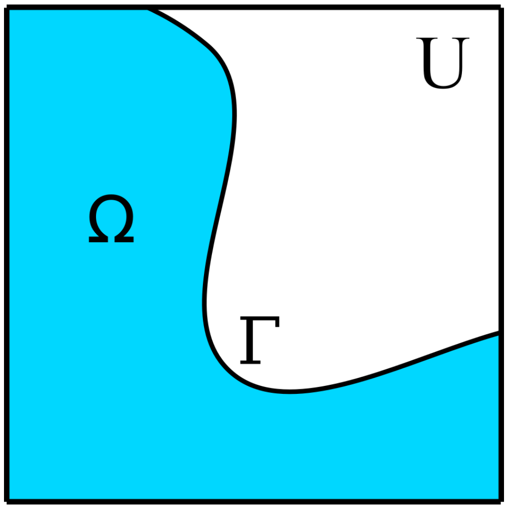
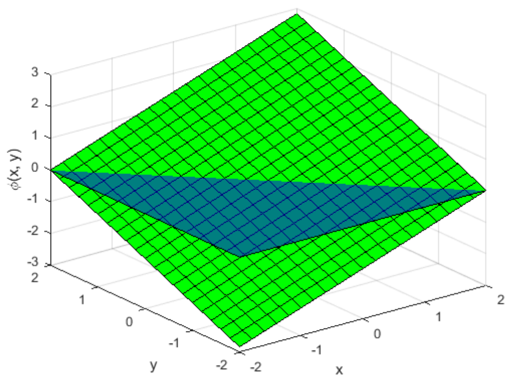
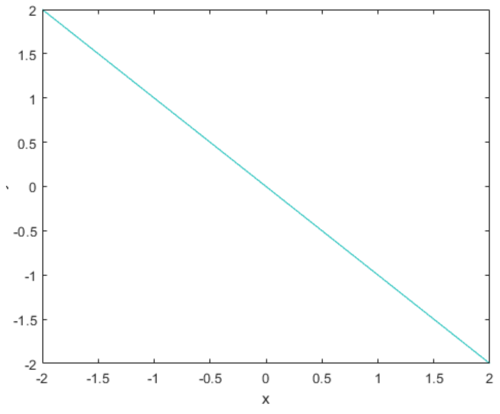


Figure 1.3: A single computational cell in 2D

This leads to a set of $n+1$ parameters which determine the position of the linear interface approximation. Within our formulation of MOF-PLIC, we choose to represent the interface between fluids numerically through the use of an auxiliary signed distance function whose zero level set represents the interface, referred to as the *level set function*, ϕ. An example of such a ϕ is presented in figure 1.4. While this construction is largely unnecessary for MOF as described by Dyadechko and Shashkov, as centroids and volume fractions can in principle be computed using purely geometric properties [4], it meshes nicely with the specifics of the implementation used to handle parabolic level set functions, described in detail later.



(a) Linear level set function



(b) Zero level set

Figure 1.4: An example of an linear-interface-defining level set function

Given a level set function $\phi : \mathbb{R}^n \to \mathbb{R}$, define the approximation to the fluid $\overline{\Omega}$ and the approximation to its interface $\overline{\Gamma}$ where

$$\overline{\Omega} = \{\vec{x} : \phi(\vec{x}) < 0\} \cup U,$$

$$\overline{\Gamma} = \{\vec{x} : \phi(\vec{x}) = 0\}$$

With these definitions, $\overline{\Omega}$ has its own set of $n+1$ moment data given by the following formulas

$$\overline{M} = \int_{\overline{\Omega}} d\vec{x} \qquad \overline{M_{x_i}} = \frac{\int_{\overline{\Omega}} x_i d\vec{x}}{\overline{M}} \quad \text{for } i = 1, \dots, n.$$

By the structure of the moment of fluid algorithm, the optimal level set function ϕ is one that satisfies the following conditions.

$$\phi(\vec{x}) = \phi(a_1, \dots, a_n, d) = \sum_{i=1}^{n} a_i x_i + d, \quad ||\vec{a}|| = 1 \tag{1}$$

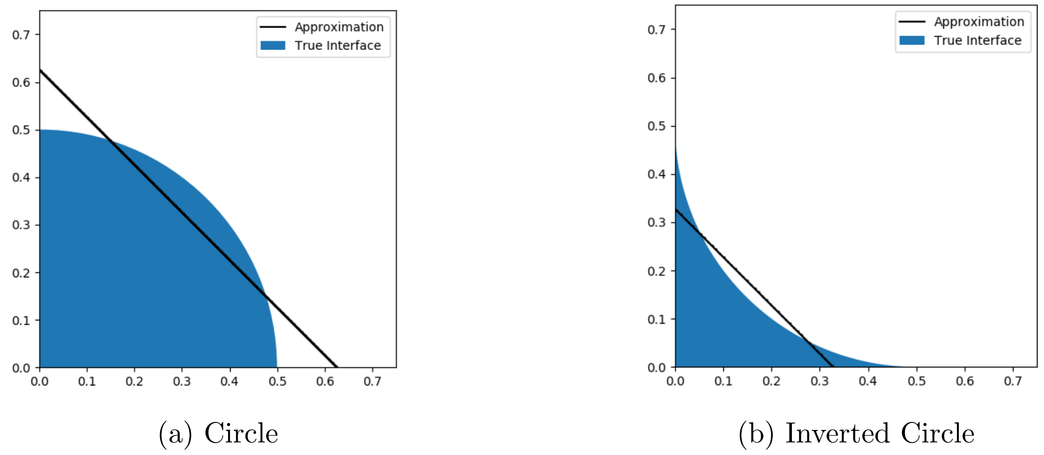
$$\overline{M} = M \tag{2}$$

$$\phi = \arg\min_{\phi} \left\{ \sum_{i=1}^{n} ||M_{x_i} - \overline{M_{x_i}}|| \right\} \tag{3}$$

The first of these three conditions enforces that the interface defined by ϕ is linear, being analogously described by its n coefficients and single intercept term. However, this does not uniquely define an interface, as a hyperplane in n-dimensions can be defined by exactly n parameters, rather than the $n+1$ parameters of the function $\phi(a_1, \dots, a_n, d)$. However, because there exist $n+1$ total components of moment data to match (one volume fraction and n components of the centroid), the problem is overdetermined. That is to say, the n parameters of the interface cannot in general satisfy the $n+1$ components of reference data exactly. Therefore, we add an additional constraint. We enforce that the volume $\overline{M}$ matches the reference volume M exactly, and optimize the n parameters to best fit the remaining n components of the centroid. This constraint is primarily motivated by the significance of volume preserving methods in practical applications. It can be shown that there exists a unique such ϕ, which can be calculated through a variety of optimization methods [4]. We introduce a simple such method for the 2D case later.

One of the benefits of this method over standard VOF-PLIC methods is the ability to compute the interface for a single cell without the use of its neighbors. Moreover, the physical geometry of the mesh is disconnected from the topology of the fluid, making this an ideal algorithm to apply to adaptive grid techniques. These properties combine into a concise black-box algorithm that can be applied to arbitrary cells individually and identically.

Noting that straight line interfaces can be captured exactly, examples of the MOF algorithm applied to common curved are presented in Figure 1.5. These represent interface shapes that can be accurately captured with suitable refinement along the interface. Moreover, the ability of the MOF algorithm to determine the accuracy of a linear approximation based on the distance between the true centroid and that of the reference provides a convenient metric on which to perform such a refinement procedure [1].

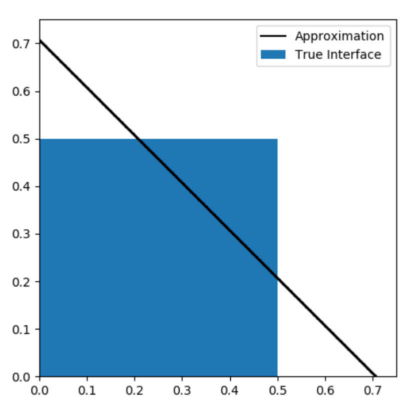


(a) Circle (b) Inverted Circle

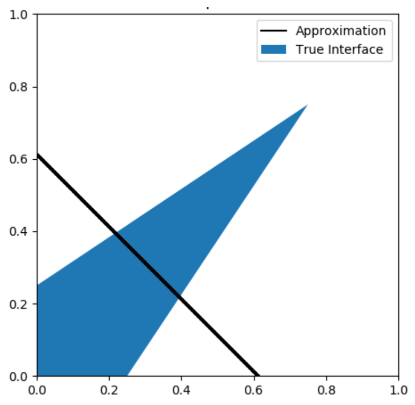
Figure 1.5: Linear MOF as applied to curved surfaces

While MOF-PLIC can be considered optimal among linear reconstructions [4], any PLIC method on a nonlinear interface will fail to capture subgrid details that otherwise define the fluid topology. In the below examples, a linear reconstruction fails to accurately represent the interface entirely at any refinement level save for those that, by coincidence or otherwise, align perfectly with the grid. For regions with corners, a linear approximation can only capture the orientation of the cell, as shown in Figure 1.6. Note that as far as the approximation is concerned, these cells are identical to those in Figure 1.5.

In other examples, it is not possible for a linear approximation to even capture the orientation of the fluid correctly, as seen in Figure 1.7. These interfaces, both sharp corners and filamentary regions of the fluid, are similarly common in practical problems. Such interfaces stand to benefit the most from the introduction of a quadratic approximation. It is with these shortcomings in mind that we introduce a quadratic moment of fluid method, one that produces a piecewise parabolic interface calculation.

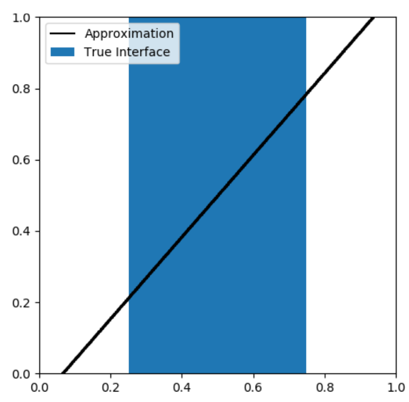


(a) Corner

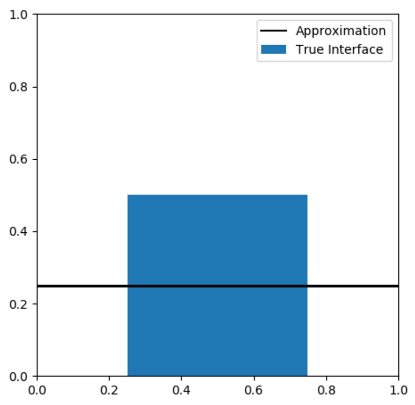


(b) Sharp Corner

Figure 1.6: Linear MOF as applied to sharp corners



(a) Filament



(b) Filament End

Figure 1.7: Linear MOF as applied to filamentary regions

1.4 Quadratic Moment-of-fluid Interface Reconstruction

A natural extension of this method involves a relaxation of one of our constraints in hopes of obtaining a more accurate interface. Namely, we no longer require that ϕ define a linear interface, and instead restrict ϕ to a level set function that is at most second order in degree. This necessitates the use of additional data tracked for each cell, as there are now greater than n parameters that define our interface. An obvious choice is tracking higher order moment data for our reference fluid. Although these moments do not have as nice a physical interpretation as the volume or centroid, we nevertheless introduce formulae for higher order moment data for Ω (with analogous formulas existing for $\overline{\Omega}$). For $k > 0$, define the set of k^{th}-order moments in R^n:

$$M_{r(x)} = \frac{\int_\Omega r(x) d\vec{x}}{M} \quad \text{for} \quad r(x) \in \left\{ \prod_{i=1}^{n} x_i^{j_i} \,\middle|\, \sum_{i=1}^{n} j_i = k,\ j_i \in \mathbb{N} \right\}$$

Essentially, each of the k^{th} order moments is the integral of a product of monomials whose exponents sum to k. As a result of this construction, the total number of k^{th} order moments is equivalent to the number of monomials of n variables of degree k, which is equal to the multicombination of n elements being selected k times. Written with multiset notation, there are $\left(\!\binom{n}{k}\!\right) = \binom{n+k-1}{k}$ k^{th} order moments in $\mathbb{R}^n$.

While this formulation could conceivably be used to extend the moment-of-fluid to arbitrary order interface reconstruction, we remain focused on expanding a linear interface to one that is second order. This necessitates the use of second order moment data, resulting in a Quadratic Moment-of-fluid Method (QMOF). For simplicity of demonstration, we restrict our discussion of QMOF method to 2D, noting that analogous definitions and methods exist for higher dimensions. We then describe for QMOF analogous conditions for the standard MOF method described in the previous chapter:

$$\phi(x, y) = \phi(\vec{a}) = a_0 x^2 + a_1 y^2 + a_2 xy + a_3 x + a_4 y + a_5 \tag{1}$$

$$\text{where} \quad \vec{a} = [a_0, a_1, a_2, a_3, a_4, a_5]^T$$

$$\overline{M} = M \tag{2}$$

$$\phi = \arg\min_{\phi} \{ ||M_{x^2} - \overline{M_{x^2}}|| + ||M_{y^2} - \overline{M_{y^2}}|| + ||M_{xy} - \overline{M_{xy}}|| + ||M_x - \overline{M_x}|| + ||M_y - \overline{M_y}|| \} \tag{3}$$

with first and second moments defined as follows

$$M_x = \frac{\int_\Omega x dx dy}{M} \quad M_y = \frac{\int_\Omega y dx dy}{M}$$

$$M_{x^2} = \frac{\int_\Omega x^2 dx dy}{M} \quad M_{xy} = \frac{\int_\Omega xy dx dy}{M} \quad M_{y^2} = \frac{\int_\Omega y^2 dx dy}{M}$$

We note that the degree of each term in our level set function is at most 2, resulting in a zero level set that defines a quadratic curve. Moreover, the number of terms in the level set function of degree k is equal to the number of k^{th} order moments by the definition of $r(x)$ in the above formulation. Similar to the MOF-PLIC case, we note that the number of parameters for the level set function ϕ is one fewer than the number of terms in ϕ, making the problem once again overdetermined. In order to enforce uniqueness, we constrain the set of allowable level set functions to those whose volume exactly matches that of the reference fluid. The resulting optimization problem then seeks to minimize the error in each other component of moment data, weighted equally. In doing so, we expand the acceptable level sets in interface reconstruction to include those that generate quadratic curves for their interface, greatly enhancing the ability of $\overline{\Omega}$ to approximate Ω for complicated fluids.

While solutions to this optimization problem are more expensive to compute, there are many benefits that outweigh out this additional cost. The valuable properties of mass conservation in VOF methods and locality in the MOF-PLIC method are each preserved, but with a significant increase in approximation accuracy compared to a linear approximation. Additionally, there exist many common fluid geometries that are represented poorly by a linear approximant, such as corners and filaments. Properly representing such interfaces with a PLIC scheme requires additional computational cost in structuring the underlying grid. By maintaining higher order moment data, however, a significantly coarser grid can be used to obtain comparable errors on fluids with complicated interface topologies. This in turn results in a cheaper algorithm overall despite the more expensive optimization problem. Additionally, curvature calculations are very difficult for linear reconstructions, as straight lines have zero curvature by definition. As a result, curvature can only be approximated by combining data cross multiple cells, acting against any existing locality properties. However, under a QMOF scheme, a more accurate curvature approximation can be computed directly from the non-linear interface within each cell. In either case, a quadratic moment of fluid method meaningfully addresses the drawbacks of a piecewise linear interface calculation.

Chapter 2

Piecewise Parabolic Interface Calculation

In order to derive a Piecewise Quadratic interface calculation, we must derive the optimal level set function ϕ from the second order moment data using a numerical optimization technique. We first define an objective function C, referred to as the cost function, that encapsulates the minimality constraints of the QMOF method:

$$C(\vec{a}) = (M_{x^2} - \overline{M_{x^2}})^2 + (M_{y^2} - \overline{M_{y^2}})^2 + (M_{xy} - \overline{M_{xy}})^2 + (M_x - \overline{M_x})^2 + (M_y - \overline{M_y})^2$$

where $\vec{a}$ is a vector of parameters for the level set function ϕ that produces moment data $\overline{\mathbf{M}} = [\overline{M_{x^2}}, \overline{M_{y^2}}, \overline{M_{xy}}, \overline{M_x}, \overline{M_y}, \overline{M}]^T$. An overview of the algorithm is presented below, which when applied to each computational cell in a domain, produce a Piecewise Quadratic Interface Reconstruction. The algorithms used in each component of this procedure are further described in chapter 3. In particular, the algorithm as presented performs its optimization through a steepest descent algorithm, but this scheme can easily be substituted for as of yet unknown and more effective methods. Note that in the most general case, it is convenient to take the vector of level set parameters $\vec{a}$ to be the coefficients of an arbitrary second order polynomial function.

2.1 Quadratic vs. Parabolic Implementation

Because the set of quadratic curves in 2D is large, it is convenient to reduce the collection of acceptable curves from arbitrary second order level quadratic curves to parabolas exclusively. This produces a Piecewise Parabolic Interface Reconstruction, in which the interface in each cell is approximated with a parabolic curve. While this means that some sub-grid interface

Algorithm 1 Quadratic Moment of Fluid Interface Reconstruction with Steepest Descent. Given a computational cell $U = [x_l, x_u] \times [y_l, y_u]$, reference data $\mathbf{M} = [M_{x^2}, M_{y^2}, M_{xy}, M_x, M_y, M]^T$, and cost function C, this algorithm produces optimal level set parameters $\vec{a}$.

```
1: Transform $\mathbf{M}$ to $\widetilde{\mathbf{M}}$: equivalent moment data on the unit square $\widetilde{U} = [0,1]^2$.
2: Intelligently produce initial level set data $\vec{a}_0$.
3: while Error tolerance is not reached do
4:     Choose a search direction $\nabla C$, step length $\gamma$
5:     $\vec{a}_{n+1} \leftarrow \vec{a}_n - \gamma \nabla C(\vec{a}_n)$
6:     Perform flooding algorithm on $\vec{a}_{n+1}$ to maintain the volume constraint.
7: end while
8: Transform level set parameters $\vec{a}_n$ on $\widetilde{U}$ back to equivalent parameters on $U$.
```

features cannot be captured through the use of ellipses and hyperbolas, the majority of interfaces on a sufficiently refined grid do not have these structures, most typically because Ω is composed of a single component. Additionally, the numerical boundaries between different types of quadratic curves is obscured by paramaterizing the curve through the coefficients of its level set function, which causes instability in the algorithm. Finally, the specific parabolic parameterization used ameliorates the uniqueness problem that arises when $\vec{a}$ represents the coefficients of the level set function (as $\phi(c \cdot \vec{a})$ and $\phi(\vec{a})$ have identical zero level sets).

In two dimensions, a parabola is the collection of all points equidistant from a line (the directrix), and a single point (the focus). As shown in Figure 2.1, we numerically parameterize this parabola with the following values:

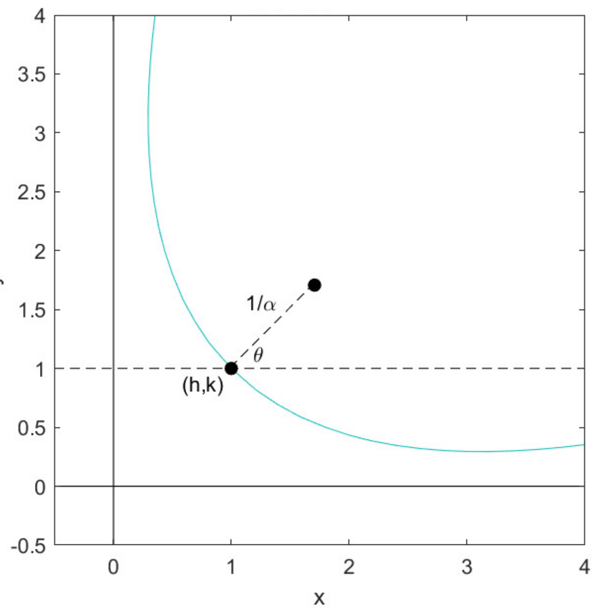


Figure 2.1: Parameters for 2D Parabola

- (h, k): The vertex of the parabola
- α: The inverse of the distance from the focus to the vertex, which is proportional to the curvature of the interface at the vertex
- θ: The counter-clockwise angle of the focus

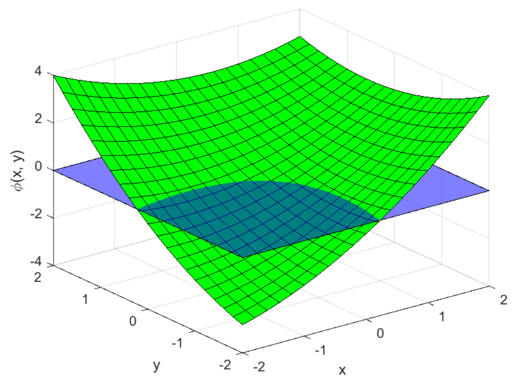
Given these parameters for the interface, we can derive the level set function that generates it. This construction begins with the standard form equation of a parabola opening in the positive x direction, i.e. $\theta = 0$. Note the substitution $a = 1/\alpha$. Geometrically, a straight line is equivalent to a parabola with a focus at infinity. Because this is inconvenient numerically, as straight lines are a common interface topology, we elect to parameterize our parabola with the reciprocal of the focus distance instead. We then let this equation become the zero level set of the level set function, with the positive x-axis being interior of its interface.

$$\begin{aligned}
(y-k)^2 &= 4a(x-h)\\
\frac{\alpha}{4}(y-k)^2 &= (x-h)\\
0 &= (x-h) - \frac{\alpha}{4}(y-k)^2\\
\phi(x,y) &= \frac{\alpha}{4}(y-k)^2 - (x-h)
\end{aligned}$$

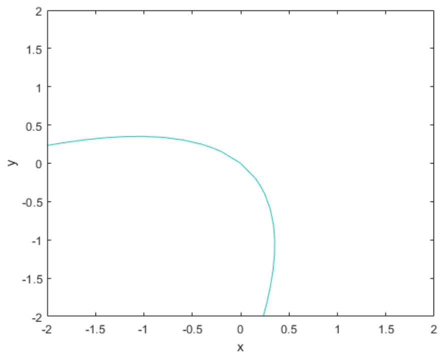
We then apply a rotation transformation around the vertex (h, k) so that the parabola opens in the direction of positive θ.

$$\begin{aligned}
x &\to (y-k)\sin(\theta) + (x-h)\cos(\theta) + h\\
y &\to (y-k)\cos(\theta) - (x-h)\sin(\theta) + k\\
\phi(x,y) &= \frac{\alpha}{4}((y-k)\cos(\theta) - (x-h)\sin(\theta))^2 - ((y-k)\sin(\theta) + (x-h)\cos(\theta))
\end{aligned}$$

Under this construction, our level set function $\phi(\vec{a})$ can be equivalently defined by $\phi(\alpha, h, k, \theta)$. An example of such a parabolic interface is presented in Figure 2.2. Note that while every parabolic interface is defined by a parabolic level set function, not every parabolic level set function defines a parabolic interface. In either case, the steepest descent algorithm implemented within acts identically. Empirical trials reveal, however, this parameterization is much more stable in practice, and leads to a more direct interpretation of the level set function. For example, θ is now periodic, and (h, k) are in the majority of cases located to within the computational cell. This parameterization is convenient to use for a MOF implementation as well, as when combined with a flooding algorithm, the straight line interface can be parameterized entirely by the value of θ (with the caveat that θ denotes the angle of an inward-facing normal, as opposed to one that points outwards).



(a) Second order level set function



(b) Zero level set

Figure 2.2: An example of an parabola-interface-defining level set function

2.2 Time Advancing of Moment Data

While not implemented within this paper, interface reconstructions are especially useful when describing fluids under the influence of some kind of velocity field. Moment-of-fluid techniques are particularly well-suited for this task, as moment data can be advanced individually in time on a cell-by-cell basis. For zeroth and first moments (volume fractions and centroids respectively) the procedure is identical to that described in the original paper describing a linear Moment-of-fluid method [4]. However, a more general procedure can be described as follows, as given by Shashkov and Dyadechko:

1. For each computational cell U_i, compute its *lagrangian prototype* $\widehat{U}_i$ by tracing the boundary of U_i backwards in time according to the velocity field.

2. For each prototype $\widehat{U}_i$, compute each component of moment data for its intersection with the reference fluid.

3. Transform each component of moment data from the prototype $\widehat{U}_i$ domain to the current time domain U_i. Similar transformations are described in section 3.3.

Certain geometric properties of the volume fraction and centroid greatly simplify the third step of this procedure for the zeroth and first order moment data. For example, the necessary transformation to move volume data from the prototype to the true cell is simply the identity. Because the fluid is assumed to be incompressible, the volume fractions in $\widehat{U}_i$ and U_i are identical. Moreover, because the centroid represents a single point in physical space that is maintained in each timestep, it can be advanced forwards in time according to the same techniques that generate the prototype. It is likely that similar properties exist for second order moment data, but are unexplored in this paper.

In any case, the primary computational difficulty in tracing the computational cell backwards in time and the construction of each lagrangian prototype. For non-linear fluid flows in particular, an at least second order scheme must be used in order to preserve the accuracy of the reconstruction itself. As demonstrated by [9], practical implementation of such methods exist for advancing volume and centroid data, and it is strongly suspected that a similar flux-based algorithm for interface advection can constructed according to the same principles.

Chapter 3

Numerical Implementation in 2D

3.1 Domain Transformation

Because our method is completely local to each computational cell, it is convenient to begin our algorithm with a transformation of the given moment data from being defined on an arbitrary rectangle $U = [x_l, x_u] \times [y_l, y_u]$ to the unit square $\widetilde{U} = [0,1]^2$. Doing so provides uniformity to many of the algorithms described later, with the use of machine learning in approximating initial conditions being a particularly useful application of such a transformation. Moreover, because this is a linear transformation, a direct mapping of the necessary integrals onto the new domain can be derived analytically.

$$\begin{aligned}
\widetilde{M} &= \frac{M}{(x_u - x_l)(y_u - y_l)} \\
\widetilde{M_x} &= \frac{M_x - x_l}{x_u - x_l} \\
\widetilde{M_y} &= \frac{M_y - y_l}{y_u - y_l} \\
\widetilde{M_{x^2}} &= \frac{M_{x^2} - 2x_l(x_u - x_l)\widetilde{M_x} - x_l^2}{(x_u - x_l)^2} = \frac{M_{x^2} - 2x_l M_x + x_l^2}{(x_u - x_l)^2} \\
\widetilde{M_{y^2}} &= \frac{M_{y^2} - 2y_l(y_u - y_l)\widetilde{M_y} - y_l^2}{(y_u - y_l)^2} = \frac{M_{y^2} - 2y_l M_y + y_l^2}{(y_u - y_l)^2} \\
\widetilde{M_{xy}} &= \frac{M_{xy} - y_l(x_u - x_l)\widetilde{M_x} - x_l(y_u - y_l)\widetilde{M_y} - x_l y_l}{(x_u - x_l)(y_u - y_l)} = \frac{M_{xy} - y_l M_x - x_l M_y + x_l y_l}{(x_u - x_l)(y_u - y_l)}
\end{aligned}$$

An inverse of this transformation can also be derived, one that maps unit reference data to the original domain. However, such a transformation is largely unneeded for the QMOF algorithm. Instead, after performing the QMOF-PPIC algorithm for a given computational cell, we must transform the coefficients of our level set function on the unit square to our original domain. These coefficients $\vec{\widetilde{a}} = [\widetilde{a}_0, \widetilde{a}_1, \widetilde{a}_2, \widetilde{a}_3, \widetilde{a}_4, \widetilde{a}_5]^T$ define an interface on $\widetilde{U} = [0,1]^2$, and the following set of functions map them back to $U = [x_l, x_u] \times [y_l, y_u]$.

$$
\begin{aligned}
a_0 &= \frac{\widetilde{a}_0}{(x_u - x_l)^2} \\
a_1 &= \frac{\widetilde{a}_1}{(y_u - y_l)^2} \\
a_2 &= \frac{\widetilde{a}_2}{(x_u - x_l)(y_u - y_l)} \\
a_3 &= \frac{-2\widetilde{a}_0 x_l}{(x_u - x_l)^2} - \frac{\widetilde{a}_2 y_l}{(x_u - x_l)(y_u - y_l)} + \widetilde{a}_3 \\
a_4 &= \frac{-2\widetilde{a}_1 y_l}{(y_u - y_l)^2} - \frac{\widetilde{a}_2 x_l}{(x_u - x_l)(y_u - y_l)} + \widetilde{a}_4 \\
a_5 &= \frac{\widetilde{a}_0 x_l^2}{(x_u - x_l)^2} + \frac{\widetilde{a}_1 y_l^2}{(y_u - y_l)^2} + \frac{\widetilde{a}_2 x_l y_l}{(x_u - x_l)(y_u - y_l)} - \frac{\widetilde{a}_3 x_l}{x_u - x_l} - \frac{\widetilde{a}_4 y_l}{y_u - y_l} + \widetilde{a}_5
\end{aligned}
$$

3.2 Quadrature Method for Implicitly Defined Volumes

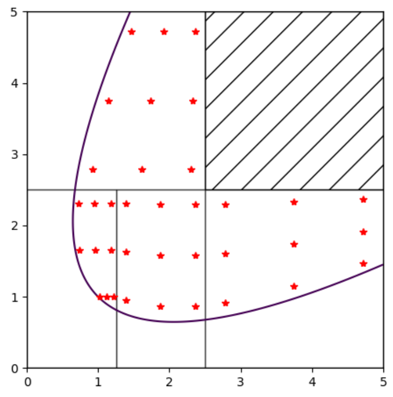
Throughout the QMOF algorithm, it is necessary to evaluate integrals whose domain is implicitly defined by the intersection of the interior of a level set function ϕ and a hyperrectangle U, producing integrals of the form

$$\int_\Omega f d\vec{x} \quad \text{where} \quad \Omega = \{\vec{x} \,|\, \phi(\vec{x}) < 0\} \cap U$$

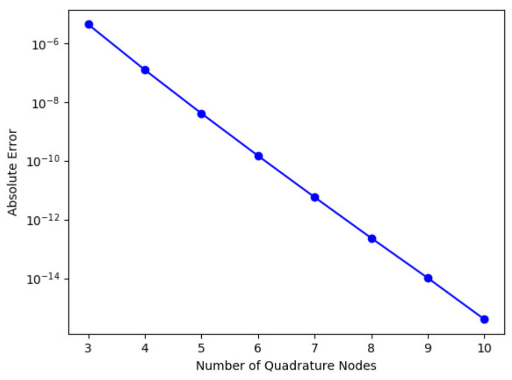
These integrals are computed through a specialization of an algorithm defined by Saye [10]. In brief, this algorithm recursively decomposes the hyperrectangular domain U into subrectangles until several implicitly defined height functions of one dimension can be defined. In 2D, these height functions $h : \mathbb{R} \to \mathbb{R}$ are created by fixing one direction of the level set function ϕ such that $h(x) = \phi(x, y_0)$ or $h(y) = \phi(x_0, y)$. These height functions are then integrated with Gaussian quadrature. When a hyperrectangle is found to be "full," in the sense that it is contained entirely interior to the level set, the integral is approximated with a full Gaussian tensor product. An example of this decomposition is provided in Figure 3.1, and a full description of both the algorithm and its C++ implementation are discussed in the appendix.

While the algorithm in principle can be applied to an arbitrary level set function in any dimension, there are a few notable simplifications to the original algorithm are possible due to restricting the allowable level sets to only second order polynomials in 2D. The majority of these involve maintaining explicit formulas for the various partial derivatives alongside the polynomial itself. For example, picking a suitable height direction involves finding the extrema of partial derivatives along rectangular domains. Because the derivatives of second order polynomials in a 2D domain are linear, these extrema must be located on the corners of the rectangular domain, saving greatly on computation time. Additionally, it is necessary to compute roots of the height functions h so that they are only integrated along the interior to the level set function. When the coefficients of the level set function are known explicitly, this becomes as simple as solving quadratic equations in 1D.

Because the integration itself is done using standard Gaussian quadrature for each height function, its convergence properties mirror those of a 1D Gaussian quadrature method. As seen in the convergence plot below, which computes the error between area interior to $\phi(x, y) = x^2 + 4y^2 - 1$ and the true area of $\pi/2$, the method converges to machine epsilon rapidly.



(a) Quadrature Nodes



(b) Quadrature Convergence

Figure 3.1: Properties of quadrature for implicitly defined volumes

3.3 Level Set Construction and Constraint

In either the PQIC or PPIC case, we must account for our volume constraint $M = \overline{M}$. This is accomplished through the use of a function $V : \mathbb{R}^6 \to R$ that adjusts the level set function to maintain a constant volume throughout the optimization algorithm. This function replaces the constant term in our coefficient parameterization, and has the practical effect of moving the intercept of ϕ without changing the shape of the interface it generates. Taking now $\vec{a} = [a_0, a_1, a_2, a_3, a_4]^T$,

$$\phi(x,y) = \phi(\vec{a}, M) = a_0x^2 + a_1y^2 + a_2xy + a_3x + a_4y + V(\vec{a}, M)$$

While an explicit formulation of V is incredibly obscure, evaluating it is equivalent to solving the root-finding problem

$$g(a_5) = \overline{M} - M = \int_{\overline{\Omega}} dxdy - M$$

where $a_5 = V(\vec{a}, M)$ is the intercept of the level set function ϕ. Within our implementation, a_5 is computed using a modified bisection method, outlined below. While other root finding methods are possible, this bisection method is chosen to take advantages of the properties of g. In particular, because g is a monotonic function (as increasing the intercept a_5 necessarily decreases $\overline{M}$), an initial bracketing interval can be calculated algorithmically. Taking after Dyadechko and Shashkov, this is referred to as a flooding algorithm, as it changes the volume of our reference fluid without changing its shape [4]. Note that this algorithm uses notation introduced in 6, where $\mathcal{I}(\phi, 1)$ computes the area of ϕ interior to U.

When the set of allowable interfaces are restricted to exclusively parabolic curves, such that the level set function is parameterized by (α, h, k, θ), the same flooding algorithm can be used by first computing equivalent polynomial coefficients for ϕ.

$$\begin{aligned}\phi(x,y) = \phi(\alpha, h, k, \theta) = &\left(\frac{\alpha}{4}\sin^2(\theta)\right)x^2 + \left(\frac{\alpha}{4}\cos^2(\theta)\right)y^2 - \left(\frac{\alpha}{2}\cos(\theta)\sin(\theta)\right)xy \\ &+ \left(\frac{\alpha}{2}\sin(\theta)(k\cos(\theta) - h\sin(\theta)) - \cos(\theta)\right)x \\ &- \left(\frac{\alpha}{2}\cos(\theta)(k\cos(\theta) - h\sin(\theta)) - \sin(\theta)\right)y \\ &+ \left(h\cos(\theta) + k\sin(\theta) + \frac{\alpha}{4}(k\cos(\theta) - h\sin(\theta))^2\right).\end{aligned}$$

However, after performing the flooding algorithm, it is necessary to recover the correct parameterization of the adjusted ϕ. While α and θ remain unchanged by construction, the new vertex must be obtained explicitly. This can be accomplished using geometric properties of the intercept of a parabolic level set function. In particular, adjusting the coefficient a_5 by a value of b translates the curve b units in the direction of θ. Therefore the vertex (h, k) is also shifted in the direction of θ, giving a new vertex coordinate $(b\cos\theta, b\sin\theta)$.

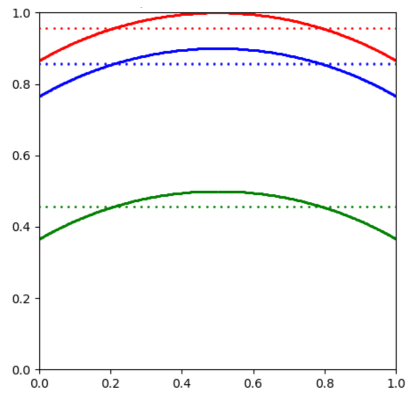
Algorithm 2 Flooding Algorithm. Given level set function coefficients $\vec{a}$, volume M, error tolerance τ, evaluate $a_5 = V(\vec{a}, M)$

```
 1: // Calculate Initial Bracketing Interval
 2: if $\mathcal{I}(\phi, 1) < M$ then
 3:     $a_5^{upper} = a_5$
 4:     while $\mathcal{I}(\phi, 1) < M$ do
 5:         $a_5 = a_5 - 1$
 6:     end while
 7:     $a_5^{lower} = a_5$
 8: else
 9:     $a_5^{lower} = a_5$
10:     while $\mathcal{I}(\phi, 1) > M$ do
11:         $a_5 = a_5 + 1$
12:     end while
13:     $a_5^{upper} = a_5$
14: end if
15: // Calculate Root with Bisection Method
16: while $|M - \mathcal{I}(\phi, 1)| < \tau$ do
17:     $a_5 = 0.5(a_5^{upper} + a_5^{lower})$
18:     if $\mathcal{I}(\phi, 1) > M$ then
19:         $a_5^{lower} = a_5$
20:     else
21:         $a_5^{upper} = a_5$
22:     end if
23: end while
```

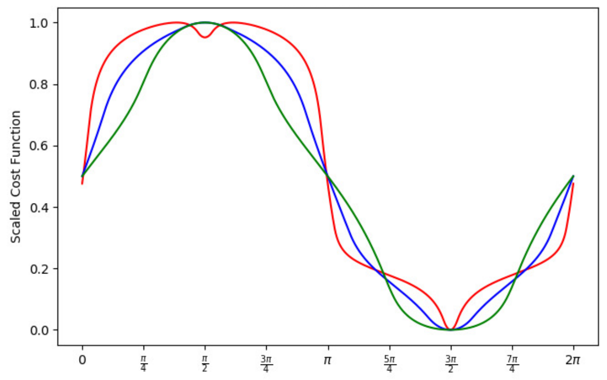
3.4 Initial Guess Methodology

We now focus entirely on a parabolic interface calculation, on where our interface is parameterized by four values. Because our cost function C is a function of four dimensions, and is defined in terms of implicitly defined integrals, many of its analytical properties are obscured. For example, we can reasonably assume C to be continuous, as small changes in the parameters result in small changes in the interface, and small changes in the interface result in small changes in the moment data. This is one of the principle benefits of a parabolic configuration of the interface, as a parameterization through the coefficients of an arbitrary second order level set functions allow the interface to shift between hyperbolas, ellipses, and even degenerate conic sections such as parallel lines.

However, because our cost function is highly non-linear, it is likely to have many local minima which would cause any optimization procedure to find suboptimal solutions. To compensate for this, we exert additional computational effort to find a good initial guess. One such method takes advantage of the relative inexpense of the standard moment-of-fluid method algorithm to find an optimal linear approximation of the interface. Because we enforce a strict volume conservation property for each computational cell, the linear interface in a cell is defined uniquely by its outward facing normal vector. This normal vector is then optimized to minimize the distance between the centroid of the fluid approximation it defines and the true, refernce centroid. Thus for a 2D, two-phase medium, optimizing each cell in MOF-PLIC reduces to solving a one dimensional optimization problem whose sole parameter θ is bracketed by $[0, 2\pi]$. In accordance with our construction of the parabolic level set function ϕ, we instead consider θ to be the direction of the inward-facing normal vector without any loss of generality. We choose a simple Golden Section search method to compute this optimal θ. However, the 1D cost function is not necessarily unimodal in θ, particularly when the volume fraction is close to 1, as shown by Figure 3.2. As a result, this method is not guaranteed to converge with an initial bracketing interval of $[0, 2\pi]$, so several other bracketing intervals are attempted (such as $[\pi/3, 7\pi/3]$) with the minimum value from each being taken as the true minimum.



(a) Optimal Linear vs. True Interface



(b) Cost vs. Angle of Linear Interface

Figure 3.2: When the volume fraction is nearly full, the 1D optimization has local minima

However, empirical trials demonstrate that in many cases the unique global minimum for MOF is by itself a poor initial guess for QMOF. Much of this can be attributed to the fact that parabolas whose parameterization has $\alpha = 0$ are not uniquely represented, as the vertex (h, k) can be shifted anywhere along the linear interface and produce identical results in C. This can lead to instability when the volume fraction is close to 1, and interfaces that "open" in the incorrect direction. Moreover, we observe that the values for h, k, and θ are much more stable than α. In particular, the value of θ often does not change substantially between the MOF and QMOF phases of the optimization algorithm presented.

As a result of these observations, a hill-climb heuristic on α immediately following the MOF optimization step is suggested and explored. This involves performing a steepest descent algorithm on α exclusively, finding the value for α given fixed values for the rest of the parameters. Practically, this has the effect of ensuring that the parabola "opens" in the correct direction, and does not get caught in local minimums due to an unstable starting position. Of course, if the interface is linear to begin with, then this hill-climbing algorithm preserves the optimality of the interface, as in this case the optimal MOF solution is also the optimal QMOF solution.

Another more precise yet less certain procedure for generating initial guesses involves using a Radial Basis Function Network (RBFN) in order to infer a reasonable parabolic curve from the moment data alone. This process is adapted from Wu, et al., [11] and uses a three-layer (6-N_n-4) network to approximate a function $F(\vec{x}) = \vec{y}$, where F is a function that maps moment data $x \in \mathbb{R}^6$ to parabola parameters $y \in \mathbb{R}^4$. In order to standardize this process across all possible computational cells, the trained RBFN used in the QMOF assumes that its input has already been mapped to the unit square domain of $[0,1]^2$. This process is performed in four primary steps. Briefly, step 1 creates training data for the network, steps 2-3 train the network, and step 4 evaluates the network to obtain the parabola parameterization $F(\vec{x})$ for a given vector of moment data.

1. **Acquire Training Data:** The parameter space for parabolas is sampled N_s times as training data. Each component of this parameter space is sampled independently according to the following distributions, where $\text{Norm}(\mu, \sigma)$ is a normal distribution with mean μ and standard deviation σ, and $\text{Unif}(a, b)$ is a uniform distribution on the interval (a, b):

   - $\alpha \in 5\text{Norm}(0, 1)$
   - $h, k \in \text{Unif}(0, 1)$
   - $\theta \in \text{Unif}(0, 2\pi)$

   These distributions are chosen heuristically to best represent the subset of parabolas whose interfaces are commonly seen in practical problems. For example, the vast majority of optimal parabolas have vertices interior to the cell. Moreover, with sufficiently high resolution, there are many more interfaces that are nearly straight than sharply curved, reflected in the choice of a normal distribution from which to select α.

   Additionally, if a parabola that leaves the computational cell either entirely full or entirely empty is generated, it is rejected and another candidate is computed. The parameters for these parabolas are stored computationally in a $4 \times N_s$ matrix $\mathbf{Y}$. Then for each parabola generated, its vector of moment data is calculated and stored in a $6 \times N_s$ matrix $\mathbf{X}$.

2. **Compute Centers**: The first stage of training the RBFN is an unsupervised learning step that clusters the data $\mathbf{X}$ into N_n clusters. These clusters make up the centers of the radial basis functions, which themselves are the nodes in the network's single hidden layer. This clustering is necessary for good performance of the RBFN. Although the parabolas themselves are generated randomly, there is no guarantee that the moment data that results is organized in any meaningful way, which is instead accomplished by this clustering step. This clustering step is performed with a standard k-means clustering algorithm, described below:

Algorithm 3 k-means Clustering Algorithm. Given N_s samples produce N_n centers that best represent the data.

```
1: Select $\vec{c}_0, \vec{c}_1, \ldots, \vec{c}_{N_n}$ randomly out of the $N_s$ samples to serve as initial clusters.
2: energy $\leftarrow \infty$
3: **while** Error tolerance not reached **do**
4:     Assign each sample $\vec{x}_i$ to the closest center $c_j$ according to a Euclidean norm.
5:     Replace center $\vec{c}_j$ with centroid of all $\vec{x}_i$ assigned to that center $\vec{c}_j$
6:     energy $\leftarrow \sum_{j=0}^{N_n} \sum_{\vec{x} \in \text{Clus}(\vec{c}_j)} ||\vec{x}_i - \vec{c}_j||^2$ where Clus($\vec{c}_j$) is the set of all samples assigned
       to center $\vec{c}_j$.
7: **end while**
```

In practice, as few as between 3 and 5 iterations of the above algorithm are needed to dramatically decrease the energy of the clustering.

3. **Compute Weights**: One of the principle benefits of an RBFN, and the primary reason for its inclusion here, is that its supervised training stage involves only the solution of a linear system. For a single input $\vec{x} \in \mathbb{R}^6$, each component output of the RBF network is given by

$$y_i(\vec{x}) = \sum_{k=1}^{N_s} w_{ki} \psi(||\vec{x} - \vec{c}_k||), \quad \text{for } i = 1, \ldots, 4$$

where $\psi(r) = (\sigma^2 + r^2)^{\frac{1}{2}}$ is one particular radial basis function suggested by Wu, et al. Here, this is chosen for being cheaper to compute than the standard Gaussian radial basis function $e^{-r^2/2\sigma^2}$ while producing similar, if not slightly more optimal results. In either case, σ represents a smoothing parameter, typically set to $\sigma = 1$.

Importantly, when applied to each of the N_s sample points, this equation can be expressed in the matrix form

$$\mathbf{Y} = \mathbf{W}^T \mathbf{\Psi}$$

where $\mathbf{W}$ is a $N_n \times 4$ matrix of weights, $\mathbf{\Psi}$ is a $N_n \times N_s$ matrix of outputs of the hidden layer acting on the sample data, and $\mathbf{Y}$ is a $4 \times N_s$ matrix of outputs for the sample data. In this way, optimal weights for the RBFN can be computed so that it best approximates F for each of the samples and ideally interpolate between them. This is because this matrix of weights $\mathbf{W}$ minimizes the mean squared error function

$$\frac{1}{N_s} \sum_{p=1}^{N_s} ||\vec{y}_p - \mathbf{W}^T \vec{\psi}_p||^2,$$

where $\vec{\psi}_p$ is the output of the hidden layer for the p^{th} sample [11].

We note here that because we have transformed any possible domain for our moment data into $[0,1]^2$, the training data, centers, and weights can each be precomputed. Therefore, no attempt is made to make the solution to this system computationally efficient, as the centers and weights can be stored between evaluations of the RBFN. In this case, the weights are found directly through a Gaussian-elimination solution of the equation

$$\mathbf{W} = (\mathbf{\Psi}\mathbf{\Psi}^T)^{-1}\mathbf{\Psi}\mathbf{Y}^T$$

4. **Evaluating the RBF Network**: In this step, we simply load the matrices of weights and centers from memory, and evaluate

$$y_i(\vec{x}) = \sum_{k=1}^{N_s} w_{ki}\psi(||\vec{x} - \vec{c}_k||), \quad \text{for}\, i = 1, \ldots, 4$$

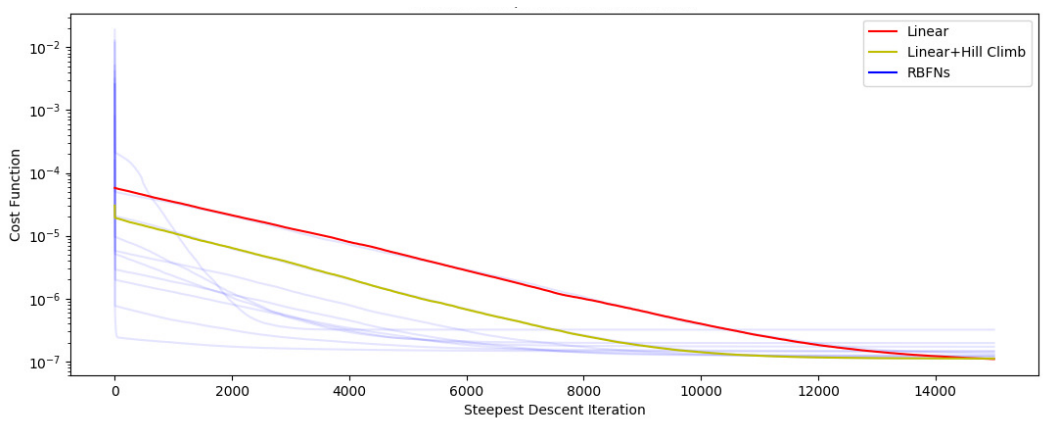
The computational inefficiency of loading these matrices from memory is offset by needing to perform this step only once for each execution of the QMOF algorithm. In general, the cost of the QMOF algorithm is dominated by the steepest descent portions, and any effort spent in finding a reasonable initial guess is necessarily offset by requiring fewer iterations of steepest descent.

We now informally compare each of the three initial condition techniques introduced: MOF-PLIC, MOF-PLIC with Hill Climbing, and a trained RBFN. The following figures demonstrate for a single interface the relative performance of the three techniques, using 10 independently trained RBFNs to account for their inherent stochastic nature. These results are presented qualitatively in Figure 3.3. Categorically, this probabilistic character strongly distinguishes the two methods. For example, the training phases as introduced and used herein are determined completely by the structure of the randomly generated set of input data to the RBFN. While this randomness can be mitigated in some sense by properly managing and clustering the data, it is present nonetheless in the convergence properties of the steepest descent portions of the algorithm. In particular, because the network is trained according to the closeness of parameters, a parabola with "near" to those of the optimal parabola may have quite different moment data. This is reflected in the figures below, where initial curves appear substantially far from any optimal structure, but in fact converge rapidly to a solution with optimal moment data. And while unlikely, it is entirely possible that such an initial guess can over time settle on a wildly incorrect local minimum. Most commonly, these minima "open" in the incorrect direction when the network is applied to nearly linear interfaces.

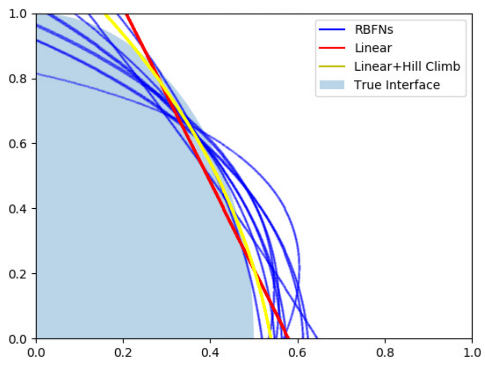
We also observe the effects of the hill-climbing algorithm when applied to the optimal linear solution to the optimization problem. In particular, we observe that this additional step does not necessarily increase the rate of convergence of the steepest descent. However, the cost of the solution obtained with hill climbing is substantially lower than the optimal linear solution. This effect becomes even more pronounced when applied to interfaces with very severe curvature.

3.5 Steepest Descent

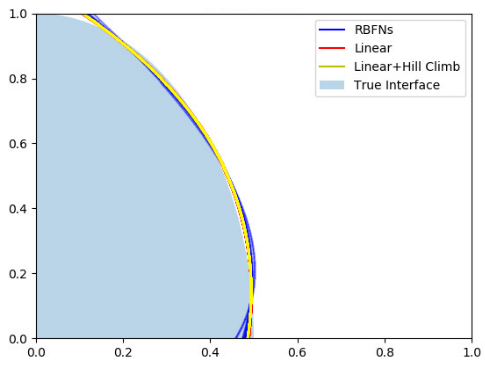
The primary method of optimization used in this implementation of QMOF is a steepest descent algorithm with a backtracking line search, provided below. While slow among algorithmic optimization procedures, steepest descent was chosen as a demonstration that such a global minima exists within some convex neighborhood. In a practical implementation of the QMOF algorithm, a more efficient algorithm like Newton's Method or BFGS would be used in its place. Note that this approach to optimization functions identically for different parameterization of the level set functions. In particular, only the size of the input vector $\vec{a}$ changes when the level set function is parameterized by its coefficients as opposed to the parabolic representation discussed thus far.



(a) Cost vs. Steepest Descent Iteration



(b) Initial Curves



(c) Final Curves

Figure 3.3: Initial Condition vs. Final Curve found through steepest descent

Algorithm 4 Steepest Descent with Backtracking Line Search. Given initial level set function parameters $\vec{a}_0$, cost function $C(\vec{a}; \mathbf{M})$, initial step size γ_0 and backtracking parameter $\beta \in (0,1)$ compute optimal level set parameters $\vec{a}$

```
1: $\vec{a} \leftarrow \vec{a}_0$
2: $\gamma \leftarrow \gamma_0$
3: while $||\nabla C|| >$ error tolerance do
4:     while $C(\vec{a} - \gamma \nabla C(\vec{a})) \geq C(\vec{a})$ do
5:         $\gamma \leftarrow \beta \cdot \gamma$
6:     end while
7:     $\vec{a} \leftarrow \vec{a} - \gamma \nabla C(\vec{a})$
8:     $\gamma \leftarrow \gamma_0$
9: end while
```

Additionally, we must compute the gradient of C in order to determine a search direction.

As a result of the implied continuity of C, we use a simple 5-point finite difference scheme in order to compute the necessary partial derivatives. Such a high order stencil is chosen compared to, say, a 3-point centered difference formula, in order to enforce a wider stencil with a high degree of accuracy, as the sign of this gradient calculation is in many ways more significant than the value itself.

$$f'(x) \approx \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h}$$

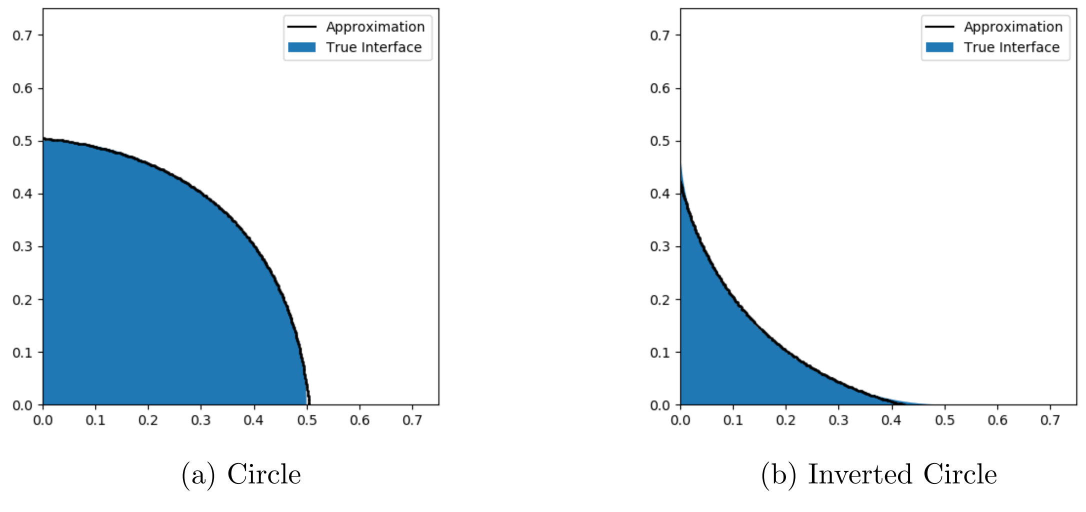
Applied to the partial derivatives of C, we have the following formula for the partial derivative, where $C(a_j)$ is shorthand for $C(a_0, \ldots, a_j, \ldots, a_4)$:

$$\frac{\partial C(\vec{a})}{\partial a_j} \approx \frac{-C(a_j+2h) + 8C(a_j+h) - 8C(a_j-h) + C(a_j-2h)}{12h}$$

Chapter 4

Numerical Results

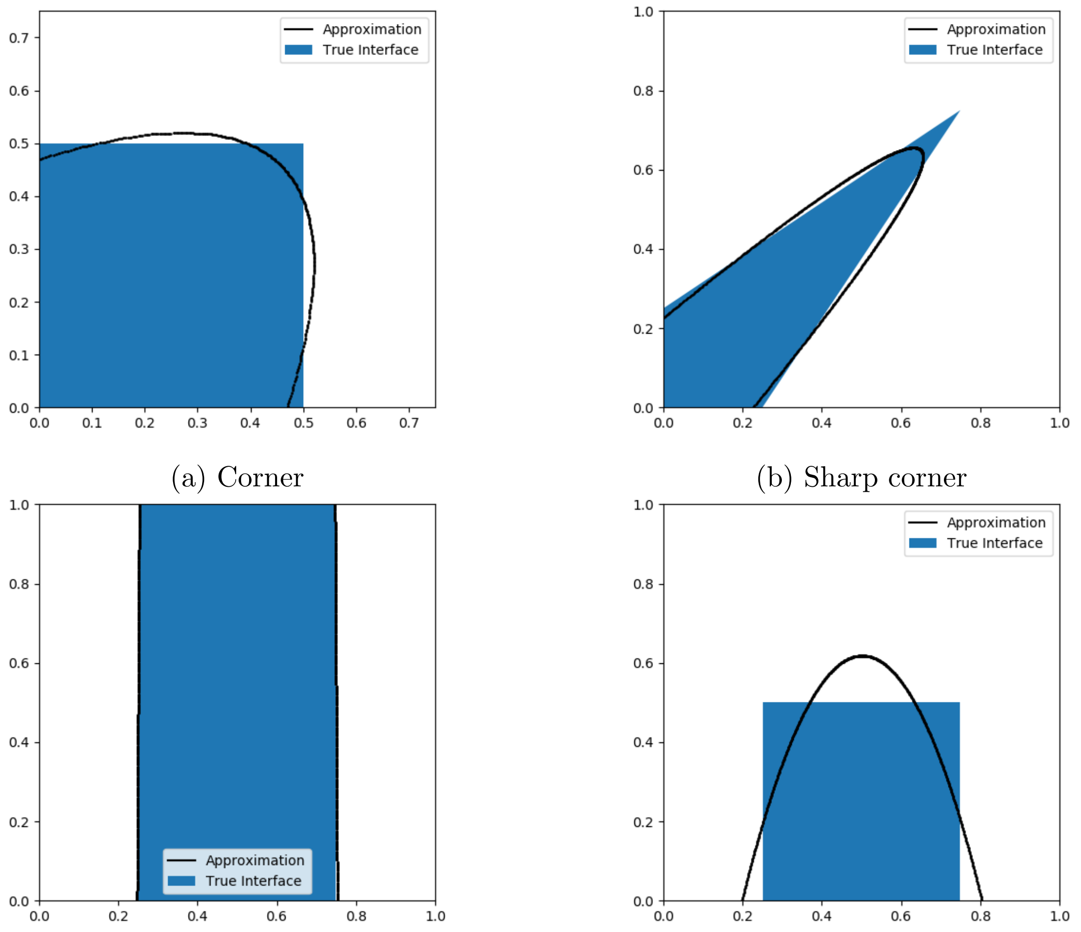
For fluids with perfectly linear interfaces, the quadratic Moment-of-fluid method produces an exact reconstruction, just as in the linear MOF case. The MOF-PPIC method can also locate exact reconstruction of parabolic interfaces, as expected. However, although the majority of curved interfaces are not strictly parabolic, they can still be reasonably approximated through a second order level set function, as seen in Figure 4.1. In each of these cases, analogous results can be produced if the fluid is "inverted," in the sense that its exterior and interior sections are swapped. In general, however, we see that the parabolic interface is able to easily capture simple curved geometries common in practical problems.



(a) Circle (b) Inverted Circle

Figure 4.1: Parabolic MOF on simple curves

Moreover, the MOF-PPIC method can effectively capture corners and filamentary regions for which a linear approximation provides a poor reconstruction, as seen in Figure 4.2. It must be noted, however, that such interfaces represent the most numerically difficult cases, as they require large values for α. This makes the reconstruction much more sensitive to noise in the moment data, and more generally take much longer to converge. This time can be often offset when a hill-climbing step is applied, as it can accelerate the process of reaching a local neighborhood for the optimal value of α.



(a) Corner (b) Sharp corner

(c) Filament (d) Filament End

Figure 4.2: Parabolic MOF on difficult interfaces

4.1 Grid Refinement Tests

For these tests, we use the symmetric difference error between the reconstruction and true interface, summed across each computational cell U_i, as the primary measure of effectiveness of an approximation.

$$E_{\text{sym}} = \sum_{U_i} \int_{\Omega \oplus \overline{\Omega}} dx dy \quad \text{where} \quad \Omega \oplus \overline{\Omega} = \left\{ x | x \in (\Omega - \overline{\Omega}) \cup (\overline{\Omega} - \Omega) \right\}$$

When the true interface can be represented as the zero level set of a second order function, as is the case for a circular interface, this integral can be computed more or less exactly using the same quadrature method outlined in section 3.2. For interfaces that cannot be represented in such a way, a high resolution Monte-Carlo integration is performed instead. The clear qualitative benefits of a parabolic interface reconstruction, in which the parabolic curves more closely resemble the true interface, are reflected in the dramatic decrease in symmetric difference error, as seen in Figure 4.4. In the circular case, the rate of convergence as a function of the refinement of the grid is identical, shown by Figure 4.3. However, a given parabolic interface approximation produces errors comparable to a linear approximation on a four times as refined grid.

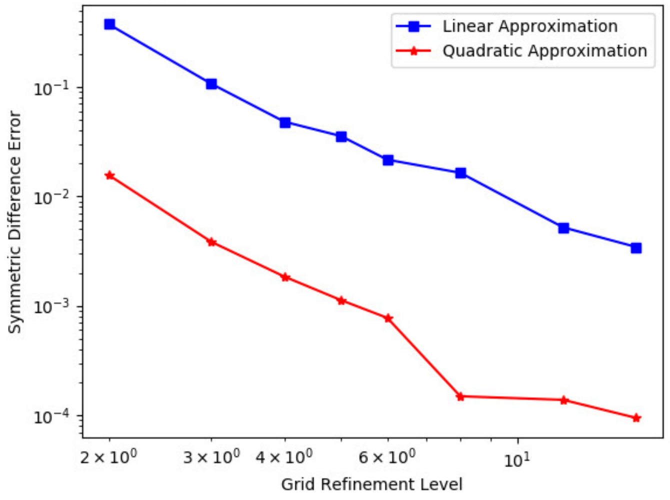
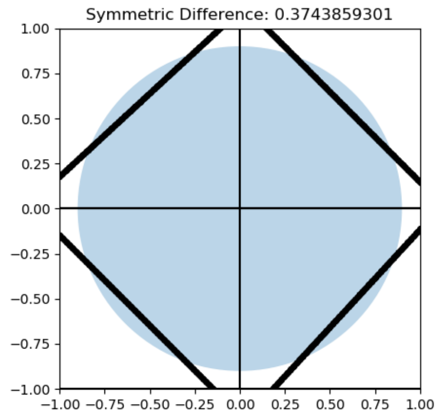
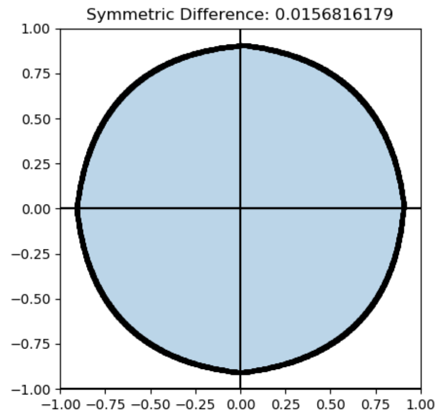


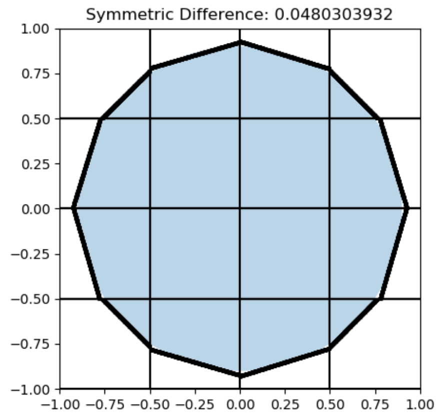
Figure 4.3: Symmetric Difference Error vs. Grid Refinement



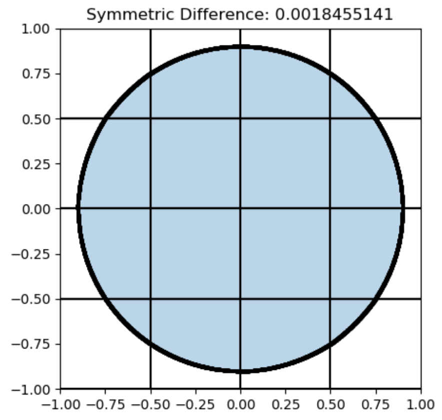
(a) 2×2 MOF-PLIC



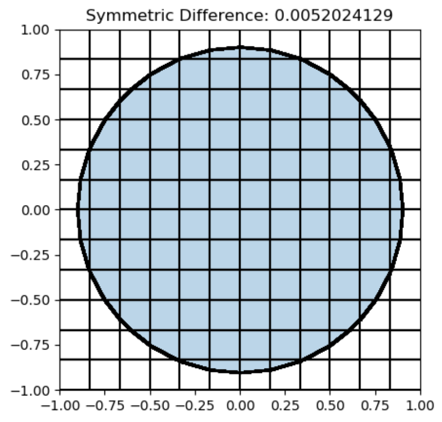
(b) 2×2 QMOF-PPIC



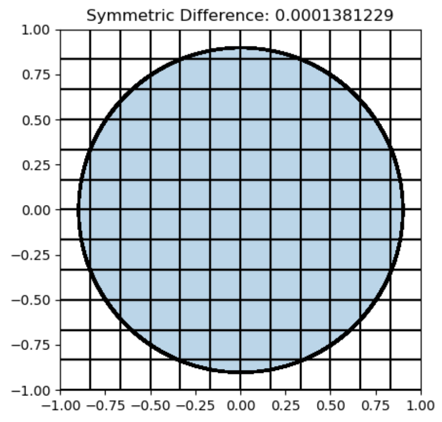
(c) 4×4 MOF-PLIC



(d) 4×4 QMOF-PPIC



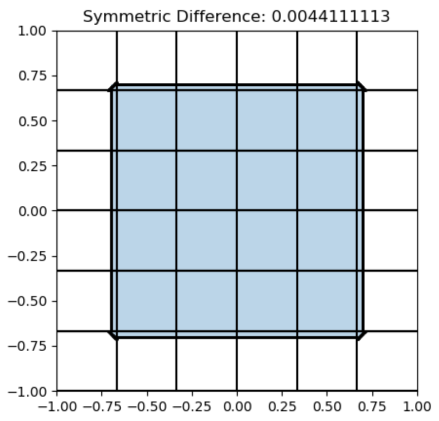
(e) 12×12 MOF-PLIC



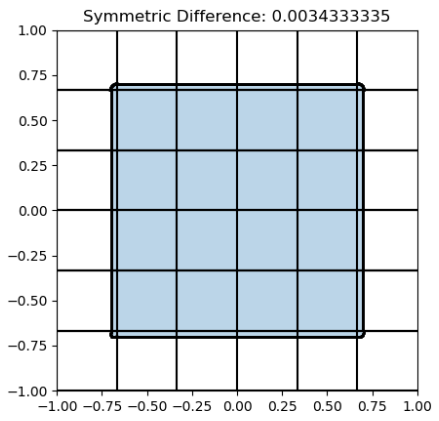
(f) 12×12 QMOF-PPIC

Figure 4.4: Effectiveness of PLIC vs PPIC under grid refinement

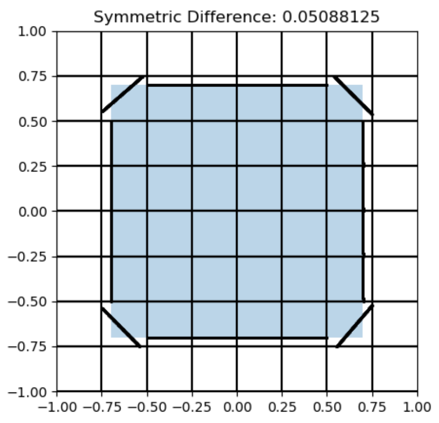
One of the difficulties in any interface reconstruction scheme is that its effectiveness depends heavily on the structure of the grid in relation to the topology of the true interface. As a trivial example, square interfaces whose vertices align exactly with a square grid can be approximated exactly, regardless of the actual refinement of the grid. In some respects, this represents one of the unique advantages of the QMOF-PPIC scheme, as corners can be reasonably approximated regardless of their closeness to cell boundaries. However, this makes an objective comparison with MOF-PLIC schemes difficult in this case. For example, Figure 4.5 presents a situation in which the quadratic approximation results in a better approximation of the interface regardless of the refinement level, but the comparative benefits vary. Here, a 6×6 grid actually produces less significant benefits over an 8×8 grid due to the closeness of the corners to boundaries of the computational cells.



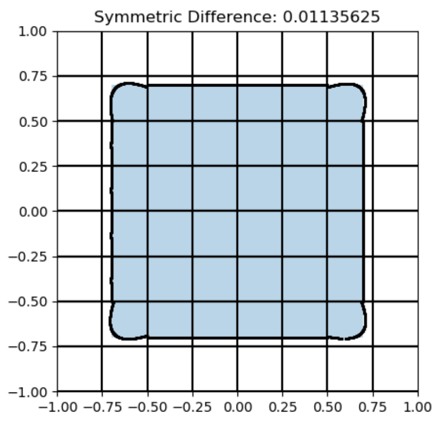
(a) 6×6 MOF-PLIC



(b) 6×6 QMOF-PPIC



(c) 8×8 MOF-PLIC



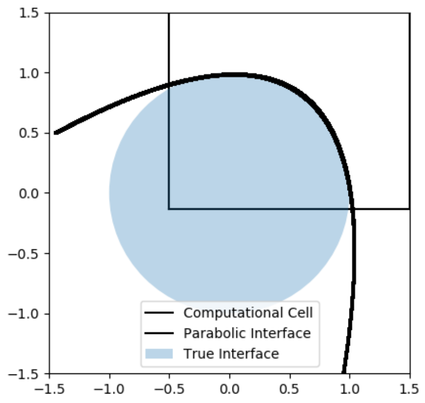
(d) 8×8 QMOF-PPIC

Figure 4.5: PLIC vs PPIC Applied to Square Interface

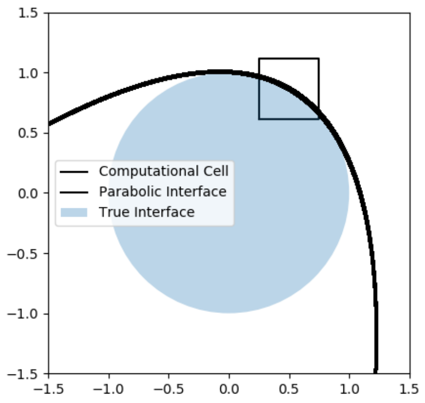
4.2 Curvature Tests

We now explore the ability of a QMOF-PPIC to approximate curvature. Such calculations are important for a variety of applications, most notably the study of fluids with significant surface tension effects. This is one of the most significant advantages to a parabolic reconstruction as opposed to one that is only linear, as linear approximations by construction all have curvature zero. This means that any curvature approximation using a PLIC cannot have any significant locality properties. By comparison, the QMOF-PPIC presented allows for an approximation of curvature that comes directly from the interface within a cell.

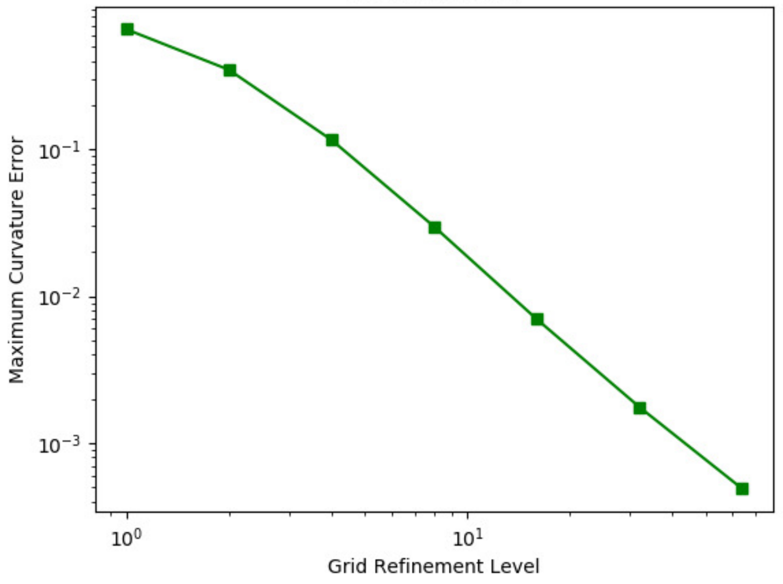
The following tests demonstrate that increasingly refined grids produce increasingly accurate curvature approximations. This was accomplished by shrinking the computational cell for which the parabolic approximation is obtained and computing the maximum error in curvature. We are able to compute the maximum error by computing the curvature at the vertex and each of the two intersections with the cell. The greatest error among these three points is therefore the greatest error within the entire cell, as the curvature for a parabola has a single locally maximal value at its vertex.



(a) Curvature test with refinement level 1



(b) Curvature test with refinement level 4



(c) Maximum curvature error vs. Grid refinement level

Figure 4.6: Accuracy of Curvature Calculation with PPIC

Chapter 5

Conclusion

This paper presents a novel method of Piecewise Parabolic Interface Calculation that uses time advanced moment data in a manner consistent with previous efforts in Moment-of-fluid methods.

This method has clear advantages over standard Moment-of-fluid methods that produce a linear interface reconstruction. In all cases, a second order interface can more closely approximate the reference fluid in terms of the symmetric difference error, even when the set of allowable curves are restricted to parabolas. Moreover a parabolic representation allows the reconstruction to capture certain sub-grid details, which allows for coarser grids to be used in the reconstruction. Certain interface topologies that are impossible to accurately capture with a linear reconstruction can also be modeled accurately, in particular filamentatary regions. The use of a second order reconstruction also allows for an approximation of curvature within the cell, which is impossible to do locally for a PLIC method.

These locality properties represent the primary benefits of the QMOF-PPIC methods over other second order interface techniques. This is possible because the necessary moment data can be calculated and maintained for each cell. This locality also has algorithmic and computational benefits as well. Just as in the original moment-of-fluid literature [4], the QMOF method can be applied as a black-box algorithm to each cell in a discretization in a manner that is largely independent of the geometry of the cell.

Chapter 6

Future Work

This paper presents an algorithmic framework which can be readily expanded in future research, expansions which fall into one of three main categories.

The first involves the formulation of dynamic tests for the quadratic Moment-of-fluid method. This would be implemented in much the same way as described in section 2.2, reflecting the techniques described in the original MOF paper [4]. While such time advancement of moment data is in principle possible, it remains to be seen how effective the PPIC method is when the interface is under the influence of a velocity field.

The second involves the Radial Basis Function Network described in this paper. One of its primary downsides is that the stochastic nature of its construction often leads to misinformed predictions of interface geometry. This is most likely caused by the fact that the network is trained only according to the ”closeness” of the vectors that represent the parameters of the parabolas. While this could be solved by training the data according to a metric that takes into consideration the symmetric difference error, such an adjustment would dramatically increase the cost of the training phase, as it can no longer be solved with a simple linear system. Regardless, the potential of the RBFN to dramatically decrease convergence time and effectively eliminate the possibility of settling at a local minimum greatly motivates further study.

Finally, several simple algorithmic improvements to the optimization phases of the method could dramatically improve the interface reconstruction as a whole. Within the discussed implementation, the primary method of optimization is a simple steepest descent algorithm, which is slow among other high-dimensional optimization schemes. Therefore it is likely that there exist other numerical optimization schemes that would bring significant computational benefits. In particular, there exist a variety of gradient-free optimization methods that can take advantage of the fact that several of the level set parameters (namely θ, h, and k) are all bounded for the majority of problems.

Bibliography

[1] Hyung Taek Ahn and Mikhail Shashkov. Adaptive moment-of-fluid method for multi-material flow. In Haecheon Choi, Hyong Gwon Choi, and Jung Yul Yoo, editors, *Computational Fluid Dynamics 2008*, pages 567–572, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[2] Alexandre Joel Chorin. Curvature and solidification. 1983.

[3] SV Diwakar, Sarit K Das, and T Sundararajan. A quadratic spline based interface (quasi) reconstruction algorithm for accurate tracking of two-phase flows. *Journal of Computational Physics*, 228(24):9107–9130, 2009.

[4] Vadim Dyadechko and Mikhail Sashkov. Moment-of-fluid interface reconstruction. *Los Alamos Report LA-UR-05-7571*, 2005.

[5] James Glimm, John W Grove, XL Li, and N Zhao. Simple front tracking. *Contemporary Mathematics*, 238(2):133–149, 1999.

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

[7] Glenn Robert Price. *A piecewise parabolic volume tracking method for the numerical simulation of interfacial flows.* Calgary, 2000.

[8] Yinghe Qi, Jiacai Lu, Ruben Scardovelli, Stphane Zaleski, and Gretar Tryggvason. Computing curvature for volume of fluid methods using machine learning. 07 2018.

[9] Yuriko Renardy and Michael Renardy. Prost: a parabolic reconstruction of surface tension for the volume-of-fluid method. *Journal of computational physics*, 183(2):400–421, 2002.

[10] RI Saye. High-order quadrature methods for implicitly defined surfaces and volumes in hyperrectangles. *SIAM Journal on Scientific Computing*, 37(2):A993–A1019, 2015.

[11] Yue Wu, Hui Wang, Biaobiao Zhang, and K-L Du. Using radial basis function networks for function approximation and classification. *ISRN Applied Mathematics*, 2012, 2012.

[12] David Youngs. *Time-Dependent Multi-material Flow with Large Fluid Distortion*, volume 24, pages 273–285. 01 1982.

Appendix A

Quadrature Algorithm Implementation Details

This algorithm was initially outlined by Saye [10] with the specific implementation used within this paper taking advantage of the presupposition that the level set functions ψ are at most quadratic in degree. Moreover, we are interested in only computing integrals across volumes, rather than surfaces. As a result of these changes, the algorithm can be simplified in many ways, and is substantially cheaper to execute than the most general case.

This algorithm was implemented in object-oriented C++ in order to properly maintain level set functions ψ_i in multiple dimensions. The library is built around the class `Phi`, which contains the coefficients of the base second order level set function, methods to find roots along a specific interval using the quadratic formula, and methods to evaluate the function and its partial derivatives. Although a specific derived class, `Phi2D`, is used for the above implementation, similar derived classes for higher order level set functions are simple to add. Through the addition of such classes, the quadrature algorithm as presented can be extended into arbitrarily high dimensions.

As suggested by Saye in the discussion of the original numerical implementation, the library generates height functions by fixing coordinate directions with a specific vallue when constructing the functions ψ_i^L and ψ_i^U. This is accomplished by using objects `Psi`, which inherit properties of the original `Phi` class, that also maintain a list of fixed directions and values. When such a function is evaluated, these directions and values are fed back into the base function in the reverse order of their addition to reconstruct the original base function with the proper parameters.

Other notable software features are a `Box` class that represents a single domain of integration, a lightweight `Point` class that maintains coordinates of up to three dimensions without needing dynamically allocated memory, and a class `F_params` that facilitates the necessary recursive function calls. These features are all implemented in a robust software library that uses minimal outside libraries beyond standard C++, using only the `vector` STL class.

Within the original work and the algorithm's reproduction here, take $\vec{x} + ye_j$ as an alternate notation for the vector $[x_0, \dots, x_{j-1}, y, x_j, \dots, x_n]^T$. Additionally, let and $x_{q,i}$ be the i^{th} of q Gaussian quadrature nodes with corresponding weight $w_{q,i}$.

Algorithm 5 Integrand evaluation for quadrature. Given a point $\vec{x} \in \mathbb{R}^d$, a function $f : \mathbb{R}^d \to \mathbb{R}$, level set functions $\psi_i : \mathbb{R}^d \to \mathbb{R}$ and sign conditions s_i, $i = 1, \dots, n$, an interval (x_1, x_2), a height direction k, and the order of quadrature q, evaluate $\mathcal{F}(\vec{x}; f, \{\psi_i, s_i\}_{i=1}^n, (x_1, x_2), k, q)$

```
1: Calculate the set of roots and union with the interval endpoints:
       $$\mathcal{R} = \{x_1, x_2\} \cup \bigcup_{i=1}^{n} \{\text{roots of } g(y) = \psi_i(\vec{x} + ye_k) \text{ on the interval } (x_1, x_2)\}$$
2: Sort $\mathcal{R}$ into ascending order such that $r_1 \leq \cdots \leq r_\ell$.
3: Set $I = 0$.
4: for $j = 1$ to $\ell - 1$ do
5:     Define $L = r_{j+1} - r_j$ and $\vec{x}_c = \vec{x} + 0.5(r_j + r_{j+1})e_k$
6:     if $s_i\psi_i(\vec{x}_c) \geq 0$ for $\forall i$ then
7:         Update $I := I + L\sum_{i=1}^{q} w_{q,i} f(\vec{x} + (r_j + Lx_{q,i})e_k)$.
8:     end if
9: end for
10: return $I$
```

Algorithm 6 Quadrature algorithm on implicitly defined volume. Given integrand $f : \mathbb{R}^d \to \mathbb{R}$, level set functions $\psi_i : \mathbb{R}^d \to \mathbb{R}$ and sign conditions s_i, $i = 1, \ldots, n$, a hyperrectangular domain $U = (x_i^L, x_i^U) \times \cdots \times (x_d^L, x_d^U)$, and the order of quadrature q, approximate $\int_\Omega f, d\vec{x} = \mathcal{I}(f; \{\psi_i, s_i\}_{i=1}^n, U, q)$

```
1: if d = 1 then
2:     return $\mathcal{F}(0; f, \{\psi_i, s_i\}_{i=1}^n, (x_1^L, x_1^U), 0, q)$, where 0 is a zero-dimensional vector.
3: end if
4: Define $\vec{x}_c \in \mathbb{R}^d$ as the center of $U$, so that $x_{ci} = 0.5(x_i^L + x_i^U)$.
5: for $i = n$ to $i = 1$ do
6:     Evaluate bounds on the value of $\psi_i$ on $U$ such that $\sup_{\vec{x} \in U} |\psi_i(\vec{x}) - \psi_i(\vec{x}_c)| \leq \delta$
7:     if $|\psi_i(\vec{x}_c)| \geq \delta$ then
8:         Remove $\psi_i$ from the list and decrement $n$ by one.
9:     else
10:        return 0 {The domain of integration is zero}
11:    end if
12: end for
13: if $n = 0$ then
14:     The domain of integration is "full," apply a tensor-product Gaussian quadrature
        scheme:
15:     return
        $$I = |U| \sum_{i_1,\ldots,i_d=1}^{q} w_{q,i_1} \cdots w_{q,i_d} f\left(\sum_{j=1}^{d} (x_j^L + (x_j^U - x_j^L) x_{q,i_j}) e_j\right)$$
16: end if
17: $k = \arg\max_j |\partial_{x_j} \psi_1(\vec{x}_c)|$ and initialize $\tilde{\psi} = \emptyset$.
18: for $i = 0$ to $n$ do
19:     Evaluate $g := \nabla \psi_i(\vec{x}_c)$
20:     Determine bounds $\delta \in \mathbb{R}$ such that $\sup_{\vec{x} \in U} |\partial_{x_j} \psi(\vec{x}) - g_j| \leq \delta$ for each $j$.
21:     if $|g_k| \geq \delta$ and $(\sum_{j=1}^{d} (g_j + \delta_j)^2 / (g_k - \delta_k)^2 < 20$ then
22:         Define $\psi_i^L : \mathbb{R}^{d-1} \to \mathbb{R}$ and $\psi_i^U : \mathbb{R}^{d-1} \to \mathbb{R}$ by
            $$\psi_i^L(\vec{x}) := \psi_i(\vec{x} + x_k^L e_k) \quad \psi_i^U(\vec{x}) := \psi_i(\vec{x} + x_k^U e_k).$$
23:         Evaluate signs: $s_i^L = \mathrm{sgn}_L(\mathrm{sign}(g_k), s_i)$, $s_i^U = \mathrm{sgn}_U(\mathrm{sign}(g_k), s_i)$.
24:         Add to the collection: $\tilde{\psi} := \tilde{\psi} \cup \{(\psi_i^L, s_i^L), (\psi_i^U, s_i^U)\}$.
25:     else
26:         Split $U$ into halves along its largest extent such that $U = U_1 \cup U_2$. {No suitable
            height direction}
27:         return $\mathcal{I}(f; \{\psi_i, s_i\}_{i=1}^n, U_1, q) + \mathcal{I}(f; \{\psi_i, s_i\}_{i=1}^n, U_2, q)$
28:     end if
29: end for
30: Define a new integrand $\tilde{f} : \mathbb{R}^{d-1} \to \mathbb{R}$ by
        $$\tilde{f} := \mathcal{F}(\vec{x}; f, \{\psi_i, s_i\}_{i=1}^n, (x_k^L, x_k^U), k, q)$$
31: return $\mathcal{I}(\tilde{f}; \tilde{\psi}, (x_1^L, x_1^U) \times \cdots \times (x_{k-1}^L, x_{k-1}^U) \times (x_{k+1}^L, x_{k+1}^U) \times \cdots \times (x_d^L, x_d^U), q)$
```