INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

**A PIECEWISE LINEAR INTERFACE TRACKING METHOD FOR
3-D EULERIAN INCOMPRESSIBLE FLUID FLOWS**

by

Vikram Digambar Kulkarni

B.E. Mechanical, University of Poona

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Engineering

Department of Mechanical and Aerospace Engineering
Ottawa - Carleton Institute for Mechanical and Aerospace Engineering

Carleton University

Ottawa, Canada

Dec, 1998

© Copyright, 1998

Vikram Digambar Kulkarni

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-36910-2

The undersigned recommend to the Faculty of Graduate Studies
and Research acceptance of the thesis

**“A Piecewise Linear Interface Tracking Method for
3-D Eulerian Incompressible Fluid Flows.”**

submitted by

Vikram Digambar Kulkarni

in partial fulfilment of the requirements for

the degree of

Master of Engineering

A handwritten signature in black ink, appearing to read 'J. Goldak', is written over a horizontal line.

Thesis Supervisor

A handwritten signature in black ink, appearing to read 'P. Desmully', is written over a horizontal line.

Chair, Department of Mechanical and Aerospace Engineering

Carleton University

Abstract

An algorithm for tracking liquid-air interfaces in three dimensions has been developed. The method uses local discrete material volume and velocity data to track interfaces that can have complex topology. In an individual element the interface is represented by the equation of a plane. Material volume fluxes are systematically computed by a set of well defined geometric algorithms. Mass is conserved exactly to precision. The method is interrogated by testing its ability to track interfaces in simple analytic velocity fields and more complex flows with high vorticity. The method has been applied to study the flow of liquid metal in a metallic die to simulate the filling process in die casting. The interface is visualized by a rigorous algorithm. Numerical results for these strenuous test problems and a large scale analysis provide evidence that the algorithm is a robust and reliable design with competitive performance.

To Ajoba, my grandfather,

he made us fearless men.

Acknowledgments

Dr. John A. Goldak is my *guru*. He taught me how to deal with situations involving high uncertainty by becoming proactive. His passion for explaining complex ideas in physics and mathematics to anyone, may he be a fellow scientist or a person on the road and *making sure* that the recipient understood the idea, makes me rich today. I thank him for having worked shoulder to shoulder with me on several occasions. I would like to thank Dr. Frederic Montoya for setting up the “MRCOShell”, a computing environment that was the backbone of this thesis. I thank Ms. Andrea Zypchen for her kindness and invaluable support in introducing me to the fascinating world of scientific computing. My best wishes to her children Emma, Samantha and Nicholas. I thank Ms. Mihaela Mocanita for providing me insight on computational weld mechanics. Mr. Daniel Downey taught me the art of visualization, knock! knock! thanks Dan. My father, Major D. V. Kulkarni and mother, Mrs. Chitra Kulkarni gave me the vision to look ahead and the power to distinguish right from wrong, thanks Ma, thanks Dad. I wish to thank H. E. Vannatter Limited, designers and manufacturers of die casting dies, for letting me use their “Transmission Case” for my thesis.

Contents

Acceptance Sheet	ii
Abstract	iii
Acknowledgements	v
Table of Contents	vi
List of Figures	xii
List of Tables	xvii
Nomenclature	xix
1 Introduction	1
1.1 Organization of Thesis	3
2 Literature Review	4
2.0.1 Developments in Interface Tracking Methods	7

3	Algorithm Summary	12
4	Interface Slope	16
4.1	VOF Values in a 3x3x3 Stencil	17
4.1.1	Transformation Maps for an Advecting Face	17
4.1.2	Use of a Rotation Map	21
4.2	Computing the Gradient of VOF ∇f	21
4.3	Given ∇f Compute the Interface Normal	25
5	Interface Distance	26
5.1	Co-ordinate System for Interface Reconstruction	28
5.1.1	Transform from $(\hat{x}, \hat{y}, \hat{z})$ Space to (x_1, x_2, x_3) Space	31
5.1.2	Reverse the Interface Normal $\bar{n}$	37
5.2	Components of $\bar{n}$ in x_1, x_2 and x_3 Space	39
5.3	Interface Intercepts h_1, h_2, h_3 in x_1, x_2, x_3 Space	40
5.4	Interface Polygon Type	40
5.4.1	Volume Of a Tetrahedron	42
5.5	The Triangular Interface Polygon	45
5.5.1	Interface Constraint Triangular Interface Polygon . . .	46
5.6	The Quadrilateral Type A Interface Polygon	48

5.6.1	Interface Constraint Quadrilateral Type A	51
5.7	The Pentagonal Interface Polygon	53
5.7.1	Interface Constraint for a Pentagonal Interface Poly- gon	55
5.8	The Hexagonal Interface Polygon	57
5.8.1	Interface Constraint for a Hexagonal Interface Polygon	59
5.9	The Quadrilateral Type B Interface Polygon	61
5.9.1	Interface Constraint for a Quadrilateral Type B Inter- face Polygon	63
5.10	Newton-Raphson Root Finding Algorithm	66
5.10.1	Implementation of Newton-Raphson	69
6	Flux of Liquid to Advect	71
6.0.2	Summary of Volume Integration with 3 Node Triangles	73
6.1	Determination of Exit Face	75
6.1.1	Implementation Strategy for Exit Face	77
6.2	Volume Integration	80
6.2.1	Equation of the Planar Interface	80
6.2.2	Edges of the Unit Cell and the Slice Box	82

6.2.3	Bounding Box of the Slice Box	84
6.2.4	Actual Slice Box and the Complement Slice Box . . .	87
6.2.5	Intersections with the Slice Box	91
6.2.6	Polygon Assembly	94
6.2.7	Slice Box Polygon Tessellation	97
6.2.8	Volume of Material Under a Three Node Triangle . . .	99
6.2.9	Volume Of Material In the Slice Box	100
6.2.10	Managing the Little Triangle	101
6.2.11	Volume of Liquid in the Slice Box	104
7	Law of Conservation of Mass	108
7.1	Computational Domain for Interface Advection	111
7.2	Time Integration	114
7.2.1	Mathematical Statement of the Operator Splitting Time Integration Algorithm	121
8	Interface Graphics	122
8.0.2	Equation of the Interface in Global Co-ordinates . . .	124
8.0.3	Interface Triangulation	128

9	Initializing VOF in a Sphere	129
9.1	Summary of the Analytic Spherical Interface Algorithm . . .	130
9.2	Identify Elements that Intersect the Sphere	132
9.2.1	Determine the Column Type	133
9.2.2	Transform Nodal Co-ordinates by Column Id	133
9.3	Intersections of the Sphere With the Computational Cell . .	136
9.4	Six Node Tessellation Algorithm	138
9.4.1	The Three Node Triangle Tessellator	140
9.4.2	Computing the Mid-Edge Node	145
9.4.3	Volume Integration for 6 Node Triangles	146
9.4.4	Six Node Triangle Basis Functions and Derivatives . .	147
9.4.5	Implementation of Volume Integration with 6 Node Triangles	149
10	Numerical Results	151
10.1	Test Case Preparation	152
10.2	Critical Assessment of the PLIC Interface	156
10.3	Rigid Body Translation Tests	160
10.4	Rigid Body Rotation of Two Spheres	161

10.5	Advection in a Wu and Currie Flow Field	164
10.6	Advection in a Smolarkiewicz Flow Field	169
10.7	Analysis of a Real Casting	176
11	Conclusion	181
12	References	184

List of Figures

2.1	Comparison of the SLIC and PLIC methods for reconstructing a circular interface.	9
2.2	Comparison of the PPIC and PLIC methods for reconstructing a circular interface.	11
3.1	A flow chart of the 4 basic steps employed in the PLIC algorithm: First (1) discrete VOF data is provided on the computational domain. Second (2) a piecewise linear interface is constructed, then (3) volume of liquid to be advected is computed and finally (4) VOF values are advanced by integrating over time.	13
4.1	$3 \times 3 \times 3$ numbering map. VOF will advect from donor cell at position 13 to the acceptor cell at position 14.	18

5.1	Eight node numbering map in the $\hat{x}, \hat{y}, \hat{z}$ co-ordinate system. .	28
5.2	Ordering of plane normal contacts.	36
5.3	The five possible ways to shave a cube in a single plane cut so that not more than half of it is shaved.	37
5.4	Volume of air and volume of liquid in a control volume are complementary.	38
5.5	Scalar triple product.	44
5.6	The triangular interface polygon.	45
5.7	The quadrilateral type A interface polygon.	48
5.8	The pentagonal interface polygon.	53
5.9	The hexagonal interface polygon.	57
5.10	The quadrilateral type B interface polygon.	61
5.11	Extrapolation of a local derivative to compute the next guess of the root by the Newton-Raphson algorithm.	67
6.1	Flux of liquid that flows across an advecting face.	72
6.2	A rectangular element can advect on six faces.	75
6.3	Exit face id map.	78
6.4	Edge numbering map.	82

6.5	Measurement of ϵ to make a slice box on an exit face.	84
6.6	Face numbering map.	86
6.7	The actual slice box and complement slice box.	87
6.8	Volume of liquid in the actual and complement slice box. . . .	88
6.9	Projection map for polygon assembly.	94
6.10	Examples of a convex and a non-convex polygon.	97
6.11	Meshing the slice box polygon into three node triangles. . . .	98
6.12	Volume integration using one point integration. The height from the projection plane to the centroid of the triangle is H . .	99
6.13	The little triangle on edge numbers 4 and 7.	102
7.1	The flow line for a particle of fluid at point x with velocity $u(x, t)$ in region D	109
7.2	The mass flow rate across ∂W is: $\frac{d}{dt}(\text{mass crossing } \partial W) =$ $\int_{\partial W} \rho(u \cdot n) dA$	110
7.3	Interface movement during operator splitting for x and y sweep directions.	118
9.1	The 3 functional spaces considered for interface projection. . .	138
9.2	Two line segments of a polygon with nodes $(x_1, y_1); (x_2, y_2); (x_3, y_3)$.	141

9.3	The value of $\sin\theta$ is +ve in two quadrants and -ve in two quadrants.	143
9.4	Meshing 2 D polygons by “cutting off ears.”	144
9.5	Node numbering map for a reference 6 node triangle.	147
10.1	Decomposition of a 6 node triangle into 4 three node triangles.	153
10.2	Numerically simulated analytic spherical interface.	155
10.3	The PLIC interface.	158
10.4	Magnified view of the PLIC interface.	159
10.5	Rigid body rotation of two spheres.	163
10.6	Wu & Currie velocity field.	165
10.7	Two spheres in an analytic Currie & Wu flow velocity field at time $t = 0.0$	166
10.8	Two spheres in an analytic Wu & Currie flow velocity field. The velocity field is constant and the time step size is 0.11091 s.	167
10.9	Two spheres in an analytic Wu & Currie flow velocity field. This is at time step number 13.	168
10.10	The Smolarkiewicz deformation velocity field.	171

10.11	A cylindrical mass of fluid in a deformation velocity field.	
	The velocity field is constant and the time step size is 0.03167 s.	173
10.12	A cylindrical mass of fluid in a deformation velocity field.	
	The velocity field is constant and the time step size is 0.03167 s.	174
10.13	Geometry of a transmission case housing.	178
10.14	The interface in a 600, 000 element mesh of transmission case at about 54% filling.	179
10.15	The interface on a transmission case at about 54% filling showing about 46, 000 triangles.	180

List of Tables

- 4.1 Untransformed donor cell and acceptor cell pairs. 19
- 5.1 Reference 8 node brick co-ordinates. 29
- 5.2 Sum vertex numbers on the diagonals of the reference cell . . . 32
- 6.1 Exit face and sum pair diagonal vertex indices on exit face. . . 77
- 6.2 Co-ordinates of points on 12 edges of a unit cell. 83
- 6.3 Minimum and maximum co-ordinates of slice boxes on exit
faces. 85
- 6.4 Minimum and maximum co-ordinates of slice boxes. 90
- 6.5 Co-ordinates of the third point on the little triangle. 103
- 8.1 End point combination and global intercept of the interface. . 126
- 10.1 Comparison of analytic and numerical volume. 153

10.2	Lower and upper bound errors in the dot product.	157
10.3	Lower and upper bound errors in the radius.	157
10.4	Comparison of sum VOF sphere for $r = 4$ elements and 4.5 elements.	161
10.5	Conservation of Mass in rigid body rotation.	162
10.6	Conservation of Mass in a fountain flow velocity field.	166
10.7	Comparison of sum VOF cylinder for $r = 8$ elements	171
10.8	Conservation of Mass in a deformation velocity field.	172

Nomenclature

dx, dy, dz – Dimensions of an element in a regular grid in Cartesian directions.

f – Volume fraction of liquid present in a cell.

VOF – Volume of fluid in a cell.

$(\hat{x}, \hat{y}, \hat{z})$ – Co-ordinate system for advection.

$(x1, x2, x3)$ – Co-ordinate system for interface re-construction.

$(x1, x2, x3)$ – Co-ordinate system for determining the exit face.

(m_0, m_1, m_2) – Vertex numbers in $(x1, x2, x3)$ co-ordinate system.

(h_1, h_2, h_3) – Interface intercepts on the $x1, x2$ and $x3$ axes respectively.

$\bar{n}$ – The interface normal in the $(\hat{x}, \hat{y}, \hat{z})$ co-ordinate system.

$(n_{\hat{x}}, n_{\hat{y}}, n_{\hat{z}})$ – Components of $\bar{n}$ in the $(\hat{x}, \hat{y}, \hat{z})$ co-ordinate system.

(n_1, n_2, n_3) – Components of the interface normal in the $(x1, x2, x3)$ co-ordinate system.

d – Interface distance from $(0, 0, 0)$.

dt – Time step size.

CN – Courant Number.

(v_x, v_y, v_z) – Components of velocity in Cartesian directions.

ϵ – Length of a slice box in a unit cell.

δV – Volume of liquid in a slice box.

V – Volume of liquid in a donor cell.

donor cell – A cell that donates/looses liquid.

acceptor cell – A cell that accepts flux of liquid from a donor cell.

Chapter 1

Introduction

If one gazes at a still pond on a bright summer afternoon, one would notice small insects that can actually *walk* on water. Amazing as it may sound the insect's weight and the manner in which it is distributed prevents the insect from breaking through the surface. The phenomenon that keeps the insect afloat is a balance between surface tension forces and the insect's weight. This simple observation introduces the idea of a liquid-air surface or interface.

When two immiscible fluids are in contact the layer that separates them forms the *interface*. In a glass that is half full of water we know that the air in the glass ends at some height and exactly at this height water starts. So, the concept of an interface is certainly quite generic. One that exists in

nature and numerous examples can be seen in our everyday lives.

In the two examples above the fluids under consideration were stationary. Niagara falls, reminds us of picturesque scenic beauty and a vast quantity of water that gushes down a significant height. It is common for tourists to take photographs of these falls. Such a photograph would clearly indicate separation of the water and air. So, interfaces exist during fluid motion.

The process of candle making is a good example of a casting process. Using the same mould to produce each candle is the reason that candles look alike. Candles produced from a mould do not need any further processing. The idea of casting is to produce a shape that is as close as possible to the final shape and requires a minimum of machining. Sand casting uses gravity as the motive force to fill a sand mould while die casting employs a pressurized flow to fill metallic die cavities that could be as thin as 4 mm. at certain locations.

The availability of cheap computing power has been harnessed to develop a computer algorithm that ultimately provides data that enables the interface to be visualized. This algorithm has been used to simulate filling of real die castings.

The objective of this thesis was the developement of an algorithm to track

interfaces during fluid motion.

1.1 Organization of Thesis

This thesis begins with a short literature review. In the first chapter we review work done by selected researchers and establish a global understanding for numerical simulation of interface tracking methods.

In the next chapter a summary of the algorithm is provided. In the following chapters the algorithm is described in detail. Figures and examples have been used to explain ideas in detail.

Finally, the performance of the algorithm is assessed using interface graphics and numerical tests.

Preparation of test cases to present numerical results that examine the validity of ideas employed in various algorithms is done next. The thesis concludes by summarizing the results of these tests.

Chapter 2

Literature Review

Modern advection algorithms can provide a piecewise linear or a piecewise parabolic representation of the interface between two or more immiscible fluids. The piecewise linear interface tracking method will be referred to as PLIC and the piecewise parabolic interface tracking method will be referred to as PPIC. The commonality in both these methods, PLIC and PPIC, is the *art* of computing the interface constants. A PLIC algorithm constructs the interface as planes in each mixed cell and the interface is completely defined by computing the four constants in the general equation of a plane. On the other hand the PPIC interface needs computation of seven interface constants. Although PPIC requires almost twice as many constants as PLIC,

the complexity of the PPIC algorithm is much more than this ratio. Note, the discussion above is strictly for 3-D problems. The term art has been used deliberately to express several methods that have been used by researchers all to finally answer the same *hard* question, what is the best method to obtain the interface constants?

A grand challenge posed by an advection algorithm is the computation of the volume of liquid that can be advected from donor cells to acceptor cells. Literature shows that either an unacceptably complex “case by case” (with each case spawning several sub-cases) approach has been used or the problem has been solved as a geometric problem. The PPIC problem has not yet been fully solved in 3D.

The final challenge is the method adopted for time integration to advance fluxes of liquid from one time step to another. To advance liquid in the computational domain the advection equation that is basically the law of conservation of mass is solved.

In this chapter we look at solutions provided by selected researchers. The purpose of providing a literature review here is to build a global understanding of interface advection techniques that have been used and those that are used in practice today.

Two notable researchers have had an impact on the author of this thesis, a brief introduction about them is first provided.

David L. Youngs (1984) of the Atomic Weapons Research Establishment (U.K), a mathematician at heart, provides the earliest applications of PLIC to 3-D situations. His method to compute the interface normal by using a central differencing scheme [15] is a landmark development. In my opinion it represents an extremely clever method for constructing a local co-ordinate system for interface re-construction. This is another landmark that has not been fully acknowledged in literature. Although the information provided is terse, there is a discussion on managing co-ordinate systems. The family of piecewise linear methods introduced by adopting Youngs algorithm are referred to as PLIC methods.

Douglas Kothe of the Los Alamos National Laboratory is undoubtedly a leader in the field of interface computation. Lucid explanations and an inherent passion for the subject are easily noticed in his published work. Careful planning of numerical simulations and proposal of a few daring tests are a feature of Kothe's work. Kothe has extended Youngs work to unstructured meshes [2] by introducing a second order Runge-Kutta method for time integration and a robust method for plane truncation of arbitrary polyhedra.

2.0.1 Developments in Interface Tracking Methods

The field of interface advection is crowded with acronyms. Each acronym represents a method. In the order of their evolution the notable ones are,

- SLIC (for Simple Line Interface Computation).
- VOF (for Volume-Of-Fluid).
- PLIC (for Piecewise Linear Interface Computation).
- PPIC (for Piecewise Parabolic Interface Computation).

SLIC Method

The SLIC algorithm devised by Noh and Woodward [6] invokes a piecewise constant approximation to the interface. In this algorithm the interface in each cell is assumed to be lines in a 2-D problem or planes in a 3-D implementation. They are aligned with one of the logical mesh co-ordinates. It is suspected that the motivation to work with a simple representation of the interface was to allow more than two fluids in the computational domain. The time integration technique used was operator splitting.

VOF Method

Hirt and Nichols' [3] VOF method uses a piecewise constant ("stair stepped") approximation of the interface. It is similar to the SLIC method. Interfaces in the VOF method are also forced to align with mesh co-ordinates, but are additionally allowed to align with more than one mesh co-ordinate ("stair step") within each cell, depending upon the local distribution of discrete volume data. A notable feature of the VOF method is that its volume fluxes can be solved as algebraic equations. This also means that exact interface re-construction is avoided. The volume fluxes can be expressed as a weighted sum of upwind and downwind contributions, depending on the orientation of the interface relative to the local flow direction. The time integration technique used was operator splitting.

A shortcoming of the VOF and the SLIC methods is the unphysical creation of what [6] call *flotsam* (floating wreckage). These algorithms produce sub-mesh size material bodies that separate from the main material body. It is suspected that the shortcoming has under-currents of a diffused interface.

PLIC Method

D. L. Youngs [15] is the pioneer of this algorithm. He published the first 3-D results in 1984. The algorithm uses a central differencing scheme to reconstruct a planar interface in each mixed cell using data values of VOF from its 26 element neighborhood. Volume Integration is done by algebraic expressions each suited for a set constraints. Time integration is done by operator splitting.

Figure 2.1: Comparison of the SLIC and PLIC methods for reconstructing a circular interface.

Literature provides examples of extensions to Youngs algorithm. Johnson

[4] extended the 2-D version of Youngs algorithm to nonorthogonal meshes. Puckett and Saltzman [8] coupled an adaptive mesh refinement algorithm to Young's 3-D algorithm. A comparison of the SLIC and the PLIC interface reconstruction schemes is shown in Figure 2.1.

PPIC Method

G.R. Price's [7] piecewise parabolic method provides a second order representation of the interface. The algorithm has been implemented in 2-D.

To reconstruct the interface, VOF values are sampled in a 3x3 block of cells. Least squares estimation is used to determine the co-efficients of the local quadratic that has a representation of the form $y = ax^2 + bx + c$. The time integration scheme used is quiet novel. The Eulerian cell vertices are advected in a Lagrangian sense by integrating along their downwind or upwind characteristics. A Lagrangian cell is formed at the new "advected" vertex locations. The vertices are connected by straight lines. This method is radically different from operator splitting and needs only one sweep over the mixed cells to complete one time step. Figure 2.2 illustrates the differences between the PPIC and PLIC reconstruction methods.

Figure 2.2: Comparison of the PPIC and PLIC methods for reconstructing a circular interface.

Solving a Cubic Spline

Alexander Z. Zinchenko [16] provides a third order representation of the interface. The problem has been solved in 3-D and the most notable feature of this work is that it has been applied to real problems to study interface stretching in colloidal suspensions.

Chapter 3

Algorithm Summary

PLIC, is an acronym for Piecewise Linear Interface Computation. This is a time dependent, Eulerian method for tracking material interfaces. In this thesis it has been implemented in 3-D rectangular grids using Strang's [12] operator splitting time integration technique.

The algorithm assumes that volume fractions and a divergence free velocity field for each cell in the computational domain are known initial data. The distribution of volume fractions is used to construct a planar interface in each mixed cell. A mixed cell has volume fraction greater than zero and less than one. The velocity component on an advecting face together with the time step size are used to compute a slice box in the mixed cell. A slice box

is a subset of the bounding box of a mixed cell. The position of the interface in such a mixed cell is used to compute the volume of liquid carried by the flux that flows across that advecting face and this enables volume fractions to be updated for the next time step. The volume fraction value has the acronym VOF (volume of fluid). A pictorial presentation of the algorithm is provided in Figure 3.1.

Figure 3.1: A flow chart of the 4 basic steps employed in the PLIC algorithm: First (1) discrete VOF data is provided on the computational domain. Second (2) a piecewise linear interface is constructed, then (3) volume of liquid to be advected is computed and finally (4) VOF values are advanced by integrating over time.

The stages involved in implementing PLIC are as follows:

- For a mixed cell get the VOF values for each element in its 26 element neighbourhood.
- Depending on the advection direction rotate the volume fraction distribution such that the donor cell is at index 13 and the acceptor cell is at index 14.
- Compute the volume fraction gradient, ∇f .
- Compute the interface normal.
- Match the volume fraction in the mixed cell to get interface distance.
- Compute a local co-ordinate system for interface construction.
- Compute the exit face in the local co-ordinate system.
- Compute the interface equation.
- Compute the intersections of the planar interface with the slice box
- Assemble the interface polygon in the slice box.
- Mesh the interface polygon into 3 node triangles.

- Sum volume under each 3 node triangle. This is the volume of liquid that flows across a cell side.
- Update volume fractions of donor and acceptor cells using operator splitting time integration technique.
- Visualize the interface

It is possible that the reader is not completely aware of all terms used in the steps mentioned above. All terms are explained when we look at each step in detail. The purpose of writing out the steps above is to provide a quick overview of the algorithm.

This algorithm is a variation of Youngs' algorithm. The variation involves the implementation of a *unified, case independent and computationally efficient* method for computing the flux of liquid that is advected through a face of a mixed cell. Secondly, we separate analysis from post-processing for visualization and compute the global equation of the interface plane. Lastly, the computational sub-domain is dynamic and is a small subset of the full domain.

Chapter 4

Interface Slope

In this chapter we look at the first four steps as described in the summary of the algorithm. For clarity we write these four steps once again.

- For a mixed cell get volume fraction distribution in its 26 element neighbourhood. This is a 3x3x3 neighborhood.
- Depending on the advection direction rotate the volume fraction distribution to maintain consistency in the position of the mixed cell (donor cell) and the corresponding acceptor cell.
- Compute the volume fraction gradient, ∇f .
- Compute the interface normal.

The algorithm used in determining the interface normal is crucial and distinguishes various volume tracking algorithms [9].

4.1 VOF Values in a 3x3x3 Stencil

To construct the interface in the core mixed cell the values of the VOF in the 26 element neighbourhood of the mixed cell were needed. The core mixed cell is the donor cell and the cell that accepts flux that flows from the donor to a face adjacent neighbour is an acceptor cell.

In order to freeze the positions of the 27 cells in discussion here, a map for the $3 \times 3 \times 3$ element numbering system is presented in Figure 4.1 on page 18. Note the labels for the $+\hat{x}$ and $+\hat{y}$ axes. The $+\hat{z}$ axis, which is not shown in the figure points out from the paper to form a right handed co-ordinate system.

4.1.1 Transformation Maps for an Advecting Face

The advection directions belong to the set:

$$\{ (+\hat{x}, 0, 0); (-\hat{x}, 0, 0); (0, +\hat{y}, 0); (0, -\hat{y}, 0); (0, 0, +\hat{z}); (0, 0, -\hat{z}) \}.$$

The transformation map for advection in the $+\hat{x}$ direction is as shown in the

Figure 4.1: $3 \times 3 \times 3$ numbering map. VOF will advect from donor cell at position 13 to the acceptor cell at position 14.

$3 \times 3 \times 3$ numbering system, Figure 4.1 on page 18. For advection in all directions other than the $+\hat{x}$ direction we first write out pairs, these pairs are donor and acceptor cell pairs, see Table 4.1 on page 19. The position of the acceptor cell which is not lexically ahead for the donor cell except for advection in the $+x$ advection direction is a specification for setting up

transformation maps for the $-\hat{x}$, $+\hat{y}$, $-\hat{y}$, $+\hat{z}$ and $-\hat{z}$ advection directions. The requirement is to have the acceptor cell in the $+\hat{x}$ transformation map to be on the right of the donor cell. The position of the donor cell is always the core cell or central cell in the 27 element neighbourhood; the thirteenth cell counting from zero.

Advection Direction	Donor Cell Index	Acceptor Cell Index
$+\hat{x}$	13	14
$-\hat{x}$	13	12
$+\hat{y}$	13	16
$-\hat{y}$	13	10
$+\hat{z}$	13	22
$-\hat{z}$	13	4

Table 4.1: Untransformed donor cell and acceptor cell pairs.

An example for creating a transformation map for the $-\hat{x}$ advection direction is given. In the $3 \times 3 \times 3$ numbering system, Figure 4.1 on page 18, note that the acceptor cell for $-\hat{x}$ advection is lexically lagging with respect to the donor cell. Table 4.1 shows untransformed donor cell and acceptor cell

pairs. To position it lexically ahead for the donor we would like a rotation about the $\hat{y}$ axis so that the $-\hat{x}$ axis is transformed to the $+\hat{x}$ axis and the $-\hat{z}$ axis is transformed to the $+\hat{z}$ axis. The reasoning for the transformation is derived from the right hand rule. In this way we produce the following transformation maps.

Transformation Map $+\hat{x}$

[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26]

Rotation Map $-\hat{x}$

[8 7 6 5 4 3 2 1 0 17 16 15 14 13 12 11 10 9 26 25 24 23 22 21 20 19 18]

Rotation Map $+\hat{y}$

[2 5 8 1 4 7 0 3 6 11 14 17 10 13 16 9 12 15 20 23 26 19 22 25 18 21 24]

Rotation Map $-\hat{y}$

[6 3 0 7 4 1 8 5 2 15 12 9 16 13 10 17 14 11 24 21 18 25 22 19 26 23 20]

Rotation Map $+\hat{z}$

[2 11 20 5 14 23 8 17 26 1 10 19 4 13 22 7 16 25 0 9 18 3 12 21 6 15 24]

Rotation Map $-\hat{z}$

[8 9 0 21 12 3 24 15 6 19 10 1 22 13 4 25 16 7 20 11 2 23 14 5 26 17 8]

4.1.2 Use of a Rotation Map

It is clear from the transformation maps that they can be created once, cached and then used as and when required. It has been mentioned that the maps are constructed to position the acceptor cell lexically ahead of the donor. It is clear from this arrangement that in the PLIC algorithm the advection direction is frozen to be the $+\hat{x}$ advection direction. To use the maps above one obtains volume fractions in the $3 \times 3 \times 3$ stencil as supplied by the $3 \times 3 \times 3$ numbering system and then re-positions the values of volume fractions in each cell by the index specified in a transformation map applied for a specific advection direction.

4.2 Computing the Gradient of VOF ∇f

The ideas of a gradient are in no way restricted to interface advection, the concept is general and is discussed briefly below.

We consider a scalar field in space given by a scalar function $f(P) = f(x, y, z)$. A scalar function is a function that is defined at each point on a

certain set of points in space and whose values are real numbers depending only on the points in space but *not* on the particular choice of the of the co-ordinate system.

Kreyszig [5] gives the definition stated above. To understand it better an example is provided. If one drives ten poles of different heights in the ground and decides to construct a polynomial that passes through the tips of each of the poles, then this polynomial is a scalar function. The function depended only on the individual height of each pole and the relative distances between poles. These are all real *real numbers*. The function did *not* depend on where the ten poles were located i.e., there is no dependence on the co-ordinate system. Interestingly, a scalar function is unaltered by co-ordinate transformations is an emerging fact.

The first partial derivatives of f with respect to co-ordinate components are the rates of change of f in the directions of the co-ordinate axes. We want to determine the the rate of change of f in *any* direction. This is called a *directional derivative*.

The next task is to determine the gradient of volume fraction in the 26 element neighbourhood. In Figure 4.1 on page 18 note labels for North (N), South (S), East (E), West (W), Upper Level and Lower Level in the $3 \times 3 \times 3$

numbering system. For simplicity we refer to the Upper Level as Top (T) and the Lower Level as Bottom (B). A discrete approximation to the gradient of VOF is a vector ∇f given by:

$$\nabla f = \begin{pmatrix} \nabla_{\hat{x}} f \\ \nabla_{\hat{y}} f \\ \nabla_{\hat{z}} f \end{pmatrix}.$$

where,

$$\nabla_{\hat{x}} f = \left(\frac{\sum f_E - \sum f_W}{2dx} \right).$$

$$\nabla_{\hat{y}} f = \left(\frac{\sum f_N - \sum f_S}{2dy} \right).$$

$$\nabla_{\hat{z}} f = \left(\frac{\sum f_T - \sum f_B}{2dz} \right).$$

We recall that an advection direction other than the $+\hat{x}$ advection needed a transformation. It is required that the interface slope, i.e. the gradient of volume fraction is unaltered by such transformations, thus a symmetric difference approximation must be used. This is a Central Difference Approximation and uses the notion of weighted averages.

$$\sum f_E = \frac{f_2 + f_8 + f_{20} + f_{26} + \beta(f_5 + f_{11} + f_{17} + f_{23}) + \gamma f_{14}}{4 + 4\beta + \gamma}$$

$$\sum f_W = \frac{f_0 + f_6 + f_{18} + f_{24} + \beta(f_3 + f_9 + f_{15} + f_{21}) + \gamma f_{12}}{4 + 4\beta + \gamma}$$

$$\sum f_N = \frac{f_6 + f_8 + f_{24} + f_{26} + \beta(f_7 + f_{15} + f_{17} + f_{25}) + \gamma f_{16}}{4 + 4\beta + \gamma}$$

$$\sum f_S = \frac{f_0 + f_2 + f_{18} + f_{20} + \beta(f_1 + f_9 + f_{11} + f_{19}) + \gamma f_{10}}{4 + 4\beta + \gamma}$$

$$\sum f_T = \frac{f_{18} + f_{20} + f_{24} + f_{26} + \beta(f_{19} + f_{21} + f_{23} + f_{25}) + \gamma f_{22}}{4 + 4\beta + \gamma}$$

$$\sum f_B = \frac{f_0 + f_2 + f_6 + f_8 + \beta(f_1 + f_3 + f_5 + f_7) + \gamma f_4}{4 + 4\beta + \gamma}$$

It has been shown by Youngs [15] that choosing values of the weights β and γ as 2 and 4 respectively determines the exact interface when the real interface is planar. It has also been shown that interface reconstruction is not very sensitive to selection of values of β and γ when the real interface is not planar.

4.3 Given ∇f Compute the Interface Normal

The interface normal is a vector given by :

$$\bar{n} = \begin{pmatrix} n_{\hat{x}} \\ n_{\hat{y}} \\ n_{\hat{z}} \end{pmatrix} = \begin{pmatrix} \frac{\nabla_{\hat{x}} f}{|\nabla f|} \\ \frac{\nabla_{\hat{y}} f}{|\nabla f|} \\ \frac{\nabla_{\hat{z}} f}{|\nabla f|} \end{pmatrix}$$

where,

$$|\nabla f| = \sqrt{(\nabla_{\hat{x}} f)^2 + (\nabla_{\hat{y}} f)^2 + (\nabla_{\hat{z}} f)^2}$$

Chapter 5

Interface Distance

In the previous discussion the interface normal $\bar{n}$ was computed. Here we discuss how to compute the distance at which a planar interface is placed from the origin of a computational cell such that volume fraction of the donor cell is conserved. In many algorithms that deal with advection of interfaces, this action of matching the volume fraction of the donor by computing the interface distance from a fixed point in the computational cell is in-fact an action that *re-constructs* the interface.

A local co-ordinate system is devised for convenience. It is defined by the three mutually perpendicular vectors x_1 , x_2 and x_3 . The next task is to establish a relation between the $\hat{x}$, $\hat{y}$, $\hat{z}$ space, the space in which $\bar{n}$ was

computed and the space in which we reconstruct the interface, i.e., the x_1 , x_2 and x_3 space.

In the co-ordinate system, x_1 , x_2 and x_3 , we do the following:

- Reverse the interface normal $\bar{n}$ if volume fraction of the donor is greater than 0.5.
- Determine components for the interface normal $\bar{n}$.
- Determine the interface polygon type.
- Compute the interface distance.
- Determine the exit face.

5.1 Co-ordinate System for Interface Reconstruction

This PLIC algorithm is designed for a regular grid, so all elements in the grid have the same dimensions in Cartesian directions. In order to simplify computation the computational cell is transformed to a unit cell. The vertex map for such a unit cell in the $\hat{x}$, $\hat{y}$, $\hat{z}$ space is shown in Figure 5.1.

Figure 5.1: Eight node numbering map in the $\hat{x}$, $\hat{y}$, $\hat{z}$ co-ordinate system.

For clarity a table with the node number m and the corresponding point in the $\hat{x}$, $\hat{y}$, $\hat{z}$ is shown in Table 5.1. Note that the corner number or node number $m \in \{1, 2, 3, 4, 5, 6, 7, 8\}$.

Node Number, m	$\hat{x}$	$\hat{y}$	$\hat{z}$
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

Table 5.1: Reference 8 node brick co-ordinates.

In the author's opinion this method to define the local co-ordinate system for interface construction is very cleverly devised by Youngs[15]. While reviewing literature on interface advection it is found that only Youngs provides a discussion on the co-ordinate system in which the interface is recon-

structed. With all respect for Youngs work, the author is most fascinated by the method that is explained below. The simplicity of the method is not obvious. However a full attempt is made to provide a clear explanation.

It is important that while one conducts research one is able to separate what is to be done from how it is to be done. If we agree with this statement, some simple questions are posed with reference to the eight node numbering map as seen in Figure 5.1 on page 28.

Question 1: Given that the centroid of the unit cell is located at $(1/2, 1/2, 1/2)$ what are the vertex numbers that can possibly form the body diagonal of the unit cube?

Answer 1:

- node 1 , node 8
- node 5 , node 4
- node 7 , node 2
- node 3 , node 6

Question 2: What is the sum of the node numbers for the four possible choices of the body diagonal of the unit cell?

Answer 2: Interesting as it may seem the answer for all the four possible choices is 9. We note that we are doing only integer arithmetic here.

Question 3: The unit cell has 6 faces. We associate a *min* face in the $\hat{x}$, $\hat{y}$, $\hat{z}$ space because the point $(0, 0, 0)$, i.e., node number 1 is a member of the face. Similarly we associate a *max* face in the $\hat{x}$, $\hat{y}$, $\hat{z}$ space because the point $(1, 1, 1)$, i.e., node number 8 is a member of the face. For each of the 6 faces in the $\hat{x}$, $\hat{y}$, $\hat{z}$ what is the sum of the node numbers that are members of the 2 face diagonals?

Answer 3: The answer is provided in Table 5.2 on page 32.

Note that on the $\hat{x}_{min}$ and $\hat{x}_{max}$ faces the node numbers sum to 5 and 13 respectively.

5.1.1 Transform from $(\hat{x}, \hat{y}, \hat{z})$ Space to (x_1, x_2, x_3) Space

The interface normal $\bar{n}$ has been defined in the $\hat{x}$, $\hat{y}$, $\hat{z}$ space. The definition of a plane is complete if one knows the normal to the plane and a point on the plane. This is a very general definition of a plane and is globally true. One representation of such an equations is:

Face Description	Diag 1	Diag 2	Sum Members Diagonals
$\hat{x}_{min}$	1, 4	2, 3	5
$\hat{x}_{max}$	5, 8	6, 7	13
$\hat{y}_{min}$	2, 5	6, 1	7
$\hat{y}_{max}$	8, 3	7, 4	11
$\hat{z}_{min}$	1, 7	5, 3	8
$\hat{z}_{max}$	2, 8	4, 6	10

Table 5.2: Sum vertex numbers on the diagonals of the reference cell

$$P_m = n_{\hat{x}}\hat{x}_m + n_{\hat{y}}\hat{y}_m + n_{\hat{z}}\hat{z}_m$$

We recall that m belongs to the set $\{ 1, 2, 3, 4, 5, 6, 7, 8 \}$. For a fixed $\bar{n}$ and m being one of the eight points associated with the eight node numbers one obtains eight possible values of P_m . The eight values of P_m denote the distance of node 1, i.e., point $(0, 0, 0)$ to each of the eight planes that passes through each of the eight corners. We note that each plane passing through each vertex node is a different plane and it is the normal that was kept constant. Now the eight values of P_m are sorted so that the plane closest to

node 1 is the first plane and the plane furthest from node 1 is the last plane. For $m = 1$ the point is $(0, 0, 0)$ so $P_m = 0.0$, similarly for $m = 8$ is $(1, 1, 1)$ so $P_m = n_{\hat{x}} + n_{\hat{y}} + n_{\hat{z}}$. From the set of eight sorted values of P_m the first three values are queried for the node numbers that provided values for $\hat{x}_m$, $\hat{y}_m$ and $\hat{z}_m$. Now we have three node numbers that we denote as m_0, m_1, m_2 .

The origin of x_1, x_2 and x_3 space is m_0 .

The x_1 axis is defined by node numbers m_0 and m_1 .

The x_2 axis is defined by node numbers m_0 and m_2 .

Now the only job that remains is to define the x_3 axis. The requirement is that x_1, x_2, x_3 *must* form a right handed Cartesian co-ordinate system. We have a plane defined by node numbers m_0, m_1 and m_2 . We recall that the sum of the diagonally opposite node numbers of a face produced unique numbers that belonged to the set $\{ 5, 13, 7, 11, 8, 10 \}$ see Table 5.2 on page 32. For clarity we present an example for the $\hat{y}_{min}$ face. The diagonal node numbers associated are $(2, 5)$ and $(6, 1)$. We can write,

$$2 + 5 = 6 + 1 = 7$$

It is equally true that,

$$2 + 5 - 6 = 1,$$

$$2 + 5 - 1 = 6,$$

$$6 + 1 - 2 = 5 \text{ and}$$

$$6 + 1 - 5 = 2$$

This tells us that given three node numbers that belong to a face the fourth node number that is a member of a face is easily determined. We define m_3 as the fourth corner associated with the face that has node numbers m_0 , m_1 and m_2 . We now write,

$$m_3 = m_1 + m_2 - m_0$$

The x_3 axis is yet to be defined. We recall that the sum of node numbers that were members for the 4 possible choices of body diagonal of the unit cell always sum to 9 , see answer on pages 31. Let m_4 be the node number that pairs with m_3 to form a body diagonal of the unit cell. We can write

$$m_3 + m_4 = 9$$

So,

$$m_4 = 9 - m_3.$$

m_0, m_4 is guaranteed to be *perpendicular* to axes defined by m_0, m_1 and m_0, m_2 .

We have now fully defined the co-ordinate system $(x1, x2, x3)$ in which the interface construction will be done. The co-ordinate system is as follows.

The origin of $x1, x2$ and $x3$ space is m_0 .

The $x1$ axis is defined by node numbers m_0 and m_1 .

The $x2$ axis is defined by node numbers m_0 and m_2 .

The $x3$ axis is defined by node numbers m_0 and m_4 .

One could obtain values of m_0, m_1, m_2, m_4 as 8, 7, 6, 4. This shows that the origin of the $x1, x2$ and $x3$ space is node 8 i.e., (1, 1, 1,). The $x1$ axis is defined by a vector from node 8 to node 7. The $x2$ axis is defined by a vector from node 8 to node 6 and the $x3$ axis is defined by a vector from node 8 to node 4. This method of co-ordinate transformation is an elegant algorithm that does not explicitly use any transformation matrices and yields right handed co-ordinate systems.

To make it easier to visualize how eight planes are made at each vertex

of the unit cell and to explain the interaction of the interface slope with the associated vertices see Figure 5.2. In this figure the plane first touches the vertex P_0 then P_1 and so on.

Figure 5.2: Ordering of plane normal contacts.

5.1.2 Reverse the Interface Normal $\bar{n}$

A simple experiment is proposed to clarify the statement that if the volume fraction of the donor is less than 0.5 there are only *five interface types* to be considered.

Figure 5.3: The five possible ways to shave a cube in a single plane cut so that not more than half of it is shaved.

If one begins to shave a wooden cube with a jack plane without shaving

more than half the volume of the cube, the resulting shapes of the cut will produce one of the possible shapes shown in Figure 5.3 on page 37. Note that the one plane experiment did *not* impose any restriction on the angle of the cut.

In the experiment described above the removal of wood is imagined as an increase in the volume fraction in a mixed cell. The resulting polygon from the cut is the shape of the interface.

Figure 5.4: Volume of air and volume of liquid in a control volume are complementary.

To reduce the complexity in the code we reverse the sign of the normal $\bar{n}$ if the volume fraction of the donor exceeds 0.5. In other words instead

of the interface normal pointing into air it now points into the liquid. All computation is done for air instead of liquid if the volume fraction of the donor exceeds 0.5. The volume fraction of air in a mixed cell is $(1.0 - \text{volume fraction of liquid})$ in the mixed cell. Volume of air and volume of liquid in a mixed cell are shown in Figure 5.4 on page 38 as complementary.

5.2 Components of $\bar{n}$ in $x1$, $x2$ and $x3$ Space

The components of $\bar{n}$ in the $\hat{x} \hat{y} \hat{z}$ space are known and they are $n_{\hat{x}}, n_{\hat{y}}$ and $n_{\hat{z}}$. The components of $\bar{n}$ in $x1, x2$ and $x3$ space are denoted by $n1, n2$ and $n3$. They are chosen by sorting

$$|n_{\hat{x}}|; |n_{\hat{y}}|; |n_{\hat{z}}|$$

such that the smallest component in the sort appears as the first component and the biggest component appears as the last component. To summarize, $n1, n2$ and $n3$ are the minimum, intermediate and maximum values in the set

$$\{|n_{\hat{x}}|, |n_{\hat{y}}|, |n_{\hat{z}}|\}$$

This guarantees that the $x3$ axis is the axis with which the interface is most perpendicular. Note that the projection plane for the interface can be the

plane defined by x_1 and x_2 axes.

5.3 Interface Intercepts h_1, h_2, h_3 in x_1, x_2, x_3 Space

The origin of the x_1, x_2, x_3 space is at node m_0 . Let the intercepts made by the planar interface with the x_1, x_2 and x_3 axes be h_1, h_2 and h_3 , see Figure 5.5. Let d be the distance of the planar interface from m_0 . Then,

$$h_1 = d/n_1; \quad h_2 = d/n_2; \quad h_3 = d/n_3$$

Note that since $n_1 \leq n_2 \leq n_3$,

$$h_1 \geq h_2 \geq h_3$$

5.4 Interface Polygon Type

The values of n_1, n_2 and n_3 are known and that is all that is needed to declare the type of the interface polygon. By declaring an interface shape we mean to associate an interface with proper nouns for common polygonal

shapes observed in nature. Below are written out the number of characteristic nodes associated with a convex polygonal shape.

- A polygon with 3 nodes is a triangle.
- A polygon with 4 nodes is a quadrilateral.
- A polygon with 5 nodes is a pentagon.
- A polygon with 6 nodes is a hexagon.

Let us assume for a moment that the distance d of the interface from m_0 is known. If d is known then together with values of n_1 , n_2 and n_3 the intercepts of the planar interface h_1 , h_2 and h_3 can be determined. We recall that the computational cell under consideration is a unit cell. Therefore h_1 , h_2 and h_3 can have values less than 1.0 or greater than 1.0 or equal to one, which means that each intercept of the planar interface can be inside the unit cell, outside the unit cell or on the boundary of the unit cell. For each of the five interface types two tasks are performed.

- Compute interface constraints and declare an interface type.
- Given an interface type compute the distance from the origin m_0 to the interface so that volume fraction of the donor is conserved.

A proof for the volume of a tetrahedron formed by the intercepts of the planar with the local co-ordinate system is presented below because we will need to use ideas from tetrahedrons in building constraints to determine interface polygon types.

5.4.1 Volume Of a Tetrahedron

The volume of a tetrahedron is computed from a **scalar triple product** or a *mixed triple product* . A scalar triple product of three vectors **a**, **b** and **c** is given by,

$$\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})$$

With respect to any right handed Cartesian co-ordinate system (**i**, **j**, **k**), let,

$$\mathbf{a} = a_1\mathbf{i} + a_2\mathbf{j} + a_3\mathbf{k}$$

$$\mathbf{b} = b_1\mathbf{i} + b_2\mathbf{j} + b_3\mathbf{k}$$

$$\mathbf{c} = c_1\mathbf{i} + c_2\mathbf{j} + c_3\mathbf{k}$$

then,

$$\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = (a_1 \mathbf{i} + a_2 \mathbf{j} + a_3 \mathbf{k}) \cdot \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix}.$$

We rewrite the above as

$$\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = \begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix}.$$

We denote the scalar triple product $\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})$ by $(\mathbf{abc})$.

then $(\mathbf{abc}) = |\mathbf{a}| |\mathbf{b} \times \mathbf{c}| \cos \beta$,

$(\mathbf{b} \times \mathbf{c})$ is a vector by itself and β is the angle produced by the dot product.

For the local co-ordinate system x_1, x_2, x_3 we can set β to zero so $\cos \beta =$

1.0. The intercepts are h_1, h_2 and h_3 so,

$$(\mathbf{abc}) = \begin{vmatrix} h_1 & 0 & 0 \\ 0 & h_2 & 0 \\ 0 & 0 & h_3 \end{vmatrix} = h_1 h_2 h_3$$

$h_1 h_2 h_3$ is a scalar and is the volume of the rectangular parallelepiped as shown in the figure 5.5.

Figure 5.5: Scalar triple product.

Six congruent tetrahedrons are contained in the rectangular parallelepiped. So the volume of a tetrahedron having the intercepts are h_1 , h_2 and h_3 is $\frac{1}{6}h_1h_2h_3$.

In five sections that follow each of the five interface polygons will be discussed in detail. The interface distance and the interface constraint will be derived.

5.5 The Triangular Interface Polygon

Figure 5.6: The triangular interface polygon.

For the triangular interface polygon shown in Figure 5.6 the interface intercepts h_1 , h_2 and h_3 obey the following constraints.

- h_1 , the intercept on the x_1 axis is ≤ 1.0
- h_2 , the intercept on the x_2 axis is ≤ 1.0
- h_3 , the intercept on the x_3 axis is ≤ 1.0

The volume of a tetrahedron with edge lengths h_1 , h_2 and h_3 is,

$$volume = \frac{1}{6}h_1h_2h_3$$

Now substitute for h_1 , h_2 and h_3 as,

$$h_1 = \frac{d}{n_1}; \quad h_2 = \frac{d}{n_2}; \quad h_3 = \frac{d}{n_3}$$

This is rewritten as,

$$volume = \frac{1}{6} \frac{d}{n_1} \frac{d}{n_2} \frac{d}{n_3} = \frac{d^3}{6n_1n_2n_3}$$

The volume of the tetrahedron is in-fact the volume fraction of the donor f . For a unit cube. So,

$$f = \frac{d^3}{6n_1n_2n_3}$$

$$\boxed{d = \sqrt[3]{6n_1n_2n_3f}}$$

5.5.1 Interface Constraint Triangular Interface Polygon

In the limiting case the interface intercept h_1 can be 1.0 and can still continue to provide a triangular interface. Since,

$$h_1 = \frac{d}{n_1}$$

In the limiting case $h_1 = 1.0$.

$$d = n_1$$

Substituting this value of d in the expression

$$d = f(6n_1n_2n_3f)^{\frac{1}{3}}$$

and respecting the global constraints $h_1 > h_2 > h_3$

The interface constraint for a triangular polygonal section is

$$\boxed{6n_1n_2n_3f < n_1^3}$$

Interestingly, for the biggest tetrahedron that can be formed in the unit cell, the values of n_1, n_2, n_3 are $\{1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3}\}$. It is also known that for such a case the volume fraction of liquid in the cell is $1/6$ and the volume fraction of air in the cell is $5/6$. This provides a test case for checking code as it is written. Note, the interface shape has been declared by only knowing the values of n_1, n_2, n_3 and volume fraction of the donor. Determination of the interface distance d is provided by an explicit expression, i.e., the solution of an algebraic equation

5.6 The Quadrilateral Type A Interface Polygon

Figure 5.7: The quadrilateral type A interface polygon.

The interface can have two types of quadrilateral sections. The type discussed here is shown in Figure 5.7 and has the interface intercepts h_1 , h_2 and h_3 that obey the following constraints.

- h_1 , the intercept on the x_1 axis is $\gg 1.0$
- h_2 , the intercept on the x_2 axis is ≤ 1.0
- h_3 , the intercept on the x_3 axis is ≤ 1.0

The volume enclosed by the intercepts h_1, h_2 and h_3 is again a tetrahedron. The volume fraction of liquid in the unit cell is obtained by subtracting the volume of the tetrahedron outside the unit cell from the tetrahedron formed by the interface intercepts. We first derive a relation for the volume of the tetrahedron outside the unit cell. The edge lengths of the little tetrahedron say e_1, e_2 and e_3 are computed as follows,

$e_1 = h_1 - 1$, this is the edge length on the x_1 axis.

e_2 is the edge length on the edge of the unit cube which is parallel to the x_2 axis.

$$\frac{h_1 - 1}{h_1} = \frac{e_2}{h_2}$$

$$e_2 = \frac{h_2(h_1 - 1)}{h_1}$$

e_3 is the edge length on the edge of the unit cube which is parallel to the x_3 axis.

$$\frac{h_1 - 1}{h_1} = \frac{e_3}{h_3}$$

$$e_3 = \frac{h_3(h_1 - 1)}{h_1}$$

This gives the volume of the little tetrahedron as

$$e_1 e_2 e_3 \frac{1}{6} = (h_1 - 1) \left(\frac{h_2(h_1 - 1)}{h_1} \right) \left(\frac{h_3(h_1 - 1)}{h_1} \right) \frac{1}{6} = h_1 h_2 h_3 (h_1 - 1)^3 \frac{1}{6h_1^3}$$

$$\text{volume of little tetrahedron} = h_1 h_2 h_3 (h_1 - 1)^3 \frac{1}{6h_1^3}$$

$$\text{volume of big tetrahedron} = h_1 h_2 h_3 \frac{1}{6}$$

This is rewritten as,

$$\text{volume of liquid in the unit cell } f = (\text{volume of big tetrahedron}) - (\text{volume of little tetrahedron})$$

$$f = \frac{1}{6}(h_1 h_2 h_3) \left\{ 1 - \left(\frac{h_1 - 1}{h_1} \right)^3 \right\}$$

Now substitute for h_1 , h_2 and h_3 as,

$$h_1 = \frac{d}{n_1}; \quad h_2 = \frac{d}{n_2}; \quad h_3 = \frac{d}{n_3}$$

so,

$$f = (d^3/6n_1n_2n_3)[1 - \{((d/n_1) - 1)/(d/n_1)\}^3]$$

This is a quadratic equation that has 2 roots,

$$d = \frac{1}{6}\{3n_1 - (\sqrt{3} \sqrt{-n_1^2 + 24fn_2n_3})\}$$

and

$$d = \frac{1}{6}\{3n_1 + (\sqrt{3} \sqrt{-n_1^2 + 24fn_2n_3})\}$$

The first value of d makes d negative which is not feasible. So the second value of d is accepted. The $\sqrt{3}$ is re-arranged to get

$$\boxed{d = \frac{1}{6}\{3n_1 + (\sqrt{72fn_2n_3 - 3n_1^2})\}}$$

5.6.1 Interface Constraint Quadrilateral Type A

In the limiting case the interface intercept $h_2 = 1.0$ and can still continue to provide a quadrilateral type A interface polygon.

$$h_2 = \frac{d}{n_2}$$

In the limiting case $h_2 = 1.0$, so

$$d = n_2$$

Substituting this value of d in the expression

$$f = (d^3/6n_1n_2n_3)[1 - \{((d/n_1) - 1)/(d/n_1)\}^3]$$

Since the volume fraction here must be more than that found in the triangular interface polygon, the interface constraint for a quadrilateral type A interface polygon is given by,

$$n_1^3 < 6n_1n_2n_3f < n_2^3 - (n_2 - n_1)^3$$

Note, the interface shape has been declared only knowing the values of n_1, n_2, n_3 and volume fraction of the donor. Determination of the interface distance d is provided by an explicit expression, i.e., the solution of an algebraic equation.

5.7 The Pentagonal Interface Polygon

Figure 5.8: The pentagonal interface polygon.

For the pentagonal interface polygon shown in Figure 5.8, the interface intercepts h_1 , h_2 and h_3 obey the following constraints.

- h_1 , the intercept on the x_1 axis is $\gg 1.0$
- h_2 , the intercept on the x_2 axis is $\gg 1.0$

- h_3 , the intercept on the x_3 axis is ≤ 1.0

The strategy for computing the volume of liquid below the pentagonal interface polygon is the same as that was used is the quadrilateral type A. The difference is in the presence of 2 little tetrahedrons outside the unit cell for a pentagonal interface as opposed to one for the quadrilateral type A interface. For the pentagonal polygon we directly write the volume of liquid under the interface polygon f as ,

$$f = \frac{1}{6}(h_1 h_2 h_3) \left\{ 1 - \left(\frac{h_1 - 1}{h_1} \right)^3 - \left(\frac{h_2 - 1}{h_2} \right)^3 \right\}$$

Now substitute for h_1 , h_2 and h_3 as,

$$h_1 = \frac{d}{n_1}; \quad h_2 = \frac{d}{n_2}; \quad h_3 = \frac{d}{n_3}$$

so,

$$f = \frac{1}{6n_1 n_2 n_3} \{ d^3 - (d - n_1)^3 - (d - n_2)^3 \}$$

We denote this cubic polynomial by $f_{nct_{PLICpentagon}}$

$f_{nct_{PLICpentagon}} = \frac{1}{6n_1 n_2 n_3} \{ d^3 - (d - n_1)^3 - (d - n_2)^3 \} - f = 0.0$

5.7.1 Interface Constraint for a Pentagonal Interface

Polygon

For a pentagonal interface polygon the upper bound on h_3 is $h_3 = 1.0$

$$h_3 = \frac{d}{n_3}$$

In the limiting case $h_3 = 1.0$, so $d = n_3$

Substituting this value of d in the expression

$$f = \frac{1}{6n_1n_2n_3} \{d^3 - (d - n_1)^3 - (d - n_2)^3\}$$

The interface constraint developed is

$$6n_1n_2n_3f < \{n_3^3 - (n_3 - n_1)^3 - (n_3 - n_2)^3\}$$

Note that the volume of liquid under a pentagonal interface polygon must be more than that under a quadrilateral type A interface polygon. This leads to a lower bound constraint for the pentagonal interface. The final constraint takes the form,

$$\boxed{\{n_1 + n_2 > n_3\} \text{ and } \{n_2^3 - (n_2 - n_1)^3\} < 6n_1n_2n_3f < \{n_3^3 - (n_3 - n_1)^3 - (n_3 - n_2)^3\}}$$

The value of d , the interface distance, is computed using the Newton-Raphson algorithm. This is a non-linear problem. However experience shows that only 3 to 5 iterations are required for Newton-Raphson to reach a convergence of 10^{-12} . A detailed explanation of the Newton-Raphson algorithm has been provided after discussion on the hexagonal interface section. The hexagonal interface section also uses the Newton-Raphson algorithm to compute the interface distance d .

5.8 The Hexagonal Interface Polygon

Figure 5.9: The hexagonal interface polygon.

For the hexagonal interface polygon shown in Figure 5.9, the interface intercepts h_1 , h_2 and h_3 obey the following constraints.

- h_1 , the intercept on the x_1 axis is $\gg 1.0$
- h_2 , the intercept on the x_2 axis is $\gg 1.0$

- h_3 , the intercept on the x_3 axis is $\gg 1.0$

The strategy of computing the volume of liquid below the interface polygon is similar to that was used in the quadrilateral type A. The difference is in the presence of 3 little tetrahedrons outside the unit cell for a hexagonal interface as opposed to one for the quadrilateral type A interface section and two for the pentagonal interface polygon. For the hexagonal section we directly write the volume of liquid under the interface polygon f as,

$$f = \frac{1}{6}(h_1 h_2 h_3) \left\{ 1 - \left(\frac{h_1 - 1}{h_1} \right)^3 - \left(\frac{h_2 - 1}{h_2} \right)^3 - \left(\frac{h_3 - 1}{h_3} \right)^3 \right\}$$

Now substitute for h_1 , h_2 and h_3 as,

$$h_1 = \frac{d}{n_1}, h_2 = \frac{d}{n_2}, h_3 = \frac{d}{n_3}$$

so,

$$f = \frac{1}{6n_1 n_2 n_3} \{ d^3 - (d - n_1)^3 - (d - n_2)^3 - (d - n_3)^3 \}$$

We denote this cubic polynomial by $f_{nct_{PLIHexagon}}$

$f_{nct_{PLIHexagon}} = \frac{1}{6n_1 n_2 n_3} \{ d^3 - (d - n_1)^3 - (d - n_2)^3 - (d - n_3)^3 \} - f = 0.0$

The value of d , the interface distance is computed by Newton-Raphson root finding algorithm. This is a non-linear problem. However as seen in the pentagonal interface polygon type 3 to 5 iterations are required for Newton-Raphson to reach a convergence of 10^{-12} .

5.8.1 Interface Constraint for a Hexagonal Interface Polygon

The value of h_3 has to be $\gg 1.0$ for the interface polygon to be hexagonal in shape. The volume of liquid under the hexagonal interface polygon has to be greater than that obtained with $h_3 = 1.0$.

$$h_3 = \frac{d}{n_3}$$

Set $h_3 = 1.0$, so $d = n_3$

Substituting this value of d in the expression

$$f = \frac{1}{6n_1n_2n_3} \{d^3 - (d - n_1)^3 - (d - n_2)^3 - (d - n_3)^3\}$$

The interface constraint developed is,

$$6n_1n_2n_3f > \{n_3^3 - (n_3 - n_1)^3 - (n_3 - n_2)^3\}$$

Note the $>$ sign here, for the pentagonal was $<$.

The final interface constraint takes the form,

$$\{n_1 + n_2 > n_3\} \text{ and } \{n_2^3 - (n_2 - n_1)^3\} > 6n_1n_2n_3f < \{n_3^3 - (n_3 - n_1)^3 - (n_3 - n_2)^3\}$$

5.9 The Quadrilateral Type B Interface Polygon

Figure 5.10: The quadrilateral type B interface polygon.

The equation of the planar interface shown in Figure 5.10 in the (x_1, x_2, x_3) space is,

$$n_1x_1 + n_2x_2 + n_3x_3 = d$$

Note that this equation is true for all interface types. It follows that every point on the interface polygon satisfies the equation.

The interface intersecting all the vertical edges of the unit cube is a characteristic feature of such an interface. The co-ordinates of point A are given by $(1, 0, p_1)$. Now substitute this point in the equation of the plane to obtain a relation

$$n_1 + p_1 n_3 = d$$

Similarly substituting the co-ordinates of points C, D and E obtain

$$n_1 + n_2 + p_2 n_3 = d;$$

$$n_2 + p_3 n_3 = d;$$

and

$$p_4 n_3 = d.$$

The volume of liquid f under the interface polygon is given by,

$$f = \frac{1}{4}(p_1 + p_2 + p_3 + p_4)$$

Now substitute values of (p_1, p_2, p_3, p_4) in the expression for f .

$$f = \frac{1}{4n_3}(d - n_1 + d - n_1 - n_2 + d - n_2 + d)$$

$$f = \frac{1}{2n_3}(2d - (n_1 + n_2))$$

$$2d = 2n_3f + (n_1 + n_2)$$

this gives,

$$\boxed{d = n_3f + \frac{1}{2}(n_1 + n_2)}$$

5.9.1 Interface Constraint for a Quadrilateral Type B

Interface Polygon

Since $p_2 > 0$, we find a lower bound if we set $p_2 = 0$.

This implies that $n_1 + n_2 = d$

First write out the volume of liquid as the volume of the large tetrahedron minus the volume of the two little tetrahedrons that protrude from the cube.

$$f = \frac{h_1 h_2 h_3}{6} - \frac{(h_1 - 1)^{\frac{d-n_1}{n_3}}}{6} - \frac{(h_2 - 1)^{\frac{d-n_2}{n_3}}}{6}$$

$$6f = h_1 h_2 h_3 + \frac{(h_1 - 1)(d - n_1)}{n_3} + \frac{(h_2 - 1)(d - n_2)}{n_3}.$$

$$6f = \frac{d^3}{(n_1 n_2 n_3)} + \frac{(h_1 - 1)(d - n_1)}{n_3} + \frac{(h_2 - 1)(d - n_2)}{n_3}.$$

Now assume that $p_2 = 0$, then $(n_1 + n_2) = d$.

$$6f > \frac{(n_1 + n_2)^3}{(6n_1 n_2 n_3)} + \frac{\{\frac{d}{n_1} - \frac{n_1}{n_1}\}(d - n_1)}{n_3} + \frac{\{\frac{d}{n_2} - \frac{n_2}{n_2}\}(d - n_2)}{n_3}.$$

$$6f > \frac{(n_1 + n_2)^3}{(6n_1 n_2 n_3)} + \left\{ \frac{(n_1 + n_2)}{n_1} - \frac{n_1}{n_1} \right\} \frac{(n_1 + n_2) - n_1}{n_3} +$$

$$\left\{ \frac{(n_1 + n_2)}{n_2} - \frac{n_2}{n_2} \right\} \frac{(n_1 + n_2) - n_2}{n_3}.$$

Simplifying, we obtain,

$$2(n_1)^3 + 3(n_1)^2 n_2 + 3n_1(n_2)^2 + \frac{2(n_2)^3}{(n_1 n_2 n_3)}.$$

The above can be written in the form,

$$6n_1 n_2 n_3 f > (n_1 + n_2)^3 - n_1^3 - n_2^3.$$

It can also be expressed as,

$$2n_3f > n_1 + n_2.$$

The final interface constraint takes the form,

$$\boxed{\{n_1 + n_2 < n_3\} \text{ and } 2n_3f > n_1 + n_2.}$$

This concludes discussion on the determination of the interface distance and the interface constraint for all five interface types. In the next section the Newton-Raphson root finding algorithm is discussed.

5.10 Newton-Raphson Root Finding Algorithm

In the case of the pentagonal and hexagonal interface polygon a cubic equation has to be solved in order to compute the interface distance d . The Newton-Raphson algorithm is used to do this. We first explain the method, then the reason for choosing this method and finally outline the steps for implementing it for the two specific polygonal shapes.

The Newton-Raphson method requires the evaluation of the function $f(x)$ and the derivative $f'(x)$. The method extends the tangent at a current point x_i until it crosses zero, then setting the next guess x_{i+1} to the abscissa of that zero crossing. Algebraically, the method derives a Taylor series expansion for a function in the neighborhood of a point,

$$f(x + \delta) \approx f(x) = f'(x)\delta + \frac{f''(x)}{2}\delta^2 + \dots$$

For small enough values of δ and well behaved functions, the terms beyond the second order term in the expansion given above are ignored because their contribution is insignificant, hence $f(x + \delta) = 0$ implies,

$$\delta = -\frac{f(x)}{f'(x)}.$$

To summarize, the Newton-Raphson root finding scheme extrapolates the

Figure 5.11: Extrapolation of a local derivative to compute the next guess of the root by the Newton-Raphson algorithm.

local derivative to find the next estimate of the root [13].

For the quizzical mind, a certain question would be, why use the Newton-Raphson algorithm?

Some of the popular root finding schemes are:

- Simple Bracketing and Bisection.
- Secant Method and False Position method.
- Newton-Raphson method.

The Newton-Raphson root finding scheme has a *quadratic* rate of convergence. This means that the number of guesses or iterations that are needed to get close to the real root are fewer than those needed for the other methods cited above. By close, we mean for an acceptable error of 10^{-10} between the real root and the computed root. Let the error be η and the root of $f(x)$ be x . Note that $(x + \eta)$ is in the close neighborhood of x . For the function $f(x + \eta)$ a Taylor series expansion is,

$$f(x + \eta) = f(x) + \eta f'(x) + \eta^2 \frac{f''(x)}{2} + \dots$$

the derivative of the function above is

$$f'(x + \eta) = f'(x) + \eta f''(x) + \dots$$

By the Newton Raphson algorithm,

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)},$$

so that,

$$\eta_{i+1} = \eta_i - \frac{f(x_i)}{f'(x_i)}.$$

When a trial solution x_i differs from the true root by η_i , we can use the expressions for $f(x + \eta)$ and $f'(x + \eta)$ to express $f(x_i)$ and $f'(x_i)$ in terms of

η_i and the derivatives at the root itself. The result is a recurrence relation for the derivatives of the trial solutions

$$\eta_i = -\eta^2 \frac{f''(x)}{2f'(x)}.$$

5.10.1 Implementation of Newton-Raphson

We recall that the functions developed for the pentagonal and hexagonal interface polygons were as follows,

$$f_{nct_{PLICpentagon}} = \frac{1}{6n_1n_2n_3} \{d^3 - (d - n_1)^3 - (d - n_2)^3\} - f = 0.0$$

and,

$$f_{nct_{PLIChexagon}} = \frac{1}{6n_1n_2n_3} \{d^3 - (d - n_1)^3 - (d - n_2)^3 - (d - n_3)^3\} - f = 0.0$$

The requirement of the root returned for the two functions above is that the volume fraction of the computational cell is conserved. The convergence criteria is set at 10^{-10} . This is a measure of the accuracy of what is accepted as a distance that conserves mass in the mixed cell. The derivatives of the functions w.r.t to the interface distance d are,

$$\frac{\partial f_{nct_{PLICpentagon}}}{\partial d} = \frac{d^2 - (d - n_1)^2 - (d - n_2)^2}{2n_1n_2n_3}$$

$$\frac{\partial f_{nct_{PLIC_{hexagon}}}}{\partial d} = \frac{d^2 - (d - n_1)^2 - (d - n_2)^2 - (d - n_3)^2}{2n_1n_2n_3}$$

The initial guess of distance for the pentagonal interface polygon is n_2 and that for a hexagonal interface is n_3 . For each new value of d the functions and their derivatives are evaluated. The new guess for distance is given by Newton-Raphson as,

$$d_{GuessNew} = d_{GuessOld} - \frac{f_{nct_{PLIC_{GuessOld}}}}{\frac{\partial f_{nct_{PLIC_{GuessOld}}}}{\partial d}}$$

From this obtain a new quantity *deltaDistance*, as,

$$deltaDistance = abs(\frac{d_{GuessNew} - d_{GuessOld}}{d_{GuessNew}})$$

Continue iterations till *deltaDistance* becomes smaller than 10^{-10} . The last value of $d_{GuessNew}$ is the interface distance.

Chapter 6

Flux of Liquid to Advect

In Figure 6.1 on page 72 the cell under consideration is the core element in the 26 element neighbourhood, the thirteenth element counting from zero, see Figure 4.1 on page 18. The exit face is shown. Let the volume under the pentagonal interface polygon (chosen as an example) ABCDE be V . Let the volume of liquid that flows from the donor, cell index 13, to the acceptor, cell index 14, be δV . This is the volume under the polygon labeled as FCDG. For now we assume that δV is known because we want to define a third quantity ϵ . This quantity ϵ is the length of the slice box. A slice box is a subset of the donor cell.

So,

- V = volume of cell 13.
- δV = volume of liquid advected from cell 13 to cell 14.
- ϵ = slice box length $\delta V/V$.

Figure 6.1: Flux of liquid that flows across an advecting face.

The next job is to determine the volume of liquid in the slice box δV . The efficiency and robustness of the methods employed in computing the volume of liquid that is advected across a cell side will eventually determine the correctness and consequent results produced by this algorithm.

We recall that one of five interface types had to be dealt with when the interface distance was computed. In this thesis the interface type can be *forgotten* once the interface distance has been computed because a method that computes the volume of liquid under the slice box polygon independent of the interface type has been implemented. The method draws inspiration from the author's involvement in Piecewise Parabolic Interface Computation where he integrated the volume under six node triangles. In the PLIC algorithm we implement three node triangles. The algorithm is summarized below.

6.0.2 Summary of Volume Integration with 3 Node Triangles

- Compute the exit face.
- Compute the equation of the interface.

- Determine the bounding box of the slice box.
- Compute intersections of the interface with the slice box edges.
- Assemble the interface polygon in the slice box.
- Mesh the slice box polygon into 3 node triangles.
- Compute the volume under each 3 node triangle.

It is possible that all terms may not be understood by the reader at this time. However, in forthcoming sections we will deal with each part of the algorithm in detail. The purpose of providing the summary is to provide a quick overview of the algorithm.

6.1 Determination of Exit Face

Figure 6.2: A rectangular element can advect on six faces.

In Figure 6.2 it has been shown that an element can advect on any of its six faces. The interface construction was done in the (x_1, x_2, x_3) co-ordinate system. So the exit face has to be determined in the (x_1, x_2, x_3) co-ordinate system. The exit face belongs to the set,

$$ExitFace \in \{x_{1min}, x_{1max}, x_{2min}, x_{2max}, x_{3min}, x_{3max}\}.$$

We recall that the values of volume fractions in the 27 element stencil were transformed to ensure that the advection direction was the $\hat{x}$ direction. Also from the 8 node numbering map, Figure 5.1 we know that the sum of the node numbers that were members of the face diagonals for the $\hat{x}_{min}$ and $\hat{x}_{max}$ is 5 or 13 respectively. This tells us that the only integers involved in determining the exit face in the (x_1, x_2, x_3) co-ordinate system are 5 and 13.

The interface distance was measured from a fixed point (0,0,0) in the (x_1, x_2, x_3) co-ordinate system.

Establishing the (x_1, x_2, x_3) local co-ordinate system did not help in fixing the point (0,0,0), it is the determination of the exit face where the use of the local co-ordinate system in terms of the associated node numbers in the $(\hat{x}, \hat{y}, \hat{z})$ co-ordinate system is made.

The local co-ordinate system was described by four node numbers m_0, m_1, m_2, m_4 . The origin was at m_0 and x_1, x_2 and x_3 are vectors from m_0 to m_1, m_2 and m_4 . respectively. We now set up pairs of node numbers that we sum. The pairs are,

$$\{m_1 + m_2\}, \{m_2 + m_4\}, \{m_1 + m_4\}.$$

The six combinations of the vertex node numbers that determine an exit face

Pair	Sum Pair	Exit Face
$m_2 + m_4$	5	$x1_{min}$
$m_1 + m_4$	5	$x2_{min}$
$m_1 + m_2$	5	$x3_{min}$
$m_2 + m_4$	13	$x1_{max}$
$m_1 + m_4$	13	$x2_{max}$
$m_1 + m_2$	13	$x3_{max}$

Table 6.1: Exit face and sum pair diagonal vertex indices on exit face.

are presented in Table 6.1

6.1.1 Implementation Strategy for Exit Face

The implementation of the idea expressed above is done as follows.

The sum $m_2 + m_4$ is associated with exit faces on the $x1$ axis. The sum $m_1 + m_4$ is associated with exit faces on the $x2$ axis. The sum $m_1 + m_2$ is associated with exit faces on the $x3$ axis.

Now set up a list,

$$[\{m_1 + m_2\}, \{m_2 + m_4\}, \{m_1 + m_4\}].$$

Figure 6.3: Exit face id map.

The only job that remains to be done is to find out in which position one is able to find a 5 or a 13. Note 5 and 13 cannot co-exist, also either 5 or 13 has to exist in the list mentioned above. 5 can exist at indices that belong to the set $\{0, 1, 2\}$. Similarly 13 can exist at indices that belong to the set $\{0, 1, 2\}$. It is not sufficient to know that 5 or 13 exist but one must also determine their positions. To do this add the index at which 5 or 13 exist

to 5 or 13. This gives the six possible choices of exit faces. So once again we say,

$$ExitFace \in \{5, 6, 7, 13, 14, 15\}$$

To make it easy to deal with the integers computed above, re-map them to a set,

$$ExitFace \in \{0, 1, 2, 3, 4, 5\},$$

The map simply says that if the computed exit face is 14 return 4. This re-mapping was done to ensure uniformity in coding and to establish consistency with conventions and in this thesis for denoting the 6 faces of a rectangular element. The map is shown in Figure 6.3.

The determination of the exit face did not require information on the interface polygon.

6.2 Volume Integration

The algorithm discussed here evaluates the volume of liquid in the slice box. All computation is done in the bounding box of the slice box because it is only the volume of liquid in the slice box that gets advected from a donor cell to an acceptor cell.

6.2.1 Equation of the Planar Interface

The equation of a plane in a general co-ordinate system (x, y, z) is given by the expression,

$$a_0x + a_1y + a_2z = d$$

In the (x_1, x_2, x_3) co-ordinate system, which is the co-ordinate system for interface construction and determination of the exit face, the equation of the plane is,

$$n_1x_1 + n_2x_2 + n_3x_3 = d$$

Among (n_1, n_2, n_3) , n_3 is chosen to be the maximum. This aids in writing out the equation of the plane as,

$$x_3 = \frac{\{d - (n_1 x_1) - (n_2 x_2)\}}{n_3}$$

We parameterize the equation above and rewrite it as,

$$x_3 = \{a_0 + (a_1 x_1) + (a_2 x_2)\}$$

This is done because in the local co-ordinate system (x_1, x_2, x_3) , n_3 is most perpendicular to the interface. It follows that the x_1, x_2 plane can be treated as a projection plane for the interface.

The values of n_1, n_2, n_3 are known. The interface distance d is also known. The constants a_0, a_1 and a_2 are computed as follows,

$$a_0 = d/n_3$$

$$a_1 = -n_1/n_3$$

$$a_2 = -n_2/n_3$$

6.2.2 Edges of the Unit Cell and the Slice Box

Figure 6.4: Edge numbering map.

A map for numbering the 12 edges of a unit cell is shown in Figure 6.4. The same edge map is used when a slice box is made in the unit cell.

For each edge the values of the co-ordinates of points that may lie on the edge are given in the Table 6.2 on page 83. The subscript $_u$ indicates the co-ordinate that is unknown along that edge. For each point two co-ordinates are known and one is unknown.

Edge Number	$x1 \text{ co} - \text{ord}'$	$x2 \text{ co} - \text{ord}'$	$x3 \text{ co} - \text{ord}'$
Edge 0	$x1_u$	$x2_{min}$	$x3_{min}$
Edge 1	$x1_{max}$	$x2_u$	$x3_{min}$
Edge 2	$x1_u$	$x2_{max}$	$x3_{min}$
Edge 3	$x1_{min}$	$x2_u$	$x3_{min}$
Edge 4	$x1_u$	$x2_{min}$	$x3_{max}$
Edge 5	$x1_{max}$	$x2_u$	$x3_{max}$
Edge 6	$x1_u$	$x2_{max}$	$x3_{max}$
Edge 7	$x1_{min}$	$x2_u$	$x3_{max}$
Edge 8	$x1_{max}$	$x2_{min}$	$x3_u$
Edge 9	$x1_{max}$	$x2_{max}$	$x3_u$
Edge 10	$x1_{min}$	$x2_{max}$	$x3_u$
Edge 11	$x1_{min}$	$x2_{min}$	$x3_u$

Table 6.2: Co-ordinates of points on 12 edges of a unit cell.

6.2.3 Bounding Box of the Slice Box

Let the length of the slice box be ϵ . The exit face computed previously is needed to decide the dimensions of the bounding box of the slice box. In

Figure 6.5 we show how ϵ is measured for a given exit face.

Figure 6.5: Measurement of ϵ to make a slice box on an exit face.

An easy method of understanding the measurement of ϵ is described in two steps below.

- Imagine that one is standing on the exit face.
- Walk a distance ϵ from the exit face towards the opposite face. The slice box must be inside the unit cell.

For each slice box Table 6.3 provides the bounding box of a slice box associated with an exit face.

Exit Face	Min co-ords' Slice Box	Max co-ords' Slice Box
0	$\{(1 - \epsilon), 0, 0\}$	$\{1, 1, 1\}$
1	$\{0, 0, 0\}$	$\{\epsilon, 1, 1\}$
2	$\{0, (1 - \epsilon), 0\}$	$\{1, 1, 1\}$
3	$\{0, 0, 0\}$	$\{1, \epsilon, 1\}$
4	$\{0, 0, (1 - \epsilon)\}$	$\{1, 1, 1\}$
5	$\{0, 0, 0\}$	$\{1, 1, \epsilon\}$

Table 6.3: Minimum and maximum co-ordinates of slice boxes on exit faces.

The exit face numbering scheme is shown again in Figure 6.6 on page 86.

We recall that this is a re-map from

$$ExitFace \in \{5, 6, 7, 13, 14, 15\}$$

to

$$ExitFace \in \{0, 1, 2, 3, 4, 5\}$$

Figure 6.6: Face numbering map.

The face 0 is not shown in Figure 6.6. It is the one that is not labeled. The dotted lines are drawn for ease of making an association between the face number and the face that it belongs to.

Note that Table 6.3 on page 85 is not in a usable form yet. The bounding boxes for the 0, 2 and 4 faces are right. However for reasons that will be made clear soon the bounding boxes for the 1, 3 and 5 faces will be modified to be the complements of the slice boxes mentioned in Table 6.3.

6.2.4 Actual Slice Box and the Complement Slice Box

In a unit cell if a slice box has length ϵ the complement of this slice box has length $(1 - \epsilon)$. See Figure 6.7

Figure 6.7: The actual slice box and complement slice box.

Consider a case where the exit face is 5. We choose the interface polygon to be a pentagon and show graphically in Figure 6.8 on page 88, the volume of liquid in the actual slice box and the volume of liquid in the complement of the actual slice box.

The polyhedron in the actual slice box associated with the exit face as shown in the figure can always be decomposed with three node triangles but it would need specialized treatment. The specialized treatment is the requirement to set up *sub - lists* of points extracted by computing the intersections

Figure 6.8: Volume of liquid in the actual and complement slice box.

of the slice box with the equation of the plane. The tessellation algorithm that will be discussed in the next section is designed to provide points that directly make three node triangles. The three node triangles will be used to evaluate the volume of the prism so produced *exactly*.

It is desired that a set of points form three node triangles that can be directly used to evaluate the volume of the associated polyhedron. This is done by maintaining a consistent volume integration method. We choose to evaluate volumes of convex triangular prisms. This is done by computing the volume of liquid in the complement of the actual slice box when the intersection points in the actual slice box do not readily decompose into a

set of convex triangular prisms. This treatment is required for the faces $\{1, 3, 5\}$. Note that these are the *min* faces and have been treated in the same fashion. So if an element wants to advect on a *min* face we work in the complement of its actual slice box to compute the volume of liquid in the actual slice box.

Interestingly, is it possible to manage the case stated above for planar interfaces, but if one encounters quadratic interfaces as seen in Piecewise Parabolic Interface Computation a sophisticated tessellation algorithm, with conversion from three node to six node triangles and more complex isoparametric volume integration is needed. In the chapter 9 where we describe initialization of volume fractions for a sphere using a second order equation of the sphere this algorithm will be briefly explained.

Table 6.4 on page 90 provides the bounding boxes of the actual slice boxes for the *max* and the complements of the actual slice boxes for the *min* faces.

Note that the max co-ordinate of the slice box for all the six faces is $(1, 1, 1)$. Also the min co-ordinate of the slice box is the only the co-ordinate of the advecting face index that is non-zero and the other two are zero. For faces $\{0, 2, 4\}$ the non-zero co-ordinate is the same and it is ϵ . Similarly, for faces $\{1, 3, 5\}$ the non-zero co-ordinate is the same and it is $(1 - \epsilon)$. This

Exit Face	Min co-ords' Slice Box	Max co-ords' Slice Box
0	$\{(1 - \epsilon), 0, 0\}$	$\{1, 1, 1\}$
1	$\{\epsilon, 0, 0\}$	$\{1, 1, 1\}$
2	$\{0, (1 - \epsilon), 0\}$	$\{1, 1, 1\}$
3	$\{0, \epsilon, 0\}$	$\{1, 1, 1\}$
4	$\{0, 0, (1 - \epsilon)\}$	$\{1, 1, 1\}$
5	$\{0, 0, \epsilon\}$	$\{1, 1, 1\}$

Table 6.4: Minimum and maximum co-ordinates of slice boxes.

leads to simpler code.

In the authors opinion a computer program that sets out on a *treasure hunt* is a program that certainly can be made simpler. The term *treasure hunt* is a metaphor for a computer program that has to fight through a maze of conditional clauses to produce an answer. A computer program once written should flow from one logical procedure to another without being hindered by too many conditional clauses. The design and conceptualization of a well abstracted computer program has to be first desired before it can be implemented. Writing simple code that is resilient to mathematical complexity

has been a learning experience for the author.

6.2.5 Intersections with the Slice Box

Given that the equation of the interface polygonal plane is,

$$x_3 = a_0 + (a_1x_1) + (a_2x_2)$$

where,

$$a_0 = d/n_3; \quad a_1 = -n_1/n_3; \quad a_2 = -n_2/n_3$$

We now compute the intersection of this plane with the slice box. To do this we set up 3 equations. The first equation is used for the case where x_2 and x_3 are known, the second equation is used where x_1 and x_3 are known and lastly the third equation is used where x_1 and x_2 are known, see Table 6.2 on page 83.

- Case x_2 and x_3 known and x_1 unknown.

$$x_{1_u} = \frac{\{x_3 - a_0 - (a_2x_2)\}}{a_1}$$

- Case x_1 and x_3 known and x_2 unknown.

$$x_{2_u} = \frac{\{x_3 - a_0 - (a_1x_1)\}}{a_2}$$

- Case x_1 and x_2 known and x_3 unknown.

$$x_{3_u} = \{a_0 + (a_1x_1) + (a_2x_2)\}$$

The three equations stated above are obtained from rearranging the equation of the plane.

We are only interested in computing the polygon in the slice box. So intersection points outside the slice box are not needed. We now convert a problem that evaluated intersections of an infinite plane with an infinite line to a problem that evaluates intersections of an infinite plane with a line segment. The bounds for the line segment are the associated *min* and *max* co-ordinates of the computational slice box.

- The conditions for accepting a root when the x_1 co-ordinate is unknown are:

$$x_1 > x_{minSliceBox} \text{ and } x_1 < x_{maxSliceBox}$$

- The conditions for accepting a root when the x_2 co-ordinate is unknown are:

$$x_2 > x_{2minSliceBox} \text{ and } x_2 < x_{2maxSliceBox}$$

- The conditions for accepting a root when the x_3 co-ordinate is unknown are:

$$x_3 \geq x_{3_{minSliceBox}} \text{ and } x_3 \leq x_{3_{maxSliceBox}}$$

The last condition has the $\geq$ and $\leq$ signs because we allow the root to lie on the vertex when the x_3 co-ordinate is unknown. A vertex is shared by three edges on an eight node brick. If a root exists at a vertex then it finds membership in three edges. The same point can exist on three edges, to filter this point, the vertex, we constrain the roots to lie between the vertices for all edges except the vertical edges. On the vertical edges we allow the root to exist on the vertices. If the unknown co-ordinate is accepted store the x_1, x_2 and the edge number. Note we do not recommend storing x_3 . This is because the projection of a point (x_{1p}, x_{2p}, x_{3p}) on the projection plane x_1, x_2 is (x_{1p}, x_{2p}) .

It is always possible to filter the duplicate root that lies on a vertex but if the signs are implemented in the manner stated above we argue that *prevention is better than cure*.

At this point legal intersection points have been collected in the order of the edges. The next job is to re-order them so that a counter clockwise

closed polygon is obtained. We call this operation as polygon assembly.

6.2.6 Polygon Assembly

Figure 6.9: Projection map for polygon assembly.

The Figure 6.9 above shows a configuration that prepares 4 sublists to develop the slice box polygon. The edge number touches the associated edge. There is no rule about choosing such a configuration, but the requirement is that a

configuration must exist and that it be maintained consistently in the logic.

Note that Figure 6.4 on page 82 used the same edge map.

We now make 4 subs-lists, $sublist_1 - 4$

- $sublist_1$ has intersection points that lie on edges 1, 9, 5, 8.
- $sublist_2$ has intersection points that lie on edges 2, 6.
- $sublist_3$ has intersection points that lie on edges 3, 10, 7, 11
- $sublist_4$ has intersection points that lie on edges 4, 0.

We now invoke 4 different sorts on the 4 sublists just made.

- For $sublist_1$ sort all intersection points with increasing x_2 co-ordinate.
- For $sublist_2$ sort all intersection points with decreasing x_1 co-ordinate.
- For $sublist_3$ sort all intersection points with decreasing x_2 co-ordinate.
- For $sublist_4$ sort all intersection points with increasing x_1 co-ordinate.

Assemble the slice box polygon by concatenating the sorted intersection points in the order

$$sliceBoxPolygon \in \{sublist_1, sublist_2, sublist_3, sublist_4\}$$

The intersection points that live in the list *sliceBoxPolygon* are a set of points that have been ordered in the counter clockwise sense. In this way we obtain the slice box polygon. Note that in doing intersections of the planar interface and polygon assembly no information about the shape of the interface polygon was needed. The method of doing intersections and polygon assembly is a generic method that can be applied not just for planar slice box polygons but for quadratic slice box *surfaces* as well.

Interestingly, for the quizzical mind we make a mental note that a planar interface can intersect a given edge at only one point. A quadratic surface on the other hand is not as forgiving because two intersections on the same edge are possible. This property of intersections of lines and planes is the basis for developing a simple and computationally efficient tessellation algorithm.

In the Figure 6.9 on page 94, the bold dots indicate the intersections of the pentagonal interface that has been used as a standard interface example in this thesis.

6.2.7 Slice Box Polygon Tessellation

A slice box polygon is defined by a set of ordered points. The ordering is in the counter clockwise sense. At this point we introduce the concept of a convex polygon. The book Graphics Gems provides an excellent exposition on this issue in a “Gem” (they refer to all algorithms in the book as gems) by [10]. A set of points S is convex if for any line drawn between two points in the set S , then all points on the line segment are also in the set. Examples of a convex and non-convex polygons are as shown in Figure 6.10.

Figure 6.10: Examples of a convex and a non-convex polygon.

If the polygon in the slice box is projected on a projection plane which is the (x_1, x_2) plane in our case, it is always a convex polygon. The tessellation algorithm for a convex polygon is simple.

Figure 6.11: Meshing the slice box polygon into three node triangles.

Let the first point in the slice box polygon be a node for all the other triangles. In this way, if the slice box polygon has five points we make three triangles. In this manner we project the intersection points on the (x_1, x_2) projection plane and tessellate the slice box polygon. The Figure 6.11 above illustrates this.

6.2.8 Volume of Material Under a Three Node Triangle

A three node triangle is projected on a projection plane and the resulting solid is as can be seen in Figure 6.12.

Figure 6.12: Volume integration using one point integration. The height from the projection plane to the centroid of the triangle is H .

The volume under a three node triangle by *one point integration* is,

$volume_{3NodeTriangle} = (\text{area of the projected triangle}) \times (\text{height of the three node triangle above the centroid of the projected triangle})$

The volume computed above is exact.

The area of any triangle with vertices K, L, M is given by the determinant

$$area_{triangle} = \frac{1}{2} \begin{pmatrix} (x1_L - x1_K) & (x1_M - x1_K) \\ (x2_L - x2_K) & (x2_M - x2_K) \end{pmatrix}$$

This is the method used for computing the area of a triangle in two dimensions. The determinant comes from the cross product of two vectors and the 1/2 is half of the area of the parallelogram produced by the cross product.

The centroid in 2 dimensions for any triangle with vertices K, L, M is given by

$$x1_{centroid} = \frac{x1_L + x1_K + x1_M}{3}$$

$$x2_{centroid} = \frac{x2_L + x2_K + x2_M}{3}$$

The height of the interface at the centroid is computed by solving

$$interfaceHeight_{(x_{centroid}, y_{centroid})} = \{(a0 + (a1x1_{centroid}) + (a2x2_{centroid})) - x3_{minSliceBox}\}$$

6.2.9 Volume Of Material In the Slice Box

If the steps that compute the volume under a three node triangle are repeated for each three node triangle in the tessellation and the individual volumes are summed, we have an exact answer for the volume of material in the slice box. We choose to be generic about the name of the material at this point

because in some cases we compute the volume of liquid in the slice box and in other cases we compute the volume of air in the slice box.

6.2.10 Managing the Little Triangle

As shown in Figure 6.13 the hexagonal interface has a little triangle on the top surface that is shown. This triangle arises because $h3 > 1.0$ in the unit cell. See Figure 5.9 on page 57. The volume integration technique is the same as explained before. This little triangle might or might not occur in the slice box.

In Figure 6.13 on page 102, see that 2 points on the little triangle are on the edges 4 and 7. Edges 4 and 7 are defined on the map in Figure 6.4 on page 82. The third point has to be determined. This is done by examining the exit face. We recall that the exit face is one of the integers from 0 to 5. Let the intersection points on edges 4 and 7 after projection be denoted by $(x1_4, x2_4)$ and $(x1_7, x2_7)$ respectively. Let the third point be $(x1_{littleTriangle}, x2_{littleTriangle})$. Table 6.5 lists them for each exit face.

Note that for exit faces $\{1, 3, 5\}$ the co-ordinates given above are for the complement of the actual slice box.

Figure 6.13: The little triangle on edge numbers 4 and 7.

Once again we have a well posed three node triangle with co-ordinates

$$(x1_4, x2_4), (x1_7, x2_7), (x1_{littleTriangle}, x2_{littleTriangle})$$

The *min* and the *max* co-ordinates of the slice box are also known, they are shown in Table 6.4 on page 90.

The interface height for the little triangle is computed as follows,

Exit Face	$x1_{littleTriangle}$	$x2_{littleTriangle}$
0	$(1 - \epsilon)$	0.0
1	ϵ	0.0
2	0.0	$(1 - \epsilon)$
3	0.0	ϵ
4	0.0	0.0
5	0.0	0.0

Table 6.5: Co-ordinates of the third point on the little triangle.

$$interfaceHeight_{littleTriangle} = x3_{maxSliceBox} - x3_{minSliceBox}$$

It is true that the little triangle only exists for the hexagonal interface but again we recommend that the decision to add the volume under the little triangle be taken only on finding legal intersections on edges 4 and 7 and not by storing any other piece of information. Simply put, add the volume under the little triangle to the volume produced by the other three node triangles if legal intersections on edges 4 and 7 exist, else add zero. The prism produced by intersections on edges 4 and 7 is the third prism in Figure 6.12 on page

99.

This ends discussion on volume integration of the material in the slice slice box. In the next section we deal with computation of the volume of liquid in a slice box.

6.2.11 Volume of Liquid in the Slice Box

The advection algorithm needs to know the volume of liquid in the slice box because it is this volume of liquid in the slice box that gets advected from a donor cell to an acceptor cell.

Four cases are required to be considered.

- The exit faces are $\{0, 2, 4\}$ and volume fraction of the donor is less than 0.5.
- The exit faces are $\{0, 2, 4\}$ and volume fraction of the donor is greater than 0.5.
- The exit faces are $\{1, 3, 5\}$ and volume fraction of the donor is less than 0.5.
- The exit faces are $\{1, 3, 5\}$ and volume fraction of the donor is greater

than 0.5.

We recall that if the volume fraction of the donor exceeds 0.5 the volume of material in the slice box is the volume of air in the slice box. We also recall that for exit faces $\{1, 3, 5\}$ we work in the complement of the actual slice box. If a slice box has length ϵ and the computational mixed cell is a unit cell then the volume of the slice box is

$$1.0 \times 1.0 \times \epsilon = \epsilon$$

Let the volume of liquid in the slice box be denoted by $vLiquid_{sliceBox}$.

Let the volume of material in the slice box computed from three node triangles be denoted by $vMaterial_{sliceBox}$.

Let the volume fraction of the donor cell be denoted by V .

We first deal with the 6 exit faces when the volume fraction of the donor is less than 0.5.

- Exit face belongs to the set $\{0, 2, 4\}$.

$$vLiquid_{sliceBox} = vMaterial_{sliceBox}$$

- Exit face belongs to the set $\{1, 3, 5\}$.

$$vLiquid_{sliceBox} = (V - vMaterial_{sliceBox})$$

We now deal with the 6 exit faces when the volume fraction of the donor is exceeds 0.5.

- Exit face belongs to the set $\{0, 2, 4\}$.

$$vLiquid_{sliceBox} = (\epsilon - vMaterial_{sliceBox})$$

- Exit face belongs to the set $\{1, 3, 5\}$.

$$vLiquid_{sliceBox} = (V - (1.0 - \epsilon - vMaterial_{sliceBox}))$$

At this point we know the volume of *liquid* that can be advected from a donor cell to an acceptor cell.

If there is any difficulty in understanding the expressions stated above the reader could set the value of ϵ to one. It is hoped that the difficulty will vanish. Interestingly, this is a robust test because if $\epsilon = 1.0$ then,

$$vLiquid_{sliceBox} = V$$

also if $\epsilon = 0.0$ then,

$$vLiquid_{sliceBox} = 0.0$$

Note that these two simple tests deal with all cases. We also recall that V is known data.

Establishing the expressions stated above and in general developing this generic algorithm for volume integration is a source of great satisfaction and joy for the author of this thesis.

This ends discussion on volume integration. In the next chapter discussion on the law of conservation of mass that will lead to the development of a computational domain and a time integration technique.

Chapter 7

Law of Conservation of Mass

Algorithms that deal with transport of a fluid *must* conserve mass. In order to explain clearly how the law of conservation of mass is applied in the PLIC algorithm a mathematical treatment is first provided.

Let D be a region in three-dimensional space completely filled or partly filled with a fluid. The fluid has a constant density ρ . Our objective is to describe the motion of such a fluid. Let $x \in D$ be a point in D and consider the particle of fluid moving through x at time t . Let $u(x, t)$ denote the velocity of the particle of fluid that is moving through x at time t . This particle traverses a well-defined trajectory called a flow line. Thus, for each instant of time, u is a vector field on D , as in Figure 7.1 on page 109.

Figure 7.1: The flow line for a particle of fluid at point x with velocity $u(x, t)$ in region D .

For each instant of time t , assume that the fluid has a well defined mass density $\rho(x, y)$. Thus, if W is any sub-region of D , the mass of fluid in W at the instant of time t is given by,

$$m(W, t) = \int_W \rho(x, t) dV.$$

where dV is a volume element in space.

Let W be a fixed sub-region of D (W does not change with time). The rate of change of mass in W is,

$$\frac{d}{dt} m(W, t) = \frac{d}{dt} \int_W \rho(x, t) dV = \int_W \frac{\partial \rho}{\partial t}(x, t) dV.$$

Let ∂W denote the boundary of W . Let n denote the outward normal defined at points of ∂W . Let dA denote an area element on ∂W . The

volume flow rate across ∂W per unit area is $(u \cdot n)$ and the mass flow rate per unit area is $(\rho u \cdot n)$ where n is the outward normal. See Figure 7.2.

Figure 7.2: The mass flow rate across ∂W is: $\frac{d}{dt}(\text{mass crossing } \partial W) = \int_{\partial W} \rho(u \cdot n) dA$.

The principle of conservation of mass, i.e., *mass is neither created nor destroyed*, can be stated more precisely as: *the rate of increase of mass in W equals the rate at which mass is crossing ∂W in the inward direction [1]; i.e.,*

$$\frac{d}{dt} \int_W \rho dV = - \int_{\partial W} \rho(u \cdot n) dA.$$

Note, the left hand side is a volume integral and the right hand side is a surface integral. The principle of conservation of mass can be stated in words as,

volume added = inflow volume - outflow volume.

In the next two sections we define the domain D and then discuss time integration respectively.

7.1 Computational Domain for Interface Advection

All elements that have values of volume fractions greater than zero and less than one will be included in the computational domain. However, this thesis argues that a computation domain for interface advection need not include all elements that have values for volume fraction *equal* to zero or *equal* to one. We agree that volume integration for elements with volume fraction equal to zero and equal to one is trivial, but we insist that a lean domain that minimizes cpu time be used.

We begin by answering a simple question, given that operator splitting has been employed for time integration how many acceptor elements can a single donor cell update in one advection time step? For discussion purposes we allow $CN = 1.0$, this is the limiting case. For $CN = 1.0$ and one advection time step decomposed into three one dimensional sweeps a donor can only

advect into elements in its 26 element neighbourhood.

Extensive use of sets and boolean operations has been made to make a computational domain for interface advection. For each advection time step, values of volume fractions and velocity are known data. We partition the domain into three sets and they are:

`vofE0Set`, `vofE1Set`, `vof01Set` where all elements with volume fractions equal to zero belong to the `vofE0Set`, all elements with volume fractions equal to one belong to the `vofE1Set` and all elements with volume fractions greater than zero and less than one belong to the `vof01Set`.

The first step is to get a 26 element neighbourhood for each element in the `vof01Set`, let this be called `SetA`. Note that the core element, i.e., the element in the `vof01Set` has been included in the 26 element neighbourhood.

The next step is to make a set of nodes shared by the `vofE0Set` and the `vofE1Set`. For each node get the 2 elements that it shares. For each element above we fetch a 27 element neighborhood. Let this set be called `SetB`.

A union of `SetA` and `SetB` make a computational domain for interface advection. Let the union be called, `IfDomainSet`. Each element that resides in the `IfDomainSet` can update itself as a donor.

We now construct a layer of elements around the `IfDomainSet` that we

call the `InletOutletSet`. The `InletOutletSet` is a set of elements that only update their acceptors. They do not update themselves as donor elements.

It is a requirement that all acceptors be in the `IfDomainSet`.

There are issues on managing sub domains in the grid that are not addressed here but the core ideas that contribute to making a domain for interface advection have been explained.

Experience shows that boolean operations are very efficient and robust in managing complexity.

7.2 Time Integration

In order to advance the flux of liquid from a donor cell to an acceptor cell in an incompressible divergence free velocity field the equation for transport of the fluid is the equation for what is popularly known as *advection*. Given a flow field u , the equation that governs the evolution of volume fraction is,

$$\frac{\partial f}{\partial t} + u \cdot \nabla f = 0$$

In this thesis the equation stated above is solved in 3 D. The conditions for a cell to advect on any of its six faces are as described below. The velocity components v_x, v_y, v_z are prescribed on each node of a six node brick where each node is located at the center of the face that it occupies. A cell advects on,

- the $+x$ face if $v_x > 0$ on the $+x$ face.
- the $-x$ face if $v_x < 0$ on the $-x$ face.
- the $+y$ face if $v_y > 0$ on the $+y$ face.
- the $-y$ face if $v_y < 0$ on the $-y$ face.
- the $+z$ face if $v_z > 0$ on the $+z$ face.

- the $-z$ face if $v_z < 0$ on the $-z$ face.

The 6 conditions stated above determine which faces of a cell advect in a computational domain.

We begin by defining some terms,

- Length of a cell in a Cartesian grid in the x direction is dx .
- Length of a cell in a Cartesian grid in the y direction is dy .
- Length of a cell in a Cartesian grid in the z direction is dz .
- The volume of liquid in a donor cell is V_d .
- The volume of liquid in an acceptor cell is V_a .
- The volume of a donor cell is V_{dcell} and initialize it to 1.0.
- The volume of an acceptor cell is V_{acell} and initialize it to 1.0.
- The volume fraction of a donor cell is f_d and

$$f_d = \frac{V_d}{V_{dcell}}.$$

- The volume fraction of an acceptor cell is f_a and

$$f_a = \frac{V_a}{V_{acell}}.$$

- A time step length is dt .
- The magnitude of a velocity component for a given advection direction on an associated advecting face is v .
- The length of a slice box is ϵ , for advection in the x direction.

$$\epsilon = \frac{vdt}{dxV_{dcell}}$$

- The volume of liquid in a slice box of a donor cell is δV_d

Note, a slice box can only exists in the donor. The acceptor cell does not have a slice box.

This formulation of the PLIC algorithm is for an Eulerian grid. It means that the grid remains stationary and the liquid moves in the grid. For an Eulerian formulation the Courant Number CN must obey,

$$CN \leq 1.0$$

The Courant Number is a dimensionless quantity which if computed for advection in the x direction is given by

$$CN = \frac{vdt}{dx}$$

It is the ratio of the distance travelled by a particle with velocity v in time dt , to the length of a cell. $CN \leq 1.0$ means that in one time step a donor can advect all its liquid to only one face adjacent acceptor. In the 27 point stencil cell index 13 can advect all its liquid to cell index 14, see Figure 4.1 on page 18.

We adopt operator splitting to integrate the advection equation with respect to time. This means that one advection time step is decomposed into 3 separate advection sweeps. To break symmetry effects the sweep directions for alternating advection time steps we adopt Strang [12] splitting. Strang splitting says that if the ordering of the sweep directions for the current advection time step is (x, y, z) then for the next time step the ordering is (z, y, x) .

We advect in 3 directions and study the consequences of this action step by step. The sweep directions for this example advection time step are (x, y, z) . In this example it is assumed that the advection faces are the $+x, +y$ and $+z$. A complete mathematical formulation is provided later but

an explanation of the algorithm is given first.

Figure 7.3: Interface movement during operator splitting for x and y sweep directions.

For an x advection stage:

1. Compute the slice box length ϵ for the $+x$ face.
2. Construct the interface.
3. Compute the volume of liquid in the slice box δV_d .

4. Subtract δV_d from the volume of liquid from the donor cell, i.e., $V_d - \delta V_d$.
5. Add δV to the acceptor cell, i.e., $V_a + \delta V_d$.
6. Subtract the volume of the slice box ϵ from the donor cell, i.e., $V_{dcell} - \epsilon$.
7. Add the volume of the slice box ϵ to the acceptor cell, i.e., $V_{acell} + \epsilon$.
8. Repeat all steps above so that all $+x$ advection faces are updated for donors and acceptors.
9. Compute volume fractions in the donor and acceptor cells, i.e.,

$$f_d = \frac{V_d}{V_{dcell}}.$$

and

$$f_a = \frac{V_a}{V_{acell}}.$$

10. The final conditions of volume fractions of a $+x$ sweep are initial conditions for the next sweep which is the $+y$ sweep in this example.
11. Repeat steps 1 to 9 for y advection and then for z advection.
12. The final conditions of volume fractions of a $+y$ sweep are initial conditions for the next sweep which is the $+z$ sweep in this example.

The operations, $(V_d - \delta V_d)$ and $(V_a + \delta V_d)$ are easily conceptualized. However the operations $(V_{dcell} - \epsilon)$ and $(V_{acell} + \epsilon)$ need explanation. It is true that advection is being done but it is a requirement that the volume of the grid remain unchanged. This means that a donor cell shrinks and an acceptor cell grows between sweeps. For a divergence free velocity field, it also means that at the end of three sweeps that constitute one advection time step the cells must get back to their original volumes. Each sweep is a one dimensional advection sweep. So, three sweeps constitute a 3 D solution of the advection equation. Note that the volumes of all cells were initialized to 1.0 and it was assumed that the cells were cubic. If the cells are not cubic, then the flux of liquid that flows across a cell side has to be normalized with the face area. To reduce this computation all cells in our grid are cubic.

7.2.1 Mathematical Statement of the Operator Splitting Time Integration Algorithm

The position of a computational cell in a regular mesh (azimuthally symmetric cylindrical spatial lattice) is defined by 3 integers (i, j, k) . In 3 dimensions and for a divergence free velocity field i.e., $\nabla \cdot u = 0$, volume fractions f^n are advanced in the first sweep to $\tilde{f}$ according to,

$$\tilde{f}_{\{i,j,k\}} = f^n_{\{i,j,k\}} - \frac{\delta V_{\{i+\delta i, j+\delta j, k+\delta k\}} - \delta V_{\{i-\delta i, j-\delta j, k-\delta k\}}}{V_{i,j,k}}.$$

In the second sweep $\tilde{f}$ are advanced to f^{n+1}

$$f^{n+1}_{\{i,j,k\}} = \tilde{f}_{\{i,j,k\}} - \frac{\delta V_{\{i+\delta i, j+\delta j, k+\delta k\}} - \delta V_{\{i-\delta i, j-\delta j, k-\delta k\}}}{V_{i,j,k}}.$$

In the last sweep f^{n+1} are advanced to f^{n+2}

$$f^{n+2}_{\{i,j,k\}} = f^{n+1}_{\{i,j,k\}} - \frac{\delta V_{\{i+\delta i, j+\delta j, k+\delta k\}} - \delta V_{\{i-\delta i, j-\delta j, k-\delta k\}}}{V_{i,j,k}}.$$

For sweeps in the x direction $(\delta i, \delta j, \delta k) = (1/2, 0, 0)$.

For sweeps in the y direction $(\delta i, \delta j, \delta k) = (0, 1/2, 0)$.

For sweeps in the z direction $(\delta i, \delta j, \delta k) = (0, 0, 1/2)$.

This ends discussion on the core PLIC algorithm.

Chapter 8

Interface Graphics

Images communicate ideas. The quality of an image bridges the gap between a creator's mind and a perceivers mind. The reason for making a picture is to communicate an idea that must be said; the image must support that message and not dominate it.

In order to examine the accuracy and efficiency of the PLIC algorithm the interface has to be visualized.

In previous chapters the following operations have been explained. They are,

- For a mixed cell get volume fraction distribution in its 26 element

neighbourhood.

- Compute the volume fraction gradient, ∇f .
- Compute the interface normal.
- Match the volume fraction in the mixed cell to get interface distance.
- Compute a local co-ordinate system for interface construction.
- Compute the exit face in the local co-ordinate system.
- Compute the interface equation.
- Compute the intersections of the planar interface with the edges of the mixed element.
- Assemble the interface polygon in the slice box.

To visualize the interface we need the equation of the planar interface in global co-ordinates. The global co-ordinates are also the grid co-ordinates. To do graphics there is no need to rotate volume fraction values and also there is no need to compute volumes. There are no slice boxes to be considered. Another way of saying this is, each cell that has volume fraction greater than zero and less than one is a cell that contains the interface and the bounding

box of such a cell in the grid co-ordinates could be treated as the bounding box of the slice box for the purpose of interface graphics.

The steps summarized above are repeated and additional steps are provided below.

- Instead of computing the equation of the planar interface in the (x_1, x_2, x_3) co-ordinate system we compute the equation of the interface in the $(\hat{x}, \hat{y}, \hat{z})$ co-ordinate system.
- The *min* and *max* co-ordinates of the slice box are the *min* and *max* co-ordinates of the mixed cell in the $(\hat{x}, \hat{y}, \hat{z})$ co-ordinate system.
- The interface polygon is computed and assembled.
- Finally the interface polygon is decomposed into three node triangles.

We now look at the new steps in detail.

8.0.2 Equation of the Interface in Global Co-ordinates

The interface normal in the $(\hat{x}, \hat{y}, \hat{z})$ co-ordinate system is $\bar{n}$ and it has three components $\{n_{\hat{x}}, n_{\hat{y}}, n_{\hat{z}}\}$. Now the only job that remains to be done is to compute a point on the interface. We know that the interface makes three

intercepts with the grid. The intercepts are $(h1, h2, h3)$. Any point among these 3 points could be chosen. We choose to determine $h1$. We recall that m_0 and m_1 , the vertex numbers were computed in the grid co-ordinate system. We would like to compute a vector from m_1 to m_0 . The vertex numbers m_0 and m_1 belong to the set $\{1, 2, 3, 4, 5, 6, 7, 8\}$. From the eight node numbering scheme defined in Figure 5.1 on page 28 the associated co-ordinates of the vertices m_0 and m_1 are obtained.

Let the vector from m_1 to m_0 be denoted by $vec_{m_1m_0}$

Let the co-ordinates of the vertex m_0 be $(x0, y0, z0)$

Let the co-ordinates of the vertex m_1 be $(x1, y1, z1)$

then,

$$vec_{m_1m_0} = \{(x1 - x0), (y1 - y0), (z1 - z0)\}$$

m_0 can occupy any one of the 8 spatial positions on the reference cell. Similarly m_1 can occupy any one of the 8 spatial positions on the reference cell. m_0 and m_1 cannot occupy the same spatial position. m_0 and m_1 present 6 combinations on the reference cell.

A well abstracted method is employed to detect a combination . We present an equation that does this. Denote the edge number by *caseEdge*.

Then,

$$caseEdge = \{(x1 - x0) \times 1\} + \{(y1 - y0) \times 2\} + \{(z1 - z0) \times 3\} + 3$$

Note that *caseEdge* cannot be 3. If this happens m_1 and m_0 are the same vertex which is not feasible.

caseEdge belongs to the set $\{0, 1, 2, 4, 5, 6\}$

Let the co-ordinates of the point $h1$ in the unit cell be $\{xp, yp, zp\}$.

Table 8.1 links the vertex co-ordinates to *caseEdge* and finally computes $\{xp, yp, zp\}$ is presented.

Case Edge	m_0	m_1	$\{xp, yp, zp\}$
0	$\{0, 0, 1\}$	$\{0, 0, 0\}$	$\{x0, y0, (1 - h1)\}$
1	$\{0, 1, 0\}$	$\{0, 0, 0\}$	$\{x0, (1 - h1), z0\}$
2	$\{1, 0, 0\}$	$\{0, 0, 0\}$	$\{(1 - h1), y0, z0\}$
4	$\{0, 0, 0\}$	$\{1, 0, 0\}$	$\{h1, y0, z0\}$
5	$\{0, 0, 0\}$	$\{0, 1, 0\}$	$\{x0, h1, z0\}$
6	$\{0, 0, 0\}$	$\{0, 0, 1\}$	$\{x0, y0, h1\}$

Table 8.1: End point combination and global intercept of the interface.

The point $\{xp, yp, zp\}$ must be transformed to the $(\hat{x}, \hat{y}, \hat{z})$ co-ordinate

system.

Let the co-ordinates of the transformed point be $\{xp_{Global}, yp_{Global}, zp_{Global}\}$.

Let the *min* co-ordinates of the cell in the $(\hat{x}, \hat{y}, \hat{z})$ co-ordinate system be $\{xmin_{Global}, ymin_{Global}, zmin_{Global}\}$.

Then,

$$xp_{Global} = (xp \times dx) + xmin_{Global}$$

$$yp_{Global} = (yp \times dy) + ymin_{Global}$$

$$zp_{Global} = (zp \times dz) + zmin_{Global}$$

The equation of the plane in grid co-ordinates is

$$n_{\hat{x}}\hat{x} + n_{\hat{y}}\hat{y} + n_{\hat{z}}\hat{z} - \{n_{\hat{x}}xp_{Global} + n_{\hat{y}}yp_{Global} + n_{\hat{z}}zp_{Global}\} = 0$$

This is parameterized in the form,

$$\hat{z} = a0 + a1\hat{x} + a2\hat{y}$$

where

$$a0 = \frac{\{(n_{\hat{x}}xp_{Global}) + (n_{\hat{y}}yp_{Global}) + (n_{\hat{z}}zp_{Global})\}}{n_{\hat{z}}}$$

$$a1 = -1 \times (n_{\hat{x}}/n_{\hat{z}})$$

$$a2 = -1 \times (n_{\hat{y}}/n_{\hat{z}})$$

8.0.3 Interface Triangulation

The algorithm for computing the intersections of the interface is used without any changes. Once the intersections are computed the polygon assembly is done as explained before. The tessellation algorithm is unchanged too.

Most graphics hardware is designed for three node triangles. A visualization software usually describes its file format. The three node triangles could be written in a file format that is dependent on the visualization software. A standard file format has information on geometry (the co-ordinates) and on the topology (connectivity).

Chapter 9

Initializing VOF in a Sphere

A sphere is a solid that occupies the least surface area of all solids with the same volume. A sphere is also completely parameterized by a point and a radius. An outstanding polygon that has a second degree equation in the perfect world serves as inspiration for us to test the efficiency of an advection algorithm that is piecewise linear and discontinuous.

In order to determine if an algorithm works and to bring to light why it does not work when problems arise during development a strong and simple test case is required.

In unison with our approach for integrating volumes for planar interfaces using three node triangles, we now develop an algorithm for a second degree

interface equation. Again, our volume integration is exact because we use quadratic triangular iso-parametric elements.

9.1 Summary of the Analytic Spherical Interface Algorithm

- Specify the center and the radius of the sphere.
- Flag elements that are completely outside the sphere, completely inside the sphere and those that intersect the sphere.
- Compute the normal to the sphere for elements that intersect the sphere.
- Compute one of 6 column types to transform nodal co-ordinates for using the same representation of the sphere.
- Transform the co-ordinates of the center of the sphere.
- From the *min* and *max* co-ordinates of the transformed element, define co-ordinates of end points of each of the 12 edges of the computational cell.

- Solve the equation of the sphere on the each of the 12 edges and compute legal intersections.
- Split the intersections into 2 sublists each containing a set of indices that point to a specific intersection point.
- Tessellate into 3 node triangles using a sort on the minimum angle.
- Identify unique edges in the tessellation.
- Compute the mid-edge node on each unique edge.
- Construct 6 node triangles.
- Integrate and sum the volume under the system of 6 node triangles for an element that intersects the sphere.
- Transform the nodal co-ordinates of the 6 node triangles back to global co-ordinates for visualization.

An overview of the algorithm was provided and a detailed discussion follows. As can be seen most of this algorithm is directly applicable for Piecewise Parabolic Interface Computation with changes in the method for computing the 7 co-efficients that would be needed for a quadratic equation.

9.2 Identify Elements that Intersect the Sphere

Assume that the center and radius of the sphere have been suitably chosen.

Let the global center of the sphere be denoted by $(x_{CGlobal}, y_{CGlobal}, z_{CGlobal})$

and let the radius of the sphere be r .

To test each of the 8 nodes of an eight node brick, the equation of the sphere in global co-ordinates is given by

$$f(r) = (x - x_{CGlobal})^2 + (y - y_{CGlobal})^2 + (z - z_{CGlobal})^2 - r^2$$

If $f(r) > 0$ the node under consideration (x, y, z) is outside the sphere. Similarly, if $f(r) < 0$ the node under consideration (x, y, z) is inside the sphere. This indicates that an element may have all 8 nodes outside the sphere, all 8 nodes inside the sphere or only some nodes inside the sphere. For elements having all 8 nodes inside the sphere initialize volume fraction to 1.0. For elements having all 8 nodes outside the sphere initialize volume fraction to 0.0.

In the process of assigning the two aforesaid values for element volume fraction make a set of element id's that have some but not all nodes inside the sphere. This is the set of elements in which the spherical interface has to be computed. Let this set be denoted by **InterfaceElementSet**.

9.2.1 Determine the Column Type

A column is a row of elements in a Cartesian direction. A column type belongs to the set $\{+x, -x, +y, -y, +z, -z\}$. A element belongs to a $+x$ column if the magnitude of the n_x component of the normal $n : n \in \{n_x, n_y, n_z\}$ is maximum and the sign of n_x is > 0.0 . The global normal is computed by differencing the global center of the sphere, $(x_{C_{Global}}, y_{C_{Global}}, z_{C_{Global}})$ from the centroid of the element that has membership in the `InterfaceElementSet`. The motivation for this operation is to use a representation of the type $z = f(x, y)$ for all cases of the column when intersections of the sphere are computed and projected in a fixed edge map. This is an important operation that helps in avoiding 6 copies of the code that would have been needed otherwise. For simplicity we associate the column types to integers from 0 to 5. To summarize, the column type $\in \{0, 1, 2, 3, 4, 5\}$

9.2.2 Transform Nodal Co-ordinates by Column Id

The representation of the equation of the sphere has been chosen to be $z = f(x, y)$. There is no rule about choosing the representation but the requirement is that a representation must exist. If an element displays a

column id other than 4 (+z column), we transform nodal co-ordinates so as to achieve a rotation from the current column id $\neq 4$ because we want it equal to 4. This implies that we employ the identity transformation matrix for column id equal to 4.

Case Column Id = 0, +x Column Type

$$\begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Case Column Id = 1, -x Column Type

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

Case Column Id = 2, +y Column Type

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}$$

Case Column Id = 3, $-y$ Column Type

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Case Column Id = 4, $+z$ Column Type

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Case Column Id = 5, $-z$ Column Type

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & -1 \end{pmatrix}$$

The global center of the sphere $(x_{C_{Global}}, y_{C_{Global}}, z_{C_{Global}})$ also must be transformed with the associated transformation matrix. Let the transformed center of the sphere be denoted by (x_c, y_c, z_c) .

9.3 Intersections of the Sphere With the Computational Cell

The procedure of computing the intersections of a plane or a sphere with the computational cell is identical and it has been described before in section 6.2.5. The distinguishing feature though is that a quadratic surface can intersect an edge of an element twice.

On an edge where the x co-ordinate of an intersection is unknown solve the equation,

$$x = -r^2 + (y - y_c)^2 + (z - z_c)^2 + x_c^2$$

On an edge where the y co-ordinate of an intersection is unknown solve the equation,

$$y = -r^2 + (x - x_c)^2 + (z - z_c)^2 + y_c^2$$

On an edge where the z co-ordinate of an intersection is unknown solve the equation,

$$z = -r^2 + (x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2$$

For all three cases above we do not accept a complex root. For the edges where the x and y co-ordinates are unknown the root is constrained to lie between the *min* and *max* co-ordinates associated with that edge. In the case where the z co-ordinate is known we accept the maximum of the two roots obtained by evaluating the quadratic equation.

As explained in the section 6.2.6 on Polygon assembly we sort the intersection points to prepare for a triangular decomposition.

9.4 Six Node Tessellation Algorithm

Figure 9.1: The 3 functional spaces considered for interface projection.

Six node triangles deal with three surfaces.

The projection of the the part of the interface that is below the projection plane onto the bottom of the cell. This surface has zero height and no triangles are formed on this polygon. This polygon is not formed either.

The projection of the interface above z_{max} on the top surface of the box.

This surface has height dz and triangles are formed on this polygon.

The projection of the interface that lies below z_{max} and above z_{min} onto the projection plane. This has height $z(x, y)$ and triangles are formed on this polygon.

Except for two special cases, 3-node triangles are first formed by intersections of the surface with the edges of the cell.

The special cases will be described by special examples but the methodology is more general than the examples. Consider a sphere that intersects a bottom edge but no vertices of the cell are inside the sphere. This polygon has only two intersection points. This line is bisected and the intersection of the normal to this line on the projection plane is computed with the interface. This now provides 4 points and two 3-node triangles can be computed.

If the sphere intersects a top edge of the cell, the polygon on the top surface is the complement of the polygon described above and it must be meshed.

A sphere could intersect the bottom surface and not intersect any edges. In this case, the maximum point on the sphere is found. Two normals on the projection plane that pass through the projection of the maximum point are intersected with the sphere. This provides five points and four triangles are

constructed.

If the sphere intersects the top surface but no edges of the cell, then the minimum point is found and bisectors on the top surface of the cell are intersected with the sphere surface. This provides five points to generate four triangles for the part of the surface that protrudes into the cell. In addition, the complement of this circle on the top surface of the cell forms a polygon of height dz that must be triangulated.

9.4.1 The Three Node Triangle Tessellator

If two edges of a triangle have a small included angle then the triangle so produced is a *slivered* triangle. We would like to make as few triangles as possible and at the same time not allow the included angles in any triangle to be too small. We would like to deal with tessellation of convex and non-convex polygons as well. To do this we would like to specify a robust tessellator.

Problem Statement: Given a set of points ordered in the counter clockwise sense that form a closed polygon which could be non-convex, decompose the polygon into three node triangles by sorting the minimum angle. The

problem is to be solved in 2 D.

The first job is to compute all the included angles of the polygon. As shown in Figure 9.2 let the 3 nodes that define two line segments be

$$(x_1, y_1); (x_2, y_2); (x_3, y_3)$$

Figure 9.2: Two line segments of a polygon with nodes $(x_1, y_1); (x_2, y_2); (x_3, y_3)$.

Let the vectors defined by the 3 points above be $\bar{a}$ and $\bar{b}$.

$$\bar{a} = \{\bar{a}_x, \bar{a}_y\} = \{(x_2 - x_1), (y_2 - y_1)\}$$

$$\bar{b} = \{\bar{b}_x, \bar{b}_y\} = \{(x_3 - x_2), (y_3 - y_2)\}$$

then,

$$|\bar{a}| = \sqrt{\bar{a}_x^2 + \bar{a}_y^2}$$

$$|\bar{b}| = \sqrt{\bar{b}_x^2 + \bar{b}_y^2}$$

$$\bar{a} \times \bar{b} = \{\bar{a}_x \bar{b}_y\} - \{\bar{a}_y \bar{b}_x\}$$

$$\bar{a} \cdot \bar{b} = \{\bar{a}_x \bar{b}_x\} + \{\bar{a}_y \bar{b}_y\}$$

$$\sin\theta = \frac{(\bar{a} \times \bar{b})}{|\bar{a}| |\bar{b}|}$$

If $\bar{a} \cdot \bar{b} \geq 0$ the angle between the vectors is $\sin^{-1}\theta$.

If $\bar{a} \cdot \bar{b} < 0$ the angle between the vectors is $\pi - (\sin^{-1}\theta)$.

A distinction between the included and the exterior angle has not been made yet. We would like to compute the included angle.

Convert θ to degrees.

$$\theta_{deg} = \frac{\theta \times 180.00}{\pi}$$

The included angle is computed by examining the sign of $\sin\theta$ to determine the quadrant in which the angle is begin computed. See Figure 9.3.

If $\sin\theta < 0$

- +	+ +
--	+ -

$\cos\theta, \sin\theta = (\text{sign}, \text{sign})$

Figure 9.3: The value of $\sin\theta$ is +ve in two quadrants and -ve in two quadrants.

$$angle_{included} = 180.00 + \theta_{deg}$$

If $\sin\theta > 0$

$$angle_{included} = 180.00 - \theta_{deg}$$

It is advised to sort the integer part of the included angle, i.e., store the integer part of the included angle too. The reason to convert θ to θ_{deg} and then to just store the integer part of θ_{deg} is that a sort on integers is much more efficient than a sort on reals. The value of the angle is a pointer. As long as a pointer exists the algorithm will function. A sort on reals usually builds binary trees that could take more time.

Figure 9.4: Meshing 2 D polygons by “cutting off ears.”

Having computed the included angles for a polygon identify the smallest angle. Call this angle θ_{min} . We know that θ_{min} is shared by two edges in the polygon. Two edges that share a node have a total of three nodes. Then these three nodes make a well posed three node triangle. Remove the node where θ_{min} exists and repeat the procedure stated above. Do this till there are only three nodes on the polygon. These last three nodes are a well posed

triangle. This process is sometimes called, “cutting off ears.” Figure 9.4 shows results produced by the tessellator. Note that the algorithm remains unchanged by sorting the maximum angle.

9.4.2 Computing the Mid-Edge Node

A three node triangle is decomposed into a 6 node triangle by computing the mid-edge nodes. The decision to either solve the quadratic equation or to do a simple bisection of a line segment has to be detected algebraically. More generally, in order to determine the (x, y, z) co-ordinate of a mid-edge node the co-ordinates of the end points of the edge of the three node triangle have to be examined. If the x co-ordinate lies on the edges of the computational cell then the x co-ordinate of the mid-edge node is mid way between the x co-ordinates of the end points. On the other hand if the end points do not lie on the edges of the computational cell, the co-ordinate has to be computed by solving the quadratic equation of the sphere. For all cases the mid-edge node co-ordinate computation follows a common pattern that is,

- One of the co-ordinates is computed by bisection.
- One of the co-ordinates is computed by solving the quadratic equation.

- One of the co-ordinates is computed by assigning it a *min* or a *max* value of the transformed computational cell.

9.4.3 Volume Integration for 6 Node Triangles

For a six node triangle, we consider local coordinates (r, s, t) in 2D where $t = 1 - r - s$. One reason for including t is that the formulae become symmetric.

By 3-point Gaussian integration the volume of liquid above a 6 node triangle is

$$VOF_{6NodeTriangle} = \frac{1}{6}(\phi_4(0.5, 0, 0) \det J(0.5, 0, 0) + \\ \phi_5(0.5, 0.5, 0) \det J(0.5, 0.5, 0) + \\ \phi_6(0, 0.5, 0) \det J(0, 0.5, 0))$$

In the notation below, this is:

$$VOF_{6NodeTriangle} = \frac{1}{6}(z_4(0.5, 0, 0) \det J(0.5, 0, 0) + \\ z_5(0.5, 0.5, 0) \det J(0.5, 0.5, 0) + \\ z_6(0, 0.5, 0) \det J(0, 0.5, 0))$$

$\det J(0.5, 0.5, 0)$ means determinant of Jacobian evaluated at the point

$(0.5, 0.5, 0)$. The node numbering map is shown in Figure 9.5 on page 147.

The nodes 4, 5 and 6 are the mid-edge nodes.

9.4.4 Six Node Triangle Basis Functions and Derivatives

GaussPoints coordinates are: $((a, 0), (a, a), (0, a))$.

Node numbering is shown below:

Figure 9.5: Node numbering map for a reference 6 node triangle.

Reference Coordinates of nodes are

$$\{(0, 0, 0), (1, 0, 0), (0, 1, 0), (0.5, 0, 0), (0.5, 0.5, 0), (0, 0.5, 0)\}.$$

The six basis functions as functions of reference coordinates (r, s, t) are:

$$\phi_6 = 4s(1 - r - s)$$

$$\phi_5 = 4sr$$

$$\phi_4 = 4r(1 - r - s)$$

$$\phi_3 = s - (1/2)(s_5 + s_6)$$

$$\phi_2 = r - (1/2)(s_4 + s_5)$$

$$\phi_1 = 1 - r - s - (1/2)(s_4 + s_6)$$

The derivatives of the basis functions wrt to r are:

$$\frac{\partial \phi_6}{\partial r} = -4s$$

$$\frac{\partial \phi_5}{\partial r} = 4s$$

$$\frac{\partial \phi_4}{\partial r} = 4(1 - r - s) - 4r = 4(1 - 2r - s)$$

$$\frac{\partial \phi_3}{\partial r} = (1/2)(-4s + 4s) = 0$$

$$\frac{\partial \phi_2}{\partial r} = 1 + (1/2)(4(1 - 2r - s) + 4s) = 3 + 4(s - r)$$

$$\frac{\partial \phi_1}{\partial r} = -1 - (1/2)(4(1 - 2r - s) - 4s) = -3 + 4(r + s)$$

The derivatives of the basis functions wrt to s are:

$$\frac{\partial \phi_6}{\partial s} = 4(1 - r - 2s)$$

$$\frac{\partial \phi_5}{\partial s} = 4r$$

$$\frac{\partial \phi_4}{\partial s} = -4r$$

$$\begin{aligned}\frac{\partial \phi 3}{\partial s} &= 1 + (1/2)(4r - 4r) = 0 \\ \frac{\partial \phi 2}{\partial s} &= -(1/2)(-4s + 4r) = 2(s - r) \\ \frac{\partial \phi 1}{\partial s} &= -1 - (1/2)(-4r + 4(1 - r - 2s)) = -3 + 4(r + s)\end{aligned}$$

The Jacobian for a general curvilinear element is

$$J = \begin{pmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial s} \end{pmatrix}.$$

$$\frac{\partial x}{\partial r} = \sum_{i=1}^{No. \ Nodes} \frac{\partial \phi_i}{\partial r} \bar{x}_i.$$

$$\frac{\partial x}{\partial s} = \sum_{i=1}^{No. \ Nodes} \frac{\partial \phi_i}{\partial s} \bar{x}_i.$$

$$\frac{\partial y}{\partial r} = \sum_{i=1}^{No. \ Nodes} \frac{\partial \phi_i}{\partial r} \bar{y}_i.$$

$$\frac{\partial y}{\partial s} = \sum_{i=1}^{No. \ Nodes} \frac{\partial \phi_i}{\partial s} \bar{y}_i.$$

9.4.5 Implementation of Volume Integration with 6

Node Triangles

For each 6 node triangle and for each of the 3 Gauss points,

$$(r, s, t) = (0.5, 0, 0); (0.5, 0.5, 0); (0, 0.5, 0).$$

evaluate the following:

$$\frac{\partial s_6}{\partial r}; \frac{\partial s_5}{\partial r}; \frac{\partial s_4}{\partial r}; \frac{\partial s_3}{\partial r}; \frac{\partial s_2}{\partial r}; \frac{\partial s_1}{\partial r}.$$

Make a vector dsdr {ds6dr ds5dr ds4dr ds3dr ds2dr ds1dr}

$$\frac{\partial s_6}{\partial s}; \frac{\partial s_5}{\partial s}; \frac{\partial s_4}{\partial s}; \frac{\partial s_3}{\partial s}; \frac{\partial s_2}{\partial s}; \frac{\partial s_1}{\partial s}.$$

Make a vector dsds {ds6ds ds5ds ds4ds ds3ds ds2ds ds1ds}

Also make vectors

xCoord {x6 x5 x4 x3 x2 x1} and

yCoord {y6 x5 y4 y3 y2 y1}

Then the Jacobian is

$$J_{11} = ds \ dr \times xCoord.$$

$$J_{12} = ds \ dr \times yCoord.$$

$$J_{21} = ds \ ds \times xCoord.$$

$$J_{22} = ds \ ds \times yCoord.$$

$$\det = [J_{11} \times J_{22} - J_{12} \times J_{21}]$$

$$VOF = (1/6)(z_4 \det(0.5, 0, 0) + z_5 \det(0.5, 0.5, 0) + z_6 \det(0, 0.5, 0)$$

Chapter 10

Numerical Results

In order to assess the performance of the PLIC algorithm the following simulations were conducted.

1. Test case preparation. Use of 6 node triangles to initialize volume fractions in a sphere.
2. Detailed examination of the PLIC interface.
3. Rigid body translation of a sphere in the $+x$ direction.
4. Rigid body translation of a sphere in a direction (a, a, a) where $a = 1/\sqrt{3}$.

5. Rigid body rotation of two spheres with different diameters.
6. Advection of two spheres in an analytic fountain flow velocity field.
7. Advection of two spheres in an analytic deformation field.
8. A coupled Navier-Stokes and PLIC advection simulation on a domain that represents a transmission housing typically used in automobiles.

All results that *show the PLIC interface* have been generated by the algorithm described in the chapter on Interface Graphics. Graphics have not been provided for simulation 3. In all simulations the Courant Number, $CN = 0.95$ and the elements are cubic. Mesh details are provided for each test.

10.1 Test Case Preparation

As sphere is initialized with liquid and the results are compared with an analytic sphere having the same radius. To present the triangulation clearly a sphere having a radius of 5 elements is chosen. The elements are cubic and have a size of 1 mm. each and the grid used is $30 \times 30 \times 30$. Since graphics software used to print results in this thesis does not allow a data set for 6

node triangles. A six node triangle is decomposed into three node triangles for visualization. Each six node triangle forms four, three node triangles.

Figure 10.1: Decomposition of a 6 node triangle into 4 three node triangles.

The volume of a sphere with radius r is $(4/3)\pi r^3$. For a sphere of radius 5 mm the analytic volume is 523.598.

The sphere initialized from six node triangles reports the sum VOF for the domain 523.497. Elements that are completely outside the sphere make no contribution to sum VOF. This means that sum VOF domain is the VOF in the sphere. In the Table 10.1 the comparison is shown.

Exact volume of sphere	Volume from 6 node triangles
523.598	523.497

Table 10.1: Comparison of analytic and numerical volume.

As can be seen from Table 10.1 the values of sum VOF for the sphere generated from 6 node triangles are in excellent agreement with the analytic

sphere sum VOF computation. The sphere generated from 6 node triangles can be used to represent an ideal interface to form a test case.

In the graphics provided in Figure 10.2 on the next page the nodes that are visible in the view have been marked by diamonds. Line segments joining the nodes are the edges of 3 node triangles.

Figure 10.2: Numerically simulated analytic spherical interface.

10.2 Critical Assessment of the PLIC Interface

A macro analysis of the appearance of the PLIC interface shows that as expected, the interface has been constructed as a plane in each mixed cell. The existence of 5 interface polygon types discussed earlier is easily confirmed.

A micro analysis for comparing the interface normal computed by the central differencing scheme and that obtained from an analytic computation was performed on a sphere with radius $r = 4$ mm where the element size was 1mm. The cosine of the angle between the analytic normal and the PLIC normal by the dot product is computed. Ideally the angle is zero and the dot product is 1.0. The Euclidean distance of a node on a three node triangle is measured from the center of the sphere.

The lower bound and the upper bound signed error in the dot product are presented in Table 10.2 . The definition of error here is $1.0 - (\text{dot product})$.

The lower bound and the upper bound error in the radius measured at the nodes of each three node triangle are presented in Table 10.3 . The definition of error here is $(\text{exact radius}) - (\text{radius achieved})$.

Close examination of the interface shows that there are gaps. A highly

Lower bound error dot product	Upper bound error dot product
1.2685×10^{-10}	1.0399×10^{-4}

Table 10.2: Lower and upper bound errors in the dot product.

Lower bound error radius at node	Upper bound error radius at node
-4.966×10^{-5}	-1.3631×10^{-5}

Table 10.3: Lower and upper bound errors in the radius.

magnified graphic, Figure 10.4 that shows the gaps clearly, is provided. We suspect that the gaps visible on the magnified graphic are associated with the upper bound error in the PLIC normal seen in Table 10.2 .

Figure 10.3: The PLIC interface.

Figure 10.4: Magnified view of the PLIC interface.

10.3 Rigid Body Translation Tests

For the rigid body translation test the grid is $30 \times 30 \times 30$ with element length $1/30$ mm. The sphere is initialized at the center of the grid and has radius 4 elements. The first test was done with a velocity being prescribed on the $+x$ face, $v_x = 1\text{ m/s}$. The v_y and v_z components were maintained at zero velocity. In this simple test mass is conserved. The shape of the sphere remains spherical after advection. No distortion of the interface is observed. The test was repeated 5 times by setting only one velocity component to 1 m/s on each node in the domain for each test. The sphere was initialized at the center of the grid because we wanted to use identical initial conditions for all 6 runs. This is a robust method of detecting flaws in the logic, especially finding out any illegal directional syndromes. The success obtained in these tests leads to the next translation test where the sphere is translated diagonally across the mesh at (a, a, a) , where $a = 1/\sqrt{3}$. In this translation test the sphere translates smoothly along the body diagonal of the domain. No distortion of the interface is observed during this movement. Mass is conserved to a precision of 1×10^{-6}

10.4 Rigid Body Rotation of Two Spheres

Our method to initialize spheres needs user input of only two parameters, the center and the radius. We face no difficulty in initializing two spheres in the same grid that forms a test case for our next simulation. The two spheres have different radii. The small sphere has radius $r_{small} = 4$ elements and the big sphere has radius $r_{big} = 4.5$ elements. The grid is $30 \times 30 \times 30$ and the element length is $1/30$ mm. The centers of both spheres are offset by 7 elements from the grid center in the $+x$ and $-x$ directions.

Radius	Sum VOF 6 node Triangles	Sum VOF analytic
4.0 elements	268.0596	268.0825
4.5 elements	381.6836	381.7035

Table 10.4: Comparison of sum VOF sphere for $r = 4$ elements and 4.5 elements.

Velocity is prescribe by the expressions,

$$v_x = -1.0 \times \{y - y_c\}; \quad v_y = x - x_c; \quad v_z = 0.0$$

At the end of 21 time steps the spheres have rotated by 90° . Values for

sum VOF in the sphere are presented in Table 10.5 . Orthographic views for the initial conditions and final conditions at the end of 21 times steps are presented in Figure 10.5

Sum VOF spheres time $t = 0.0s$.	Sum VOF spheres time $t = 1.603080s$.
649.7433	649.7428

Table 10.5: Conservation of Mass in rigid body rotation.

The loss of fluid is 5 parts per 6 million parts.

Figure 10.5: Rigid body rotation of two spheres with different diameters.

Solid body rotation results add little additional insight to the translation tests. Neither the translation or rotation problems pose serious problems for a well designed interface tracking algorithm. They will expose diffusion and dispersion problems. Excellent performance on translation and rotation tests is necessary but not sufficient for a method tasked to track interfaces in complex topology. A more serious test is conducted now.

10.5 Advection in a Wu and Currie Flow Field

The objective of this test is to examine if values of VOF from the same 3x3x3 stencil can be used by two interfaces that move close to each other. In other words we want to see what separation is possible between two colliding interfaces before they merge. At the same time we want to induce topological changes in the interfaces.

H. Wu and I.G. Currie [14] present an analytic fountain flow velocity field given by the expressions

$$v_x = \cos x \sin y; \quad v_y = -\sin x \cos y; \quad v_z = 0$$

Figure 10.6: Wu & Currie velocity field.

The aforesaid velocity field has been used by [14] simulation of viscous incompressible flows. The domain extends from $(0, 0, 0)$ to $(2\pi, 2\pi, 2\pi)$. This centers the flow field in the domain. The domain is $30 \times 30 \times 30$ and the element length is $(2\pi/30.0)m$. Two spheres of equal radii $r = 4$ elements are initialized. Their centers are offset by 6 elements from the center of the grid in the x direction. The values of sum VOF for a sphere of radius 4 elements initialized from six node triangles are available in Table 10.4 . The simulation was conducted for 12 time steps. The velocity field used is shown in Figure 10.6

Loss of fluid in 12 time steps is 146 parts per 5.3 million parts.

Sum VOF spheres time $t = 0.0s$.	Sum VOF deformed spheres time $t = 1.441840s$.
536.1194	536.1048

Table 10.6: Conservation of Mass in a fountain flow velocity field.

Results for the fountain flow velocity simulation are presented in Figures 10.7 , 10.8 and 10.9 . We find that on the 13th time step the interfaces do have separation. Interestingly, in this simulation the net inflow volume and out flow volume are zero. In the case of rotation of two spheres with different diameters we found the inflow volume equal to the out flow volume within precision.

Figure 10.7: Two spheres in an analytic Currie & Wu flow velocity field at time $t = 0.0$.

Figure 10.8: Two spheres in an analytic Wu & Currie flow velocity field. The velocity field is constant and the time step size is 0.11091 s.

Figure 10.9: Two spheres in an analytic Wu & Currie flow velocity field.

This is at time step number 13.

10.6 Advection in a Smolarkiewicz Flow Field

The Smolarkiewicz [11] deformation flow field is considered to be the most stringent test for an interface tracking algorithm by Rider and Kothe [9].

The stream function,

$$\Psi = \frac{1}{4\pi} \sin\{4\pi(x + \frac{1}{2})\} \cos\{4\pi(y + \frac{1}{2})\}$$

gives velocities,

$$v_x = \frac{\partial \Psi}{\partial x}; \quad v_y = -\frac{\partial \Psi}{\partial y}; \quad v_z = 0$$

where,

$$\frac{\partial \Psi}{\partial x} = \cos\{4\pi(x + \frac{1}{2})\} \cos\{4\pi(y + \frac{1}{2})\}$$

$$\frac{\partial \Psi}{\partial y} = -\sin\{4\pi(x + \frac{1}{2})\} \sin\{4\pi(y + \frac{1}{2})\}.$$

This velocity field induces radical deformation and topological changes on the interface.

We disagree with Rider and Kothe [9] that an un-split time integration scheme uses cpu time significantly more efficiently than an operator splitting time integration scheme. Experience shows that time required to construct

our minimal computational sub-domain and the time to do three sweeps have an average ratio 0.8. An un-split time integration algorithm might produce a domain set time to solve time ratio of 1.0 to 1.3. Note that the solve time would have been significantly increased if the full domain had been solved. Often the increase would be a factor 50 times or more. We also disagree that operator splitting does not preserve symmetry. We suspect that differences in algorithm implementation are the cause of this disagreement.

In our frame work to initialize a sphere using six node triangles we initialize a cylinder of fluid that is only one element thick. This is done by making the domain $30 \times 30 \times 1$. The radius of the cylinder is 8 elements and the elements are $1/30$ mm. In this domain our objective is to run a 2 D problem on code that manages 3 D problems. It is also our objective to test the efficiency of our time integration scheme that uses Strang's [12] operator splitting. The simulation is conducted for 12 time steps. At the end of 12 time steps severe deformation has been induced in the interface. A large fraction of the circle has been entrained into the four nearest vortices.

No special efforts have been made to center the cylinder with respect to the center of 4 complete vortices. Figure 10.10 on page 171 shows the deformation velocity field used the simulation.

Figure 10.10: The Smolarkiewicz deformation velocity field.

Sum VOF cylinder 6 node Triangles	Sum VOF cylinder analytic
200.5877	201.06193

Table 10.7: Comparison of sum VOF cylinder for $r = 8$ elements

The simulation was conducted for 12 time steps and mass conservation is as shown in Table 10.8.

Our cylinder generated from six node triangles shows error this time because we have in-fact sliced a sphere to approximate a cylindrical interface. Comparison of the values for analytic and the achieved sum VOF are shown in Table 10.7 .

Sum VOF time $t = 0.0s.$	Sum VOF time $t = 0.380000s.$
200.5877	199.8675

Table 10.8: Conservation of Mass in a deformation velocity field.

Results for the deformation flow field simulation are presented in Figures 10.11 on page 173 and 10.12 on page 174.

We see that that our interface deforms very symmetrically about an axis that is parallel to the x axis. The graphics for this simulation have been done by our graphics algorithm and since the interface is only one element thick 2 D views are provided.

Figure 10.11: A cylindrical mass of fluid in a deformation velocity field.

The velocity field is constant and the time step size is 0.03167 s.

Figure 10.12: A cylindrical mass of fluid in a deformation velocity field.

The velocity field is constant and the time step size is 0.03167 s.

Analysis of Real Castings

10.7 Analysis of a Real Casting

The final goal of this thesis is to apply the PLIC advection algorithm to solve real problems so that industry benefits from research.

We provide an existence proof that our PLIC advection solver is not just capable of solving analytic velocity fields but is also capable of providing data useful to designers in the die casting industry.

The casting under study is a transmission case typical of those used in automobiles, see Figure 10.13 on page 10.13. The analysis is a filling analysis. This casting is manufactured with a die that costs a little more than 1 million dollars. The time needed to design the die is about ten person years and six to ten calendar months. This analysis provides the die designer with a realistic simulation tool to assess a filling pattern as the design evolves.

Two solvers are coupled to do this. The divergence free velocity field is generated by a transient 3-D Navier- Stokes equation solver for incompressible flow. This solver has been used as a tool and has not been coded by the author of this thesis. The second solver is the PLIC advection solver. It computes VOF as a function of time.

The mesh has 589,145 elements and the view in Figure 10.14 represents

about 54% filling. The interface is shown in figure 10.15. In Figure 10.15, there are about 46,000 triangles.

Figure 10.13: Geometry of a transmission case housing.

Figure 10.14: The interface in a 600, 000 element mesh of transmission case at about 54% filling.

Figure 10.15: The interface on a transmission case at about 54% filling showing about 46, 000 triangles.

Chapter 11

Conclusion

A Piecewise Linear Interface Computation, PLIC, was implemented that followed the basic theory outlined by Youngs [15] but substantially simplified the implementation by replacing Youngs' volume integration formulae by a scheme that integrated volumes under 3-node triangles.

The analytic velocity fields, particularly those by Wu and Currie and by Smolarkiewicz (used by Kothe [9]) provide useful insight. They show that the PLIC algorithm is capable of handling stretching and topological changes of the interface.

The analyses were halted because the discrete velocity field was not div free. If a div free projection of the velocity field had been available, it would

have been possible to run the analyses as long as desired.

The velocity field from Wu and Currie showed that VOF values from the same 3x3x3 stencil can be used by two distinct interfaces without causing distortion or diffusion of the interface. In the literature reviewed on interface tracking no example that illustrates this point was found.

The Smolarkiewicz velocity field test shows clearly that use of operator splitting does not in any manner alter the symmetry of the deforming cylindrical interface.

This PLIC scheme was coupled with a transient 3D Navier-Stokes equations solver to solve the problem of filling a casting with liquid metal. The results of the analysis for filling an automotive transmission case are most encouraging and are a strong test of the robustness of the PLIC code. The results also indicate the usefulness of the PLIC algorithm and again literature reviewed does not present an example of advection by PLIC on real castings with complex 3D geometry.

The volume integration algorithm that uses one point integration on three node triangles has proven to be effective in managing complexity robustly.

Using a dynamic minimal interface domain has proven to be successful in terms of minimizing cpu time.

The thesis has been written by providing examples as and when they were required. It is hoped that the reader has been able to conceptualize the algorithm and relate it to the associated physics.

Bibliography

- [1] A. CHORIN and J. MARSDEN. *A mathematical introduction to fluid mechanics*, chapter 1, pages 1–4. Springer-Verlag, USA, 1990.
- [2] W. J. RIDER, D. B. KOTHE and J. I. HOCHSTEIN. Volume tracking of interfaces having surface tension in two and three dimensions. *Technical Report AIAA*, (96-0859), 1996. Presented at the 34rd Aerospace Sciences Meeting and exhibit.
- [3] C. W. HIRTH and B. D. NICHOLAS. A computational method for free surface hydrodynamics. *Journal of Pressure Vessel Technology*, (103:136-140), 1981.
- [4] F. L. ADDESSIO, J. N. JOHNSON and B. A. KASHIWA. Caveat: A computer code for fluid dynamics problems with large distortion and

- internal slip. Technical Report Technical Report LA-10613-MS, Los Alamos National Laboratory, 1986.
- [5] E. KREYSZIG. *Advanced Engineering Mathematics*, chapter 10, pages 20–31. John Wiley & Sons, USA, 1979.
 - [6] W. F. NOH and P. R. WOODWARD. Slic (simple line interface method). *Lecture Notes in Physics 59*, pages 330–340, 1976.
 - [7] G.R. PRICE and C.T. READER. A piecewise parabolic interface calculation for volume tracking. *Journal of Computational Fluid Dynamics*, VIII:71–77, 1998. The Sixth Annual Conference of the Computational Fluid Dynamics Society of Canada (CFD 98), Quebec City, Canada.
 - [8] E.G. PUCKETT and J. S. SALTZMAN. A 3-d adaptive mesh refinement algorithm for multi-material gas dynamics. *Physica D*, (60:84-93), 1992.
 - [9] W. J. RIDER and DOUGLAS B. KOTHE. Reconstructing volume tracking. *Journal of Computational Physics*, 141(2):112 – 152, APR 1998.

- [10] P. SCHORN and F. FISHER. Testing convexity of a polygon. In Paul S. Heckbert, editor, *Graphics Gems IV*, pages 4–7. AP Professional, USA, 1994.
- [11] P. K. SMOLARKIEWICZ. The multi-dimensional crowley advection scheme. *Monthly Weather Review*, (110:1968-1983), 1982.
- [12] G. STRANG. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, (5:506-516), 1968.
- [13] B. P. FLANNERY SAUL, W.H. PRESS and A. TEUKOLSKY. *Numerical Recipes, The Art of Scientific Computing*, chapter 10, pages 254–256. Cambridge University Press, USA, 1979.
- [14] H. WU and I.G. CURRIE. Numerical simulation of viscous incompressible flows using posteriori error estimates. *Journal of Computational Fluid Dynamics*, 141(2):525–530, May 25-27 1997. The Fifth Annual Conference of the Computational Fluid Dynamics Society of Canada (CFD 97), Victoria, Canada.
- [15] D. L. YOUNGS. An interface tracking method for a 3-d eulerian hydrodynamics code. Technical Report AWRE/44/92/35, Atomic Weapons

Research Establishment (U.K.), 1984.

- [16] A. Z. ZINCHENKO and R. H. DAVIS. An efficient algorithm for hydrodynamical interaction of many deformable drops. *Journal of Computational Fluid Dynamics*, VIII:13–18, 1998. The Sixth Annual Conference of the Computational Fluid Dynamics Society of Canada (CFD 98), Quebec City, Canada.

IMAGE EVALUATION TEST TARGET (QA-3)

APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved