University of Cape Town

Doctoral Thesis

An Algebraic Volume of Fluid Method for Strongly Coupled Spacecraft Fuel Slosh Modelling

Author:
Bevan W. S. Jones

Supervisor:
Prof. Arnaud G. Malan

University of Cape Town

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Industrial Computational Fluid Dynamics Research Group
Department of Mechanical Engineering

Tuesday 6th October, 2020

The copyright of this thesis vests in the author. No quotation from it or information derived from it is to be published without full acknowledgement of the source. The thesis is to be used for private study or non-commercial research purposes only.

Published by the University of Cape Town (UCT) in terms of the non-exclusive license granted to UCT by the author.

University of Cape Town

Declaration of Authorship

I, Bevan W. S. Jones, declare that this thesis titled, "An Algebraic Volume of Fluid Method for Strongly Coupled Spacecraft Fuel Slosh Modelling" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Signed by candidate

Tuesday 6th October, 2020

"*Modern science has been a voyage into the unknown, with a lesson in humility waiting at every stop. Our common sense intuitions can be mistaken, our preferences don't count, we do not live in a privileged reference frame.*"

- Carl Sagen

Abstract

The increase in the number of commercial space missions has resulted in the increased need for efficient and effective spacecraft designs. A key contributor to the accuracy of space vehicle simulation is the prediction of fuel slosh loads during in-orbit manoeuvres, particularly due to the large fuel-to-solid mass ratios involved. To this end, this thesis details a high resolution mathematical model capable of predicting the dynamic interaction between fuel slosh and the rigid structure of a spacecraft.

The Volume of Fluid (VoF) method provides a framework in which Computational Fluid Dynamics (CFD) can be used to model the fluid dynamics of two phase fuel slosh in a mass conservative manner. To be applicable to industrial geometries, an unstructured finite volume median dual cell methodology is employed for spatial discretisation. This gives rise to the first novel contribution of this work, namely the development of a new volume conservative VoF initialisation method for arbitrary interfaces on unstructured meshes. The scheme, called the Arbitrary Grid Initialiser (AGI), is rigorously validated and proven conservative to machine precision [1].

An algebraic, as opposed to geometric, VoF advection method is used due to being similarly well suited to unstructured grids. Improvements to the algebraic VoF method is therefore the next contribution of this thesis; where the CICSAM [2] and HiRAC [3] VoF methods are improved, and the first conservative HiRAC method presented. The improved CICSAM and HiRAC methods are shown to be competitive with their geometric counterparts on unstructured grids while being mass conservative.

Both CICSAM and HiRAC are then coupled (HiRAC for the first time) to a well-balanced Continuum Surface Force (CSF) surface tension discretisation. The surface tension implementation, for which standard height functions are used, is shown to be well-balanced with an accuracy that compares favourably to existing methods.

In the final part of the thesis, the complete spacecraft model is constructed. A numerical rigid body code is developed for this purpose, which can additionally track its orientation. The rigid body and fluid schemes are finally coupled together in a strong, stable, and partitioned manner using the Aitken's Δ^2 method [4]. The model is demonstrated to be numerically stable for large liquid-to-solid ratios via a benchmark test case.

Acknowledgements

Professor Arnaud Malan

A number of years ago I joined the InCFD group, and at a time of great personal strife. Arnaud has focused me, and guided me through this journey. I have grown both as a researcher and personally under Prof. Malan's tutelage. I am forever grateful for his input and contributions to my work and for advice offered in the darker moments of this journey. I am certain that this project could not have been achieved without his mentorship.

Dr. Philipp Behruzi, Francesco de Rose, and Guido Schwartz

The project would not have commenced if not for the team at Ariane Group and Airbus Defence and Space. The insight and exposure to the space industry has been a deeply rewarding experience. I always found myself invigorated and motivated after a meeting with the Bremen team. My trips to Bremen, Germany, are remembered fondly. The warmness and hospitably shown by these gentlemen is truly appreciated. *Vielen Dank, meine Herren.*

InCFD Research Group

The InCFD Research Group has been my home for the past many years. I have seen a number of masters and PhD students come and go. All have improved my working environment and their attitude to work as a team has always amazed me. The support received by various members over the years is too numerous to recount individually, all of which I am grateful for. I would like to make a special mention to Niran Ilangakoon, who I have had the privilege of working closely with for about as long as I have been in the research group.

NRF-SARChi, Ariane Group, and CHPC

This work was financially supported by the South African National Research Foundation through the South African Research Chair, and by Ariane Group in Bremen, Germany. A number of caclucations were conducted on the Centre for High Performance Computing (CHPC) clusters.

Nadia and Haydn Jones

Mom and Dad, you have been a never ending source of stability and encouragement through my life. I am in debited to you. Throughout this process you have been a continued pillar of strength. This work represents the end of a long, difficult journey, and the beginning of a new one. One in which I am sure I will be able to count on your continued support.

Contents

Nomenclature

Acronyms

CFD	Computational Fluid Dynamics
CICSAM	Compressive Interface Capturing Scheme for Arbitrary Meshes
CSF	Continuum Surface Force
CSS	Continuum Surface Stress
FPI	Fixed-Point Iteration
FSI	Fluid-Structure Interaction
FVM	Finite Volume Method
HiRAC	Higher Resolution Artificial Compressive
VoF	Volume of Fluid

Operators

$\#\boldsymbol{\phi}$	cardinality of the set $\boldsymbol{\phi}$
$\boldsymbol{\phi} \otimes \boldsymbol{\varphi}$	outer product between $\boldsymbol{\phi}$ and $\boldsymbol{\varphi}$
$\boldsymbol{\phi} \times \boldsymbol{\varphi}$	vector cross product between $\boldsymbol{\phi}$ and $\boldsymbol{\varphi}$
$\boldsymbol{\phi}^{\perp\boldsymbol{\varphi}}$	orthogonal component of $\boldsymbol{\phi}$ with respect to $\boldsymbol{\varphi}$
$\ddot{\phi}_{/\varphi}$	second derivative with respect to time of ϕ as seen by the φ reference frame.
$\Delta\phi$	discrete difference of ϕ
$\dot{\phi}_{/\varphi}$	first derivative with respect to time of ϕ as seen by the φ reference frame.
$\hat{\boldsymbol{\phi}}$	unit vector of $\boldsymbol{\phi}$
$\mathcal{H}(\phi)$	Hessian of ϕ
$\nabla\boldsymbol{\phi}$	Jacobian of $\boldsymbol{\phi}$
$\nabla\phi$	gradient of ϕ
$\nabla^2\phi$	Laplacian of ϕ
${}^{\varphi}\boldsymbol{\phi}$	the vector $\boldsymbol{\phi}$ expressed in the φ basis.
$\underline{\boldsymbol{\phi}}^{\times}$	cross product matrix, $\boldsymbol{\phi}\times$

Symbols

α	volume fraction	
α^*	smoothed volume fraction	
β	CICSAM blending constant	
$\boldsymbol{\omega}$	angular velocity	$\mathrm{rad.s^{-1}}$
$\boldsymbol{\partial\Omega}$	set of control faces	
$\boldsymbol{\tau}$	torque	$\mathrm{N.m^{-1}}$
$\boldsymbol{f}$	force	N
$\boldsymbol{G}$	linear momentum	$\mathrm{kg.m.s^{-1}}$
$\boldsymbol{H}$	angular momentum	$\mathrm{kg.m^2.s^{-1}}$
$\boldsymbol{h}$	height as a vector line segment	
$\boldsymbol{k}_m$	kinematic state vector: acceleration, velocity, and displacement	
$\boldsymbol{k}_t$	kinetic state vector: force and torque	
$\boldsymbol{q}$	quaternion	
$\boldsymbol{r}_{i,j}$	direction vector from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$	m
$\boldsymbol{u}$	velocity u, v, w	$\mathrm{m.s^{-1}}$
$\boldsymbol{X}$	set of point co-ordinates	m
$\boldsymbol{x}$	point co-ordinate x, y, z	m
$\hat{\boldsymbol{e}}$	unit vector in a cardinal direction	
$\hat{\boldsymbol{n}}$	unit normal	
ι	(sub)iteration index	
κ	curvature	$\mathrm{m^{-1}}$
λ	scalar parameter	
$\mathcal{E}$	discretisation error	
$\mathcal{R}$	residual	
μ	dynamic viscosity	$\mathrm{kg.m^{-1}.s^{-1}}$
Ω	control region or domain	
Ω_p	polytope	
Ω_s	simplex	
$\partial\Omega$	control face	
$\phi/\boldsymbol{\phi}$	general scalar/vector	
ρ	density	$\mathrm{kg.m^{-3}}$

σ	surface tension coefficient	$\mathrm{N.m^{-1}}$
$\underline{\boldsymbol{I}}$	mass moment of inertia matrix	$\mathrm{kg.m^2}$
$\underline{\boldsymbol{T}}_q$	quaternion transformation matrix	
$\underline{\boldsymbol{W}}_q$	quaternion rate matrix	
ε	numerical error (absolute or relative)	
ϖ	Aitken's Δ^2 relaxation parameter	
A	area	$\mathrm{m^2}$
c	scalar constant	
D	diameter	m
G	set of implicit functions	
g	implicit function	
G_0	zero level contour of a set of implicit functions	
h	height (of a column)	
m	mass	kg
n	time step index	
p	pressure	$\mathrm{N.m^{-2}}$
t	time	s
u_{ff}	face flux	$\mathrm{m^3.s^{-1}}$
V	volume	$\mathrm{m^3}$

To Emma

Chapter 1

Introduction

1.1 Background and Motivation

Space operations have grown dramatically since the advent of chemical rocket engines powerful enough to launch humans and their payloads to space. From single infrequent launches to present day, where there is permanent human occupation in space, numerous deep space missions, and a cloud of communication, weather, and other satellites in Earth orbit. The industry continues to grow, and the number of spacecraft in service is expected to increase. This trend is furthered by the emergence of reusable hardware which has reduced the cost of the access to space.

Computer based simulation has become a quintessential part in the production of spacecraft, as it gives critical insight into the expected performance of a design. However, the simulation of spacecraft is complex and computationally demanding. Orbital craft are multidisciplinary machines and will experience extreme vibrational loads, sub and supersonic flight, and micro gravity in orbit. In addition, these crafts are expected to function continuously over long periods of time.

These phenomena impact on a vehicle with a large internal dynamic load, in the form of propellent slosh. This fuel mass is typically an order of magnitude more than the solid craft. Clearly, the capturing of the internal fluid dynamics is paramount to accurately simulate spacecraft, especially when the estimation of the dynamic loads, due to in-flight manoeuvres, is desired. Continuum mechanics, through the use of multi-physics Computational Fluid Dynamics (CFD), provides such a framework for high fidelity modelling. This will be the subject of this work which will focus on the following key aspects to the modelling of spacecraft operation:

- With respect to the liquid propellent slosh modelling:
  - An isothermal, incompressible, two-phase flow model is considered. Where unstructured approaches are favoured due industrial applicability.
  - Surface tension effects will be considered, however contact line forces are neglected.
- The continuum mechanics strongly coupled simulation of the entire space vehicle where the dry mass is considered rigid, with no flexible components or deformable parts.

The above will be expanded further in the following sections.

1.1.1 Incompressible Two Phase Flow

Two phase incompressible flow modelling via CFD consists of the simultaneous solution of a series of equations. These include the momentum and mass conservation of

the fluid,

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot \rho \boldsymbol{u} \otimes \boldsymbol{u} + \nabla p - \nabla \cdot \mu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T \right) = \rho \boldsymbol{a} + \boldsymbol{f}_\sigma, \tag{1.1}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{1.2}$$

where $\boldsymbol{u}$, p, and $\boldsymbol{a}$ are the velocity, pressure, and source acceleration of the fluid. The fluid density and viscosity are represented by ρ and μ. The time and differential operator are denoted t and ∇ respectively. For the simulation of propellent tanks a multi-phase flow model must be adopted and further consideration must be given to liquid-gas interface effects, such as the surface tension force, denoted $\boldsymbol{f}_\sigma$.

A number of two-phase approaches have been developed, namely the Ghost Fluid [5], level-set [6], front tracking [7], and Volume of Fluid (VoF) [8] methods. In this work the VoF approach is selected due to its mass conserving properties. The VoF equation to be solved reads

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot \alpha \boldsymbol{u} = 0, \tag{1.3}$$

where the volume fraction of the tracked fluid in a discrete volume is symbolised α. As a result of this representation the interface between the two fluids is smeared over discrete control volumes, where the interface region is considered to be where $\alpha \in (0, 1)$.

This introduces a number of challenges. This thesis will addresses two of these. Firstly, the conservative initialisation of α implies a volume fraction field derived from the intersection of a manifold (surface) and the control volumes of an unstructured mesh. Secondly, the conservative advection of α via an unstructured algebraic VoF method.

From a volume fraction initialisation perspective, it is desirable that the initial volume fraction be a mass conservative representation of the tracked fluid on the grid. A key contribution on this topic was by Bna et al. [9,10], where an initialisation tool for structured grids was developed, achieving machine precision in terms of mass conservation for the first time. Strobl et al. [11] were the first to develop a method for unstructured grids, however this was limited for spherical interfaces only. A significant contribution of this thesis entails the development of the first method which can robustly and conservatively initialise a volume fraction field on an unstructured grid, this for arbitrary free surface topologies [1].

Great strides have been made in the development of structured VoF methods over the past decades [8, 12–16]. Recently, geometric VoF schemes, which utilise an explicit geometric reconstruction of the interface to advect the volume fraction, have been extended to unstructured grid methodologies [17–19]. Algebraic VoF methods, which do not requiring explicit interface reconstruction, are however less well developed. The widely used algebraic Compressive Interface Capturing Scheme for Arbitrary Meshes (CICSAM) [2] scheme is prone to smearing the interface at large time steps sizes. Recent work to addresses this aspect resulted in the Higher Resolution Artificial Compressive (HiRAC) method [3]. However, the HiRAC method is not volume conserving, which negates the main advantage of the VoF method. This key deficiency will be addressed in this thesis.

1.1.2 Coupling to Rigid Bodies

The evolution of the fluid is ultimately shared by the kinematics and kinetics of the rigid space vehicle. A rigid body model, representative of the dry mass of the vehicle, must be coupled to the liquid propellent to simulate a full orbital vehicle. The coupling must resolve the shared dynamics of each model to create a consistent approach.

A spacecraft can be simulated using a six-degree-of-freedom model where the linear components of motion are known, chiefly described by Newton [20]. The angular components were later described by Euler [21, 22]. Taking into consideration inertial and non-inertial reference frames, the so-called Newton-Euler equations are expressed as

$$m\underline{\boldsymbol{I}}\ddot{\boldsymbol{r}}_{o,o'/o} - m\boldsymbol{r}^{\times}_{o',c}\dot{\boldsymbol{\omega}}_{o'/o} + m\boldsymbol{\omega}^{\times}_{o'/o}\boldsymbol{\omega}^{\times}_{o'/o}\boldsymbol{r}_{o',c} = \boldsymbol{f}, \tag{1.4}$$

$$m\boldsymbol{r}^{\times}_{o',c}\underline{\boldsymbol{I}}_c\ddot{\boldsymbol{r}}_{o,o'/o} - m\boldsymbol{r}^{\times}_{o',c}\boldsymbol{r}^{\times}_{o',c}\dot{\boldsymbol{\omega}}_{o'/o} + \boldsymbol{\omega}^{\times}_{o'/o}\left(\underline{\boldsymbol{I}}_c - m\boldsymbol{r}^{\times}_{o',c}\boldsymbol{r}^{\times}_{o',c}\right)\boldsymbol{\omega}_{o'/o} = \boldsymbol{\tau}_c - \boldsymbol{r}^{\times}_{o',c}\boldsymbol{f}, \tag{1.5}$$

where the first equation conserves linear momentum and the second angular momentum. The mass, moment of inertia, and moment about the centre of gravity of the dry craft are denoted m, $\underline{\boldsymbol{I}}_c$, and $\boldsymbol{\tau}_c$ respectively. $\underline{\boldsymbol{I}}$ is the 3×3 identity matrix. The force acting on the dry mass is symbolised by $\boldsymbol{f}$. The subscripts $/o$ and $/o'$ denote the change in time of quantities relative to the inertial and non-inertial frame respectively. The position vector from the non-inertial frame to the centre of mass and from the inertial frame to the non-inertial frame are respectively represented by $\boldsymbol{r}_{o',c}$ and $\boldsymbol{r}_{o,o'}$. The superscript '$\cdot$' denotes a derivative with respect to time, and the '$\times$' superscript, the cross-product matrix operator. Finally, the angular velocity is represented $\boldsymbol{\omega}_{o'/o}$. To track the attitude of the rigid body, and account for rotational degrees of freedom, quaternions are used.

For Fluid-Structure Interaction (FSI) calculations, a set of coupled governing equations must ultimately agree on the state of a set of shared, or interfaced, variables. It is typical at the interface, that the solid provides kinematic data, $\boldsymbol{k}_m$, containing displacement, velocity, and acceleration. The fluid responds via kinetic, $\boldsymbol{k}_t$, i.e. forces and torques. Consider a rigid body, governed by Equation (1.4) and (1.5), which maps a given kinetic state to some output kinematic state by

$$\boldsymbol{k}_m = \mathcal{S}\left(\boldsymbol{k}_t\right). \tag{1.6}$$

The fluid, through Equations (1.1) and (1.2), maps some given kinematic state to some output force and torque as

$$\boldsymbol{k}_t = \mathcal{F}\left(\boldsymbol{k}_m\right). \tag{1.7}$$

Here a simple imposition of coupling could be the kinematic state, hence,

$$\boldsymbol{k}_m = \mathcal{S}(\mathcal{F}(\boldsymbol{k}_m) + \boldsymbol{f}_{ex}), \tag{1.8}$$

where $\boldsymbol{f}_{ex}$ is an external force (e.g. a thruster). There are broadly two ways to construct a numerical model of the above system, monolithic and partitioned.

Monolithic approach embed one governing equation inside the other, or otherwise providing one solver with detailed knowledge of the state of the other. Thus creating a single or modified set of governing equation(s), where examples include [23–25]. While appealing, these approaches lack the flexibility and scalability that their counter parts, partitioned approaches, have [26].

Partitioned approaches, conversely, assemble a collection of 'black-box' physics solvers, coupled with further mathematics. This allows for a '*lego*' like construction of dynamic multi-physics analysis. Where component solvers can be added, omitted, and replaced with relative ease. Each solver has no 'knowledge' of the other, and only provides a response to a given kinematic or kinetic input. However, such methods are prone to numerical instability due to the added mass effect [27], which occurs where the mass ratio between the liquid and sold exceeds unity.

Thus care must be taken when formulating and implementing a strongly coupled partitioned solution procedure. An example is Atiken's Δ^2 method [4, 28, 29] for Fixed-Point Iteration (FPI) schemes. In this work a partitioned strongly coupled model for a spacecraft is presented. The aim is to demonstrate its robustness and generality. This in contrast to previous monolithic approaches [23, 30].

1.2 Original Contributions

The work laid out in this thesis builds on an existing code base, *Elemental*® (referred to hereafter as *Elemental*), which is capable of unstructured, incompressible [31, 32], two-phase flow modelling [3, 33]. The contributions outlined have been implemented in this base and include:

1. With regard to the CFD, the first novel contribution is the development of a new VoF initialisation tool for unstructured grids which is accurate, robust, and conservative. This work has been accepted as such through a recent publication [1].

2. Further contributions have been made to improve the popular unstructured algebraic VoF advection scheme namely CICSAM. In particular, the recent HiRAC [3] derivative is made volume conservative. Further, the HiRAC method is for the first time rigorously compared to both structured and recent unstructured geometric VoF methods.

3. Both CICSAM and HiRAC (HiRAC for the first time) are added to a well-balanced CSF surface tension model where height functions are used for curvature capturing purposes. A number of surface tension test cases are conducted to asses the use of CICSAM and HiRAC as advective methods for surface tension modelling.

4. A six degree-of-freedom numerical rigid body model is formulated. The fluid and the rigid body codes are then solved in a strongly coupled fashion. Where as other works have created spacecraft models, the novel contribution is the first high fidelity fully black box solver for spacecraft, which accounts for the orientation of the rigid body through the use of quaternions.

The above contributions are validated through a number of test cases which seek to both demonstrate that all sub-components are implemented correctly and the formulated approach is robust and has industrial value.

1.3 Outline

The thesis is organised into eight chapters, this chapter being the first. The next chapter introduces the *Elemental* incompressible model and relevant mathematics. Chapter 3 and 4 detail the work done on VoF modelling for unstructured grids,

through the creation of a new initialiser and significant improvements to algebraic VoF advective methods. To conclude work on the fluid aspects of the project, Chapter 5 deals with the inclusion of surface tension physics to the numerical model. Chapter 6 and 7 deal with the rigid body and the strong coupling of the fluid and rigid body. Finally Chapter 8 concludes the thesis, with a review of the work completed and suggests future research directions.

Chapter 2

The Elemental® Incompressible Flow Solver

2.1 Introduction

Many modern incompressible flow models can trace their origins back to work by Chorin [34–36]. Broadly, they can be dissected into two different approaches, artificial compressible methods, and split or fractional step methods. The former approach introduces an 'artifical' wave speed which relaxes the system. The method was expanded for unstructured, finite volume approaches by Malan et al. [32] where the artificial parameter was automatically selected based on the local flow characteristics.

The split approach typically involves three steps. First, an intermediate velocity projection is computed, then a Poisson like pressure equation is solved to ensure continuity, and, finally, a velocity correction step using the solved pressure. The characteristic based split (CBS) method, developed by Zienkiewicz et al. [37], builds on this while catering for a variety of temporal integration approaches and flow regimes (sub-, trans-, and supersonic flows).

Nithiarasu [38] unified the two unstructured incompressible approaches yielding a finite element split scheme with artificial compressibility, which took advantage of both approaches. The method was turned into an edge based approach by Malan and Lewis [39] for heat flow, and finally was modified to be used in the simulation of two phase flow, specifically within the VoF framework by Heyns et al. [33].

Elemental is a multi-physics simulation tool capable of compressible, incompressible, and free surface flow modelling. The preceding text laid out a brief history of their origin. *Elemental* is also capable of simulating flow coupled to flexible solid structures and, with the work of this thesis, rigid bodies. This chapter describes the *Elemental* mathematical base required for the chapters to follow.

2.2 Time Integration

2.2.1 Derivation

As with a classic fractional/split scheme the temporal term in (1.1) is split, namely

$$\frac{\rho\boldsymbol{u}|^{n+1} - \rho\boldsymbol{u}|^{n}}{\Delta t} = \frac{\rho\boldsymbol{u}|^{n+1} - \rho\boldsymbol{u}|^{*}}{\Delta t} + \frac{\rho\boldsymbol{u}|^{*} - \rho\boldsymbol{u}|^{n}}{\Delta t},$$

where

$$\frac{\rho\boldsymbol{u}|^{*}-\rho\boldsymbol{u}|^{n}}{\Delta t} = -\nabla\cdot\rho\boldsymbol{u}\otimes\boldsymbol{u}|^{n}+\nabla\cdot\mu\left(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^{T}\right)\Big|^{n}+\rho\boldsymbol{a}|^{n+1}, \tag{2.1}$$

$$\frac{\rho\boldsymbol{u}|^{n+1}-\rho\boldsymbol{u}|^{*}}{\Delta t} = -\nabla p^{n+1}+\boldsymbol{f}_{\sigma}^{n+1}. \tag{2.2}$$

The superscript $*$ denotes the projected velocity field. The surface tension and body acceleration terms are inserted into Equation (2.2) to ensure a well balanced scheme (further details are provided in Chapter 5). Equation (2.2) is rearranged, to compute the corrected velocity as

$$\boldsymbol{u}^{n+1}=\frac{\rho\boldsymbol{u}|^{*}-\Delta t\left(\nabla p^{n+1}-\boldsymbol{f}_{\sigma}^{n+1}\right)}{\rho^{n+1}}.$$

Next, setting $\rho^{*}=\rho^{n+1}$ and taking the divergence of both sides gives

$$\begin{aligned}\nabla\cdot\boldsymbol{u}^{n+1} &= \nabla\cdot\left[\boldsymbol{u}^{*}-\frac{\Delta t}{\rho^{n+1}}\left(\nabla p^{n+1}-\boldsymbol{f}_{\sigma}^{n+1}\right)\right],\\ 0 &= \nabla\cdot\left[\boldsymbol{u}^{*}-\frac{\Delta t}{\rho^{n+1}}\left(\nabla p^{n+1}-\boldsymbol{f}_{\sigma}^{n+1}\right)\right],\end{aligned} \tag{2.3}$$

where condition (1.2) has been applied, and yields the Poisson Equation. For the purpose of spatial discretization, the above is used to construct the face flux for the common face between two adjoining control volumes,

$$\begin{aligned}u_{ff} &= \boldsymbol{u}_{ff}\cdot A\hat{\boldsymbol{n}}|_{\partial\Omega_{i,j}},\\ \boldsymbol{u}_{ff}^{n+1} &= \left[\boldsymbol{u}^{*}-\frac{\Delta t}{\rho^{n+1}}\left(\nabla p^{n+1}-\boldsymbol{f}_{\sigma}^{n+1}\right)\right]_{\partial\Omega_{i,j}},\end{aligned} \tag{2.4}$$

where the control volume face coefficient is denoted $A\hat{\boldsymbol{n}}|_{\partial\Omega_{i,j}}$ and will be defined in section 2.3.3. The solution to p^{n+1} is obtained through Newton linearisation of Equation (2.3) (due the unstructured nature of the scheme [26]). To linearise, the residual of the system is defined,

$$\begin{aligned}\mathcal{R} &= \nabla\cdot\left[\boldsymbol{u}^{*}-\frac{\Delta t}{\rho^{n+1}}\left(\nabla p^{n+1}-\boldsymbol{f}_{\sigma}^{n+1}\right)\right],\\ \mathcal{R}^{\iota+1} &= \mathcal{R}^{\iota}+\Delta p\frac{\partial\mathcal{R}}{\partial p},\end{aligned} \tag{2.5}$$

where ι is the iteration index and $\mathcal{R}^{\iota+1}$ is converged to a specified user tolerance. The computation of the Jacobian, $\frac{\partial\mathcal{R}}{\partial p}$, is further detailed in section 5.2.1. The pressure at $n+1$ is then computed from

$$p^{n+1}=p^{n}+\sum_{\iota}\Delta p^{\iota}.$$

For the purpose of stability, the global time-step size, Δt, is computed as

$$\Delta t=\min\left\{\Delta t_{u},\Delta t_{\alpha},\Delta t_{\sigma}\right\},$$

where Δt_u is due the explicit treatment of momentum, Δt_α due the VoF advection equation and Δt_σ due the explicit treatment of surface tension.

The momentum restriction, Δt_u, is computed by

$$\Delta t_u = CFL \min_{i \in \Omega} \left[\sum_d \left(\frac{\Delta x_{i,\hat{\boldsymbol{e}}}}{|\boldsymbol{u}_{i,\hat{\boldsymbol{e}}}|} \right) + \frac{\varsigma \rho}{\mu} \left(\sum_d \frac{1}{\Delta x_{i,\hat{\boldsymbol{e}}}^2} \right)^{-1} \right],$$

where $CFL \in (0; 1)$ for explicit calculations [38–40] and d denotes spatial dimension. The nodal dual cell size in each dimension is represented $\Delta x_{i,\hat{\boldsymbol{e}}}$. The von Neumann number, ς, is set to 0.4.

For VoF calculations an additional stability criteria is placed,

$$\Delta t_\alpha = c_f \min_{i \in \Omega} \sum_d \frac{\Delta x_{i,\hat{\boldsymbol{e}}}}{|\boldsymbol{u}_i|},$$

where c_f is the Courant number. This time step restriction, Δt_α, limits the fraction of a volume that the interface can travel within a specific time-step. Finally, where applicable, a surface tension stability criterion is added as per Brackbill et al. [41] as

$$\Delta t_\sigma = \min_{i \in \Omega} \sqrt{\frac{(\rho_l + \rho_g)\, \Delta x_{i,\hat{\boldsymbol{e}}}^3}{4\pi\sigma}},$$

where σ is the surface tension coefficient, and l and g subscripts denote the liquid and gas phase densities.

2.2.2 Solution Procedure

The momentum equation solution procedure can be written out in semi-discretised form and is preceded by solving α^{n+1} as laid out in Chapter 4. The procedure follows:

$$\begin{aligned}
\boldsymbol{u}^* &= \frac{1}{\rho^{n+1}} \left[\rho \boldsymbol{u} - \Delta t \left(\nabla \cdot \rho \boldsymbol{u} \otimes \boldsymbol{u}|^n + \nabla \mu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T \right)\big|^n - \rho \boldsymbol{a}|^{n+1} \right) \right], \\
0 &= \nabla \cdot \left[\boldsymbol{u}^* - \frac{\Delta t}{\rho^{n+1}} \left(\nabla p^{n+1} - \boldsymbol{f}_\sigma^{n+1} \right) \right], \\
\boldsymbol{u}^{n+1} &= \boldsymbol{u}^* - \frac{\Delta t}{\rho^{n+1}} \left[\nabla p - \boldsymbol{f}_\sigma \right]^{n+1},
\end{aligned} \tag{2.6}$$

where p^{n+1} is obtained via the above outlined Newton linearisation procedure. Material properties for density and viscosity are computed respectively as

$$\rho(\alpha) = \alpha \rho_l + (1 - \alpha)\, \rho_g, \tag{2.7}$$

$$\mu(\alpha) = \alpha \mu_l + (1 - \alpha)\, \mu_g. \tag{2.8}$$

This work includes extending the above solution procedure to notionally second order in time. This was achieved using the 'predictor-corrector' method laid out by Tryggvason et al. [16] which reads

$$\begin{aligned}
\boldsymbol{u}^{\iota+1} &= \boldsymbol{u}^n + \Delta t \mathcal{L}(u^n), \\
\boldsymbol{u}^{\iota+2} &= \boldsymbol{u}^{\iota+1} + \Delta t \mathcal{L}(u^{\iota+1}), \\
\boldsymbol{u}^{n+1} &= \frac{1}{2}(\boldsymbol{u}^n + \boldsymbol{u}^{\iota+2}),
\end{aligned}$$

with $\mathcal{L}$ being some linear operator for the discrete spatial terms.

2.3 Spatial Discretisation

In Chapter 3 the geometric properties of median duel cells are called upon extensively. This section lays out the required computational geometry definitions, which follows.

2.3.1 Polytope Definitions

A polytope (polygon or polyhedron), Ω, is defined as the multiset $\Omega(\boldsymbol{X}, \boldsymbol{\partial\Omega}) = \{\boldsymbol{X}, \boldsymbol{\partial\Omega}\}$ where $\boldsymbol{X} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}\}$ represents a set of unique vertices and $\boldsymbol{\partial\Omega}$, the faces, defined as $\boldsymbol{\partial\Omega} = \{\partial\Omega_0, \partial\Omega_1, \ldots, \partial\Omega_{m-1}\}$. Here some face, $\partial\Omega_i$, is defined as a set of indices of the vertices in $\boldsymbol{X}$ which uniquely define the face.

Note that $\boldsymbol{X}$ may contain vertices not in Ω. Hence, once a vertex index is removed from all faces, in $\boldsymbol{\partial\Omega}$, it may be considered discarded without the need to remove it from $\boldsymbol{X}$.

The vertices, or indices referring to vertices, on each face are ordered specifically such that the face normals point out of the polytope. This property is derived from the gambit neutral format [42]. The outward-pointing normals are computed by

$$\hat{\boldsymbol{n}}_f = \begin{cases} \hat{\boldsymbol{r}}_{0,1} \times \hat{\boldsymbol{e}}_z & \text{if } 2D \\ \frac{\boldsymbol{r}_{0,1} \times \boldsymbol{r}_{1,2}}{|\boldsymbol{r}_{0,1} \times \boldsymbol{r}_{1,2}|} & \text{if } 3D \end{cases}, \tag{2.9}$$

where $\hat{\boldsymbol{e}}_z$ is the unit vector along the z-axis and $\boldsymbol{r}$ is an edge direction vector on a face. Note, since control volumes are decomposed into simplexes, and face normals computed from these sub-simplexes, non-planar faces are naturally accounted for (approximated). The subscripts of $\boldsymbol{r}$ denote the vertices of the face to which the edge belongs, and the direction of construction. Thus,

$$\boldsymbol{r}_{j,k} = \boldsymbol{x}_{\partial\Omega_i(k)} - \boldsymbol{x}_{\partial\Omega_i(j)} \quad j, k \in [0, \#\partial\Omega_i) \wedge j, k \in \mathbb{Z}, \tag{2.10}$$

where $\partial\Omega_i$ is a face of Ω and $\#\partial\Omega_i$ denotes the cardinality of $\partial\Omega_i$.

2.3.2 Simplex Volume

The volume of a 2- and 3- simplex ($\Omega_s(\boldsymbol{X}, \boldsymbol{\partial\Omega})$), triangle(2D) or tetrahedron(3D), is defined by

$$V_s = \begin{cases} \frac{1}{2} \left|(\boldsymbol{x}_1 - \boldsymbol{x}_0) \times (\boldsymbol{x}_2 - \boldsymbol{x}_0)\right| & \text{if 2-simplex} \\ \frac{1}{6} \left|(\boldsymbol{x}_1 - \boldsymbol{x}_0) \right. & \\ \left. \times (\boldsymbol{x}_2 - \boldsymbol{x}_0) \cdot (\boldsymbol{x}_3 - \boldsymbol{x}_0)\right| & \text{if 3-simplex} \end{cases}, \tag{2.11}$$

where the nomenclature is as previously defined.

2.3.3 Median Duel Cell Construction

The field equations are discretised on a vertex centred co-located finite volume mesh. This stands in contrast to the majority of other unstructured multi-phase codes which employ a cell centred approach to spatial discretisation. The computational cells are constructed using a median dual cell methodology as shown, for node i, in Figure 2.1.

Node i, with control volume denoted Ω_i, is decomposed in to a set of simplexes, $\Omega_{\boldsymbol{S}_i} = \{\Omega_{S_0}, \Omega_{S_1}, \ldots, \Omega_{S_m}\}$, as shown in the Figure 2.1. The volume, V_i, is computed

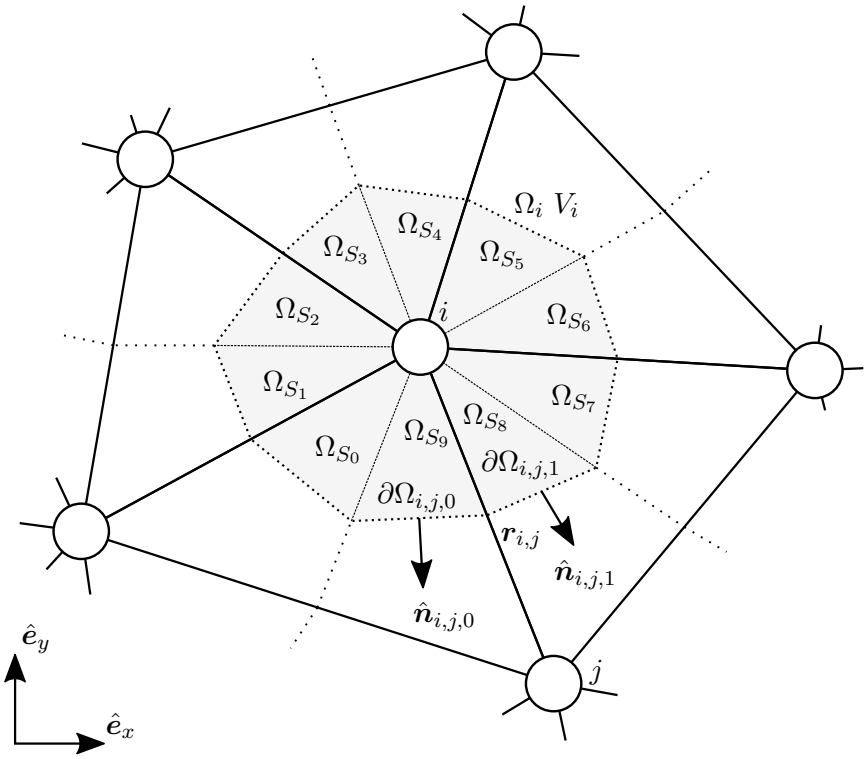


Figure 2.1: Diagram of a computational dual cell around node i.

using Equation (2.11) with $\Omega_{\boldsymbol{S}_i}$. Node i is connected to a neighbouring node j, where there are m neighbouring nodes to i, thus $j \in [0:m)$, with $\Omega_{\boldsymbol{S}_j}$ representing the set of simplexes for j.

Discretisation is conducted in an edge wise manner. For the edge $\boldsymbol{r}_{i,j}$, with length $|\boldsymbol{r}_{i,j}|$, connecting node i to j, the face coefficient, $A\hat{\boldsymbol{n}}$, is constructed from the shared facets, $\partial\Omega_{i,j}$, as

$$A\boldsymbol{n}|_{\partial\Omega_{i,j}} = \sum_{\partial\Omega \in \partial\Omega_{i,j}} A_{\partial\Omega}\hat{\boldsymbol{n}}_{\partial\Omega}, \tag{2.12}$$

where A and $\hat{\boldsymbol{n}}$ are computed as per Equations (2.11) and (2.9) respectively. Finally

$$\partial\Omega_i = \left\{\partial\Omega_{i,0}, \partial\Omega_{i,1}, \ldots, \partial\Omega_{i,j_{m-1}}\right\},$$

and represents the set of control faces for node i with face coefficients computed via Equation (2.12).

2.3.4 Velocity Projection Discretisation

The spatial derivatives, for some scalar ϕ, at nodes are approximated using Gauss-Green's Theorem,

$$\nabla\phi|_i \approx \frac{1}{V}\sum_{f\in\partial\Omega_i} \phi_f A\hat{\boldsymbol{n}}|_f,$$

where ϕ_f is, unless otherwise stated, approximated using central differencing. Where gradients at faces are required, an unstructured compact formulation [43] is used. The

gradient operator, $\nabla\phi|_{\partial\Omega_{i,j}}$, involves a central difference component enriched with the directional derivative along the edge and is expressed by

$$\nabla\phi|_{\partial\Omega_{i,j}} = \nabla\phi|_{cd} - \left[(\nabla\phi|_{cd} - \nabla_{ij}\phi)\cdot\hat{\boldsymbol{r}}_{i,j}\right]\hat{\boldsymbol{r}}_{i,j}, \tag{2.13}$$

where

$$\nabla\phi|_{\partial\Omega_{cd}} = \frac{1}{2}\left(\nabla\phi|_i + \nabla\phi|_j\right),$$
$$\nabla_{ij}\phi = \frac{\phi_i - \phi_j}{|\boldsymbol{r}_{i,j}|}\hat{\boldsymbol{r}}_{i,j},$$

and $\hat{\boldsymbol{r}}_{i,j}$ is the unit vector along the edge $\boldsymbol{r}_{i,j}$. The viscous term is discretised as

$$\nabla\cdot\mu\left(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T\right)\big|_i \approx \frac{1}{V_i}\sum_{f\in\partial\Omega_i}\mu(\alpha_f)\left(\nabla\boldsymbol{u}|_f + \nabla\boldsymbol{u}|_f^T\right)\cdot A\hat{\boldsymbol{n}}|_f \tag{2.14}$$

where the face dynamic viscosity is computed via Equation (2.8) using α_f, and the volume fraction at the face given by Equation (4.7). The advective component in Equation (2.1) is discretised by

$$\nabla\cdot\rho\boldsymbol{u}\otimes\boldsymbol{u}|_i \approx \frac{1}{V_i}\sum_{f\in\partial\Omega_i}\rho(\alpha_f)\boldsymbol{u}_f u_{ff},$$

where for consistency of mass and momentum flux it is critical to use Equation (4.7) for α_f in Equation (2.7). The upwinded velocity, $\boldsymbol{u}_f$, is computed with a 3rd order $\kappa-$scheme [44] and a van Albada flux limiter [45]. For this project the acceleration source term $\rho\boldsymbol{a}$ is added nodally in strong form, and accounts for the non-inertial nature of the fluid's reference frame,

$$\rho\boldsymbol{a}|_i = -\rho_i\left[\ddot{\boldsymbol{r}}_{o'/o} + \dot{\boldsymbol{\omega}}_{o'/o}\times\boldsymbol{r}_{o',i} + \boldsymbol{\omega}_{o'/o}\times\left(\boldsymbol{\omega}_{o'/o}\times\boldsymbol{r}_{o',i}\right) + 2\boldsymbol{\omega}_{o'/o}\times\boldsymbol{u}_i\right],$$

where the full definition of the symbols are detailed in Chapter 6, Equation (6.4). Briefly, $\boldsymbol{\omega}_{o'/o}, \ddot{\boldsymbol{r}}_{o'/o}$, and $\boldsymbol{\omega}_{o'/o}$ are the angular velocity and translational and angular accelerations of the fluid mesh with respect to the inertial reference frame. The nodal position and velocity are given $\boldsymbol{r}_{o',i}$ and $\boldsymbol{u_i}$. The divergence of the projected velocity field, $\boldsymbol{u}^*$, in Equation (2.3) is expressed discretely as

$$\nabla\cdot\boldsymbol{u}^*|_i \approx \frac{1}{V_i}\sum_{f\in\partial\Omega_i}\frac{1}{2}\left(\boldsymbol{u}_i^* + \boldsymbol{u}_j^*\right)\cdot A\hat{\boldsymbol{n}}|_f.$$

The discretisation of the remaining spatial terms will be detailed in Chapter 5.

2.3.5 Parallel Computing

To facilitated multi-core simulations the fluid domain is split into a number of sub-domains, see Figure 2.2. This is done using the METIS library [46]. At each domain boundary a single element overlap exists. The nodes in the overlapped element are then marked either as a 'master' or a 'slave'. 'Master' nodes have a complete control volume, and computations performed at master nodes yield results congruent to a serial calculation.

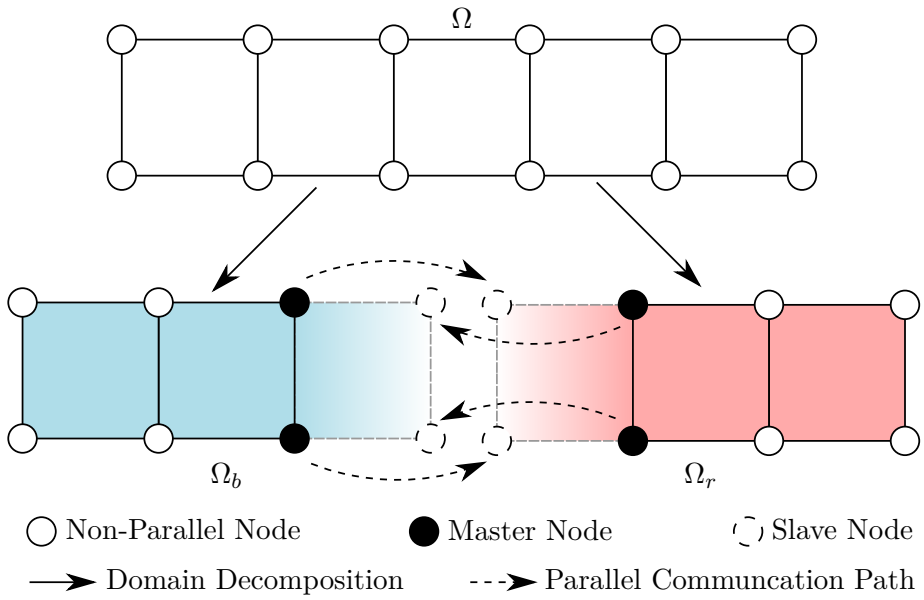


Figure 2.2: The decomposition for domain Ω into two sub-domains Ω_b and Ω_r and the resulting one element overlap with master and slave nodes drawn.

Using the Message Passing Interface (MPI) framework results that have been finalised at master nodes are communicated to their corresponding slaves. Note, while there may be multiple overlapping slave nodes, there will, due to the single element overlap, only ever be one master node. The communication in *Elemental* has been optimised to mask the cost of parallel communications. This enables linear parallel scaling down to ≈ 10000 nodes per domain.

Chapter 3

Arbitrary Grid Initialisation

3.1 Introduction

The dependence of two-phase physics on the local volume faction value highlights the importance of correctly initialising the field. For example, poor initialisation can prevent schemes such as the height function method [47–49], used for surface tension curvature calculations, from obtaining second-order accuracy after initialization. In addition, strict mass conservation may not be adhered to.

Research on methods which efficiently initialise the volume fraction scalar field accurately have therefore recently become a topic of interest [9–11], where earlier work focused on recursive local mesh refinement [48]. In the work by Bná et al. [9,10] a new initialisation technique was created to initialise arbitrary implicit surfaces on Cartesian grids via numerical integration. Primarily for cell-centred approaches, the method yielded a celebrated initialiser named the Volume of Fluids Initialiser or VOFI.

Some preliminary work on unstructured methods has been done by Strobl et al. [11], where the developed method can accurately initialise a sphere. The method works by 'cutting' away portions of the sphere using planes constructed from tetrahedral faces and accounting for the overlapped volume. However, the cited scheme, referred to as Overlap, remains specialised for spherical surfaces.

This chapter extends on the above cited work, by presenting a scheme, named the Arbitrary Grid Initialiser (AGI), which is general both in terms of the surface definition as well as the type of mesh employed. AGI builds on concepts introduced by VOFI but aims to be generic with respect to mesh topology and removes the Overlap restriction to spherical surfaces. This constitutes the first generic conservative volume of fluids initialiser.

AGI first decomposes nodal mesh volumes (finite volumes) into 2- and 3-simplexes. These are then intersected with a free surface which is defined by an arbitrary continuous implicit function. The submerged portion of the intersected simplexes form a reference polytope. A number of parametrised quadrature operators, formulated to integrate at oblique angles to the principle axes, are applied over the intersected faces of the polytope to achieve high accuracy in computing volume fractions.

3.2 Arbitrary Grid Initialiser Overview

The VoF method represents the fraction of some reference fluid occupying the control volume of node i, denoted Ω_i, as α_i. Thus $\alpha \in [0, 1]$ and is expressed as

$$\alpha_i = \frac{V_{r,i}}{V_i}, \tag{3.1}$$

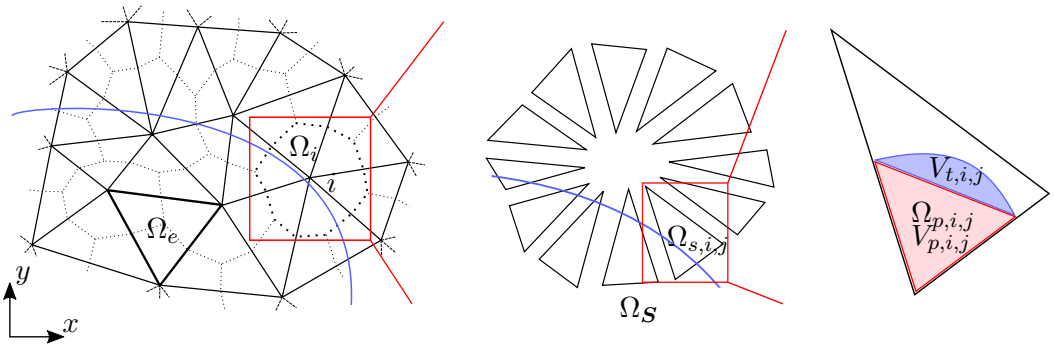


Figure 3.1: The process of computing the volume fraction, α_i, around a node; an unstructured grid with an arbitrary duel cell Ω_i (left) decomposed into a set of 2-simplexes, $\Omega_{\boldsymbol{S}}$, (middle). The computation of volume fraction for a single sub simplex, $\Omega_{i,j}$ (right).

where V_i represents the volume of Ω_i and $V_{r,i}$ the reference fluid volume contained in Ω_i. With a varying number of spatial discretisation approaches used in CFD (cell/vertex-centred finite volumes), the exact configuration of the computational cell will vary. To demonstrate generality the arbitrary median dual cell Ω_i resulting from the unstructured vertex-centred method (see Figure 3.1 left) is considered. However, the final computational cell will always be some form of polytope.

The proposed method recursively reduces complexity for both 2D and 3D, at which point a volume calculation can be performed. Grid complexity is respectively reduced by subdividing each computational cell into a set of j_m 2- or 3-simplexes, $\Omega_{\boldsymbol{S}}$, where the summed volume and reference volume are as follows:

$$V_i = \sum_{j}^{j_m} V_{i,j},\ V_{r,i} = \sum_{j}^{j_m} V_{r,i,j}. \tag{3.2}$$

Each simplex, $\Omega_{s,i,j} \in \Omega_{\boldsymbol{S}}$, is then checked for intersection with the free surface. If intersected, a reduced or reference polytope, Ω_p, is created as shown in Figure 3.1. The polytope will have an intersection line or plane, formed by the connection of the intercept vertices. The volume, V_p, is computed as the first step, of two, to computing the reference fluid volume. The second step completes the procedure by computing the trim reference fluid volume $V_{t,i,j}$ between the intersected polytope face and the free surface. The final reference volume for each simplex is given

$$V_{r,i,j} = V_{p,i,j} + V_{t,i,j}. \tag{3.3}$$

Note that volume in 2D refers to area with unit depth, and for brevity, subscripts i and j in further reference to V and Ω are omitted.

3.3 Polytope Tooling

3.3.1 Polytope Decomposition

Nodal control volumes are created, for vertex-centred approaches, by connecting edge, element, and face centres to the computational vertex to form a set of 2- or 3-simplexes, around node i. The procedure is detailed by Algorithm 1. For cell-centred discretisations, the computational vertex and element centroids are swapped. A mesh element is depicted in Figure 3.1 left as Ω_e.

Algorithm 1: Decompose Polytope to Simplexes

$\Omega_S = [\,]$; // Array of j_m Simplexes
foreach *element* $\Omega_e(\boldsymbol{X}, \boldsymbol{\partial\Omega})$ *connected to* i **do**
  foreach *face* $\partial\Omega_j$ *in* Ω_e *connected to* i **do**
    set $k \leftarrow i$'s index in $\partial\Omega_j$;
    if *2D* **then**
      add $\Omega_S \leftarrow \{\frac{1}{2}(\boldsymbol{x}_{\partial\Omega_j(k)} + \boldsymbol{x}_{\partial\Omega_j(k-1)}), \boldsymbol{x}_{\partial\Omega_j(k)}, \overline{\boldsymbol{X}}_{\Omega_e}\}$;
      add $\Omega_S \leftarrow \{\boldsymbol{x}_{\partial\Omega_j(k)}, \frac{1}{2}(\boldsymbol{x}_{\partial\Omega_j(k+1)} + \boldsymbol{x}_{\partial\Omega_j(k)}), \overline{\boldsymbol{X}}_{\Omega_e}\}$;
    else
      add $\Omega_S \leftarrow \{\boldsymbol{x}_{\partial\Omega_j(k)}, \frac{1}{2}(\boldsymbol{x}_{\partial\Omega_j(k)} + \boldsymbol{x}_{\partial\Omega_j(k-1)}), \overline{\boldsymbol{X}}_{\partial\Omega_j}, \overline{\boldsymbol{X}}_{\Omega_e}\}$;
      add $\Omega_S \leftarrow \{\boldsymbol{x}_{\partial\Omega_j(k)}, \overline{\boldsymbol{X}}_{\partial\Omega_j}, \frac{1}{2}(\boldsymbol{x}_{\partial\Omega_j(k+1)} + \boldsymbol{x}_{\partial\Omega_j(k)}), \overline{\boldsymbol{X}}_{\Omega_e}\}$;
    end
  end
end
return Ω_S;

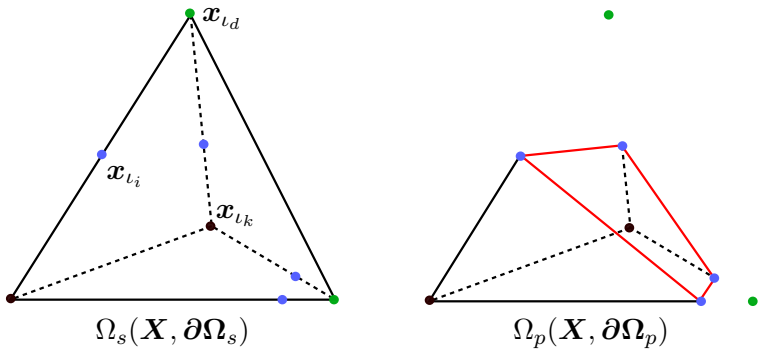


Figure 3.2: A tetrahedron, Ω_s, being cut along four edges to form an irregular triangular prism, Ω_p.

In Algorithm 1, $\overline{\boldsymbol{X}}_{\Omega_e}$ and $\overline{\boldsymbol{X}}_{\partial\Omega_j}$ are the mean co-ordinates of the polytope and face j respectively. In 3D, the face averaging process will convert non-coplanar faces into a set of 2-simplexes. These 2-simplexes are identically formed by adjoining polytopes sharing the same face, ensuring no overlapped volume is encountered in subsequent cell divisions. The total process of cell division allows for arbitrary grid connectivity.

3.3.2 Polytope Cutting

The intersection of some polytope, $\Omega(\boldsymbol{X}, \boldsymbol{\partial\Omega})$ and the free surface results in a volume reduced polytope, $\Omega'(\boldsymbol{X}', \boldsymbol{\partial\Omega}')$, with some arbitrary configuration. In this work this is exclusively the conversion of a simplex, Ω_s, to Ω_p, see Figure 3.2. In 3D, multiple intersections between polytopes and the free surface are expected, requiring an algorithm to efficiently reorganise the face tables of Ω_s. Algorithm 2 performs this operation and requires the faces, $\boldsymbol{\partial\Omega}_{\Omega_s}$, and a set containing the edge intercept data $\boldsymbol{\eta} = \{\eta_0, \eta_1, \ldots, \eta_m\}$ as inputs. Each intercept data entry, η_i, contains three integers flagging the index, in $\boldsymbol{X}_{\Omega_s}$, of the vertex to *keep*(ι_k), to *discard*(ι_d), and to *insert*(ι_i),

$$\eta_i = \{\iota_k, \iota_d, \iota_i\}\,. \tag{3.4}$$

To begin, Algorithm 2 creates an array of state flags, ς, for each vertex, which is populated via the passed intercept data. The three possible states are ς_k, ς_d, and ς_i for *keep*, *discard*, and *insert* states respectively.

Intercept vertices are then inserted into each face, to form a new set of faces. At this stage some vertices will still lack a state. A greedy type procedure is used, where vertices without a state take on a ς_k or ς_d state from their neighbours, respectively coloured black and green in Figure 3.2. The ς_i flag, coloured blue, acts as a 'boundary' preventing crossover of the other two states. These operations are carried out between lines 7 and 17 of Algorithm 2.

From lines 18 to 23, the vertices in each face which have the ς_d state are deleted from the new face table $\boldsymbol{\partial\Omega}_{\Omega_p}$. Since the inserted vertices were added between the original vertices on each face, gambit formatting of each face is maintained when vertices are discarded from the tables. In 3D, a face is deleted if the number of vertices it contains drops below 3.

In 2D the faces, which are edges (face with two vertices), of the polygon are converted to a single face at the start of Algorithm 2. After deleting all the ς_d vertices on the single face, it is converted back into a set of 'edge faces' after line 23 and returned.

For 3D, the algorithm continues by constructing a new face to close the reduced polytope. Gambit formatting dictates that two connected vertices appear in opposite order in the two faces containing both vertices. For example, if some vertex pair $\{v, w\}$ is seen on a face $\partial\Omega_i$, which is connected to some other face $\partial\Omega_j$, then in $\partial\Omega_j$ the pair will be seen as $\{w, v\}$.

To construct a new face, $\partial\Omega_n$, a list of new edges is created. The face table is constructed by linking the edges together, as the starting vertex of one edge is the end vertex of one of the other edges. This operation is conducted over lines 24 to 43. The new set of edges, $\boldsymbol{e}$, is constructed by reversing the order of every edge containing only ς_i vertices, as per line 27.

Algorithm 2 need not update $\boldsymbol{X}$, as noted in Section 2.3.1 only the face table requires an update to exclude vertices from Ω. Thus $\Omega_p(\boldsymbol{X}_p, \boldsymbol{\partial\Omega}_p)$ is congruent with $\Omega_p(\boldsymbol{X}, \boldsymbol{\partial\Omega}_p)$ so long as the inserted vertices already belong to $\boldsymbol{X}$, i.e. if $\boldsymbol{X}_p \subset \boldsymbol{X}$.

3.4 Numerical Tooling

3.4.1 Analytical Surface Definitions

AGI represents the free-surface implicitly as a level set function $g_i(x, y, z) = g_i(\boldsymbol{x})$, where the manifold $g_i(\boldsymbol{x}) = 0$ is interpreted as the position of the free surface or interface. For example, an implicit spherical definition would read as

$$g_i(\boldsymbol{x}) = a_0 \left[a_1 (x - a_4)^2 + a_2 (y - a_5)^2 + a_3 (z - a_6)^2 + 1 \right],$$

where $a_0 \rightarrow a_6$ would be the user defined coefficients. In practice, more complex surface definitions such as Zalesak's disk [50] may be required. A user can create complex shapes by creating a 'super' function which is composed of a number of level set functions, $\boldsymbol{g} = \{g_0, g_1, ...g_m\}$. The super function returns the maximum level set value in the set $\boldsymbol{g}$, which allows for the formation of squares from line equations for example. This approach is similar to VOFI [9,10]. The 'super' function G is expressed by

$$G(\boldsymbol{g}, \boldsymbol{x}) = \max(g_0(\boldsymbol{x}), g_1(\boldsymbol{x}), \ldots, g_m(\boldsymbol{x})), \tag{3.5}$$

Algorithm 2: Polytope Edge Cut

$\varsigma = []$; `// Array of states for each vertex`
foreach *intercept* η_i *in* $\boldsymbol{\eta}$ **do**
  set $\varsigma(\eta_i(0)) \leftarrow \varsigma_k$;
  set $\varsigma(\eta_i(1)) \leftarrow \varsigma_d$;
  set $\varsigma(\eta_i(2)) \leftarrow \varsigma_i$;
end
$\boldsymbol{\partial\Omega}_{\Omega_p} = []$; `// Reduced Face Table`
foreach *face* $\partial\Omega_i$ *in* $\boldsymbol{\partial\Omega}_{\Omega_s}$ **do**
  foreach *vertex* v *in* $\partial\Omega_i$ **do**
    add $\partial\Omega'_i \leftarrow v$;
    foreach *intercept* η_i *in* $\boldsymbol{\eta}$ **do**
      if $\{\eta_i(0), \eta_i(1)\} = \{v, v+1\}$ **or** $\{v+1, v\}$ **then** add $\partial\Omega'_i \leftarrow \eta_i(2)$;
    end
  end
  add $\boldsymbol{\partial\Omega}_{\Omega_p} \leftarrow \partial\Omega'_i$;
end
Greedy propagate state ς_k and ς_d;
foreach *face* $\partial\Omega'_i$ *in* $\boldsymbol{\partial\Omega}_{\Omega_p}$ **do**
  foreach *vertex* v *in* $\partial\Omega'_i$ **do**
    if $\varsigma(v) = \varsigma_d$ **then** delete v ;
  end
  if *3D and size of* $\partial\Omega'_i < 3$ **then** delete $\partial\Omega'_i$;
end
$\boldsymbol{e} = []$; `// Array of edges on the new face`
foreach *face* $\partial\Omega'_i$ *in* $\boldsymbol{\partial\Omega}_{\Omega_p}$ **do**
  foreach *vertex* v *in* $\partial\Omega'_i$ **do**
    if $\varsigma(v)$ *and* $\varsigma(v+1) = \varsigma_i$ **then** add $\boldsymbol{e} \leftarrow \{v+1, v\}$;
  end
end
while $\boldsymbol{e}$ *not empty* **do**
  $\partial\Omega_n = []$; `// New face`
  set $\{\partial\Omega_n(0), \partial\Omega_n(1)\} \leftarrow \{\boldsymbol{e}_0(0), \boldsymbol{e}_0(1)\}$;
  delete $\boldsymbol{e}_0$;
  foreach $\boldsymbol{e}_i$ *in* $\boldsymbol{e}$ **do**
    if $\partial\Omega_n(m-1) = \boldsymbol{e}_i(0)$ **or** $\boldsymbol{e}_i(1)$ **then**
      add $\partial\Omega_n \leftarrow$**if** $\partial\Omega_n(m-1) = \boldsymbol{e}_i(0)$ **then** $\boldsymbol{e}_i(1)$ **else** $\boldsymbol{e}_i(0)$;
      delete $\boldsymbol{e}_i$;
      reset $\boldsymbol{e}_i \leftarrow \boldsymbol{e}_0$;
    end
  end
  add $\boldsymbol{\partial\Omega}_{\Omega_p} \leftarrow \partial\Omega_n$;
end
return $\boldsymbol{\partial\Omega}_{\Omega_p}$;

and the free surface definition is taken to be the manifold satisfying $G(\boldsymbol{g}, \boldsymbol{x}) = 0 = G_0$.

3.4.2 Assumptions and Constants

With the interface G_0 and polytope elements Ω defined, a number of assumptions and constraints are stated. The objective of these is to maintain generality in the types of surfaces AGI can process as well as provide a set of rules to limit unreasonable complexity. The assumptions and constraints are as follows:

1. The types of surfaces specified in $\boldsymbol{g}$ are not allowed to be queried.
2. G must be continuous.
3. The analytical value of ∇g or ∇G is not provided.
4. $\#\left(G_0 \cap \Omega_E\right) > 0$ where Ω_E is the set of mesh elements.
5. $\#\left(G_0 \cap \boldsymbol{e}_i\right) \leq 2$ where $\boldsymbol{e}_i$ is an edge in some element Ω_e.
6. $G_0 \cap \partial\Omega_i$ is a single curve, where $\partial\Omega_i$ is a face in Ω_e.
7. The manifold G_0 is a single continuous manifold in Ω_e.
8. There exists a normal $\hat{\boldsymbol{n}}$ and coefficient $\beta(\boldsymbol{x})$ such that

$$[\forall \boldsymbol{x} \in \boldsymbol{X}, \exists! |\beta(\boldsymbol{x})| < \beta_{max} : \boldsymbol{x} + \beta\hat{\boldsymbol{n}} \in G_0 \wedge \text{sign}(\beta(\boldsymbol{x}_0)) = \text{sign}(\beta(\boldsymbol{x}_1))],$$

where

$$\hat{\boldsymbol{n}} = \sum_{j \in \#(G_0 \cap \boldsymbol{e})} \gamma_j \boldsymbol{R}_{\boldsymbol{e}}(j), \quad \gamma_j \geq 0$$
$$\boldsymbol{x}_0, \boldsymbol{x}_1 \in \boldsymbol{X},$$

and $\boldsymbol{R}_{\boldsymbol{e}}$ is the set of intersected unit edge direction vectors, orientated such that $\nabla G \cdot \boldsymbol{R}_{\boldsymbol{e}}(j) > 0$ along the edge. β_{max} is some maximum search length, and should be of order element size. Here, $\boldsymbol{X}$ is the union of all sets of points on every possible line (2D) or plane (3D) created from the intersection vertices in $G_0 \cap \boldsymbol{e}$. This condition ensures that the resulting trim volumes V_t are convex, or at least only 'mildly' concave.

The main objective of items 1 to 3 is to ensure generality of surfaces. Points 4 and onwards ensure that a conservative volume fraction calculation can be guaranteed via the outlined algorithm. Should one of these conditions be violated, it is recommended that the mesh be refined.

3.4.3 Root Searches

Root finding is required for two purposes, the first is to determine the intercept coordinate between G_0 and the edge of a simplex. Secondly, roots are used to construct the integration heights, H, defined in section 3.4.5. With the abundance of required root finding operations, computational speed is of importance. Additionally, root (and extrema) finding techniques must be capable of finding G_0 given that G may contain multiple surfaces.

Where $G(\boldsymbol{x}_0)G(\boldsymbol{x}_1) < 0$, a root is sought between the two vertices, $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ (see Figure 3.3). A point on the line segment between the vertices is defined as a parametrised function of λ,

$$\boldsymbol{x}(\lambda) = \boldsymbol{x}_0 + \lambda\left(\boldsymbol{x}_1 - \boldsymbol{x}_0\right) \Big| \lambda \in [0, 1]. \tag{3.6}$$

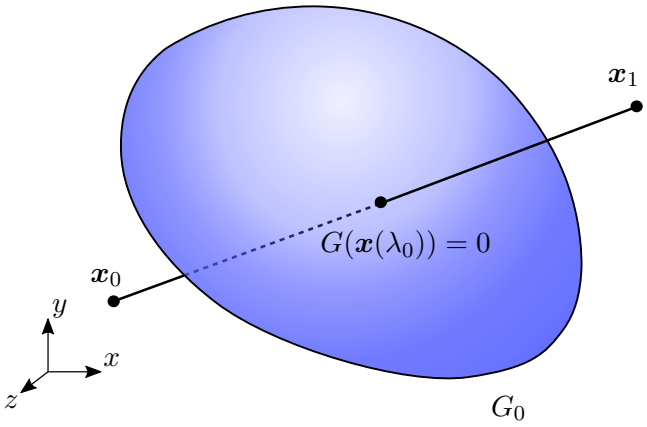


Figure 3.3: The manifold G_0, intersected by some line segment, where λ_0 is the λ value which zeroes G on $[\boldsymbol{x}_0, \boldsymbol{x}_1]$.

The parametrisation allows G to be expressed as $G(x, y, z) = G(\boldsymbol{x}(\lambda))$. To find the root one need only search for some $\lambda_0 \in \lambda$, that zeros G on $[\boldsymbol{x}_0, \boldsymbol{x}_1]$. The root is then expressed as

$$\boldsymbol{x}_r = \boldsymbol{x}(\lambda_0) \Big| G_0(\boldsymbol{x}(\lambda_0)) = 0. \tag{3.7}$$

Since ∇G is computed numerically, and G may be composed of multiple surfaces, methods relying on analytical or exact gradient information are not considered. The root finding method of Brent [51] is however a suitable candidate, balancing speed and robustness. It employs an inverse quadratic fit, with super-linear convergence, and a bisection method, with linear convergence, for cases where convergence is either slow or negative. Further, the bisection method ensures the root remains bracketed. The Brent method strikes a good balance between speed and robustness, as it guarantees a root on an intersected line segment.

3.4.4 Extrema Searches

An extrema search is conducted along an edge of a simplex when the two vertices share the same sign. This determines if a double intersection of G_0 along the edge exists. The search is conducted when $G(\boldsymbol{x}_0)G(\boldsymbol{x}_1) > 0$, with reference to Figure 3.4. For positive values of G, at the vertices, a minima is sought and, *vice versa*.

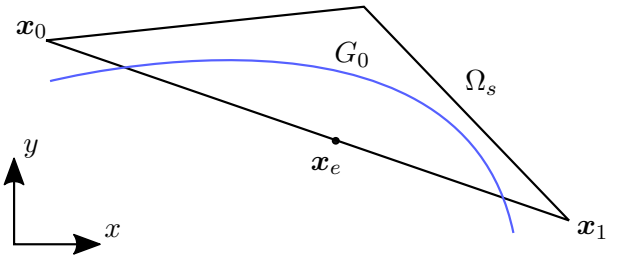


Figure 3.4: The double intersection of G_0 with the edge of a simplex(Ω_s), with edge vertices $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$. The extrema $\boldsymbol{x}_e$ of G along the edge is drawn.

Similar to root searches, it is required that the employed method comprise super-linear convergence where possible, but remains bounded around the extrema. In the work by Brent [51] an extrema search is proposed, using an inverse parabolic fit

as the super-linear method and a golden section search as the linear method. Like root searches along some parametrised line segment, $\boldsymbol{x}(\lambda)$(Equation (3.6)), a value of $\lambda_e \in \lambda$ that minimises $c_e G$ is sought such that the extrema is expressed by

$$\boldsymbol{x}_e = \boldsymbol{x}(\lambda_e) \Big| \lambda_e = \min_{\lambda} c_e G(\boldsymbol{x}(\lambda)), \tag{3.8}$$

where

$$c_e = \begin{cases} -1 & \text{if maxima} \\ 1 & \text{else} \end{cases}.$$

Finally in 3D, an intercept with only a face, *i.e.* no intersections with the edges, must be detected, see Figure 3.5. This occurs when all the vertices of a face share the same sign and all the edges contain extrema with the same sign. For face minimisations, a bounded Polak-Ribiere conjugate gradient(CG) method [52] is used. The underlying implementation follows that of Press et al. [28] which does not require the construction of a Hessian matrix. The line minimiser, used when searching for minima along conjugate directions is that of Brent.

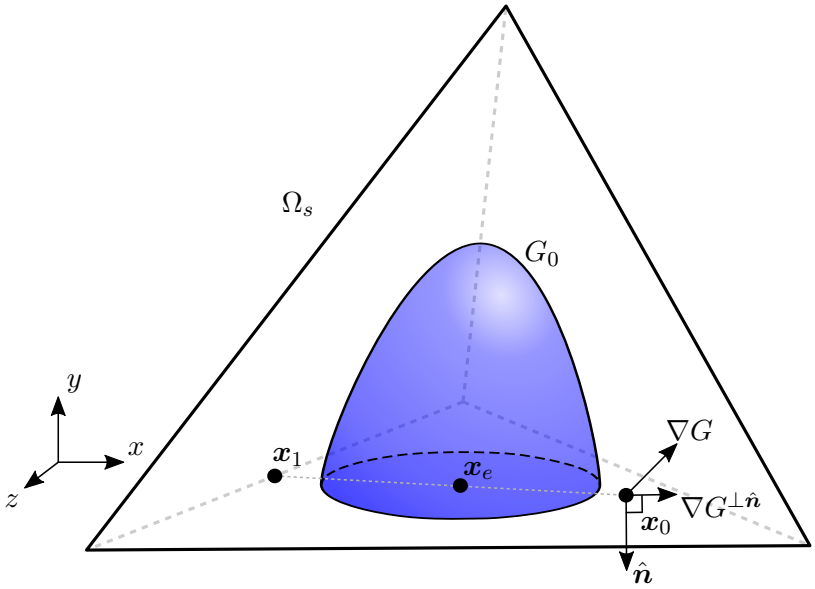


Figure 3.5: A simplex Ω_s intersected by G_0, only on a single face. The removal of face normal components of ∇G is shown as well as the bounding of the line search for the line segment $[\boldsymbol{x}_0, \boldsymbol{x}_1]$.

To solve Equation (3.8) over the simplex face, the CG minimiser is constrained. The component of the gradient of G that is normal to the simplex face is removed using

$$\nabla G^{\perp\hat{\boldsymbol{n}}} = \nabla G - (\nabla G \cdot \hat{\boldsymbol{n}})\hat{\boldsymbol{n}}, \tag{3.9}$$

where ∇G is approximated by a finite difference stencil in close proximity to the query point. Further, the end point, $\boldsymbol{x}_1$ in Figure 3.5, of the line segment for each line search of the CG minimiser is bound by the edges of the face. The solver is reset every 6 iterations to prevent solution stagnation.

3.4.5 Integral Parametrisation

To obtain high accuracy of volume calculation, AGI employs numerical quadrature over intersection lines and planes. To enable integration in arbitrary directions, a 'kernel' of predefined integration operators are created. Using the geometry of intersection, the various input parameters to the operators can be constructed, allowing the volume over the integration line or plane to be computed to high accuracy in a generic manner.

Consider a height function H, where the height is a fraction of the magnitude of some direction vector $\boldsymbol{r}$ (generally aligned with $\hat{\boldsymbol{r}}_v$, detailed shortly), starting at some point $\boldsymbol{x}$, and ending on the free surface, G_0 (given some set $\boldsymbol{g}$). Then,

$$H(G, \boldsymbol{x}, \boldsymbol{r}) = \begin{cases} |\boldsymbol{h}(\lambda_0) - \boldsymbol{x}| & \text{if } \exists G(\boldsymbol{h}(\lambda_0)) = 0 \\ 0 & \text{else} \end{cases}, \tag{3.10}$$

where

$$\boldsymbol{h}(\lambda) = \{\boldsymbol{x} + \lambda \boldsymbol{r} | \lambda \in [0, 1]\},$$

and where λ parametrises the line segment $\boldsymbol{h}$ and λ_0 is the value of λ that zeros G on $\boldsymbol{h}$, using Brent [51].

The integration operators conduct numerical integration in locally constructed, orthogonal axis systems. To provide integration limits to the operators, non-normalised direction vectors, $\boldsymbol{r}$, are used in the construction of the axis system. These axes, $\mathbb{R}'_2 = \{\boldsymbol{o}, (\boldsymbol{r}_u, \boldsymbol{r}_v)\}$ for 2D and $\mathbb{R}'_3 = \{\boldsymbol{o}, (\boldsymbol{r}_u, \boldsymbol{r}_v, \boldsymbol{r}_w)\}$ for 3D, are used extensively in the formulation of the operators. The operators and axis systems are shown in Figure 3.6. To compute the volume, H is integrated over these direction vectors, as well as additional bounding vectors. The volume in 2D (see Figure 3.6) is computed using a parametrised single integral of H with a constructed axis system, $\mathbb{R}'_2$, as

$$S(G, \mathbb{R}'_2, \hat{\boldsymbol{r}}_a) = \int_0^1 H(G, \boldsymbol{a}(\lambda), \boldsymbol{r}_v) |\boldsymbol{r}_u| d\lambda, \tag{3.11}$$

where

$$\boldsymbol{a}(\lambda) = \left\{ \boldsymbol{o} + \lambda \frac{|\boldsymbol{r}_u|^2}{\hat{\boldsymbol{r}}_a \cdot \boldsymbol{r}_u} \hat{\boldsymbol{r}}_a \middle| \lambda \in [0, 1] \right\},$$
$$\hat{\boldsymbol{r}}_a \cdot \boldsymbol{r}_u \neq 0,$$

and $\hat{\boldsymbol{r}}_a$ is a unit direction vector used to construct the parametrised line segment $\boldsymbol{a}$, which is used as the starting point for each height. The domain size along $\hat{\boldsymbol{r}}_u$ is set by the length of $\boldsymbol{r}_u$. S will have predictable behaviour, *i.e.* correctly compute the volume between G_0 and $\boldsymbol{a}$, when $\hat{\boldsymbol{r}}_a \cdot \boldsymbol{r}_u > 0$.

For 3D, two types of volume integrals are defined, both integrate S along the third axis $\hat{\boldsymbol{r}}_w$ but vary the local axis, used by S, by computing $\mathbb{R}'_2$ as a function of λ. The first operator integrates over some bounded region (in this work always a triangular polygon), namely,

$$R(G, \mathbb{R}'_3, \hat{\boldsymbol{r}}_a, \hat{\boldsymbol{r}}_{b0}, \hat{\boldsymbol{r}}_{b1}) = \int_0^1 S(G, \mathbb{R}'_2, \hat{\boldsymbol{r}}_a) \, |\boldsymbol{r}_w| \, d\lambda, \tag{3.12}$$

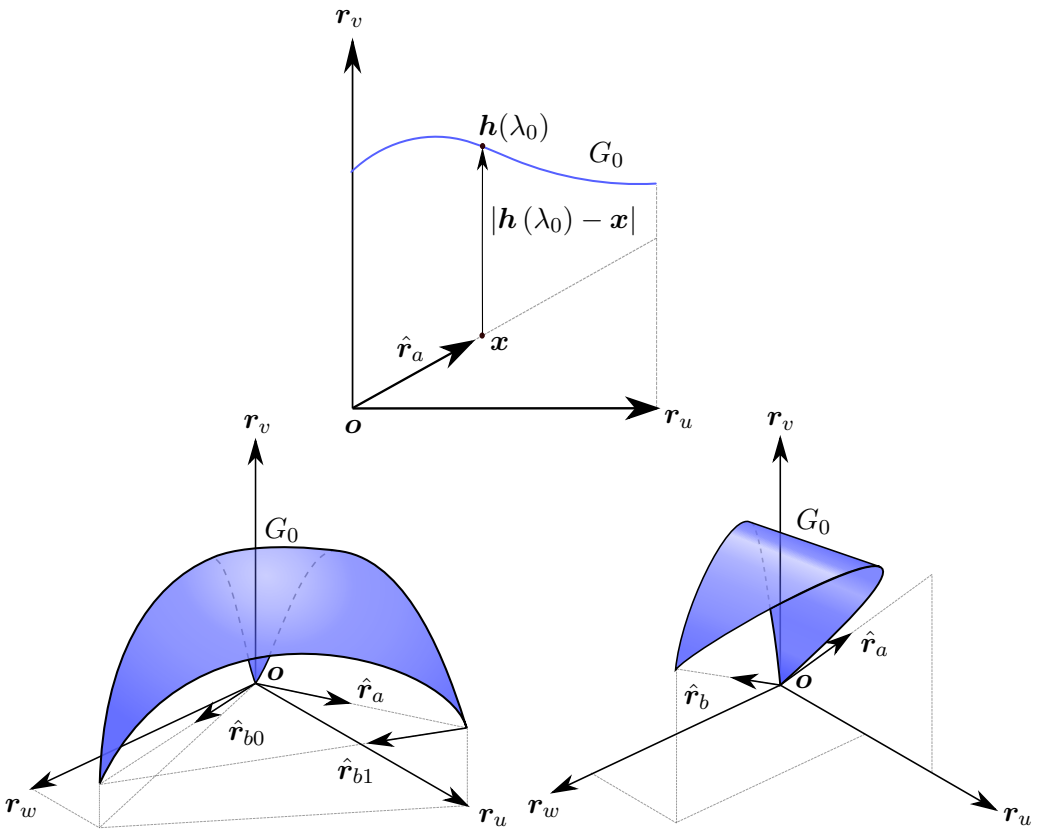


Figure 3.6: The integration operator S in $\mathbb{R}'_2$ (top). The integration operator, R, over some triangular plane (bottom left) and wedge integration operator W (bottom right) in $\mathbb{R}'_3$.

where

$$\mathbb{R}'_2 = \{\boldsymbol{b}_0(\lambda), ((\boldsymbol{b}_{1-0}(\lambda) \cdot \hat{\boldsymbol{r}}_u)\, \hat{\boldsymbol{r}}_u, \boldsymbol{r}_v)\}\,,$$
$$\boldsymbol{b}_0(\lambda) = \left\{ \boldsymbol{o} + \lambda \frac{|\boldsymbol{r}_w|^2}{\hat{\boldsymbol{r}}_{b0} \cdot \boldsymbol{r}_w} \hat{\boldsymbol{r}}_{b0} \middle| \lambda \in [0,1] \right\},$$
$$\boldsymbol{b}_1(\lambda) = \left\{ \boldsymbol{o} + \boldsymbol{r}_u + \lambda \frac{|\boldsymbol{r}_w|^2}{\hat{\boldsymbol{r}}_{b1} \cdot \boldsymbol{r}_w} \hat{\boldsymbol{r}}_{b1} \middle| \lambda \in [0,1] \right\},$$
$$\hat{\boldsymbol{r}}_a \cdot \boldsymbol{r}_u, \hat{\boldsymbol{r}}_{b0} \cdot \boldsymbol{r}_w, \hat{\boldsymbol{r}}_{b1} \cdot \boldsymbol{r}_w \neq 0,$$

where $\hat{\boldsymbol{r}}_a$, $\hat{\boldsymbol{r}}_{b0}$, and $\hat{\boldsymbol{r}}_{b1}$ are unit direction vectors and bound the planar 'region' under the surface being integrated. $\boldsymbol{b}_{1-0}$ denotes $\boldsymbol{b}_1 - \boldsymbol{b}_0$ and varies the size of the integration domain, along $\boldsymbol{r}_u$, for S. To ensure the volume is correctly computed, the lower bounding vector $\hat{\boldsymbol{r}}_a$ must be coplanar with the uv plane, and both $\boldsymbol{r}_{b0}$ and $\boldsymbol{r}_{b1}$ must point along $\boldsymbol{r}_w$, *i.e.* $\hat{\boldsymbol{r}}_{b0} \cdot \boldsymbol{r}_w, \hat{\boldsymbol{r}}_{b1} \cdot \boldsymbol{r}_w > 0$.

The second 3D volume integral is that of a wedge given by

$$W(G, \mathbb{R}'_3, \hat{\boldsymbol{r}}_a, \hat{\boldsymbol{r}}_b) = \int_0^1 S(G, \mathbb{R}'_2, \hat{\boldsymbol{r}}_a) |\boldsymbol{r}_w| d\lambda, \tag{3.13}$$

where

$$\begin{aligned}
\mathbb{R}'_2 &= \{\boldsymbol{b}(\lambda), ((H(G, \boldsymbol{b}(\lambda), \boldsymbol{r}_a)\hat{\boldsymbol{r}}_a \cdot \hat{\boldsymbol{r}}_u)\,\hat{\boldsymbol{r}}_u, \boldsymbol{r}_v)\}\,, \\
\boldsymbol{b}(\lambda) &= \left\{ \boldsymbol{o} + \lambda \frac{|\boldsymbol{r}_w|^2}{\hat{\boldsymbol{r}}_b \cdot \boldsymbol{r}_w} \hat{\boldsymbol{r}}_b \,\middle|\, \lambda \in [0,1] \right\}, \\
\boldsymbol{r}_a &= \frac{|\boldsymbol{r}_u|^2}{\hat{\boldsymbol{r}}_a \cdot \boldsymbol{r}_u} \hat{\boldsymbol{r}}_a, \\
\hat{\boldsymbol{r}}_a \cdot \boldsymbol{r}_u, \hat{\boldsymbol{r}}_b \cdot \boldsymbol{r}_w &\neq 0.
\end{aligned}$$

Here $\boldsymbol{r}_u$ is varied in $\mathbb{R}'_2$ by finding the intersection of the lower bounding vector $\boldsymbol{r}_a$ and G_0 using H (Equation (3.10)). Once again it is required that $\hat{\boldsymbol{r}}_a$ is co-planar with the uv plane and that $\hat{\boldsymbol{r}}_b \cdot \boldsymbol{r}_w > 0$ for the integral to produce a volume that is meaningful. It is worth noting that W should be used in regions close to G_0, where H will return a non-zero value, *i.e.* where $\boldsymbol{b}(\lambda) + \boldsymbol{r}_a$ contains a root of G_0.

3.4.6 Numerical Integration

AGI numerically evaluates the above parametric integrals using Legendre Gaussian Quadrature, S is approximated by

$$\begin{aligned}
S(G, \mathbb{R}'_2, \hat{\boldsymbol{r}}_a) &= \frac{|\boldsymbol{r}_u|}{2} \int_{-1}^{1} H\left(G, \boldsymbol{a}\left(\frac{1+\Lambda}{2}\right), \boldsymbol{r}_v\right) d\Lambda, \\
&\approx \frac{|\boldsymbol{r}_u|}{2} \sum_{i=1}^{m_i} w_i H\left(G, \boldsymbol{a}(\Lambda_i), \boldsymbol{r}_v\right),
\end{aligned} \tag{3.14}$$

where

$$\Lambda_i = \frac{1 + r_i}{2}.$$

Here w_i and r_i are the weights and Legendre roots. The total number of integration points is denoted m_i.

The region integral is approximated similarly via Legendre Gaussian Quadrature,

$$\begin{aligned}
R(G, \mathbb{R}'_3, \hat{\boldsymbol{r}}_a, \hat{\boldsymbol{r}}_{b0}, \hat{\boldsymbol{r}}_{b1}) &= \frac{|\boldsymbol{r}_w|}{2} \int_{-1}^{1} S\left(G, \mathbb{R}'_2, \hat{\boldsymbol{r}}_a\right) d\Lambda, \\
&\approx \frac{|\boldsymbol{r}_w|}{2} \sum_{i=1}^{m_e} w_i S\left(G, \mathbb{R}'_2, \hat{\boldsymbol{r}}_a\right),
\end{aligned} \tag{3.15}$$

where

$$\mathbb{R}'_2 = \{\boldsymbol{b}_0(\Lambda_i), ((\boldsymbol{b}_{1-0}(\Lambda_i) \cdot \hat{\boldsymbol{r}}_u)\,\hat{\boldsymbol{r}}_u, \boldsymbol{r}_v)\}\,,$$

and m_e is the external limit and is defined shortly. The wedge integral is expanded numerically as

$$\begin{aligned}
W(G, \mathbb{R}'_3, \hat{\boldsymbol{r}}_a, \hat{\boldsymbol{r}}_b) &= \frac{|\boldsymbol{r}_w|}{2} \int_{-1}^{1} S\left(G, \mathbb{R}'_2, \hat{\boldsymbol{r}}_a\right) d\Lambda, \\
&\approx \frac{|\boldsymbol{r}_w|}{2} \sum_{i=1}^{m_e} w_i S\left(G, \mathbb{R}'_2, \hat{\boldsymbol{r}}_a\right),
\end{aligned} \tag{3.16}$$

Table 3.1: Look up values for determining the number of integration points.

$\Delta\alpha$	m_*	$\lvert\boldsymbol{r}_u\rvert/\lvert\boldsymbol{r}_v\rvert$	m_i	$\lvert\boldsymbol{r}_w\rvert/\lvert\boldsymbol{r}_v\rvert$	m_e
0.0-0.1	4	0.0-0.1	$\min(m_*, 4)$	0.0-0.1	8
0.1-0.2	8	0.1-0.2	$\min(m_*, 8)$	0.1-0.3	12
0.2-0.4	12	0.2-0.4	$\min(m_*, 12)$	0.3-0.5	16
0.4-0.7	16	0.4-0.6	$\min(m_*, 16)$	0.5-1.0	20
0.4-1.0	20	0.6-1.0	$\min(m_*, 20)$	-	-

where

$$\mathbb{R}_2' = \{\boldsymbol{b}(\Lambda_i), ((H\,(G, \boldsymbol{b}(\Lambda_i), \boldsymbol{r}_a)\,\hat{\boldsymbol{r}}_a \cdot \hat{\boldsymbol{r}}_u)\,\hat{\boldsymbol{r}}_u, \boldsymbol{r}_v)\}\,.$$

The use of m_i and m_e is based on the recent work of Bná et al. [9,10], but generalised to account for the arbitrary axis system in which integration occurs. First, α defined in [10] (not to be confused with the volume fraction) is re-defined as

$$\alpha = \frac{\nabla_{\hat{\boldsymbol{r}}} G |\nabla_{\hat{\boldsymbol{r}}} G|}{2\nabla G \cdot \nabla G},$$

where $\nabla_{\hat{\boldsymbol{r}}} G$ is the directional derivative in the direction of some unit vector $\hat{\boldsymbol{r}}$. $\Delta\alpha$ is defined for every single integral S, as the absolute difference between $\alpha|_{\boldsymbol{a}(0)}$ and $\alpha|_{\boldsymbol{a}(1)}$. The non-dimensional characteristic lengths are defined as $|\boldsymbol{r}_u|/|\boldsymbol{r}_v|$ for S and $|\boldsymbol{r}_w|/|\boldsymbol{r}_v|$ for R and W. Finally, Table 3.1 is then used to select the number of quadrature points.

3.5 Reference Volume of a Simplex

3.5.1 2D Reference Volume

To begin, AGI seeds every grid point with a uniform volume fraction (α) value of unity or zero based on G at the node. To determine if a grid node requires a volume fraction computation the edges in the mesh are looped over. If the volume fraction is not uniform across the edge, the attached nodes' control volumes are decomposed into simplexes, via Algorithm 1, for further processing. A second edge loop is conducted to ensure no nodes are left unprocessed. To maintain computational efficiency, already processed nodes are skipped.

Each decomposed simplex, Ω_s, is checked for intersection with G_0 by interrogation of the sign of G at the simplex vertices. A sign change between two vertices implies an intersection along the connecting edge. An extrema search of G, along an edge, is conducted when no sign change is detected. A sign change at the extrema indicates that there exists two roots along the edge and the edge is flagged for splitting at the extrema. Ω_s is spilt into the least number of sub simplexes by only splitting those edges with splitting flags. Splitting at extrema helps to prevent further splitting.

When Ω_s contains no splitting flags but is intersected by G_0, a set of intersection vertices, $\boldsymbol{X}_{ri} = \{\boldsymbol{x}_{\partial\Omega_{ri}(0)}, \boldsymbol{x}_{\partial\Omega_{ri}(1)}\}$, is formed using a root search over the intersected edges, where the ri subscript refers to 'reference integration'. Using Algorithm 2 to cut Ω_s at the vertices in $\boldsymbol{X}_{ri}$, a fully submerged or reference polygon Ω_p is constructed (see Figure 3.7).

The volume of Ω_p is V_p from Equation (3.3) and can be computed directly with Equation (2.11) (subsequent to decomposition into simplexes where necessary), Figure 3.7 depicts Ω_p and its volume, shaded in red. The trim, V_t coloured blue in Figure

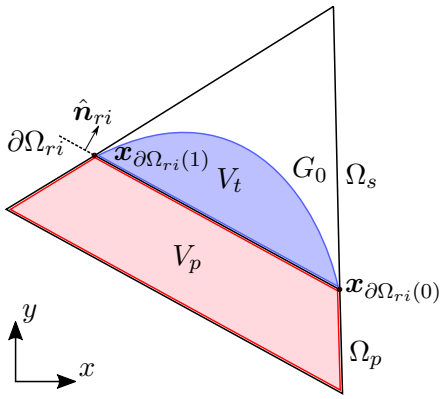


Figure 3.7: A 2-Simplex, Ω_s, intersected with the free surface G_0. The reference polygon, Ω_p, is shaded in red. The trim volume is shaded in blue.

3.7, is computed using numerical quadrature or Equation 3.11. It should be noted that simplexes not intersected by G_0, but which have vertices with $G \leq 0$, have V_p computed analytically via Equation (2.11).

To begin the integration of V_t, a plane normal, $\hat{\boldsymbol{n}}_{ri}$, is constructed to be orthogonal to the integration face $\partial\Omega_{ri}$ via Equation (2.9). In Figure 3.7, the trim is positive (or additive) to V_p since G_0 is external to Ω_p. To account for cases where G_0 protrudes into Ω_p a volume sign coefficient is introduced,

$$c_v\left(G, \boldsymbol{x}\right) = \begin{cases} 1 & \text{if } G(\boldsymbol{x}) \leq 0 \\ -1 & \text{else} \end{cases}. \tag{3.17}$$

This allows V_t to take on negative (or subtractive) volumes. In such instances, $\hat{\boldsymbol{n}}_{ri}$ must be flipped to ensure H produces heights. The general trim volume in 2D is calculated by

$$V_t = c_v S(G, \mathbb{R}'_2, \hat{\boldsymbol{r}}_{0,1}),$$

where

$$\mathbb{R}'_2 = \left\{\boldsymbol{x}_{\partial\Omega_{ri}(0)}, (\boldsymbol{r}_{0,1}, c_v l \hat{\boldsymbol{n}}_{ri})\right\},$$

and c_v is evaluated at $\overline{\boldsymbol{X}}_{ri}$, the mean of the set $\boldsymbol{X}_{ri}$. The projection length l is the hypotenuse of the bounding box of Ω_s. The subscripts to direction vectors refer to the vertices indexed by the indices in $\partial\Omega_{ri}$, such that $\hat{\boldsymbol{r}}_{0,1} = \boldsymbol{x}_{\partial\Omega_{ri}(1)} - \boldsymbol{x}_{\partial\Omega_{ri}(0)}$.

3.5.2 Considerations for Extension to 3D

In 3D, a further extrema search must be conducted after all edges have been checked for extrema. Should no edges on a face be intersected, the extrema of G on the face is sought. If the sign of G at the face extrema differs from the vertices, the face is flagged for splitting at the extrema. Further, the face and geometric centres of the simplex are pre-emptively moved to avoid subsequent splitting of the sub-simplexes by conducting extrema searches between the vertices of Ω_s and these centres and moving the centres to the extrema where required.

While in 2D, intersections of Ω_s yield two intercept vertices, 3D is more complex where two types of intersection configurations are encountered namely,

$$\boldsymbol{X}_{ri} = \{\boldsymbol{x}_{\partial\Omega_{ri}(0)}, \boldsymbol{x}_{\partial\Omega_{ri}(1)}, \boldsymbol{x}_{\partial\Omega_{ri}(2)}\},$$

or

$$\boldsymbol{X}_{ri} = \{\boldsymbol{x}_{\partial\Omega_{ri}(0)}, \boldsymbol{x}_{\partial\Omega_{ri}(1)}, \boldsymbol{x}_{\partial\Omega_{ri}(2)}, \boldsymbol{x}_{\partial\Omega_{ri}(3)}\}.$$

Further, the nature of the intersection of G_0 with the faces of Ω_s and their orientation with respect to the integration face, $\partial\Omega_{ri}$, complicates the computation of V_t. A single integral, as with 2D, is not possible and a new approach of composite integration is adopted. The trim volume is adjusted such that

$$V_t = V_R + V_W,$$

where V_R is the region volume computed via Equation (3.12) and V_W is the sum of wedge integrals, computed along the edges of the intersected face with Equation (3.13). The approach can be more clearly understood by projecting $G_0 \cap \Omega_s$ onto $\partial\Omega_{ri}$, to form a number of limit curves, and supposing no wedge integral was attempted. The limit curves are depicted in Figure 3.8 as green dashed lines and are co-planar to $\partial\Omega_{ri}$.

A limit curve will only be co-linear with an edge of $\partial\Omega_{ri}$ if the face of Ω_s, containing the edge, is orthogonal to $\partial\Omega_{ri}$. Where a limit curve falls inside $\partial\Omega_{ri}$ the manifold G_0 will overhang $\partial\Omega_{ri}$, and G_0 in this region will be external to Ω_s. Conversely, an under hang is formed when the limit curve falls outside $\partial\Omega_{ri}$. These are the shaded green sections of the manifold G_0 in Figure 3.8. If left unresolved, an overhang will unbound the volume fraction computation and an under hang will create an integration 'gap' which under-predicts the volume fraction when $c_v = 1$.

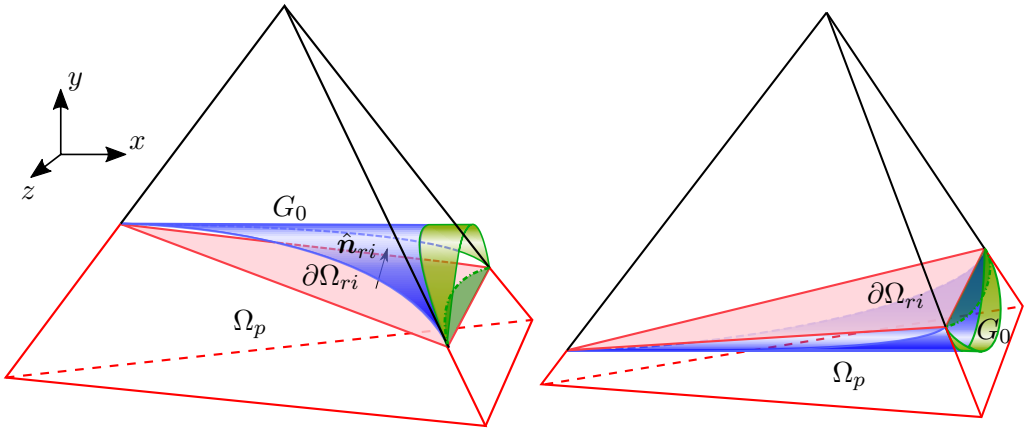


Figure 3.8: Showing cases of overhang(left) and under hang(right) for a 3-simplex intersected with a cylinder. The red face, $\partial\Omega_{ri}$, indicates the integration face in the reference volume, Ω_p. The bounded free surface is shaded blue and that part of free surface that over- or under-hangs the intersection face is coloured green, with its level curve, dotted green line, projected onto the plane containing $\partial\Omega_{ri}$.

Finally, a simple outward-pointing normal of the intersected face may not necessarily satisfy condition 8 in section 3.4.2. A new normal, $\hat{\boldsymbol{n}}'_{ri}$, is constructed from the

original normal, the mid-point of the intersected face, and the vertices $\boldsymbol{X}_{\Omega_s}$, as

$$\hat{\boldsymbol{n}}'_{ri} = \frac{\boldsymbol{x}_i - \overline{\boldsymbol{X}}_{ri}}{\left|\boldsymbol{x}_i - \overline{\boldsymbol{X}}_{ri}\right|} : \max_{\boldsymbol{x}_i \in \boldsymbol{X}_{\Omega_s}} \left(c_v \left(\overline{\boldsymbol{X}}_{ri}\right) \hat{\boldsymbol{n}}_{ri} \cdot \frac{\boldsymbol{x}_i - \overline{\boldsymbol{X}}_{ri}}{\left|\boldsymbol{x}_i - \overline{\boldsymbol{X}}_{ri}\right|} \right). \tag{3.18}$$

If the maximum value of the above dot-product is below $\sqrt{3}/2$, further candidate normals are constructed. The first set between the edge midpoints of Ω_s and $\overline{\boldsymbol{X}}_{ri}$, and the second between the closest point on the edges to $\overline{\boldsymbol{X}}_{ri}$ and $\overline{\boldsymbol{X}}_{ri}$. If any of these candidate vertices produce a higher value dot-product, they are selected to construct $\hat{\boldsymbol{n}}'_{ri}$.

3.5.3 3D Trim Volume

As mentioned, intersections of G_0 with Ω_s in 3D can yield three or four intersection co-ordinates. First the three vertex case is detailed, followed by the modifications needed to compute cases with four intersections.

The three intercepts, $\boldsymbol{X}_{ri} = \{\boldsymbol{x}_{\partial\Omega_{ri}(0)}, \boldsymbol{x}_{\partial\Omega_{ri}(1)}, \boldsymbol{x}_{\partial\Omega_{ri}(2)}\}$, are used to construct an intersection face normal $\hat{\boldsymbol{n}}_{ri}$, from which the modified normal $\hat{\boldsymbol{n}}'_{ri}$ is calculated as per Equation (3.18). A Gram-Schmidt (GS) orthogonalisation like procedure is followed to construct the local integration axes. For brevity, additional notation is introduced to truncate formulae; the orthogonal component for some vector $\boldsymbol{a}$ relative to some vector $\boldsymbol{b}$ is denoted via superscripts $\perp \boldsymbol{b}$ and defines the following operation,

$$\boldsymbol{a}^{\perp \boldsymbol{b}} = \boldsymbol{a} - \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{|\boldsymbol{b}|^2} \boldsymbol{b}.$$

The region volume can be computed directly from the intersection geometry by

$$V_R = c_v R(G, \mathbb{R}'_3, \hat{\boldsymbol{r}}_{0,2}, \hat{\boldsymbol{r}}_{0,1}, \hat{\boldsymbol{r}}_{2,1}), \tag{3.19}$$

where

$$\mathbb{R}'_3 = \left\{ \boldsymbol{x}_{\partial\Omega_{ri}(0)}, \left(\boldsymbol{r}_{0,2}^{\perp \hat{\boldsymbol{v}}}, c_v l \hat{\boldsymbol{n}}'_{ri}, (\boldsymbol{r}_{0,1}^{\perp \hat{\boldsymbol{v}}})^{\perp \hat{\boldsymbol{u}}} \right) \right\}.$$

The reader is reminded that the edge direction vector's subscripts indicate the vertices on the integration face $\partial\Omega_{ri}$. The axis must be constructed in order, such that the GS procedure produces an orthogonal $\mathbb{R}'_3$ axis, *i.e.* $\hat{\boldsymbol{v}}$ must be constructed before $\hat{\boldsymbol{u}}$, which must be constructed before $\hat{\boldsymbol{w}}$. The configuration of the axis is depicted in Figure 3.9.

Finally, the set of wedge integrals must be computed to account for over and under hangs. A new sign coefficient is introduced, c_w, to determine if G_0 over or under hangs a particular face and is expressed as

$$c_w \left(c_v, \hat{\boldsymbol{n}}'_{ri}, \hat{\boldsymbol{n}}_{\partial\Omega}\right) = \begin{cases} 1 & \text{if } c_v \hat{\boldsymbol{n}}'_{ri} \cdot \hat{\boldsymbol{n}}_{\partial\Omega} < 0 \\ 0 & \text{if } c_v \hat{\boldsymbol{n}}'_{ri} \cdot \hat{\boldsymbol{n}}_{\partial\Omega} = 0 \\ -1 & \text{else} \end{cases}, \tag{3.20}$$

where $\hat{\boldsymbol{n}}_{\partial\Omega}$ is an outward-pointing face normal of Ω_s. The volume contribution from the wedge integral is thus computed via

$$V_W = \sum_{i=0,1,2} c_v c_{w,i} W \left(G, \mathbb{R}'_{3,i}, c_v \hat{\boldsymbol{r}}_{\boldsymbol{n}_{\partial\Omega},i}, \hat{\boldsymbol{r}}_{i,i+1}\right), \tag{3.21}$$

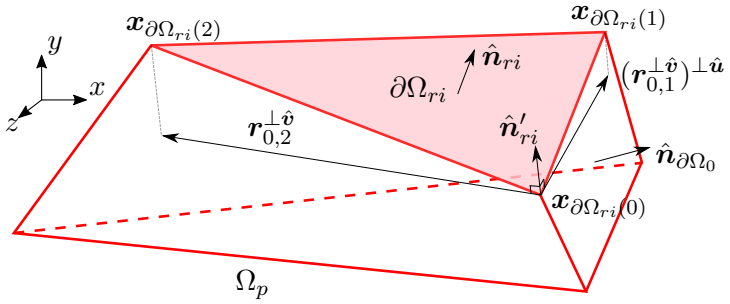


Figure 3.9: Geometry for a three vertex integration face, with the $\mathbb{R}'_3$ axis for the region operator drawn. $\hat{\boldsymbol{n}}_{\partial\Omega_0}$ will be the face normal used to construct $\hat{\boldsymbol{r}}_{\boldsymbol{n}_{\partial\Omega},0}$ for the first wedge integral.

where

$$\mathbb{R}'_{3,i} = \left\{ \boldsymbol{x}_{\partial\Omega_{ri}(i)}, \left(l\hat{\boldsymbol{r}}_{\boldsymbol{n}_{\partial\Omega},i}^{\perp\hat{\boldsymbol{v}}}, c_v l\hat{\boldsymbol{n}}'_{ri}, (\boldsymbol{r}_{i,i+1}^{\perp\hat{\boldsymbol{v}}})^{\perp\hat{\boldsymbol{u}}} \right) \right\},$$
$$\hat{\boldsymbol{r}}_{\boldsymbol{n}_{\partial\Omega},i} = \frac{\hat{\boldsymbol{n}}_{\partial\Omega_i} \times \boldsymbol{r}_{i,i+1}}{|\hat{\boldsymbol{n}}_{\partial\Omega_i} \times \boldsymbol{r}_{i,i+1}|},$$

and $\boldsymbol{r}_{i,i+1}$ represents the edge direction vector starting at the vertex, in $\partial\Omega_{ri}$, denoted by the subscript i and ending at $i+1$. To account for the last edge, where $i=2$, $i+1$ is reset to zero thus closing the face. $\hat{\boldsymbol{n}}_{\partial\Omega_i}$ is the face normal of the face in Ω_s that intersects $\partial\Omega_{ri}$ along $\boldsymbol{r}_{i,i+1}$ and is the second normal used to calculate c_w for an edge.

The intersection set of G_0 and Ω_s may yield four vertices, and further, that these vertices are not necessarily co-planar. The intersection face will be triangulated using $\overline{\boldsymbol{X}}_{ri}$ when Ω_p has its volume computed. The trim must follow this approach to ensure correctness of Equation (3.3). Thus $\overline{\boldsymbol{X}}_{ri}$ is appended to $\boldsymbol{X}_{ri}$,

$$\boldsymbol{X}'_{ri} = \boldsymbol{X}_{ri} \cup \left\{ \overline{\boldsymbol{X}}_{ri} \right\}.$$

The face $\partial\Omega_{ri}$ is broken into four sub faces,

$$\partial\Omega_{sr} = \left\{ \partial\Omega_{sr,0}, \partial\Omega_{sr,1}, \partial\Omega_{sr,2}, \partial\Omega_{sr,3} \right\},$$

where

$$\partial\Omega_{sr,j} = \{4, \partial\Omega_{ri}(j), \partial\Omega_{ri}(j+1)\} \quad j \in [0,1,2,3],$$

and where $\partial\Omega_{sr}$ is the set of the four new triangulated faces, the configuration has been drawn and shown in Figure 3.10. Again, where $j+1$ overflows the original array, it is reset to zero. The starting vertex of each sub face will be $\overline{\boldsymbol{X}}_{ri}$ which is placed at the end of $\boldsymbol{X}_{ri}$ and thus $\partial\Omega_{sr,j}(0) = \#\boldsymbol{X}_{ri} = 4$ for every sub face. Since the sub faces may not be co-planar, there may exist over or under hangs between the different sub faces. However, by maintaining a constant normal over all sub faces these over and under hangs will vanish.

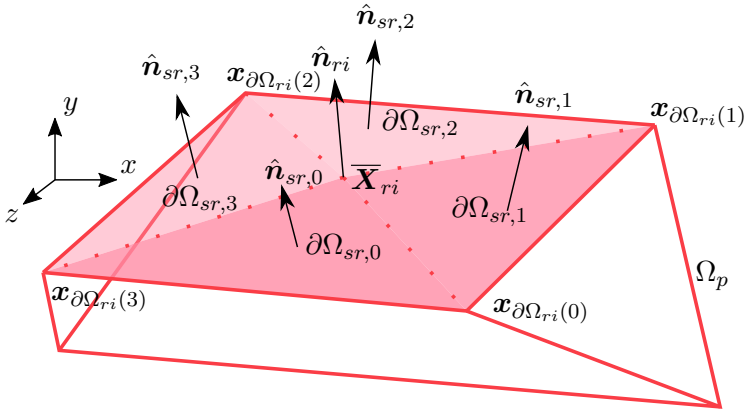


Figure 3.10: Trim geometry for a reference polytope with four intersection vertices, and the subsequent splitting of the integration face, $\partial\Omega_{ri}$, into sub-faces $\partial\Omega_{sr,0-3}$.

To compute $\hat{\boldsymbol{n}}_{ri}$, all the sub face normals, $\hat{\boldsymbol{n}}_{sr,j}$, are summed, and the resulting vector is normalised by

$$\hat{\boldsymbol{n}}_{ri} = \frac{\sum_j \hat{\boldsymbol{n}}_{sr,j}}{\left|\sum_j \hat{\boldsymbol{n}}_{sr,j}\right|}.$$

Equation (3.18) is again applied to produce $\hat{\boldsymbol{n}}'_{ri}$. The computation of V_R is computed via a sum of the region integrals over the sub faces given as

$$V_R = \sum_{j=0,1,2,3} c_v R(G, \mathbb{R}'_{3,j}, \hat{\boldsymbol{r}}_{j,4}, \hat{\boldsymbol{r}}_{j,j+1}, \hat{\boldsymbol{r}}_{4,j+1}),$$

where

$$\mathbb{R}'_{3,j} = \left\{\boldsymbol{x}_{\partial\Omega_{ri}(j)}, \left(\boldsymbol{r}_{j,4}^{\perp\hat{\boldsymbol{v}}}, c_v l \hat{\boldsymbol{n}}'_{ri}, (\boldsymbol{r}_{j,j+1}^{\perp\hat{\boldsymbol{v}}})^{\perp\hat{\boldsymbol{u}}}\right)\right\},$$

and where subscript j refers to the vertices in $\boldsymbol{X}'_{ri}$. The wedge volume can be computed via modification of Equation (3.21), thus

$$V_W = \sum_{j=0,1,2,3} c_v c_{w,j} W\left(G, \mathbb{R}'_{3,j}, c_v \hat{\boldsymbol{r}}_{\boldsymbol{n}_{\partial\Omega},j}, \hat{\boldsymbol{r}}_{j,j+1}\right),$$

where

$$\mathbb{R}'_{3,j} = \left\{\boldsymbol{x}_{\partial\Omega_{ri}(j)}, \left(l\hat{\boldsymbol{r}}_{\boldsymbol{n}_{\partial\Omega},j}^{\perp\hat{\boldsymbol{v}}}, c_v l \hat{\boldsymbol{n}}'_{ri}, (\boldsymbol{r}_{j,j+1}^{\perp\hat{\boldsymbol{v}}})^{\perp\hat{\boldsymbol{u}}}\right)\right\},$$

$$\hat{\boldsymbol{r}}_{\boldsymbol{n}_{\partial\Omega},j} = \frac{\hat{\boldsymbol{n}}_{f_j} \times \boldsymbol{r}_{j,j+1}}{|\hat{\boldsymbol{n}}_{f_j} \times \boldsymbol{r}_{j,j+1}|},$$

and need only be applied around the four edges of $\partial\Omega_{ri}$.

3.6 Results

3.6.1 Assorted 2D Surfaces

The first set of surfaces to be presented demonstrate the capability of AGI to initialise general functions on unstructured grids. To test the accuracy of AGI, the absolute error of the total initialised reference volume is computed,

$$\varepsilon = |V_n - V_a| \tag{3.22}$$

where ε and V_a are the absolute error and analytical reference volume respectively. V_n is the numerically computed reference volume and is obtained through the sum of the product of α_i and V_i, via Equation (3.1) in every cell. In all cases V_a is of order 1.

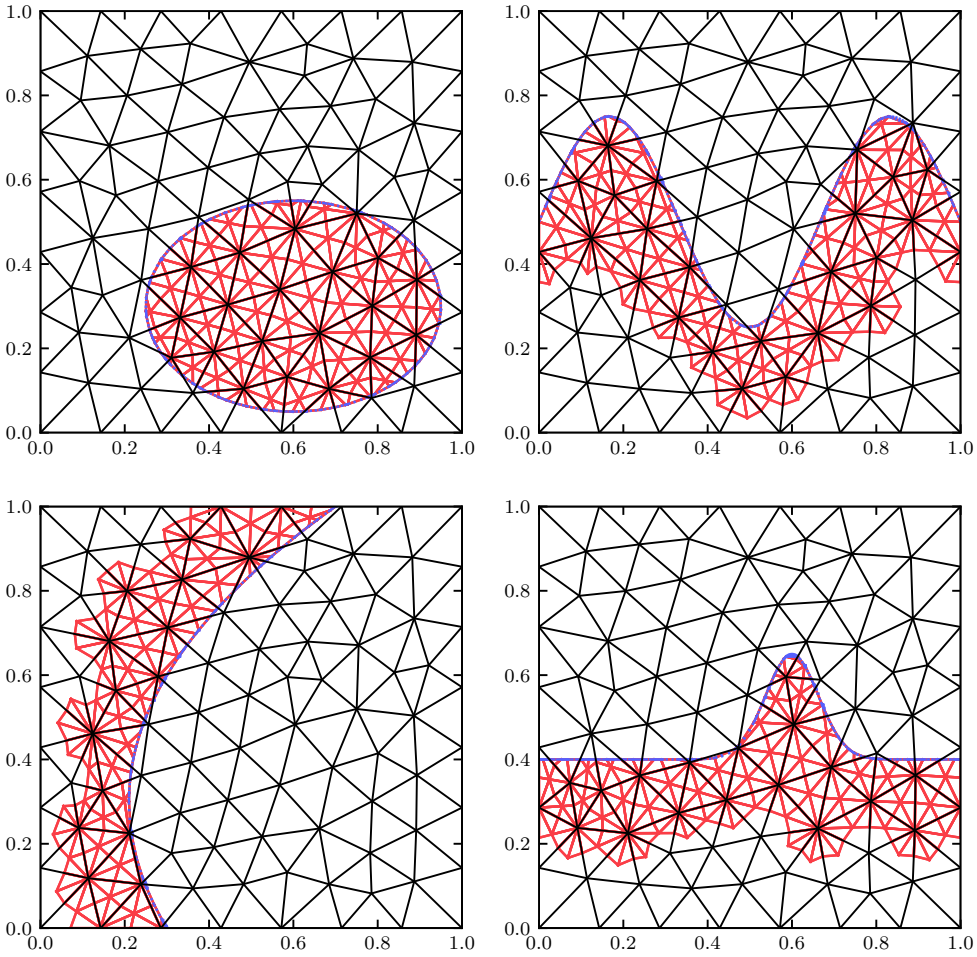


Figure 3.11: Showing the initialisation of an ellipsoidal(top left), sinusoidal(top right), parabolic(bottom left), and Gassian(bottom right) fields. The reference polygons (the wetted side of cut 2-simplexes) are depicted in red. Integration heights are drawn as blue line segments.

The fields are shown in Figure 3.11. In the figure the reference polytopes are depicted in red and the integration heights, computing the trim, are drawn in blue. The resulting absolute errors obtained were $1.18e^{-13}$ for the ellipse, $8.65e^{-12}$ for the sinusoidal field, $5.55e^{-17}$ for the parabola, and $7.36e^{-08}$ for the Gaussian field.

As depicted, reference polygons are only created in the vicinity of the interface for efficiency (See section 3.5).

Also clear in the figure are the integration heights, Equation 3.10. In regions of high curvature these are visible as a blue 'shell' where as for regions of low curvature these are not visible. The latter is due to the reference surface better approximating the interface, *i.e.* the integration height lines are extremely small.

3.6.2 2D Droplet Initialisation

Having demonstrated AGI working on general surfaces, the accuracy metrics must be evaluated. This case tests AGI's 2D performance as a function of mesh resolution (element size), number of integration points (m_i in Equation (3.14)) and element quality (equilateral triangles are considered ideal quality). This requires a large number of meshes as now described.

To begin the mesh generation process, seeding nodes were placed on each boundary in an equi-spaced manner. The interior of the domain was meshed via Delaunay triangulation. The Delaunay process was allowed to add internal nodes to optimise element spacing (equi-spacing). This constitutes a base mesh, of which six were generated by increasing the number of seed nodes on each boundary from 3 to 8.

To assess the sensitivity of AGI to element quality, the nodal co-ordinates in each base mesh were moved in a random manner. However, to retain legitimate finite volumes, nodal displacements were limited to no more that 50% of the radius of the inscribed circle of the node's control volume. Fifty such random meshes were created from each of the base meshes.

Table 3.2: Resulting mean ($\bar{\varepsilon}$), max (ε_m), and standard deviation (ε_σ) of absolute error for the 2D droplet initialization on an increasingly finer set of randomised base meshes. The set of integration points (m_i form Equation (3.14)) used was $3, 5, 7$.

No. node	ε_3	ε_5	ε_7
	$\bar{\varepsilon}$		
13	5.17e-08	8.03e-12	2.36e-15
21	4.23e-08	9.80e-12	4.21e-15
33	6.14e-09	4.25e-13	6.94e-17
48	4.64e-09	2.20e-13	3.00e-17
68	1.88e-09	5.56e-14	2.50e-17
91	2.57e-10	2.14e-15	2.16e-17
	ε_m		
13	1.64e-07	6.23e-11	3.66e-14
21	2.31e-07	1.02e-10	6.85e-14
33	2.38e-08	3.86e-12	8.88e-16
48	1.69e-08	1.28e-12	1.67e-16
68	1.16e-08	6.64e-13	8.33e-17
91	1.13e-09	2.23e-14	8.33e-17
	ε_σ		
13	2.68e-08	9.29e-12	5.18e-15
21	5.03e-08	1.85e-11	1.09e-14
33	5.55e-09	7.50e-13	1.49e-16
48	3.76e-09	3.02e-13	3.46e-17
68	1.88e-09	1.07e-13	2.10e-17
91	1.92e-10	3.40e-15	1.95e-17

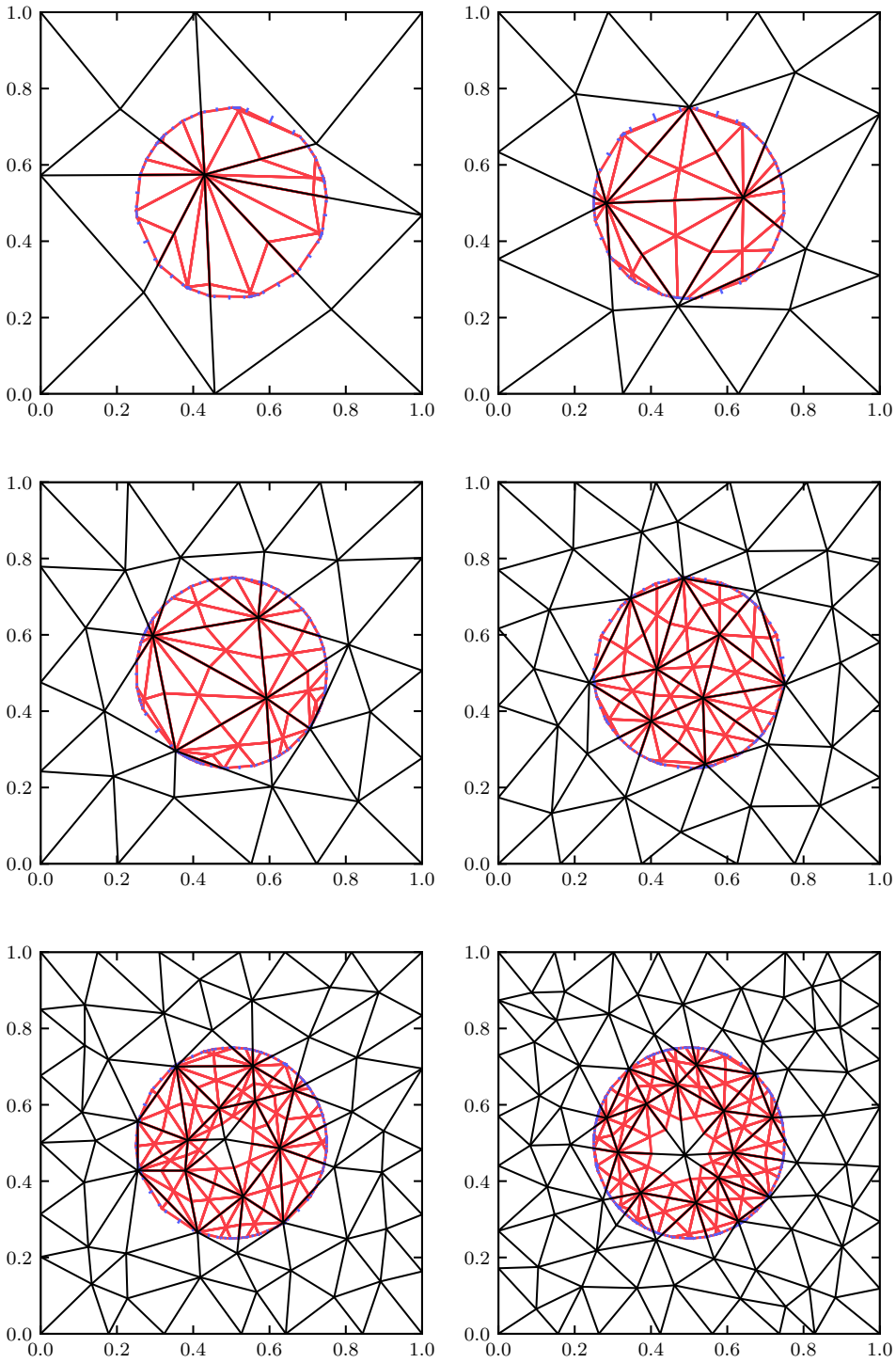


Figure 3.12: The random meshes for each refinement level, which caused the greatest error when m_i was set to 3. The red lines plot the reference polygons, while the blue lines represent integration heights on $\partial\Omega_{ri}$. In some cases splitting of the reference polygons can be seen, for example in the region in the middle left figure around the co-ordinate $[0.75, 0.4]$.

The volume fraction was computed on each of the randomised meshes for three different values of m_i, namely 3, 5, and 7. The implicit level set function selected for

describing the initial free surface was calculated by

$$g(x, y) = (x - 0.5)^2 + (y - 0.5)^2 + 0.25^2.$$

Similar to the previous case, the resulting volume and subsequent absolute errors were computed via Equation (3.22). The average ($\bar{\varepsilon}$), maximum (ε_m), and standard deviation (ε_σ) of errors were recorded for each refinement level and for each set of integration points. The results are reported in Table 3.2, where the numerical subscripts of ε denote m_i.

The results in Table 3.2 demonstrate the relative insensitivity of AGI to element quality, *i.e.* the standard deviation is of the same order as the mean. Additionally, as expected the error reduces with an increase in the number of integration points as well as a finer base mesh. It is noted that tests were conducted on a machine where *cfloat* reported double precision epsilon to be $2.22e^{-16}$. Errors of this order are therefore dominated by machine precision.

The grids resulting in the maximum absolute error from each refinement level for $m_i = 3$ have been plotted in Figure 3.12. Again the reference polygons have been coloured red and the integration heights in blue. On the coarser meshes, it is possible to see the numerical integration heights while as the grids get finer the blue 'smears' into a 'thin shell' over $\partial\Omega_{ri}$. Additionally visible is the reduction in length of the integration heights as the mesh gets finer and $\partial\Omega_{ri}$ better approximates G_0.

3.6.3 3D Droplet Initialisation

The first 3D test case is the extension of the previous 2D case. The base meshes were generated by seeding each of the faces of a unit cube with an equispaced structured grid. The size of the structured grid on each face was increased by one, from a 3×3 to a 5×5 grid, to form three base meshes. Each face was diagonalised to form triangular face elements. The volume mesh was then generated with Delaunay tetrahedralization, again allowing the procedure to add nodes to optimise for element quality. A similar randomisation process was again conducted to produce fifty grids from each base mesh. This to again assess the sensitivity of AGI to element quality.

The implicit function used to initialise the free surface was computed by

$$g(x, y, z) = x^2 + y^2 + z^2 - 0.5^2,$$

and the values of m_i and m_e varied between 3, 6, and 9. The resulting mean, maximum, and standard deviation of absolute error for each refinement level and fixed set of integration values are reported in Table 3.3. Again the trend shows that increasing the number of fixed integration points has a consistent and significant effect on the error in the computed volume. The mean error is reduced with increasing refinement, albeit slower than 2D.

Figure 3.13 shows the initialised volume for the coarsest mesh resulting in the maximum error when m_i and m_e was set to 3. The red lines depict the edges of the face $\partial\Omega_{ri}$, and the integration heights are again coloured blue. The grey lines on the top figure are the edges of the grid. The bottom, zoomed in, figure shows the integration faces, $\partial\Omega_{ri}$.

It is possible in Figure 3.13 to see integration planes with 3 and 4 intercepts. Where four intercepts occurred the density of integration heights is generally higher, owing to the integration face subdivision prior to integration (See section 3.5). When looking closely at the edges of the integration planes, the wedge integrals are also visible as blurred blue dots and lines in most cases.

Table 3.3: Resulting mean ($\bar{\varepsilon}$), max (ε_m), and standard deviation (ε_σ) of absolute error metrics for the 3D octant droplet initialization on an increasingly finer set of randomised base meshes. The set of integration points (m_i and m_e) used were $3, 6, 9$.

No. node	ε_3	ε_5	ε_7
	$\bar{\varepsilon}$		
34	1.36e-07	1.80e-10	1.12e-12
63	3.62e-08	5.30e-11	2.44e-12
112	1.04e-08	4.92e-13	4.66e-16
	ε_m		
34	4.85e-07	1.66e-09	1.70e-11
63	2.14e-07	5.58e-10	9.52e-11
112	3.76e-08	5.38e-12	1.33e-14
	ε_σ		
34	1.13e-07	3.27e-10	3.26e-12
63	4.21e-08	1.23e-10	1.35e-11
112	9.46e-09	8.44e-13	1.93e-15

Finally, one can see the faces of split simplexes (See section 3.5) which occurred in the front left corner of the sphere (Figure 3.13) where an increase in the number of integration faces is apparent. This test case demonstrates the robustness of AGI in 3D, where even for extremely poor aspect ratios, such as the split simplexes, it is still able to produce a result relatively close to machine precision when $m_i = m_e = 9$.

3.6.4 3D Sphere and Paraboloid Intersection Initialisation

The final tests case demonstrates the ability to use intersecting surfaces. The first intersecting surface was a sphere with its centre placed in the middle of a unit domain. The sphere's radius was chosen to be 0.25 and was intersected by a paraboloid, the second surface, along the plane $z = 0.5$. The implicit definition of the paraboloid was defined by

$$g(x, y, z) = -4\left(x^2 + y^2\right) + 4\left(x + y\right) + z - 2.25.$$

The mesh used was that of the base mesh which was seeded with a 5×5 structured grid on each face. The resulting Delaunay tetrahedralization of the volume yielded 112 nodes. The reference integration faces produced by AGI are plotted in red in Figure 3.14 and the ability of the algorithm to handle surfaces with discontinuous gradients is made apparent.

As with the previous case, the increase in accuracy was assessed via varying the number of integration points per case. Table 3.4 contains the resulting volume errors. The reduction in error is less dramatic than that of the spherical case, due to the discontinuity in gradient at the surface intersections. However, there is still a significant reduction in error with AGI proving accurate and robust.

3.7 Conclusion

In this chapter a new scheme is proposed to initialise the volume fraction field on an arbitrary grid given some arbitrary implicit surface definition of the liquid-gas interface. The method first simplifies grid complexity by decomposition of control

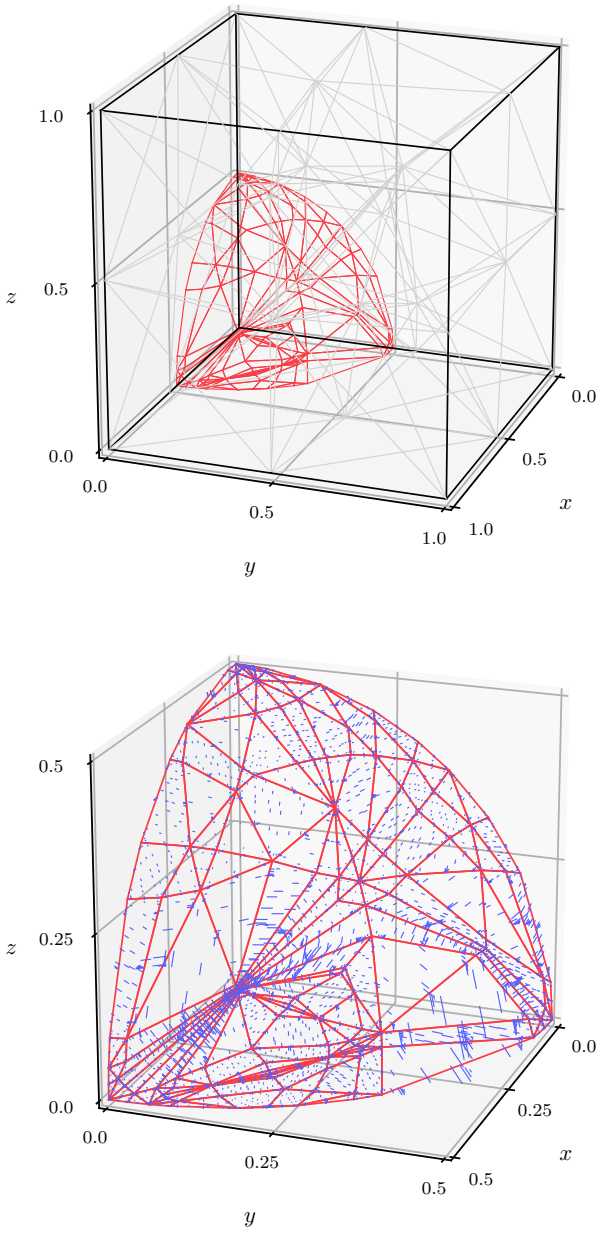


Figure 3.13: Showing the random grid on the coarsest refinement level causing the worst error when using $m_i = m_e = 3$. The top figure depicts the integration faces, $\partial\Omega_{ri}$, of the reference polyhedra, surrounded by the internal mesh edges coloured in grey. The outer 'bounding' edges of the domain have been coloured in black. The bottom figure depicts a close up view of the integration planes and includes the integration heights shaded in blue.

volumes into simplexes. Each simplex is then cut to create a wetted or reference polytope which is computed analytically while the volume between the intersected face and the free surface is computed to high accuracy via numerical quadrature *vis-à-vis* Legendre Gaussian quadrature.

The scheme was applied to a number of challenging test cases, involving a vast

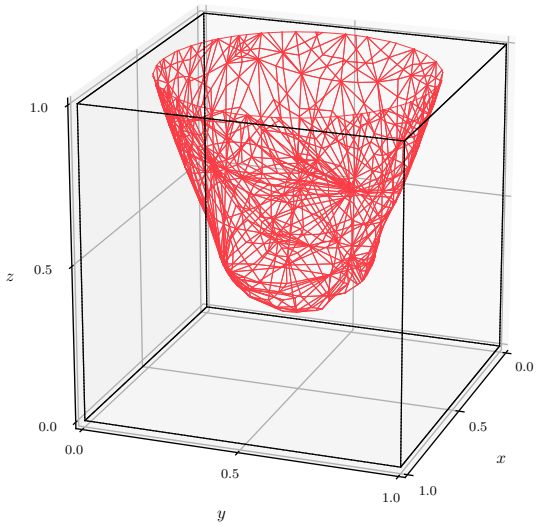


Figure 3.14: The intersection of a sphere and paraboloid at $z = 0.5$ in a unit cube. The integration faces $\partial\Omega_{ri}$ are drawn in red.

Table 3.4: The absolute volume error for different values of m_i and m_e.

$m_i = m_e$	ε
4	2.60E-04
8	6.53E-05
14	2.55E-05
22	1.13E-06

array of unstructured grids. AGI performed robustly and accurately, achieving consistent results throughout. This demonstrates the capability of AGI to reliably compute the initial volume fraction for arbitrary free surfaces on unstructured grids.

Chapter 4

The Volume of Fluid Method

4.1 Introduction

4.1.1 The Volume of Fluids Method

The Volume of Fluid (VoF) Method [8] has become one of the most popular methods for modelling two phase flow due to its volume (mass) conserving nature. From a multi-phase perspective the VoF method is an interface-capturing, single fluid, approach. These methods employ an Eulerian interface formulation where a single set of governing equations are used for both phases of the fluid [53]. The key aspect of the VoF method is the advection or transport of the volume fraction field, which is still an active area of research [18,19]. Broadly, the objective is to advect the fraction field while minimizing numerically induced distortions to the interface geometry. While seemingly simple it is a difficult feat to achieve, especially on unstructured grids.

4.1.2 Geometric Approaches

For the purpose of this discussion, VoF methods are subdivided into geometric and algebraic variants. In the case of the former, interface reconstruction is performed at a local cell level. The reconstructed interface together with the cell geometry and velocity field are then used to compute the amount of volume flux occurring at cell faces [54]. Geometric approaches can be further broken down into split and unsplit approaches. Split approaches advect in each cardinal direction separately and then reconcile the final face fluxes [12]. Unsplit methods advect in a single step but are generally more computationally costly [55, 56]. However, it is these unsplit methods that, naturally, lend themselves to unstructured methods. Until recently geometric methods were limited in scope in that there was little to no effort placed on unstructured applications. However, recently the area has advanced significantly [17–19, 57–59].

A key strength of geometric VoF is the ability to maintain a sharp interface. This allows for large Courant numbers which has a significant effect on the overall computational cost of such methods. The unstructured variant is significantly more complex and costly than the structured version.

4.1.3 Algebraic Approaches

The second class of VoF method is the algebraic variant, which also contains two sub-approaches: the hypobolic-tangent method of Xiao [60] and the compressive methods. Examples of the latter are HRIC [61], CICSAM [2], STACS [62], and FBICS [63]. The compressive methods typically blend some form of downwinding, providing interface compression, with a higher resolution upwinding scheme. In the case of CICSAM, HYPER-C [64] and ULTIMATE-QUICKEST [65] are used as the downwinding and

upwinding components respectively. The second component to these methods is the blending, or weighting, of the two computed face values. Generally the blending uses the alignment of the volume fraction gradient, a proxy for the free surface normal, and the cell face normal to determine the ratio of downwinding and upwinding.

A number of reviews have been conducted on the methods [62, 63, 66–68], especially comparing HRIC and CICSAM (which are the more popular methods). Theses reviews made a number of critical findings. The most crucial was that at higher Courant numbers, c_f, the schemes tend to become overly diffusive and lose their ability to maintain a sharp interface. Further, while naturally applicable to unstructured grids, these methods typically yield inferior results, in terms of accuracy, when compared to geometric methods on structured grids. Of the algebraic methods, CICSAM was found to perform better than HRIC in studies where the two were compared. However, while STACS and FBICS performed similarly or worse to CICSAM at low c_f numbers ($<\sim 0.4$) they proved superior at higher c_f numbers [68].

Some new entrants to the algebraic VoF methods include HiRAC [3] and M-CICSAM [68]. The former introduces a new compressive term to the VoF governing equation which increases the accuracy of CICSAM at higher c_f numbers. Additionally, HiRAC introduces a filtered volume fraction field for the blending operation, as it provides a more accurate representation of the interface. M-CICSAM combines the work over the preceding decade to reformulate CICSAM with more accurate face operators based on local c_f values and interface alignments. M-CICSAM also takes into account the orientation of the velocity vector at a cell face, during the blending operation. Finally, M-CICSAM modified the unwinding projection of the volume fraction field to improve unstructured accuracy.

Both the aforementioned methods improve on the CICSAM base scheme from which they are derived, especially at high c_f values (> 0.4). M-CICSAM and HiRAC feature the same test cases and a comparitive analysis suggests that the two methods are equivalent for $c_f < 0.7$ and that M-CICSAM is marginally better for $c_f > 0.7$. However, this comes with additional complexity and cost. Finally, a comparison between CICSAM and HiRAC against the latest geometric methods has not been conducted.

4.1.4 Closer

Historically, it is well known that geometric VoF methods are generally more accurate, for a wider range of c_f values. On arbitrary meshes this is however at the expense of computational cost and complexity as compared to the algebraic approaches [69]. In addition, until the recent works on unstructured girds [17–19, 57–59] geometric methods were limited to structured meshes.

While the work conducted by Heyns et al. [3] on HiRAC provided a valuable contribution to the algebraic VoF community it lacked a modified volume fraction face value. This ultimately leads to essentially a non-conservative scheme due to an inconsistency between face and nodal densities. In this chapter a consistent volume fraction face value will be formulated for the HiRAC method which guarantees conservation. Typically, the CICSAM (and HiRAC) methods are integrated in time using a Crack-Nicholson approach, and where bounding is desired a predictor-corrector method is applied [2]. However this has two drawbacks. The first is the high cost of an implicit VoF solve. The second difficulty is due to the velocity flux at the face not being available at $n + 1$. Correcting this would require multiple passes between the incompressible and VoF solvers, which would be prohibitively expensive. A predictor-corrector method [16] has therefore been adopted in this work, and due

to the trapezoidal rule of integration achieves second order accuracy with respect to time. This overcomes the two aforementioned mentioned drawbacks.

Finally, the CICSAM and corrected HiRAC method will be compared in terms of accuracy to the latest geometric methods. Since these methods have been evaluated at a $c_f = 0.5$, numerical analysis in this work will also be presented at this c_f value allowing for direct comparison. Further, at lower Courant numbers CICSAM and HiRAC produce similar results [3]. This will be on both structured and unstructured methods in 2D and 3D. Note also that the use of larger c_f values is where the overall scheme cost is best as this implies fewer solution time-steps. Additionally, this work represents the first quantitative accuracy assessment for HiRAC in 3D, since all preceding 3D work was qualitative in nature.

4.2 Temporal Discretisation

For convenience, Equation (1.3), from Chapter 1 is restated,

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot \alpha \boldsymbol{u} = 0,$$

where α is the volume fraction and $\boldsymbol{u}$ is the fluid velocity. The recent work by Heyns [3], on HiRAC, introduced a compressive term to the above namely,

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot \alpha \boldsymbol{u} + \nabla \cdot \alpha(1-\alpha)\boldsymbol{u}_{hr} = 0, \tag{4.1}$$

where $\boldsymbol{u}_{hr}$ is a compressive velocity detailed shortly. The semi-discrete equations can then be expressed as a 2nd order predictor-corrector scheme, as per Tryggvason [16]. Here the value at $n+1$ is obtained by taking two explicit steps per time step to yield the update set of equations:

$$\begin{aligned}
\alpha^{\iota+1} =& \alpha^n - \Delta t \left[\nabla \cdot \alpha \boldsymbol{u} + \nabla \cdot \alpha(1-\alpha)\boldsymbol{u}_{hr}\right]^n, \\
\alpha^{\iota+2} =& \alpha^{\iota+1} - \Delta t \left[\nabla \cdot \alpha \boldsymbol{u} + \nabla \cdot \alpha(1-\alpha)\boldsymbol{u}_{hr}\right]^{\iota+1}, \\
\alpha^{n+1} =& \frac{1}{2}\left(\alpha^n + \alpha^{\iota+2}\right).
\end{aligned} \tag{4.2}$$

The spatial discretisation of the volume fraction at the cell face requires an interface normal. For this purpose a smoother volume fraction field, $\alpha*$, is created. The α^* field is initialised to α after which two Laplacian smoothing iterations are performed:

$$\begin{aligned}
\alpha^{*,0} &= \alpha, \\
\alpha^{*,\iota'+1} &= \alpha^{*,\iota'} - \Delta\iota' \nabla \cdot \nabla \alpha^{*,\iota'}, \\
\alpha^{*,\iota+1} &= \alpha^{*,\iota'+2},
\end{aligned} \tag{4.3}$$

where ι' represents the sub-iteration number. $\Delta\iota'$ represents the pseudo-time step size, computed via Malan et. al. [32]. The face gradients for the spatial term in Equation (4.3) are discretised using Equation (2.13). Finally, for the update to $\alpha^{*,n+1}$ the same trapezoidal predictor-corrector method, Equation 4.2, is applied.

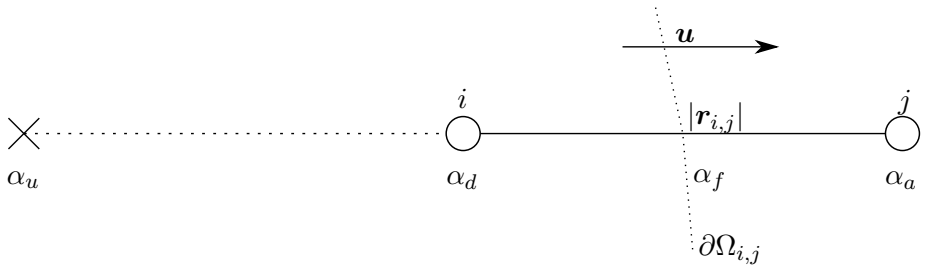


Figure 4.1: An edge discretisation where, due to the orientation of $\boldsymbol{u}$, node i is the donor node and node j is the acceptor node.

4.3 Spatial Discretisation

The first VoF spatial term in Equation (4.1) is approximated over the faces as

$$\nabla \cdot \alpha \boldsymbol{u}|_i \approx \frac{1}{V_i} \sum_{f \in \partial\Omega_i} \alpha_f u_{ff},$$

where α_f results from the CICSAM discretisation [2,70], $\partial\Omega_i$ and u_{ff} (Equation 2.4) are as defined in Chapter 2. CICSAM utilises a *donor-acceptor* approach to edge based discretisation. This requires the inspection of the orientation of the velocity at the shared face between the two connected nodes, i and j, as depicted in Figure 4.1. The donor node is the upwind node of the face, along an edge, and the acceptor node is the down wind node. Using the donor and acceptor alpha values an upwind alpha α_u is constructed to normalise the donor value:

$$\begin{aligned} \alpha_u &= \min\left(1, \max\left(0, \alpha_a - 2\,\nabla\alpha|_d \cdot \hat{\boldsymbol{r}}_{d,a}|\boldsymbol{r}_{i,j}|\right)\right), \\ |\alpha_d| &= \frac{\alpha_d - \alpha_u}{\alpha_a - \alpha_u}, \end{aligned} \tag{4.4}$$

where α_d and α_a represent the donor and acceptor node's alpha value respectively. Note that the use of Equation (4.4) is due to the nature of unstructured grids, where the value for the upwind node is not readily available. Using the normalised donor value the HYPER-C, $|\alpha_{hc}|$, and ULTIMATE-QUICKEST, $|\alpha_{uq}|$, approximations can be computed by

$$|\alpha_{hc}| = \begin{cases} \min(1, \frac{|\alpha_d|}{c_d}) & \text{if } 0 \leq |\alpha_d| \leq 1 \\ |\alpha_d| & \text{else} \end{cases},$$

$$|\alpha_{uq}| = \begin{cases} \min\left(|\alpha_{hc}|, |\alpha_d| c_d + \frac{(1-c_d)(6|\alpha_d|+3)}{8}\right) & \text{if } 0 \leq |\alpha_d| \leq 1 \\ |\alpha_d| & \text{else} \end{cases},$$

where c_d is the local donor cell Courant value, computed for each node by

$$c_d = \sum_{f \in \partial\Omega_i} \max\left(0, \frac{u_{ff}\Delta t}{V_i}\right).$$

Next the blending ratios are determined, using the smoothed volume fraction $\nabla\alpha^*$, as

$$\gamma = \min\left(\left(\frac{|\nabla\alpha^* \cdot \boldsymbol{r}_{i,j}|}{|\nabla\alpha^*||\boldsymbol{r}_{i,j}|}\right)^m, 1\right),$$

where the value of m is set to 2 and as per Heyns et al. [3]. Note that it is the smoothed volume fraction gradient which is used in the blending ratio, and differs from the original formulation of CICSAM. Finally, the normalised and final value of α_f are computed by

$$|\alpha_f| = \gamma|\alpha_{hc}| + (1-\gamma)\,|\alpha_{uq}|,$$

or, setting

$$\beta = \frac{|\alpha_f| - |\alpha_d|}{1 - |\alpha_d|}, \tag{4.5}$$

$$\alpha_f = \min(1, \max(0, (1-\beta)\alpha_d + \beta\alpha_a)). \tag{4.6}$$

The second term in Equation (4.1) reads discretely as

$$\nabla \cdot \alpha(1-\alpha)\boldsymbol{u}_{hr}|_i \approx \frac{1}{V_i}\sum_{f\in\partial\Omega_i} \alpha_f(1-\alpha_f)c_\alpha \frac{|u_{ff}|}{A_f}\left.\frac{\nabla\alpha^*}{|\nabla\alpha^*|}\right|_f \cdot A\hat{\boldsymbol{n}}|_f,$$

where c_α is a compressive factor for which a value of 0.1 is used in this work. α_f is as computed by Equation (4.6).

For the conservative discretisation of the momentum equation, a face α value must be supplied which satisfies the following relation for incompressible flow,

$$\frac{\partial\alpha_i}{\partial t} + \frac{1}{V_i}\sum_{f\in\partial\Omega_i}\alpha'_f\boldsymbol{u}_{ff} = 0.$$

The above published HiRAC relation however does not lend itself well to this (as the spatial term is not written in this form). Nor is there any formulation available for α'_f. This thesis therefore proposes such a formulation by equating the above to the spatial component of Equation (4.1), as

$$\begin{aligned}
\nabla\cdot\alpha'_f\boldsymbol{u}_{ff} &= \nabla\cdot\alpha_f\boldsymbol{u}_{ff} + \nabla\cdot\alpha_f(1-\alpha_f)\boldsymbol{u}_{hr} \\
\frac{1}{V_i}\sum_{f\in\partial\Omega_i}\alpha'_f\boldsymbol{u}_{ff} &= \frac{1}{V_i}\sum_{f\in\partial\Omega_i}\alpha_f\boldsymbol{u}_{ff} + \alpha_f(1-\alpha_f)c_\alpha\frac{|\boldsymbol{u}_{ff}|}{A_f}\left.\frac{\nabla\alpha^*}{|\nabla\alpha^*|}\right|_f\cdot A\hat{\boldsymbol{n}}|_f \\
&= \frac{1}{V_i}\sum_{f\in\partial\Omega_i}\left(\alpha_f + \alpha_f(1-\alpha_f)c_\alpha\frac{\mathrm{sign}(u_{ff})}{A_f}\left.\frac{\nabla\alpha^*}{|\nabla\alpha^*|}\right|_f\cdot A\hat{\boldsymbol{n}}|_f\right)\boldsymbol{u}_{ff}, \\
\therefore\ \alpha'_f &= \alpha_f\left(1 + (1-\alpha_f)c_\alpha\mathrm{sign}(u_{ff})\frac{\nabla\alpha^*}{|\nabla\alpha^*|}\cdot\hat{\boldsymbol{n}}_f\right),
\end{aligned} \tag{4.7}$$

where the α'_f is the conservative face volume fraction.

4.4 Volume Fraction Bounding

In the original work by Ubbink and Issa [2] a mechanism for correcting unbounded volume fractions was created. This is clearly of importance to industrial codes as un-bounding in α leads to unbounded material properties. Left uncorrected, unbounded volume fractions can lead to unphysical negative densities and viscosities. Generally, the procedure involved determining whether the donor cell's unbounded value is an over- or undershoot. Then, by looping over faces an update β value from Equation (4.5) is computed to correct the unbounded cell. This correction procedure

was originally designed for an implicit Crank-Nicholson time integration scheme and is here adapted for the explicit predictor-corrector time stepping case.

4.4.1 CICSAM

The bounding error for an over- or undershoot of the volume fraction at a node are respectively defined by

$$\mathcal{E}^+ = \max(\alpha_d^{n+1} - 1, 0),$$

and

$$\mathcal{E}^- = \max(-\alpha_d^{n+1}, 0).$$

The corrected face fraction, formulated by Ubbink and Issa [2], is expressed as

$$\alpha_f^\star = \alpha_f + \frac{\mathcal{E}^+}{c_{ff}}, \tag{4.8}$$

where

$$c_{ff} = \frac{u_{ff}\Delta t}{V_i},$$

and c_{ff} is the face Courant number or volume flux across a face. Finally, the new face volume fraction has the same form as Equation (4.6) namely,

$$\alpha_f^\star = (1 - \beta^\star)\alpha_d^n + \beta^\star \alpha_a^n, \tag{4.9}$$

and

$$\beta^\star = \beta - \beta' \wedge 0 \leq \beta' \leq \beta. \tag{4.10}$$

Substituting Equations (4.10) and (4.8) into (4.9) yields

$$\begin{aligned} \alpha_f + \frac{\mathcal{E}^+}{c_{ff}} &= (1 - \beta + \beta')\alpha_d + (\beta - \beta')\alpha_a, \\ \Rightarrow \beta' &= -\frac{\mathcal{E}^+}{\Delta\alpha c_{ff}}, \end{aligned}$$

where $\Delta\alpha = \alpha_a - \alpha_d$. Following a similar procedure for undershoot errors, where $\mathcal{E}^- > 0$, β' can be computed as

$$\beta' = \frac{\mathcal{E}^-}{\Delta\alpha c_{ff}}.$$

The above operations are performed on the donor cell and as such the following split function is created for β'

$$\beta' = \begin{cases} \min\left(-\frac{\mathcal{E}^+}{\Delta\alpha c_{ff}}, \beta\right) & \text{if } \mathcal{E}^+ > 0 \wedge \Delta\alpha < -\mathcal{E}^+, \\ \min\left(\frac{\mathcal{E}^-}{\Delta\alpha c_{ff}}, \beta\right) & \text{else if } \mathcal{E}^- > 0 \wedge \Delta\alpha > \mathcal{E}^-, \\ 0 & \text{else,} \end{cases} \tag{4.11}$$

after which β' is used to update $\beta^\star$ to then obtain $\alpha^\star$.

4.4.2 HiRAC

The original work on HiRAC contained no mechanism for dealing with unbounded volume fractions. This work addresses this shortcoming for the first time by developing a bounding scheme. As before Equation (4.7) is updated with the corrected face fraction $\alpha_f^\star$ as

$$\alpha_f'^\star = \alpha_f^\star \left(1 + (1 - \alpha_f^\star)k_h\right),$$

where

$$k_h = c_\alpha \text{sign}(u_{ff}) \frac{\nabla\alpha^*}{|\nabla\alpha^*|} \cdot \hat{\boldsymbol{n}}_f.$$

Substituting Equations (4.8) and (4.9) into the above gives

$$\begin{aligned}\alpha_f' + \frac{\mathcal{E}^+}{c_{ff}} &= \alpha_f - \Delta\alpha\beta' + \alpha_f k_h - \Delta\alpha\beta' k_h - \left(\alpha_f^2 - 2\alpha_f\Delta\alpha\beta' + \Delta\alpha^2\beta'^2\right)k_h, \\ &= \alpha_f' - \Delta\alpha\left(1 + k_h - 2\alpha_f k_h\right)\beta' - \Delta\alpha^2 k_h\beta'^2.\end{aligned}$$

Algebraic manipulation then yields the following quadratic expression:

$$\Delta\alpha^2 k_h\beta'^2 + \Delta\alpha\left(1 + k_h - 2\alpha_f k_h\right)\beta' + \frac{\mathcal{E}^+}{c_{ff}} = 0,$$

for which the two β' roots are expressed as

$$\beta' = \frac{-\Delta\alpha\left(1 + k_h - 2\alpha_f k_h\right) \pm \sqrt{\left(\Delta\alpha\left(1 + k_h - 2\alpha_f k_h\right)\right)^2 - 4\Delta\alpha^2 k_h \frac{\mathcal{E}^+}{c_{ff}}}}{2\Delta\alpha^2 k_h}. \tag{4.12}$$

An analysis of the above is prudent both to determine if real roots exist, and whether they are positive and in an appropriate range (Equation (4.10)). First the expected ranges of the above variables are considered: $\alpha_f \in [0, 1]$, $k_h \in [-c_\alpha, c_\alpha]$, $\Delta\alpha \in [-1, 1]$, $c_{ff} \in (0, 1)$. This leaves $\mathcal{E}^+$ which is strictly positive, however, it can be reasonably assumed $\mathcal{E}^+ \ll 1$.

Upon further consideration of the occurrence of an overshoot the aforementioned ranges can be further narrowed. The overshoot occurs on an advancing interface, where the over full donor cell, at $n + 1$, did not advect enough material to acceptor cells. This implies $u_{ff} > 0$ and $\nabla\alpha^* \cdot \hat{\boldsymbol{n}}_f < 0$ thus $k_h \in [-c_\alpha, 0]$. This ensures a strictly positive discriminant in Equation (4.12).

To determine if valid values of β' exist, Equation (4.6) is rewritten in the following form:

$$\alpha_f = \alpha_d^n + \beta\Delta\alpha,$$

using the relationship $\Delta\alpha$. The solution space for β' can now be mapped for the appropriate ranges of $\beta, k_h, \mathcal{E}^+$, and $\Delta\alpha$ and determine if Equation (4.10) holds. The valid solution space for $c_{ff} = 0.5$ and $\alpha_d^n = 0.95$ is illustrated in Figure 4.2. The α_d^n is selected as a representative value of a node likely to overshoot.

Clearly, for the vast majority of the possible solution space, valid solutions are available. Where no valid β' can be found, the scheme reverts to CICSAM to ensure a bounded solution. Note that for $\beta = 0$ there exists no valid solution, however this is inferred by Equation (4.10). Thus the minimum value of β in Figure 4.2 starts at

0.01.

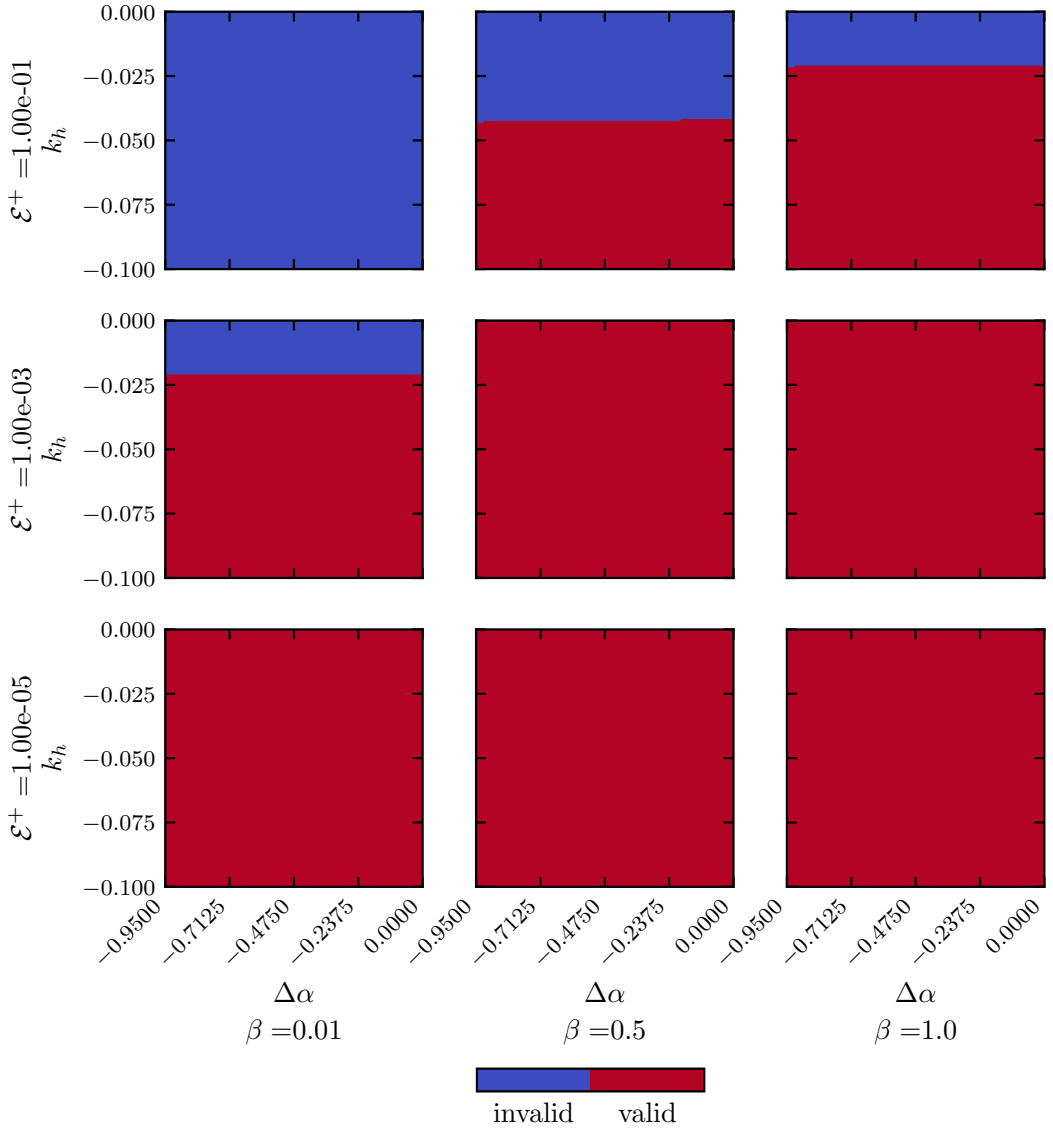


Figure 4.2: Map of HiRAC β' roots satisfying Equation (4.10) for $c_{ff} = 0.5$ and $\alpha_d^n = 0.95$.

Finally the expression for the undershoot case's ($\mathcal{E}^-$) β' roots are computed as

$$\beta' = \frac{-\Delta\alpha\left(1 + k_h - 2\alpha_f k_h\right) \pm \sqrt{\left(\Delta\alpha\left(1 + k_h - 2\alpha_f k_h\right)\right)^2 + 4\Delta\alpha^2 k_h \frac{\mathcal{E}^-}{c_{ff}}}}{2\Delta\alpha^2 k_h}.$$

With a similar analysis of the expected ranges of the variables in the above, it is possible to demonstrate that the discriminant is strictly positive. Further, the valid solution map of β' for $\mathcal{E}^-$ yields a similar plot to the overshoot case. Finally, to evaluate the boundedness of the schemes, for this chapter, the outlined bounding algorithms are not employed (but are in subsequent chapters).

4.5 Results

A number of metrics are used to determine the performance of a VoF advection scheme. The shape or geometric error, which measures the absolute difference between the analytical and numerical solution at a given node, is expressed as

$$\varepsilon_g(\boldsymbol{x}_i, t) = |\alpha_n(\boldsymbol{x}_i, t) - \alpha_a(\boldsymbol{x}_i, t)|\,,$$

where $\varepsilon_g(\boldsymbol{x}_i, t)$ is the error at time t and $\boldsymbol{x}_i$ is the node coordinate. The numerical and analytical values of the volume fraction are denoted α_n and α_a respectively. The analytical volume fraction field is computed in this work by using *AGI* [1].

The L_1 norm of the geometric error is computed,

$$\|\varepsilon_g\|_1 = \int_\Omega \varepsilon_g dV = \sum_i V_i \varepsilon_g(\boldsymbol{x}_i),$$

where V_i is the dual cell volume associated with node i. To determine the boundedness of the volume fraction a bounding error ε_b is introduced as

$$\varepsilon_b = \max_{\boldsymbol{x}_i \in \boldsymbol{X}_\Omega} \left(0, \max\left(0, \alpha_n(\boldsymbol{x}_i) - 1\right), \max\left(0, -\alpha_n(\boldsymbol{x}_i)\right)\right).$$

Volume conservation is assessed as follows via

$$\varepsilon_v = \frac{|\sum_i V_i \alpha_n(\boldsymbol{x}_i, t_m) - \sum_i V_i \alpha_a(\boldsymbol{x}_i, t_0)|}{\sum_i V_i \alpha_a(\boldsymbol{x}_i, t_0)},$$

where ε_v represents the volume error and t_0 and t_m represent the initial and final state of the volume fraction field respectively.

Finally, as a note, unless otherwise specified the test cases presented in this chapter impose an analytical velocity field which is strictly divergence free (incompressible).

4.5.1 Circle In Constant Flow

The first test case selected is that of a circle with radius 0.2 which is placed at $\boldsymbol{x} = \{0.25, 0.25\}$ and advected through a unit domain by an oblique velocity field, $\boldsymbol{u} = \{1.0, 1.0\}$ for 0.5s. The shape error is then calculated by creating an analytical circle, also of radius 0.2, using AGI at $\boldsymbol{x} = \{0.75, 0.75\}$ (the expected final position of the circle).

The test case is repeated several times for structured (Cartesian) meshes varying in size, ranging from 10^2 to 80^2. Additionally, a set of isotropic unstructured meshes (triangular elements) are also employed with a similar number of nodes as the structured grids. Finally, the test is repeated for a number of different c_f values, so as to compute the temporal convergence of the VoF schemes.

Figure 4.3 depicts the results for both the structured and unstructured meshes for both CICSAM and HiRAC. Note, the graphs are plotted against $\frac{1}{c_f}$ to allow for comparative analysis as per recent works [18, 19, 71]. The plots on the right column of Figure 4.3 show a zoomed in portion of the left column plots, i.e. for $c_f > 0.2$. From the conducted test cases it is clear that the HiRAC scheme generally performs better than the CICSAM scheme.

The magnified plots, in Figure 4.3, show that both developed (HiRAC and CICSAM) schemes exhibit superior temporal convergence properties for $c_f \leq 0.5$ as compared to the unsplit geometric methods [18, 71, 72] while being comparable to the Iso-Advector [19] scheme. Further, with reference to the geometric methods, [18, 19],

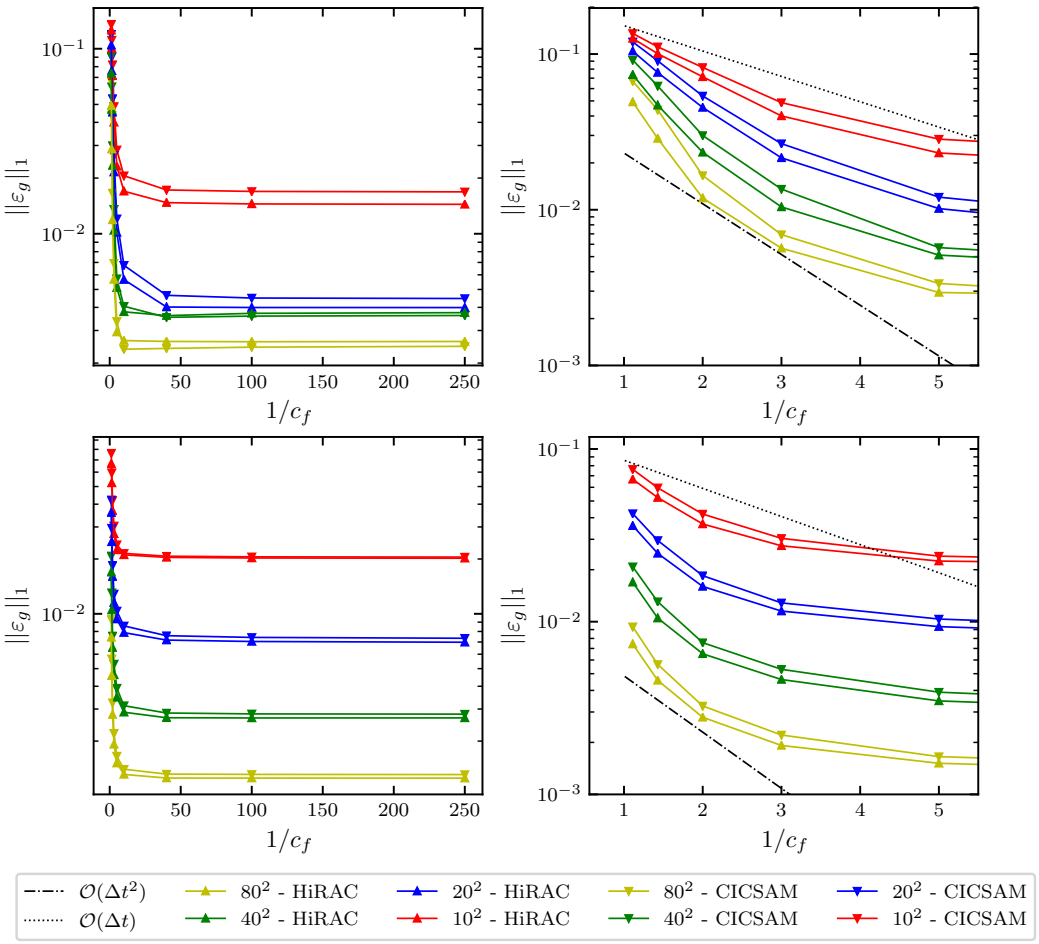


Figure 4.3: L_1-norm of the geometric error for a circle advected in a constant flow field for different c_f values. The left column plots depict the results for the Cartesian (top) and triangular (bottom) meshes respectively, and the right column plots depict a magnified section of the results for $c_f \geq 0.2$.

the spatial errors for both CICSAM and HiRAC are similar for lower structured mesh resolutions and are approximately no more than an order of magnitude worse on the finest resolutions. Unfortunately, the cited works did not perform the test on unstructured grids, prohibiting an unstructured comparison.

The second component of this test case is to demonstrate the consistency of the HiRAC discretisation using the conservative volume face faction, Equation (4.7). Figure 4.4 represents a repeated advection test, on the unstructured 40^2 grid, where the pressure field is depicted for both the original [3] and corrected face values at the end of the simulation. As depicted, the former method results in spurious pressure oscillations across the interface. The proposed conservative approach however completely eliminates this down to the pressure solver tolerance ($1.0e^{-12}$).

4.5.2 Shear Flow

The shear flow test case, proposed by Rider and Kothe [73], subjects a 2D circular interface to large shear flows in a smooth velocity field. Due to the convoluted interface motion this tests many of the sub-components of a VoF scheme. The circular drop or bubble of radius $r = 0.15$ is placed with its centre half way along the domain's x-axis and three quarters along the y-axis, where the domain is of unit size. An analytical

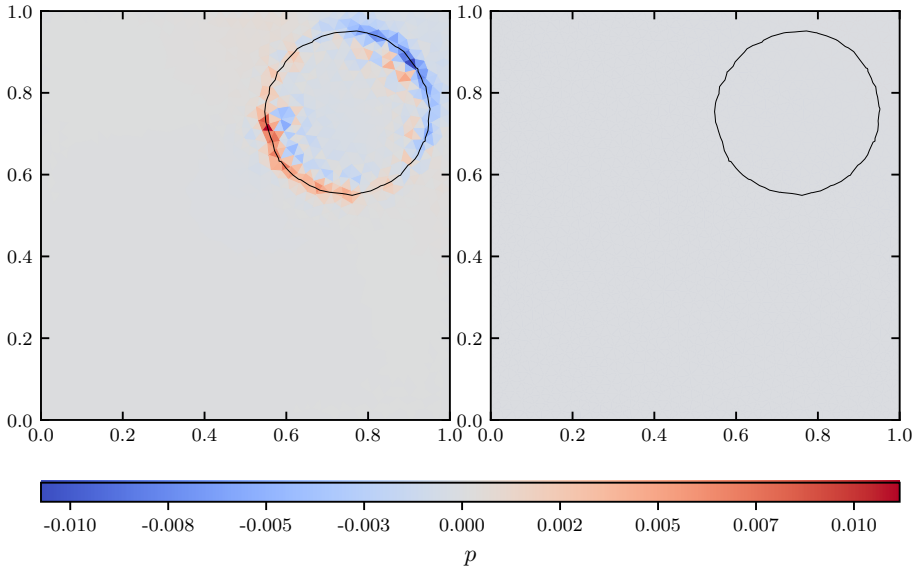


Figure 4.4: The pressure fields for the uncorrected (left) and corrected (right) volume fraction face values for HiRAC. The black contour represents a volume fraction value of 0.5.

time varying shear velocity is applied to the domain and is given by

$$\boldsymbol{u}(x,y,t) = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sin(2\pi y)\sin^2(\pi x)\cos\left(\frac{\pi t}{t_m}\right) \\ -\sin(2\pi x)\sin^2(\pi y)\cos\left(\frac{\pi t}{t_m}\right) \end{bmatrix},$$

where $t_m = 8$s and is the time domain. At the half way mark, where $t = 4$s, the velocity field reverses direction and by t_m the volume fraction field *should* have returned to its original position.

Importantly, central differencing of the velocity at the median dual cell faces can cause a violation of the divergence free condition (of the discretised VoF equation). To ensure this is avoided, the above velocity must be applied weakly over the sub-facets of each face, as follows:

$$\begin{aligned}\int_{\partial\Omega_{i,j}} \boldsymbol{u}\cdot\hat{\boldsymbol{n}}dA &= \int_0^1 \boldsymbol{u}(\boldsymbol{r}(\lambda))\cdot\hat{\boldsymbol{n}}_{\partial\Omega i,j}\left|\boldsymbol{r}'(\lambda)\right|d\lambda \\ &= \frac{|\boldsymbol{r}'|\,c_t}{8\pi}\Bigg[n_x|_{\partial\Omega_{i,j}}\left(\frac{2\cos(2\pi\left(r_x(\lambda)+r_y(\lambda)\right))}{\Delta x+\Delta y}-\frac{\cos(2\pi r_y(\lambda))}{\Delta y}\right. \\ &\qquad\left.-\frac{\cos(2\pi\left(r_x(\lambda)-r_y(\lambda)\right))}{\Delta x-\Delta y}\right) \\ &\quad - n_y|_{\partial\Omega_{i,j}}\left(\frac{2\cos(2\pi\left(r_x(\lambda)+r_y(\lambda)\right))}{\Delta y+\Delta x}-\frac{\cos(2\pi r_x(\lambda))}{\Delta x}\right. \\ &\qquad\left.-\frac{\cos(2\pi\left(r_y(\lambda)-r_x(\lambda)\right))}{\Delta y-\Delta x}\right)\Bigg]_0^1,\end{aligned}$$

where the derivation for the above can be found in Appendix A.1. Briefly, $\boldsymbol{r} = \{r_x, r_y\}$ and is a line parametrisation of the face using $\lambda \in [0,1]$. The size of the face in x and y are respectively denoted Δx and Δy. Where $\Delta x = 0$ or $\Delta y = 0$, causing division by zero, the respective face normal component, $n_x|_{\partial\Omega_{i,j}}$ and $n_y|_{\partial\Omega_{i,j}}$ are zero, thus

avoiding undefined behaviour. For brevity, strict split function notation has been omitted. Finally c_t is the temporal part of $\boldsymbol{u}$, namely,

$$c_t = \cos\left(\frac{\pi t}{t_m}\right).$$

As stated in the introduction, the objective is to compare with recent geometric VoF methods [17–19,59]. These works are mainly reported at a Courant number of 0.5, as this provides a good balance between computational cost and accuracy. Thus here, and for the remainder of the test cases conducted in this chapter, a value of $c_f = 0.5$ has been selected to enable direct comparison with the most recent developments in the field.

The qualitative results for this test case are presented in Figure 4.5. The simulation is repeated on a number of Cartesian and Delaney grids, starting with a node count of 64^2 and increasing to a maximum grid resolution of 512^2. Plotted in the figure is the volume fraction field half way through the simulation, when the velocity field reverses. In addition, level contours of the final interface have been drawn for the volume fraction for $\alpha = \{0.01, 0.5, 0.99\}$, where black contours represent the analytical solution and white the numerical.

Figure 4.6 depicts a zoomed in portion of the final volume fraction field plot for the structured mesh 256×256 mesh. This provides a clearer picture, compared to that of Figure 4.5, of the degree to which HiRAC prevents excessive smearing for shear type flows. From both plots, it is clearly demonstrated that both the two-phase interface is sharper and that the bulk of tracked phase consistency maintains a volume fraction above 0.99.

As expected an increase in refinement of the grid leads to less numerical diffusivity of the interface. As shown on the second finest mesh, and excluding the CICSAM structured mesh which is still overly diffused, the analytical and numerical solutions are well matched. Additionally, it is clear that HiRAC does maintain a much sharper interface through the simulation, both at $t = 4$s but especially at the end of the simulation.

Finally, the plots show that both methods tend to be less diffusive on unstructured grids. This is clearly true for CICSAM, where the volume fraction is still substantially 'smeared.' on the structured grids. This difference between structured and unstructured, while still present, is less pronounced for HiRAC.

Tables 4.1 and 4.2 present the qualitative results obtained from the test cases and include data from recent geometric works. For comparison it is noted that even using the above face formulation of the velocity the field still contains a deviation in the divergence free condition of approximately $1.0e^{-13}$ on some meshes, as such the presented volume errors, ε_v, are within round off error. That said, the volume errors compare well with the other geometric methods. It should be noted that at present there is no VoF fraction redistribution algorithms running for these tests, which is not true for some of the cited works. The bounding errors, ε_b are also within machine precision tolerances, and as with the volume error, no clipping algorithm has been added.

Finally, the consideration of perhaps the most critical metric: the geometric error ε_g. As expected for structured meshes, the geometric methods outperform the algebraic. As for unstructured meshes, HiRAC has a similar magnitude error as compared to the latest to geometric methods. However, geometric methods become noticeably more accurate on finer meshes.

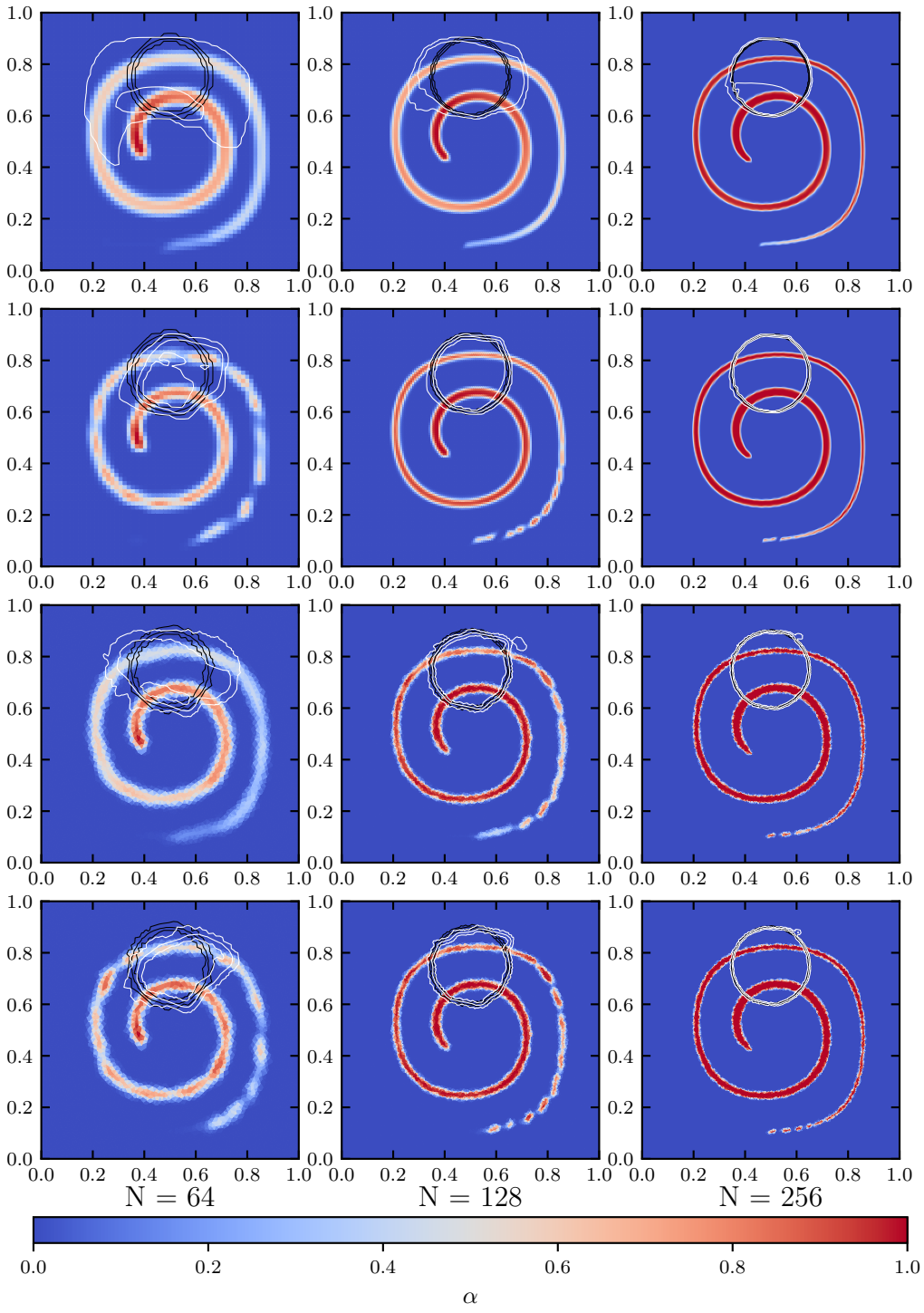


Figure 4.5: Volume fraction field plots for $t = 4$s and the white interface contours drawn for $t = 8$s and black interface contours for the analytical solution. The contour values are $\alpha = \{0.01, 0.5, 0.99\}$. The top row depicts a structured grid with CICSAM and the 2nd row with HiRAC. The 3rd row depicts the unstructured triangular grids with CICSAM, and the final row HiRAC.

4.5.3 Deformation Flow

A droplet undergoing deformation in 3D, proposed by Leveque [76], and repeated in several works [17, 19, 74] is presented as the final test case for the chapter. The

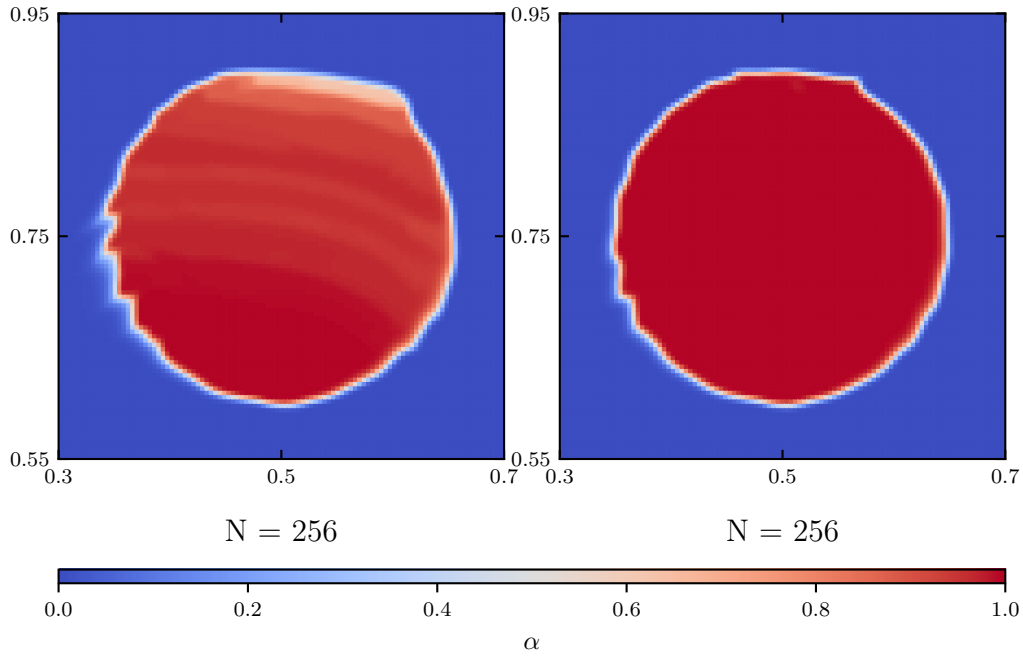


Figure 4.6: Volume fraction field plots CICSAM (left) and HiRAC (right) at $t = 8$s on the finest structured mesh.

Table 4.1: Comparative error results for a disc in shear flow for the formulated CICSAM and HiRAC schemes on a structured Cartesian mesh for $c_f = 0.5$.

Scheme	$N^{\frac{1}{2}}$	ε_v	ε_b	ε_g	$\mathcal{O}(\varepsilon_g)$
CICSAM	64	3.34e-15	3.14e-34	7.61e-02	-
HiRAC	64	5.89e-16	2.00e-45	2.41e-02	-
Owkes and Desjardins [74]	64	9.76e-15	6.51e-17	7.75e-03	-
gVoFoam [75]	64	-	-	1.44e-02	-
isoaAdvector-plicRDF [19]	64	1.66e-16	1.51e-20	1.26e-02	-
CICSAM	128	3.08e-14	9.38e-59	3.90e-02	0.963
HiRAC	128	3.06e-14	4.47e-46	4.66e-03	2.370
Owkes and Desjardins [74]	128	1.29e-14	6.33e-17	1.87e-03	2.040
gVoFoam [75]	128	-	-	2.78e-03	-
isoaAdvector-plicRDF [19]	128	9.71e-17	4.69e-13	2.61e-03	2.270
CICSAM	256	1.77e-15	1.98e-69	5.53e-03	2.820
HiRAC	256	2.08e-14	1.53e-14	1.39e-03	1.749
Owkes and Desjardins [74]	256	1.74e-14	9.33e-17	4.04e-04	2.210
isoaAdvector-plicRDF [19]	256	1.21e-15	5.27e-12	5.71e-04	2.190
CICSAM	512	1.77e-13	1.18e-99	9.06e-04	2.608
HiRAC	512	1.26e-14	9.81e-12	6.44e-04	1.106
Owkes and Desjardins [74]	512	1.74e-14	9.32e-17	8.32e-05	2.280
isoaAdvector-plicRDF [19]	512	9.59e-15	4.14e-14	1.04e-04	2.440

case tests a method's ability to maintain a thin 'sheet' of fluid under stretching and diverging flow. The case involves placing a droplet of radius $r = 0.15$ with its centre at

Table 4.2: Comparative error results for a disc in shear flow for the formulated CICSAM and HiRAC schemes on an unstructured triangular mesh for $c_f = 0.5$.

Scheme	$N^{\frac{1}{2}}$	ε_v	ε_b	ε_g	$\mathcal{O}(\varepsilon_g)$
CICSAM	64.016	7.85e-16	3.36e-27	4.22e-02	-
HiRAC	64.016	3.93e-15	7.50e-29	3.18e-02	-
isoaAdvector-plicRDF [19]	64	9.71e-17	2.73e-20	2.21e-02	-
CICSAM	128.012	2.16e-15	4.49e-34	6.16e-03	2.777
HiRAC	128.012	5.89e-16	3.29e-36	6.76e-03	2.233
isoaAdvector-plicRDF [19]	128	3.47e-16	8.11e-21	3.58e-03	2.620
CICSAM	255.996	1.85e-14	1.46e-47	1.57e-03	1.970
HiRAC	255.996	1.94e-14	2.57e-46	1.51e-03	2.167
isoaAdvector-plicRDF [19]	256	3.59e-15	1.16e-21	7.51e-04	2.250
CICSAM	511.996	6.05e-14	7.16e-66	4.68e-04	1.749
HiRAC	511.996	2.93e-14	9.89e-67	4.44e-04	1.762
isoaAdvector-plicRDF [19]	512	5.85e-14	5.32e-22	1.31e-04	2.520

the position $\boldsymbol{x}_c = \{0.35, 0.35, 0.35\}$ and a time varying analytical velocity is applied, namely,

$$\boldsymbol{u}(x,y,z,t) = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos\left(\frac{\pi t}{t_m}\right) \\ -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos\left(\frac{\pi t}{t_m}\right) \\ -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos\left(\frac{\pi t}{t_m}\right) \end{bmatrix},$$

and where the flow is evolved over 3 seconds ($t_m = 3$s). Due to the cosine function the droplet *should* return to its original position at t_m. This allows the initial configuration to provide the analytical solution, from which geometric errors may be computed. To create a divergence free velocity field the below formulation is applied to each sub-facet, k, of the face shared by the two nodes i and j,

$$\oint_{\partial\Omega_{i,j}} \boldsymbol{u}(\boldsymbol{x},t)\cdot\hat{\boldsymbol{n}}_{i,j}dA = \sum_{k}^{k_m} R'(\boldsymbol{u}, \boldsymbol{X}_{\partial\Omega_{i,j,k}}, \hat{\boldsymbol{n}}_{i,j,k}, t)$$

where there are k_m sub-facets. R' represents a modification of the region integral developed in Chapter 3. $\boldsymbol{X}_{\partial\Omega_{i,j,k}}$ represents a set of 3 points for each facet, namely the edge, face, and element centroids. Finally $\hat{\boldsymbol{n}}_{i,j,k}$ represents the facet normal. Further details for the modification of the integral used to compute the face flux are provided in Appendix A.2.

The advected droplets for CICSAM and HiRAC for the test case are depicted in Figure 4.7. A similar trend emerges to that of the 2D case, with HiRAC clearly outperforming CICSAM at the $t = 1.5$s. The thin 'sheet' has considerably smaller gaps in it. The disparity between the two methods, however, is especially clear at t_m, where HiRAC produces a less wrinkled surface on finer resolution. Further, on course resolution HiRAC is able to maintain a 'spherical' surface, while CICSAM has clearly distorted the original shape.

The quantitative results for the cases are presented in Tables 4.3 and 4.4. The

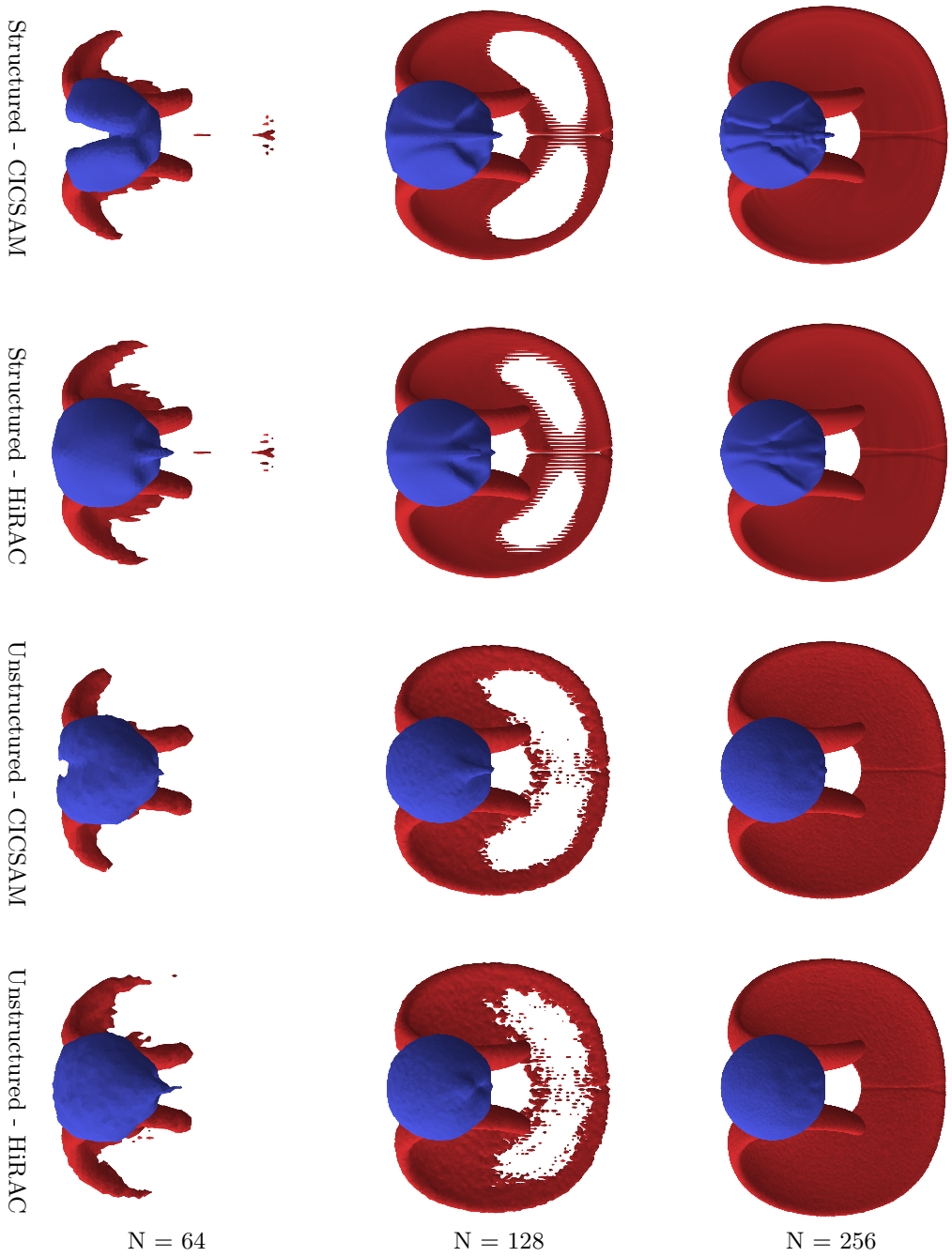


Figure 4.7: Contour plot for $\alpha = 0.5$ on increasingly finer meshes. The red surfaces represents the interface at $t = 1.5$s and the blue the droplet at $t = 3$s.

reader is reminded that the velocity field is divergence free to the order of $1.0e^{-13}$. The volume errors, ε_v, for the algebraic scheme compare well to the cited methods. The bounding errors, ε_b, similar to the 2D results are within machine precision tolerances.

The Cartesian grids show that geometric methods, expectedly, are more accurate than their algebraic counterparts, in regard to the shape error, ε_g. However, HiRAC does clearly outperform CICSAM, both in terms of accuracy and rate of convergence and while being similar to geometric methods in many cases (and never worse than an order of magnitude). Finally, the results for the shape errors, on the tetrahedral meshes, show that CICSAM, HiRAC, and isoAdvector [19] have both similar

Table 4.3: Comparative error results for a sphere in deformation flow for the formulated CICSAM and HiRAC schemes on a structured Cartesian mesh for $c_f = 0.5$.

Scheme	$N^{\frac{1}{3}}$	ε_v	ε_b	ε_g	$\mathcal{O}(\varepsilon_g)$
CICSAM	32	1.06e-14	2.73e-34	1.42e-02	-
HiRAC	32	1.28e-14	3.49e-23	1.05e-02	-
Owkes and Desjardins [74]	32	1.19e-15	1.20e-17	6.80e-03	-
isoaAdvector-plicRDF [19]	32	1.08e-16	2.12e-14	8.36e-03	-
CICSAM	64	1.45e-14	2.57e-26	8.58e-03	0.73
HiRAC	64	8.96e-15	1.97e-30	4.82e-03	1.13
Owkes and Desjardins [74]	64	2.48e-15	2.34e-17	2.10e-03	1.73
isoaAdvector-plicRDF [19]	64	9.75e-16	6.41e-20	3.25e-03	1.36
CICSAM	128	1.21e-13	2.11e-28	4.38e-03	0.97
HiRAC	128	1.39e-13	4.37e-19	1.66e-03	1.54
Owkes and Desjardins [74]	128	1.68e-14	2.75e-17	5.62e-04	1.89
isoaAdvector-plicRDF [19]	128	3.71e-15	1.11e-15	6.57e-04	2.31
CICSAM	256	2.51e-12	3.54e-13	1.61e-03	1.44
HiRAC	256	2.13e-12	9.85e-13	6.82e-04	1.28
Owkes and Desjardins [74]	256	3.87e-14	4.69e-17	1.01e-04	2.47
isoaAdvector-plicRDF [19]	256	2.03e-14	2.37e-16	9.54e-05	2.78

Table 4.4: Comparative error results for a sphere in deformation flow for the formulated CICSAM and HiRAC schemes on an unstructured tetrahedral mesh for $c_f = 0.5$.

Scheme	$N^{\frac{1}{3}}$	ε_v	ε_b	ε_g	$\mathcal{O}(\varepsilon_g)$
CICSAM	31.95	9.57e-15	2.26e-21	1.49e-02	-
HiRAC	31.95	1.23e-15	9.69e-24	1.33e-02	-
isoaAdvector-plicRDF [19]	32	1.65e-16	4.50e-18	1.31e-02	-
CICSAM	63.09	9.51e-14	1.23e-21	7.76e-03	0.95
HiRAC	63.09	9.40e-14	8.97e-24	6.13e-03	1.14
isoaAdvector-plicRDF [19]	64	9.56e-16	4.14e-19	6.34e-03	1.06
CICSAM	128.39	2.60e-13	3.35e-14	2.57e-03	1.56
HiRAC	128.39	2.42e-13	2.66e-15	1.45e-03	2.03
isoaAdvector-plicRDF [19]	128	7.75e-15	7.66e-21	1.31e-03	1.45
CICSAM	250.64	1.15e-12	6.44e-15	4.45e-04	2.62
HiRAC	250.64	1.04e-12	1.53e-14	3.49e-04	2.13
isoaAdvector-plicRDF [19]	256	7.21e-14	2.52e-20	1.93e-04	2.77

accuracies and spatial convergence rates.

4.6 Conclusion and Summary

The contribution to the VoF method has centred around the improvement of the algebraic HiRAC method, where it has been reformulated into a conservative approach. Both CICSAM and HiRAC were for the first time discretised in time using a second

order predictor-corrector method, this allowing for face flux approximations to be made at the correct time level. Finally, for CICSAM the volume fraction bounding corrector was updated for the new time marching procedure, and for HiRAC a bounding corrector was introduced. For HiRAC, it was shown that there exists large range of interface configurations for which a bounded volume fraction can be recovered.

Conducted test cases demonstrated that the proposed time marching approach exhibits good temporal convergence properties. The conservative formulation of HiRAC was also further demonstrated. With regard to comparison to the structured methods both CICSAM and HiRAC demonstrated competitiveness on courser mesh resolutions but fall behind as mesh spacing decreased. HiRAC consistently outperformed CICSAM in all test cases.

Chapter 5

Surface Tension Modelling

5.1 Introduction

5.1.1 Surface Tension Modelling

The numerical analysis of surface tension on Eulerian grids involves two aspects. The first is the inclusion of the surface tension source term, $\boldsymbol{f}_\sigma$, into the governing equations. This term effects the Laplace pressure jump over the two-fluid interface. The second aspect involves computing the local curvature of the interface geometry.

The description of the surface tension force, for pure VoF methods, is generally achieved through the volumetric Continuum Surface Force (CSF) method of Brackbill et al. [41]. The surface tension force is mathematically expressed by

$$\boldsymbol{f}_\sigma = \sigma\kappa\delta_s\hat{\boldsymbol{n}} = \sigma\kappa\nabla\alpha, \tag{5.1}$$

where κ, $\hat{\boldsymbol{n}}$, and δ_s represent the two-phase local interface curvature, the interface unit normal, and the Dirac delta function, which acts as a 'switch' in the region of the interface. The surface tension coefficient, a material property of the interfacing fluids, is denoted σ. For VoF approaches $\nabla\alpha$ is volumetricly equivalent to $\delta_s\hat{\boldsymbol{n}}$ [77].

One of the most infamous challenges faced by the VoF surface tension community was that of the generation of spurious or parasitic currents [53]. These artificial velocities were present even in cases where the equilibrium conditions were set as the initial condition, for example the classic static drop/bubble test case [12, 41]. A number of works eventually shed light on the phenomena which can be understood by analysis of the Young-Laplace equilibrium condition, namely

$$-[p] + \sigma\kappa = 0, \tag{5.2}$$

where $[p]$ denotes the Laplace pressure jump. For the VoF formulation the above results in

$$-\nabla p + \sigma\kappa\nabla\alpha = 0, \tag{5.3}$$

with nomenclature previously defined.

5.1.2 Curvature Capturing

The first, and more obvious reason for parasitic current generation is errors in the curvature computation. This results in a variation of the discrete curvature, over a sphere for example. Abadie el al. [78] formalised this mathematically, by taking the curl of Equation (5.3), as

$$\nabla \times (-\nabla p + \sigma\kappa\nabla\alpha) = \nabla\kappa \times \nabla\alpha = 0, \tag{5.4}$$

which implies that the curvature gradient must be zero for the parasitic currents to vanish, assuming σ is constant. This underlies the importance of accurate free-surface interface initialisation. Inadequate initialisation can add additional errors to the topology of the initial volume fraction field thereby worsening parasitic currents at the start of a simulation.

Brackbill et al. [41], in their work on the CSF method, constructed a smoothed volume fraction field and used the field's gradient as an approximation for the free surface normal. The curvature is computed as the divergence of the constructed normal. The method however has been shown not to have essential spatial convergent properties [16, 48, 79, 80], and is considered outdated for structured grids [81]. However, the classic CSF approach is still almost exclusively employed for estimating curvature in 2D and 3D on unstructured grids. Alternatives include interpolating to a background structured grid, which has been considered by Ivey and Moin [82], however this is computationally expensive and still does not resolve irregular boundary geometries, a common feature of industrial problems.

A number of regression approaches have been developed for curvature estimation [49, 80, 83–87]. These schemes involve some underlying method to obtain a set of points which lie on the interface, and in many cases this is performed in a conservative manner. Typically, the points are used to fit a local parabolic curve (2D) or paraboloid surface (3D), from which the curvature at a particular grid point is obtained [49, 80, 85–87]. Recently progress has been made with a 2D second order method for unstructured grids by Evrard et al. [80], and a 3D super first order method by Jibben et al. [87]. However, neither have yet been coupled to the momentum equations nor been applied to dynamic moving interfaces.

Perhaps the most successful approach to curvature capturing is the second order height function approach by Sussman [47] and Cummins et al. [48]. The method involves forming columns across the interface, where the volume fraction can be summed to determine the 'fill' or height of the tracked fluid in the column. The height, together with adjacent heights, are used to compute curvature using a finite difference stencil. The accuracy of the method was improved by López and Hernández [88], and further by using a five node stencil (2D) the method was made fourth order by Bornia et al. [89]. Unfortunately, the approach is limited to structured meshes which limits its general applicability [81]. It is however employed here to demonstrate consistent coupling with the improved algebraic VoF schemes of Chapter 4.

5.1.3 Surface Tension Discretisation

The second cause for the generation of spurious currents is due to an imbalance between the discrete pressure and volume fraction gradients. To obtain a well-balanced scheme the discrete expression of Equation (5.3) must satisfy the equilibrium condition of Equation (5.2) numerically, assuming constant curvature. This is achieved by using the same spatial gradient operator for p and α. Consider the weak form of Equation (5.3),

$$-\sum_{f\in\partial\Omega_i} p\hat{\boldsymbol{n}}A|_f - \mathcal{E}_p + \sigma\kappa \sum_{f\in\partial\Omega_i} \alpha\hat{\boldsymbol{n}}A|_f + \mathcal{E}_\alpha = 0, \tag{5.5}$$

where $\mathcal{E}_p$ and $\mathcal{E}_\alpha$ are the errors associated with the discrete numerical face operator for the pressure and volume fraction respectively. To recover the Laplace condition these errors must cancel. Failing this will cause the pressure to impart non-physical oscillations to the velocity field.

Work by Renardy and Renardy [85], Popinet [49], and Denner and Wachem [90] have yielded a number of methods which involve a balanced force discretisation for pure VoF methods, while Francois et al. [91] and Herrmann [92] have developed balanced methods for the Ghost Fluid and level-set methods respectively. Notably by combining height function curvature computation and a balanced-force approach Popinet [49] presented perhaps the first purely VoF CSF method to completely eliminate spurious velocities. The underlying scheme utilised a cell-centred oct-tree method with a structured geometric VoF advection scheme.

5.1.4 Closer

By introducing a surface tension method to HiRAC [3,33] in a consistent manner this chapter aims to expand the algebraic VoF method. The inclusion and discretisation of the surface tension term into the governing equations is achieved using harmonic means, ensuring consistency in order to obtain a well balanced CSF method. This work represents the first vertex centred approach. To the authors' knowledge the only other method to use harmonic means has been the cell-centred method by Denner and Wachem [90].

For purpose of demonstrating the consistent coupling of the pressure and surface tension terms the structured height functions method is employed. However, a finite volume approach to discretisation is retained for all of the discrete expressions. This to allow for the transition to unstructured curvature computation methodologies when a suitable unstructured curvature technique is developed.

5.2 Spatial Discretisation

5.2.1 Consistent Pressure and Surface Tension Inclusion

The discontinuity in the gradient of pressure over the interface can lead to spurious pressures and velocities if not treated correctly, especially in the gas phase [93]. To overcome this Panahi et al. [94] assumed that in a dual cell the density remains constant and the pressure varies linearly, see Figure 5.1.

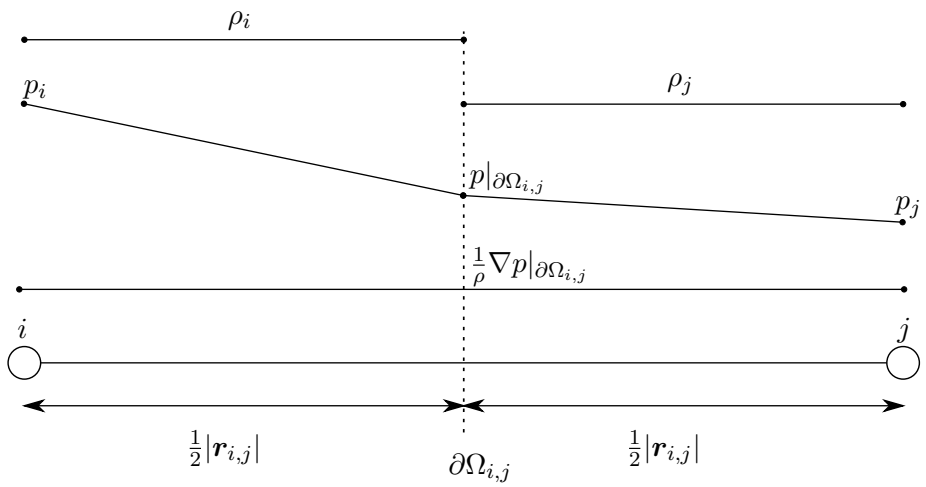


Figure 5.1: The discretisation of pressure along an edge, with a piece-wise constant density gradient. Note: that p can be replaced by α for the surface tension term.

Along an edge, $\boldsymbol{r}_{i,j}$, and employing the aforementioned assumptions, a piecewise linear discretisation operator for the pressure term in Equation (2.3), is given as

$$\frac{1}{\rho}\nabla p\Big|_{\partial\Omega_{i,j}} \cdot \hat{\boldsymbol{r}}_{i,j} = c,$$

where c is constant over the edge. By integrating along the edge, from node i to the face $\partial\Omega_{i,j}$, the difference in pressure is expressed as

$$p|_{\partial\Omega_{i,j}} - p|_i = \frac{|\boldsymbol{r}_{i,j}|\rho_i}{2}c,$$

and similarly from $\partial\Omega_{i,j}$ to j as

$$p|_j - p|_{\partial\Omega_{i,j}} = \frac{|\boldsymbol{r}_{i,j}|\rho_j}{2}c.$$

Adding the above two equations the pressure gradient over the edge is obtained as

$$c = \frac{1}{\rho}\nabla p\Big|_{\partial\Omega_{i,j}} \cdot \hat{\boldsymbol{r}}_{i,j} = \frac{2(p_j - p_i)}{|\boldsymbol{r}_{i,j}|(\rho_i + \rho_j)}. \tag{5.6}$$

Finally by subtracting the two equations the face pressure is formulated as

$$p|_{\partial\Omega_{i,j}} = \frac{\rho_j p_i + \rho_i p_j}{\rho_i + \rho_j}.$$

The pressure gradient, ∇p, is computed using Gauss-Green, with $p|_{\partial\Omega_{i,j}}$ approximated via the above expression. Note that Equation (5.6) can be computed via the compact derivative, Equation (2.13), with the additional density bias multiplied through.

Additionally, the pressure Jacobian, $\partial\mathcal{R}/\partial p$ of Equation (2.5), is now expressed by taking the derivative with respect to pressure of Equation (5.6). The resulting Jacobian is constructed for row i and column j using

$$\frac{\partial\mathcal{R}}{\partial p_\mathrm{i}}\Big|_{\mathrm{i,j}} \approx \frac{1}{V_\mathrm{i}}\frac{2\Delta t}{\rho_\mathrm{i}^{n+1} + \rho_\mathrm{j}^{n+1}}\frac{|A\hat{\boldsymbol{n}}|_{\partial\Omega_{\mathrm{i,j}}}|}{|\boldsymbol{r}_{i,j}|} + \frac{c_\tau}{\rho_\mathrm{i}\Delta t},$$

where c_τ is the pseudo-acoustic wave speed (Jacobian factor) [38,95], and is only non-zero on diagonal entries. Typically Δt is divided through Equation (2.5) to improve matrix conditioning.

The surface tension force, Equation (5.1), is expanded with the CSF approach to read

$$\boldsymbol{f}_\sigma = \sigma\kappa\delta\hat{\boldsymbol{n}} = \sigma\kappa\nabla\alpha. \tag{5.7}$$

Using $\nabla\alpha$ with central differenced face estimates would violate the discrete balance of Equation (5.5). To overcome this a new volume fraction gradient is introduced, $\nabla\alpha^\sigma$.

To ensure a balanced-force surface tension discretisation the identical piece-wise linear approach is adopted, essentially p in Figure 5.1 is replaced with α. The same harmonic mean discrete face operator is formulated. First, the surface tension term

for Equation (2.3), is expressed as

$$\frac{1}{\rho}\nabla\alpha^{\sigma}\Big|_{\partial\Omega_{i,j}} \cdot \hat{\boldsymbol{r}}_{i,j} = c, \tag{5.8}$$

where c is again some edge wise constant and σ, κ, and Δt are omitted. Integrating along the edge yields two equations,

$$\alpha|_{\partial\Omega_{i,j}} - \alpha|_i = \frac{\rho_i|\boldsymbol{r}_{i,j}|}{2}c,$$
$$\alpha|_j - \alpha|_{\partial\Omega_{i,j}} = \frac{\rho_j|\boldsymbol{r}_{i,j}|}{2}c,$$

and adding the equations yields

$$\frac{1}{\rho}\nabla\alpha^{\sigma}\Big|_{\partial\Omega_{i,j}} \cdot \hat{\boldsymbol{r}}_{i,j} = \frac{2(\alpha_j - \alpha_i)}{|\boldsymbol{r}_{i,j}|(\rho_i + \rho_j)}.$$

The normalised volume fraction face value is computed via

$$\alpha^{\sigma}|_{\partial\Omega_{i,j}} = \frac{\rho_j\alpha_i + \rho_i\alpha_j}{\rho_i + \rho_j}. \tag{5.9}$$

As with the pressure gradient, a Gauss-Green discretisation is used for $\nabla\alpha^{\sigma}$ where the face value requires Equation (5.9) be used. Following the suggestion of Popinet [49] the implementation for the weights and biases utilise the same function calls in code to ensure machine consistency. For completeness, the consistent discretisation of surface tension reads

$$\nabla\cdot\frac{\Delta t}{\rho^{n+1}}\nabla p\Big|_i \approx \frac{1}{V_i}\sum_{f\in\partial\Omega_i}\frac{2\Delta t}{\rho_i^{n+1}+\rho_j^{n+1}}\nabla p^{\iota}|_f \cdot A\hat{\boldsymbol{n}}|_f,$$
$$\nabla\cdot\frac{\Delta t\sigma\kappa}{\rho^{n+1}}\nabla\alpha^{\sigma}\Big|_i \approx \frac{1}{V_i}\sum_{f\in\partial\Omega_i}\frac{\Delta t\left(\sigma\kappa|_i + \sigma\kappa|_j\right)}{\rho_i+\rho_j}\nabla\alpha^{\sigma}\Big|_f^{n+1} \cdot A\hat{\boldsymbol{n}}|_f,$$

with ∇p_f and $\nabla\alpha_f^{\sigma}$ updated as described in this section.

5.3 Interface Curvature

5.3.1 Height Functions

Height functions in this work are constructed in a similar fashion to previously cited works. However, due to the vertex centred nature of the grid, height functions are built over edges. To begin the interface must be identified; Node i is considered on the interface where it or one of its neighbours, j, has $\alpha \in (0, 1)$. The column is constructed over interface along a cardinal direction, $\hat{\boldsymbol{e}}_x, \hat{\boldsymbol{e}}_y, \hat{\boldsymbol{e}}_z$. The selection of which cardinal direction to construct along is detailed shortly.

A column, at node i, is defined as set of consecutive co-linear nodes which are located over the interface and is denoted $\boldsymbol{\zeta}_{\hat{\boldsymbol{e}},i}$. Additionally, the column is oriented such that the node at the base is fully submerged in the reference fluid, $\alpha = 1$, and at the top in the untracked phase, $\alpha = 0$, see Figure 5.2. The height of a column along

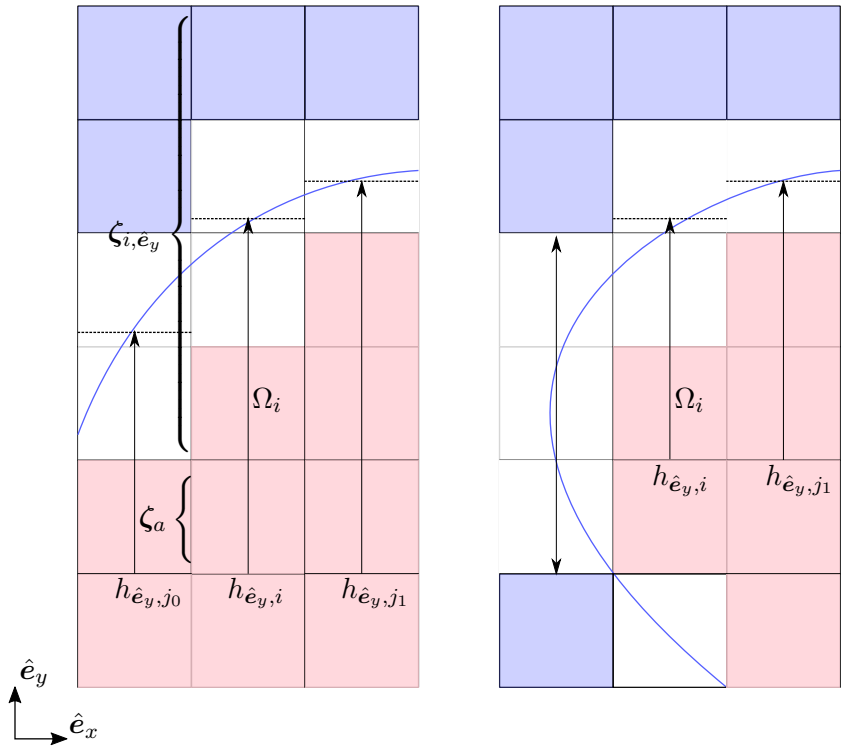


Figure 5.2: Left, a typical curvature computation for node i, with control volume Ω_i. Right, a failed construction of a column in the $\hat{\boldsymbol{e}}_y$ for the j_0 column.

one of the cardinal directions, $\hat{\boldsymbol{e}}$, is computed as

$$h_{\hat{\boldsymbol{e}},i} = \sum_{j \in \boldsymbol{\zeta}_{\hat{\boldsymbol{e}},i}}^{\#\boldsymbol{\zeta}_{\hat{\boldsymbol{e}},i}-1} \alpha_j |\boldsymbol{r}_{j,j+1}|, \tag{5.10}$$

where $j{+}1$ is the index of the connected node and $\#\boldsymbol{\zeta}_{\hat{\boldsymbol{e}},i}$ is the cardinality of the column set. In practise the column construction and height calculation occur simultaneously. The outlined approach uses a similar algorithm to that outlined by López et al. [96] and Popinet [49] which dynamically controls the column size to the local region of the interface.

5.3.2 Parallel Height Functions

Elemental's single-element-overlap parallel architecture is not adequate to support the presented column building methodology. Further, simply extending the number of overlapped elements would be, from a solver perspective, a significant increase in the memory and computational cost. Thus, a new type of parallel node, a 'ghost' node, is introduced as shown in Figure 5.3. Ghost nodes extend beyond the single-layer overlap, however, critically only contain the most essential connectivity, volume fraction, and position data.

The additional ghost node layers are created using a greedy algorithm, which is performed after the normal single overlapping layer has been constructed. The sub-domains are initialised with their typical 'master/slave' architectures, and a separate parallel object is initialised with 'master/slave' ghost nodes. The volume fractions are communicated to the ghost nodes, enabling parallel height construction.

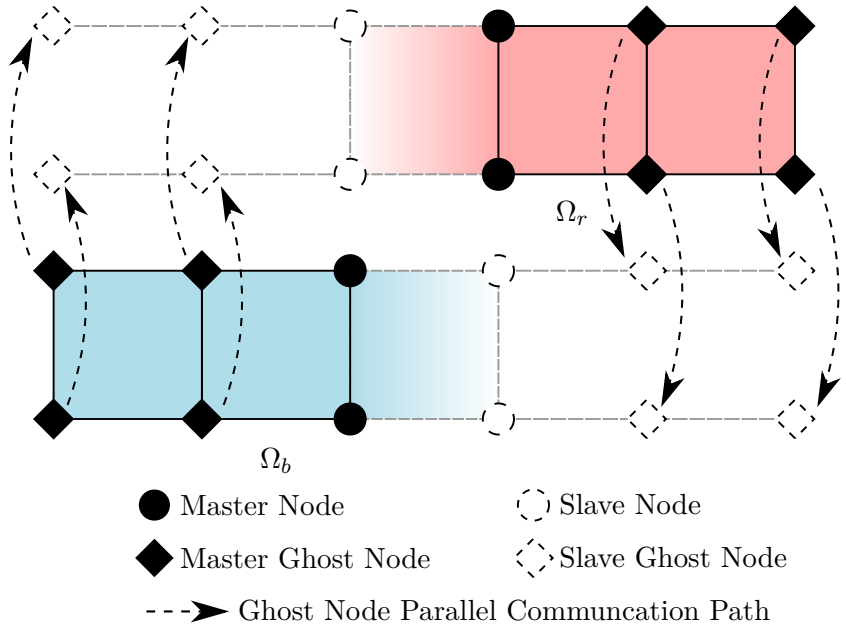


Figure 5.3: Depiction of the updated red (Ω_r) and blue (Ω_b) sub-domains, from Figure 2.2, enabling parallel height column construction.

This solution has a number of advantages, firstly it is constructed with existing infrastructure within *Elemental*'s tested parallel framework. Secondly, this approach allows for local concavity of domain intersections, which is not traditionally found with structured decomposed meshes. Concavity occurs due to the limited constrains placed on the shape of the decomposed domain's interface geometry in *Elemental* (other than to minimise thread interface area and balance node counts).

5.3.3 Computing Curvature

The gradients of the heights are computed using a finite volume approach, which allows for varying mesh spacing (although still structured). While slightly less computationally efficient, it naturally caters for the handling of vertex centred boundary control volumes. The height gradient is discretised for node i as

$$\nabla h_i \approx \frac{1}{V_i} \sum_{f \in \partial\Omega_i} \frac{1}{2} \left(h_i + h_j\right) A\hat{\boldsymbol{n}}|_f,$$

where $\hat{\boldsymbol{e}}$ has been omitted from h. It is possible for the neighbouring node, j, to not have a column constructed, or to have a column which does not share the same base ordinate as h_i. For the former, a column is constructed, starting at the neighbouring node. Where the base ordinates differ, additional nodes are appended to $\boldsymbol{\zeta}_j$ such that the base ordinates become congruent, see Figure 5.2 (left) where $\boldsymbol{\zeta}_a$ is an added node to the column $\boldsymbol{\zeta}_i$. The Hessian of h_i is computed by

$$\mathcal{H}(h_i) \approx \frac{1}{V_i} \sum_{f \in \partial\Omega_i} \nabla h|_f \otimes A\hat{\boldsymbol{n}}|_f \,,$$

and $\nabla h|_f$ is computed via the compact derivative, Equation (2.13). Note that this is identical to a finite difference approximation when nodes are equispaced. The

curvature is computed in 2D by

$$\kappa = \frac{\mathcal{H}(h)_{1,1}}{(1+\nabla h_1)^{\frac{3}{2}}},$$

and in 3D by

$$\kappa = \frac{\mathcal{H}(h)_{1,1} + \mathcal{H}(h)_{2,2} + \mathcal{H}(h)_{1,1}\nabla h_2^2 + \mathcal{H}(h)_{2,2}\nabla h_1^2 + \nabla h_1^2 \nabla h_2^2 \left(\mathcal{H}(h)_{1,2} + \mathcal{H}(h)_{2,1}\right)}{(1+\nabla h_1 + \nabla h_2)^{\frac{3}{2}}},$$

where subscript i has been omitted for brevity. The numerical subscripts refer to the row and column, in $\mathcal{H}(h)$, corresponding to the cardinal direction vectors $\hat{\boldsymbol{e}}_1$ and $\hat{\boldsymbol{e}}_2$ which represent the first and second cardinal direction(s) orthogonal to the height's construction $\hat{\boldsymbol{e}}$. The axis in which to construct the columns is selected as the axis closest aligned to the interface normal, $\hat{\boldsymbol{n}} = \nabla\alpha/|\nabla\alpha|$.

It is possible that during construction a column may become degenerate. This normally occurs where the top and bottom of the column are in the same bulk fluid, see Figure 5.2 (right). In such cases construction of columns in other directions will be attempted, in order of descending alignment with $\hat{\boldsymbol{n}}$. For curvature to be calculated, via height functions, node i and all orthogonal (to the direction of column construction) neighbours must have consistent heights.

If all directions fail to yield a height curvature calculation the curvature of the node is computed as the average of the surrounding neighbours' curvature value. If there are no neighbours from which to average curvature the curvature is set to zero. This approach is similar to work by Popinet [49], however, presently there is no quadratic fitting technique employed in this thesis.

5.4 Results

5.4.1 Evaluation of Curvature Computation

The first test case conducted is that featured in a number of previous works [47–49,80], and is used to evaluate the accuracy of the curvature computation. Here the implemented height function technique is tested to ensure correctness of implementation by assessing if 2nd order spatial accuracy is achieved. These tests are conducted in parallel to validate the parallel implementation.

The test involves initialising the VoF field for a circle to approximately machine precision, using AGI, in a unit domain. The interface curvature is computed using the outline height function algorithm. This is repeated 100 times during which the centre of the circle is located at $\boldsymbol{x} = \{0.5 + \epsilon_x \Delta x, 0.5 + \epsilon_y \Delta y\}$, where ϵ_x and ϵ_y are randomly generated and range between $[-1, 1]$ for each test. This process is repeated, each time increasing the number of grid points per circle radius to ultimately validate the order of accuracy of the method.

The relative error of curvature at a node is expressed as

$$\varepsilon_\kappa(\boldsymbol{x}_i) = \left| \frac{\kappa(\boldsymbol{x}_i) - \kappa_a}{\kappa_a} \right|,$$

where

$$\kappa_a = \frac{1}{r},$$

and the analytical curvature and circle radius are denoted κ_a and r respectively. The L_2 and L_∞ norms for the curvature are defined as

$$||\varepsilon_\kappa||_2 = \sqrt{\frac{\sum_s \sum_i \varepsilon_\kappa(\boldsymbol{x}_i)^2}{N_i N_s}},$$
$$||\varepsilon_\kappa||_\infty = \max_{s \in N_s} \max_{i \in N_i} \varepsilon_{\kappa,s}(\boldsymbol{x}_i), \tag{5.11}$$

where N_s and N_i are the number of simulations conducted (100) and the number of interface nodes, i.e. where $0 < \alpha(\boldsymbol{x}_i) < 1$, respectively. Note that as per previous works [48, 49], for low grid point to radius ratios $\kappa_a \Delta x \lesssim 8$, the height functions in many cases fail to produce consistent heights, where this occurs the node is excluded from the data set.

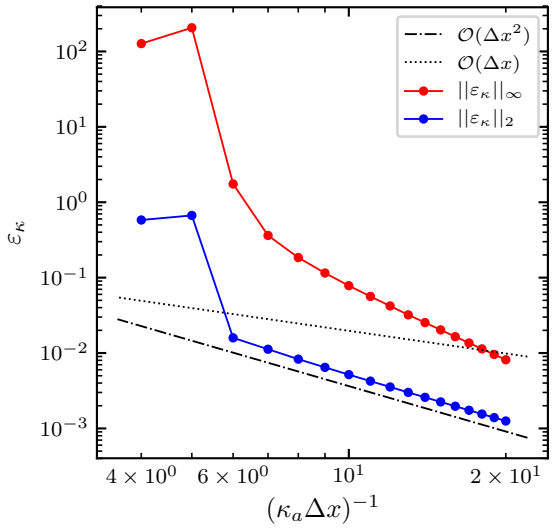


Figure 5.4: L_2 and L_∞ norms for the height function curvature capturing technique.

The results are depicted in Figure 5.4 and demonstrate that implemented height function method has the expected second order spatial convergence for both norms. It is thus concluded that the height function method has be correctly implemented.

5.4.2 Static Bubble

To validate the consistency of the proposed surface tension discretisation a static bubble test case is employed. Principally the case seeks to demonstrate that a static circular bubble, in the absence of external forces (e.g. gravity), remains stationary and recovers the Laplace pressure jump over the interface to the precision of the Poisson solver.

It is typical for this test case to be non-dimensionalised using the Laplace number, *La*, defined by

$$La = \frac{\sigma \rho D}{\mu^2},$$

where D is the bubble diameter and other symbols have been previously defined. The *La* number provides a relationship between the inertial, viscous and surface tension forces. Where higher numbers (> 10000) infer less viscous damping, and can become unstable in the case of inconsistent discretisation.

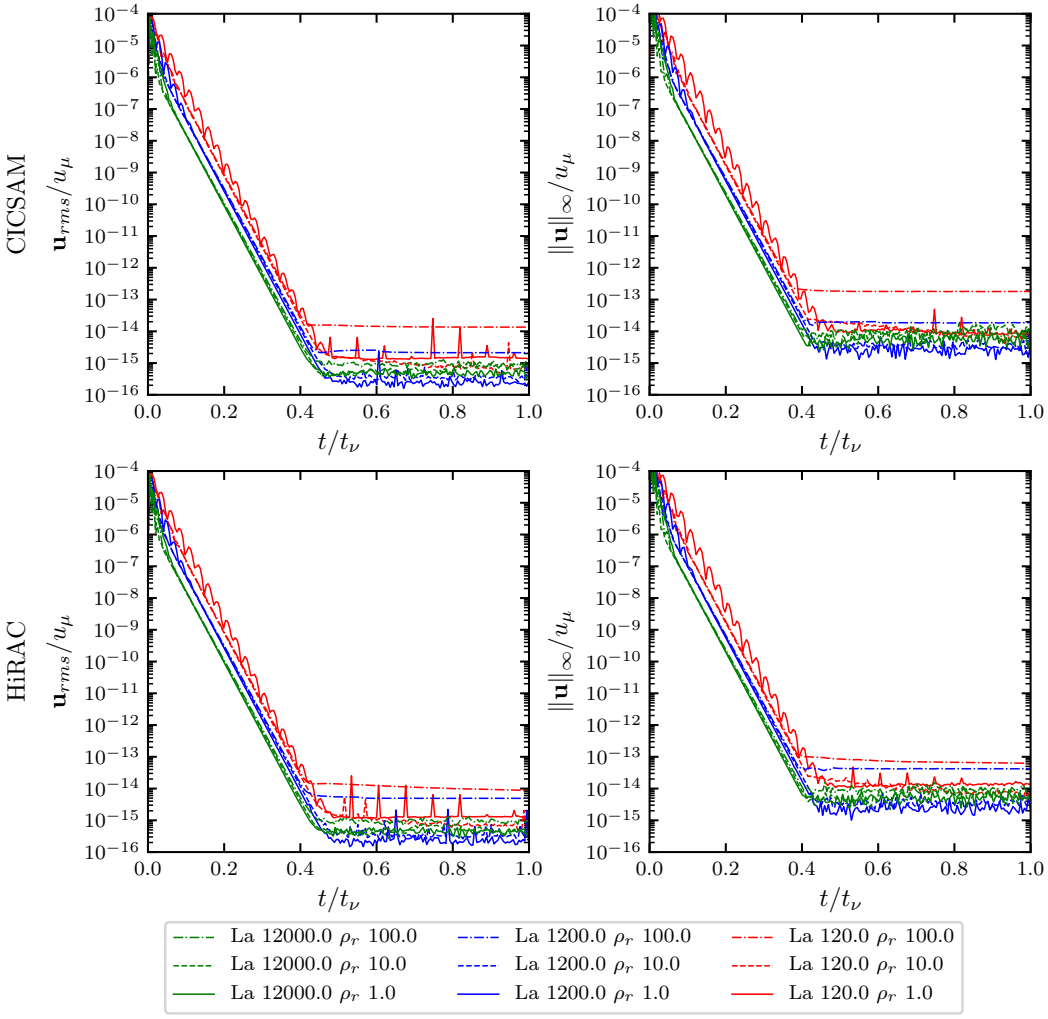


Figure 5.5: Graph of the evolution of the non-dimensional *rms* velocity (left) and maximum velocity (right) against non-dimensional time for a range of *La* numbers and density ratios.

The velocities here are made non-dimensional using a characteristic velocity defined by

$$u_{\sigma,i} = \sqrt{\frac{\sigma}{\rho_i D}},$$

as used in [49]. To ensure that the momentum has had enough time to diffuse throughout the domain the time is non-dimensionalised with

$$t_\nu = \frac{\rho_l D^2}{\mu_l}.$$

In these tests a bubble of the diameter 0.8, is placed with its centre at the origin, using AGI. Symmetry boundary conditions are applied to the left and bottom boundaries of a unit domain, which contains the first quarter of the bubble. No-slip conditions are applied to the remaining boundaries. A 32×32 grid was selected to perform the analysis. The test was repeated for a range of both *La* numbers, VoF advective schemes, and density ratios, denoted $\rho_r = \rho_l/\rho_g, \rho_r = \{1, 10, 100\}$.

It is important to note that initial velocities are expected in the calculation. These currents are attributable to variations in the curvature field due discretisation errors [49]. However, as the simulation proceeds the curvature is driven to a constant, thus eliminating the gradient in curvature, and satisfying Equations (5.4) and (5.2). The inability to reach this steady-state condition points to an inconsistent, or ill balanced, surface tension discretisation.

The evolution of the computed magnitudes of the velocity field are depicted in Figure 5.5, for the range of parameters used. For the period $0 < t/t_\nu < 0.4$ the initial capillary waves are damped out. The final magnitude of the spurious velocities is as a result of the pressure solver tolerance ($1.0e^{-12}$), and round off error. Thus, these results demonstrate that the scheme is well balanced.

The obtained $\|\boldsymbol{u}\|_\infty$ are comparable to the best results obtained by Abadie et al. [78] (for $La = 12000, \rho_r = 1$ and adjusting for differences in non-dimensionalisation). Of the various interface and surface tension models presented in that work [78], there were only three schemes which were able to damp spurious currents to below $1.0e^{-12}$. Of these, two used the height function-CSF method and of those two only one used a (geometric) VoF method. The results presented here thus represent, to the authors knowledge, the first height function-CSF algebraic VoF approach to demonstrate a well balanced surface tension discretisation.

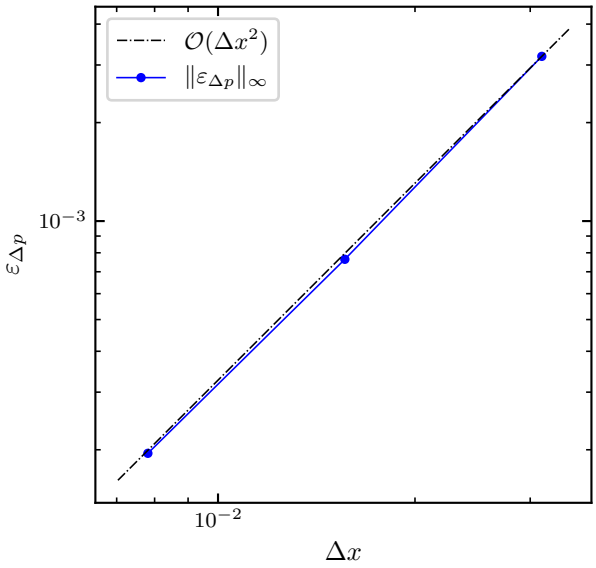


Figure 5.6: Spatial convergence plot for the Lapalce pressure jump using $La = 120, \rho_r = 1$.

It should be noted, that to achieve this with algebraic VoF requires close attention to sources of numerical noise in the simulation. For example, the termination criterion for deciding what volume fraction constitutes the interface must be selected for the height function algorithm. This was selected as $1.0e^{-12}$. However, in a simulation with greater volume transport this number has to be increased due to trailing noise around the interface. This is an inherent deficiency of algebraic VoF and will be discussed further in the next test case.

To complete the study a mesh convergence analysis was conducted, using a $La = 120$, $\rho_r = 1$. The L_∞ norm for pressure jump is computed via

$$\|\varepsilon_{\Delta p}\|_\infty = \frac{|\Delta p_n - \Delta p_a|}{\Delta p_a},$$

Table 5.1: Material properties for the rising bubble test case in 2D.

Case No.	Bo	Re	ρ_l/ρ_g	μ_l/μ_g
1	10	35	10	10
2	125	35	1000	100

where Δp_a is the Laplace pressure jump and Δp_n is the difference between the maximum and minimum pressure in the domain. The results of the study are illustrated in Figure 5.6, and demonstrates second order spatial convergence for the estimation of the pressure jump.

5.4.3 Rising Bubble 2D

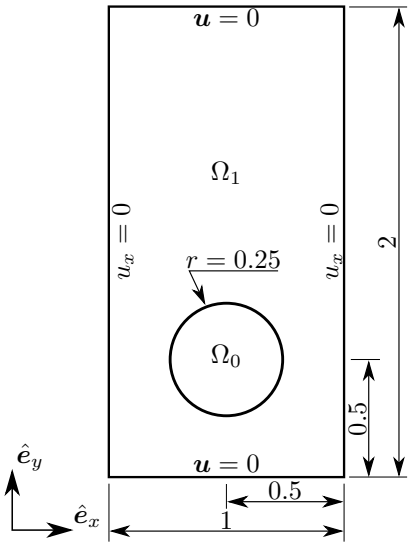


Figure 5.7: Initial conditions for a the rising bubble test case in 2D, Ω_1 represents the liquid and Ω_0 represents the gas.

The final two test cases evaluate the performance of the coupling between interface transport and the HiRAC-CSF scheme for a moving interface via a rising bubble in 2 and 3D. To begin this test case is that of a 2D bubble rising in a viscous fluid, where a benchmark study conducted by Hysing et al. [97] is employed. A bubble is placed at the bottom half of a rectangular domain subject to gravity as shown in Figure 5.7, and its buoyancy driven rise modelled to $t = 3s$. As the bubble rises its shape is deformed where the final shape is a function of the Reynolds, Re, and Bond, Bo, numbers respectively defined as

$$Re = \frac{\rho_l u_g D}{\mu_l}, \quad Bo = \frac{\rho_l u_g^2 D}{\sigma},$$

where $u_g = \sqrt{g_y D}$ and is the characteristic gravitational velocity. The gravitational constant is represented by g_y and set to 0.98. The radius, r, of the bubble is set to 0.25. The fluid properties for the two cases simulated are provided in Table 5.1. Note that to save computational time only half the domain was simulated and a symmetry plane was created along $x = 0.5$. Finally, for this case HiRAC was used as the advective VoF method.

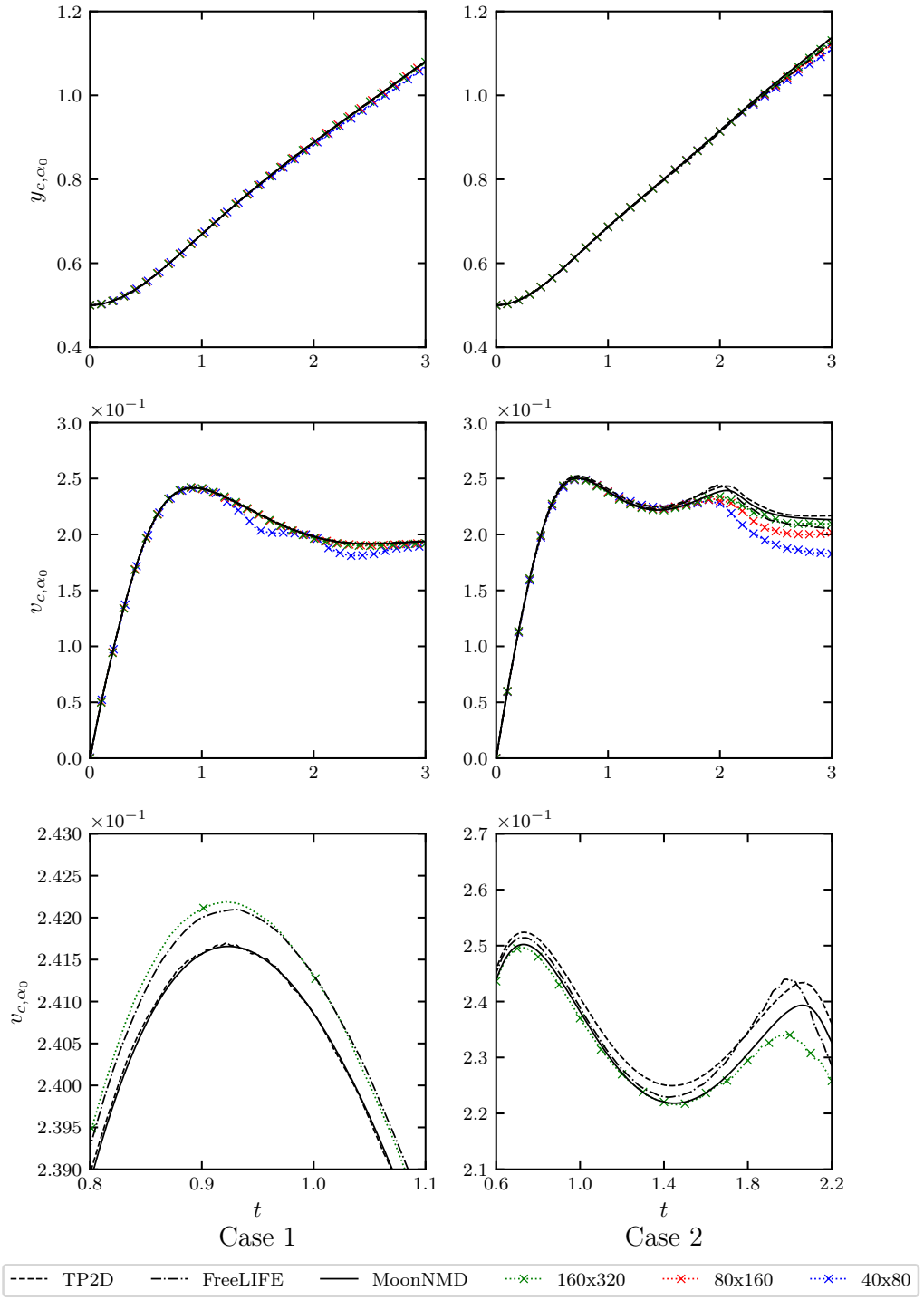


Figure 5.8: Plots of the evolution of the y component of the position (top), velocity (middle), and zoomed in section of the velocity on the finest meshes (bottom) for the 2D rising bubble.

Benchmark solutions were created by Hysing et al. [97] using three codes namely, TP2D [98, 99], FreeLife [100, 101], and MooNMD [102]. The use of benchmarked solutions is due to the lack of analytical or experimental data for this case. For both test cases a number of metrics are proposed to allow for benchmarking. The first

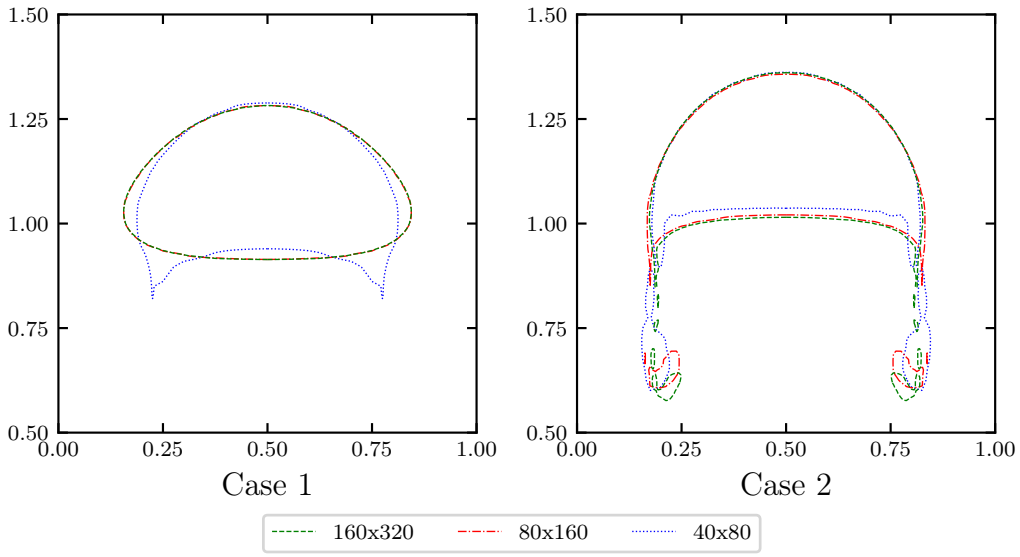


Figure 5.9: Contour plots for $\alpha = 0.5$ of the 2D bubble at $t = 3$s.

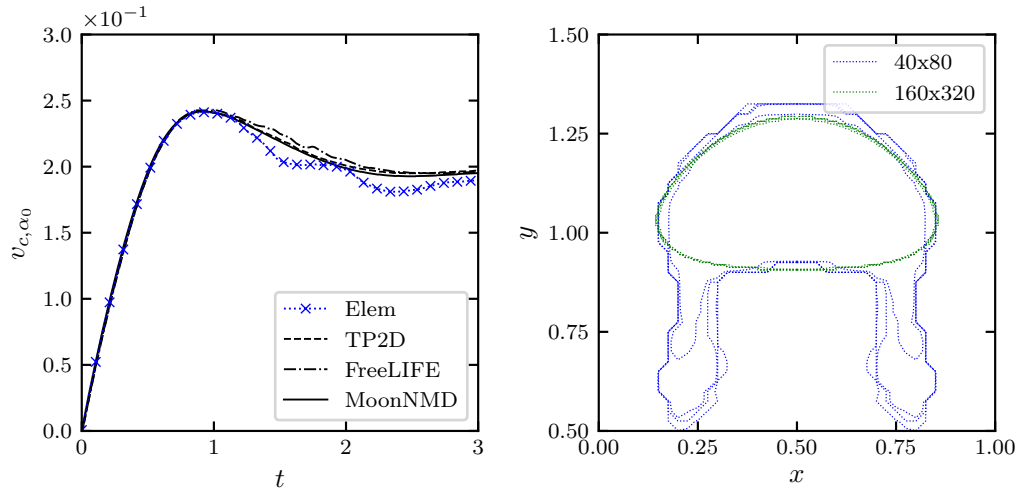


Figure 5.10: Comparison of the evolution of velocity for the coarsest meshes (left) and contour plots for $\alpha = 1.0e^{-n} | n \in \{1, 3, 5, 7\}$ of the 2D bubble at $t = 3$s on the coarsest and finest meshes (right) for Case 1.

metric is the evolution of the centre of mass of the bubble, $\boldsymbol{x}_{c,\alpha_0}$, expressed as

$$\boldsymbol{x}_{c,\alpha_0} = \frac{\sum_i (1-\alpha_i)\boldsymbol{x}_i V_i}{\sum_i (1-\alpha_i) V_i}, \tag{5.12}$$

where $\alpha_i, \boldsymbol{x}_i$, and V_i symbolise the nodal volume fraction, co-ordinate, and volume respectively. The rise velocity of the bubble, $\boldsymbol{u}_{c,\alpha_0}$, is formulated as

$$\boldsymbol{u}_{c,\alpha_0} = \frac{\sum_i (1-\alpha_i)\boldsymbol{u}_i V_i}{\sum_i (1-\alpha_i) V_i}, \tag{5.13}$$

where $\boldsymbol{u}_i$ symbolises the nodal velocity.

Figure 5.8 plots the evolution of the y component of the position and velocity of the bubble, as well as zoomed in section for the velocity evolution. Also depicted are

the results from the other benchmarked codes at their respective finest resolutions, equivalent to the finest mesh resolution ($\Delta x = 6.25e^{-3}$) used here. For Case 1 the plots agree well with the benchmark data, both in terms of position and velocity. The zoomed-in velocity plot shows a similar prediction for the peak rise velocity as compared to the benchmark codes.

On the coarsest resolution there is a clear deviation of the rise position and velocity of the bubble. Figure 5.9 illustrates how the coarsest bubble severely, and non-physically, deformed compared with the higher resolution data. This is attributable to an inability of the height function method to form consistent heights at the points of maximum curvature, due to insufficient spatial resolution.

Figure 5.10 left shows the non-physical deviation of the coarse result as compared to the coarse data from literature [97]. The field plot on the right provides further insight, and can be used to continue the discussion started in the previous case: The cut-off threshold for determining if a node is on the interface, namely $\alpha \in (0+\epsilon_\alpha, 1-\epsilon_\alpha)$. For the static bubble case a strict tolerance could be applied, $\epsilon_\alpha = 1.0e^{-12}$. Due to the erroneous trailing noise caused by HiRAC on coarse meshes (Figure 5.10 right), this value had to be significantly increased, $\epsilon_\alpha = 1.0e^{-6}$, to prevent the interface being detected in the bulk fluid. Therefore, at low resolutions the trailing noise around the interface is still relatively high, and the column construction algorithm produces inconsistent heights.

For Case 2 the results obtained, see Figure 5.8, predict a similar magnitude and temporal position for the first velocity peak when compared to the benchmark codes. From *circa* $t = 1.5$s, where all codes begin to differ in prediction, the data begins to deviate somewhat from the benchmark codes. The final position of the bubble is similar to the comparitive data.

In summary, the schemes performance on this test case is similar to previous works and thus demonstrates the ability of the scheme to model dynamic moving interfaces where surface tension is present. However, this is limited to meshes where there is sufficient spatial resolution to reduce trailing noise in the volume fraction field.

5.4.4 Rising Bubble 3D

To conclude the results section a 3D rising bubble is presented using HiRAC, this to demonstrate working robustness in 3D. For comparison, the work by Safi et al. [103] is employed for the extension of the previous case to 3D. In this case the circle is extended to a sphere of the same radius and placed at $\boldsymbol{x} = \{0.5, 0.5, 0.5\}$. For this case the domain is extended by one unit along the z-axis, and a no-slip condition applied to all boundaries.

The material properties which are provided in Table 5.1 Case 1 are used here. The error metrics, defined by Equation (5.12) and (5.13), are again used for quantitative comparison. To save computational cost, only a quarter of the domain is simulated, with symmetry condition applied to the internal boundaries.

The evolution of the y component of the position and velocity of the bubble are plotted in Figure 5.11. The FeatFlow [104] data was collected by digitising results obtained by Safi et al. [103] with a spatial resolution of $\Delta x = 7.8125e^{-3}$. The three grid spacings employed for this work were $\Delta x = \{2.5e^{-2}, 1.25e^{-2}, 6.25e^{-3}\}$.

The coarsest mesh again deviates significantly from the two finer meshes, see Figure 5.12 which shows conic features trailing under the bubble. This is no-doubt a 3D version of the skirt that was observed for the coarse 2D Case 1 result. This feature is again a result of the volume fraction field noise in regard to height construction, discussed earlier.

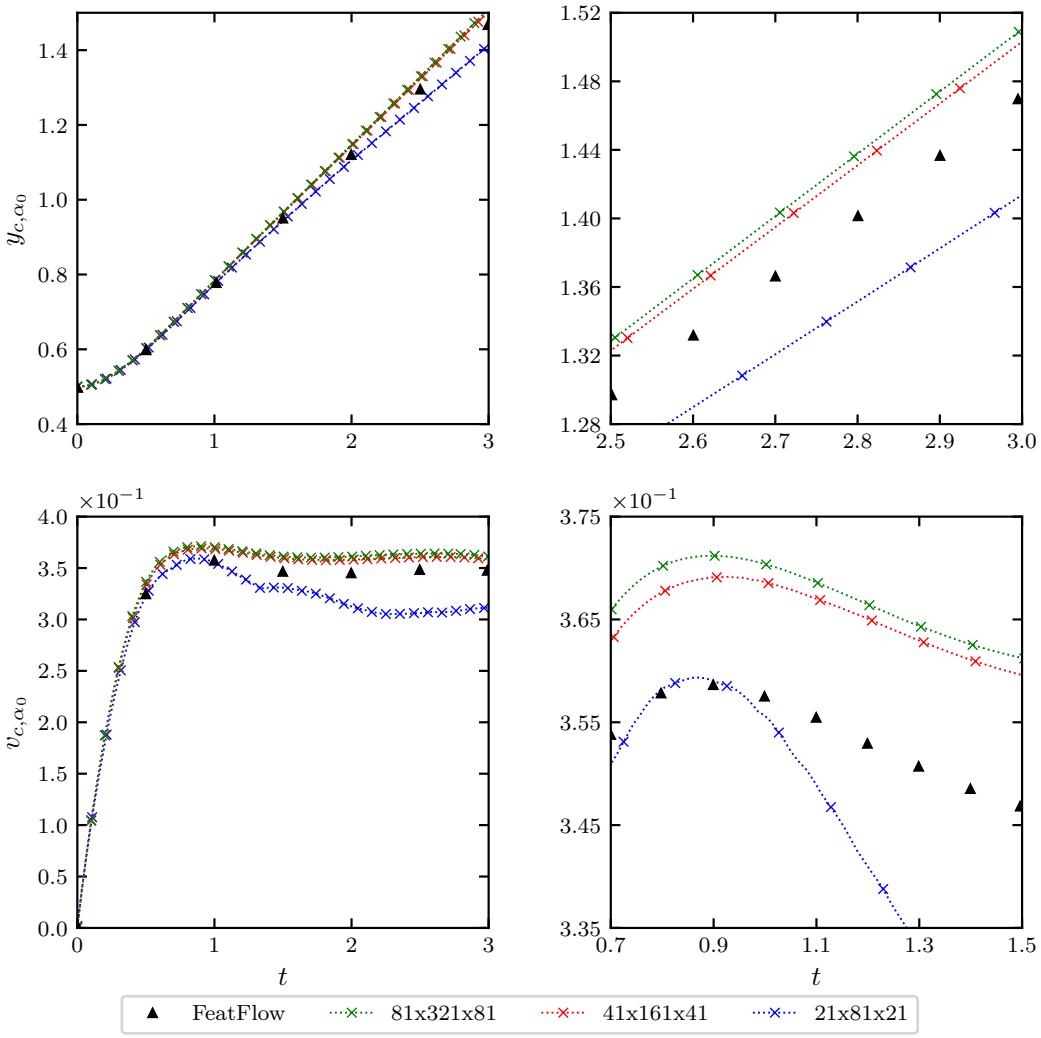


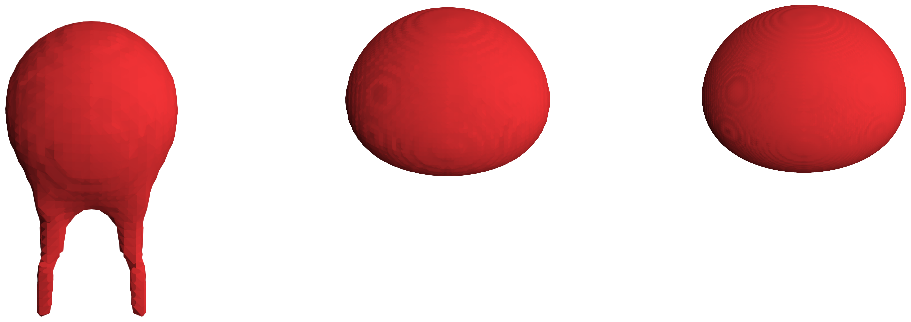
Figure 5.11: Plots of the evolution of the y component of the position (top) and velocity (bottom) of the 3D rising bubble.

The FeatFlow and HiRAC results compare relatively well in terms of peak velocity, with a deviation of 3.4%. Both methods also show this peak to occur at similar points in time *circa* $t = 0.9$s. Additionally, FeatFlow predicts the final bubble position to be $\approx 2.2\%$ lower than the current scheme.

5.5 Conclusion and Summary

The height function method was implemented to compute two-phase interface curvature for the algebraic VoF advected methods of HiRAC (in addition CICSAM on the static bubble test case). The curvature scheme was incorporated into the conservation equations in a manner that ensures a well balanced formulation, which additionally is able to handle non-unit density ratios. While the height function algorithm requires a structured grid the remaining components of the outlined algorithm are readily applicable to an unstructured approach.

The conducted static bubble test case demonstrated that the inclusion of the surface tension term was completed in a well balanced manner. Test cases with interface transport were conducted with HiRAC and revealed that on coarser mesh



N=20×80×20 N=40×160×40 N=80×320×80

Figure 5.12: Contour plots for $\alpha = 0.5$ of the 3D rising bubble at $t = 3$s for the three mesh resolutions.

resolutions it is not competitive with the geometric VoF methods cited in literature. This owing to numerical noise caused in the volume fraction field which necessitated a case specific determination of a suitable α 'cut-off' value for height function calculations. However, on finer resolution the method preform nominally compared to benchmarked cases.

Chapter 6

Numerical Rigid Body Dynamics

6.1 Introduction

This chapter details the derivation and numerical implementation of the dynamics governing rigid body motion. It is assumed that the reader is more familiar with CFD than with rigid body dynamics. As a rigid body dynamics code was not present in *Elemental* and had to be developed from the ground up, a brief derivation of the governing equations employed is presented. The preliminary mathematics will also lay notational frame work used throughout the Chapter.

To track the orientation of the rigid body as well as to transform vectors between the inertial, global, and non-inertial, body, axis quaternions are used. Their formulation is provided before the governing equations for the conservation of linear and angular momentum. Then Newton-Euler equations are presented. Finally, a number of test cases are presented to show 4th order time accuracy of the numerical implementation.

6.2 Preliminary Mathematics

6.2.1 Co-ordinate Basis and Reference Frames

The comprehension of rigid body dynamics requires the understanding of two different components. The first is the mathematical description of a vector within a certain axis system or basis. The second is the way that a vector appears to change, to an observer, in time. Here the observer could represent some different axis system.

Consider Figure 6.1, where there are a number of position vectors $\boldsymbol{r}$ describing the position of various entities in relation to two different sets of axis: $\mathbb{R}_o$ and $\mathbb{R}_{o'}$. Noting from Section 3.4.5 that a reference frame consists of an origin and three cardinal axis direction vectors. For convenience Equation (2.10) is repeated here, detailing the notation for a position vector as

$$\boldsymbol{r}_{i,j} = \boldsymbol{x}_j - \boldsymbol{x}_i,$$

where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ represent two points in space.

These position vectors could be expressed in either the $\mathbb{R}_o$ or $\mathbb{R}_{o'}$ basis, but require a transformation to be expressed in the other frame's basis. It is here that the quaternions provide the transformation capability and will be expanded shortly. From a notational point of view, the basis in which a vector is written appears as a pre-superscript. For example, ${}^{o'}\boldsymbol{r}_{o',c}$ is the displacement vector from the $\mathbb{R}_{o'}$ origin to the centre of mass expressed in the $\mathbb{R}_{o'}$ basis. However, the same vector expressed in the $\mathbb{R}_o$ basis is denoted ${}^{o}\boldsymbol{r}_{o',c}$.

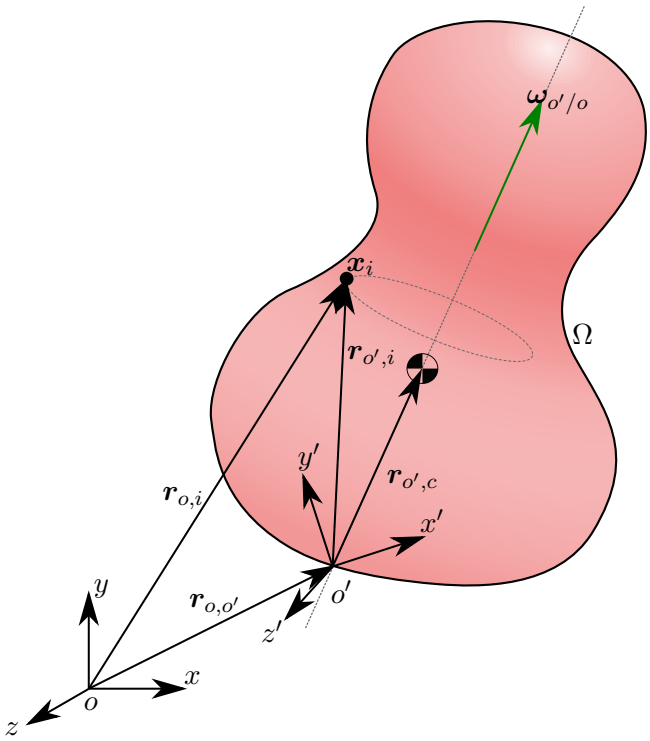


Figure 6.1: An arbitrary rigid body rotating about some axis with angular velocity $\boldsymbol{\omega}_{o'}$.

Now consider $\mathbb{R}_{o'}$ which is rotating, with some angular velocity $\boldsymbol{\omega}_{o'}$, thereby forming a non-conservative or non-inertial frame. Importantly, the way $\boldsymbol{r}_{o',c}$ appears to change if one were standing at $\mathbb{R}_{o'}$ or $\mathbb{R}_o$ is different. Note that this is irrespective of the basis in which the vector is mathematically expressed. From a notational perspective the change of a vector, $\boldsymbol{r}_{o',c}$ for example, as seen by (an observer), or relative to, $\mathbb{R}_{o'}$ is denoted $\dot{\boldsymbol{r}}_{o',c/o'}$ and when observed relative to $\mathbb{R}_o$ as $\dot{\boldsymbol{r}}_{o',c/o}$.

6.2.2 Attitude and Quaternion Transformations

Beyond a simple desire to know the attitude, or orientation, of a rigid body, the orientation of the body allows for the transformation of vectors quantities between the inertial and non-inertial frames. This is critical, for example, to the application of inertially defined forces (e.g. gravity) to the non-inertial basis in which the governing equations are expressed.

For linear motion, the final displacement of a rigid body can be directly deduced from the linear velocity and acceleration history. Equivalence for the angular counterparts does not apply, as the vector axioms do not hold for angular displacements. For example, angular displacements are not commutative; a rotation about one axis and then a second, is not congruent with first a rotation around the second and then the first axis.

A number of methods have been developed to track the attitude of a rigid body. Euler angles are one of the oldest such methods. However, Euler angles can suffer the loss of a degree of freedom, through gimbal lock. Due to this shortcoming, a number of alternatives were developed. Euler parameters, more commonly referred to as unit quaternions, are one such example and are used to track the attitude of a rigid body.

Quaternion mathematics, discovered by William Hamilton in the 19th century [105], was developed to describe the mechanics of 3-space, and introduces two additional imaginary axes to the standard complex-real axis. A feature Hamilton algebra,

denoted $\mathbb{H}$, is its non-commutativity where multiplication of the three imaginary axis unit vectors $\boldsymbol{i}, \boldsymbol{j}$, and $\boldsymbol{k}$ results in the following:

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1,$$

which implies the asymmetry,

$$\begin{aligned} \boldsymbol{ij} &= \boldsymbol{k} \quad \boldsymbol{ji} = -\boldsymbol{k}, \\ \boldsymbol{jk} &= \boldsymbol{i} \quad \boldsymbol{kj} = -\boldsymbol{i}, \\ \boldsymbol{ki} &= \boldsymbol{j} \quad \boldsymbol{ik} = -\boldsymbol{j}. \end{aligned}$$

A quaternion, $\boldsymbol{q}$, is made up of four components, three imaginary and one real,

$$\boldsymbol{q} = [q_0, q_1, q_2, q_3] = q_0 + q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k}.$$

Of practical relevance here is their ability to perform transformations between different frames, i.e. conversion between axis bases. For a given quaternion $\boldsymbol{q}$, the mapping between the inertial and non-inertial frames is given by

$$\begin{aligned} {}^{o'}\boldsymbol{r} &= \underline{\boldsymbol{T}}_q(\boldsymbol{q})\,{}^{o}\boldsymbol{r}, \\ {}^{o}\boldsymbol{r} &= \underline{\boldsymbol{T}}_q(\boldsymbol{q})^T\,{}^{o'}\boldsymbol{r}, \end{aligned}$$

where

$$\underline{\boldsymbol{T}}_q(\boldsymbol{q}) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix},$$

and $\underline{\boldsymbol{T}}_q$ is the transformation matrix and is as a function of the quaternion. The relation between the angular velocity and a quaternion is given by the following rate equations:

$$\begin{aligned} {}^{o}\boldsymbol{\omega}_{o'/o} &= 2\underline{\boldsymbol{W}}_q^o(\boldsymbol{q})\dot{\boldsymbol{q}}, \\ {}^{o'}\boldsymbol{\omega}_{o'/o} &= 2\underline{\boldsymbol{W}}_q^{o'}(\boldsymbol{q})\dot{\boldsymbol{q}}, \end{aligned}$$

where

$$\underline{\boldsymbol{W}}_q^o = \begin{bmatrix} -q_1 & q_0 & -q_3 & q_2 \\ -q_2 & q_3 & q_0 & -q_1 \\ -q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \text{ and } \underline{\boldsymbol{W}}_q^{o'} = \begin{bmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{bmatrix}.$$

Here $\dot{q}$ and $\underline{\boldsymbol{W}}_q$ represent the quaternion rate and the quaternion rate matrix respectively. Rearranged the quaternion rates $\boldsymbol{q}$ can be expressed as

$$\begin{aligned} \dot{\boldsymbol{q}} &= \frac{1}{2}\left(\underline{\boldsymbol{W}}_q^o(\boldsymbol{q})\right)^T {}^{o}\boldsymbol{\omega}_{o'/o}, \\ \dot{\boldsymbol{q}} &= \frac{1}{2}\left(\underline{\boldsymbol{W}}_q^{o'}(\boldsymbol{q})\right)^T {}^{o'}\boldsymbol{\omega}_{o'/o}. \end{aligned} \tag{6.1}$$

A comprehensive source of information and derivations on the conversion between the various attitude systems as well as between frame bases has been compiled by Diebel [106].

6.2.3 Kinematic Relationships

The *Transport Theorem*, not be be confused with the *Reynolds Transport Theorem*, provides a way to reconcile the varying observations made by observers situated in two different reference fames. Consider some vector $\boldsymbol{\phi}$, observed by two sets of axes with a relative angular velocity between them, $\boldsymbol{\omega}_{o'/o}$. The following relation describes rate of change of $\boldsymbol{\phi}$ as seen by the two observers (standing at o and o'),

$$\frac{d}{dt}(\boldsymbol{\phi})_{/o} = \frac{d}{dt}(\boldsymbol{\phi})_{/o'} + \boldsymbol{\omega}_{o'/o} \times \boldsymbol{\phi}, \tag{6.2}$$

or more compactly as

$$\dot{\boldsymbol{\phi}}_{/o} = \dot{\boldsymbol{\phi}}_{/o'} + \boldsymbol{\omega}_{o'/o} \times \boldsymbol{\phi}. \tag{6.3}$$

Consider now Figure 6.1, where $\mathbb{R}_{o'}$ is fixed to the rigid body, Ω. The position vector $\boldsymbol{r}_{o,i}$ can be described as the vector sum of the position vectors $\boldsymbol{r}_{o,o'}$ and $\boldsymbol{r}_{o',i}$, namely,

$$\boldsymbol{r}_{o,i} = \boldsymbol{r}_{o,o'} + \boldsymbol{r}_{o',i}.$$

First the case where the point $\boldsymbol{x}_i$ is not fixed to the rigid body is considered. This to derive the general kinematic expression for the acceleration of $\boldsymbol{x}_i$, used in the fluid governing equations. Finally, the simplification for the case of $\boldsymbol{x}_i$ fixed to the rigid body will be presented.

Using the Equation (6.2) the velocity of $\boldsymbol{r}_{o,i}$ can be expressed by

$$\begin{aligned}
\dot{\boldsymbol{r}}_{o,i/o} &= \dot{\boldsymbol{r}}_{o,o'/o} + \dot{\boldsymbol{r}}_{o',i/o}, \\
\dot{\boldsymbol{r}}_{o,i/o} &= \dot{\boldsymbol{r}}_{o,o'/o} + \dot{\boldsymbol{r}}_{o',i/o'} + \boldsymbol{\omega}_{o'/o} \times \boldsymbol{r}_{o',i}.
\end{aligned}$$

Differentiating again with respect to time for an observer at $\mathbb{R}_o$, and applying Equation (6.2), gives the expression for the inertial acceleration of $\boldsymbol{r}_{o,i}$ as

$$\begin{aligned}
\ddot{\boldsymbol{r}}_{o,i/o} &= \ddot{\boldsymbol{r}}_{o,o'/o} + \frac{\partial}{\partial t}\left(\dot{\boldsymbol{r}}_{o',i/o'} + \boldsymbol{\omega}_{o'/o} \times \boldsymbol{r}_{o',i}\right)_{/o}, \\
\ddot{\boldsymbol{r}}_{o,i/o} &= \ddot{\boldsymbol{r}}_{o,o'/o} + \ddot{\boldsymbol{r}}_{o',i/o'} + 2\boldsymbol{\omega}_{o'/o} \times \dot{\boldsymbol{r}}_{o',i/o'} + \dot{\boldsymbol{\omega}}_{o'/o} \times \boldsymbol{r}_{o',i} + \boldsymbol{\omega}_{o'/o} \times \left(\boldsymbol{\omega}_{o'/o} \times \boldsymbol{r}_{o',i}\right).
\end{aligned} \tag{6.4}$$

Then for a rigid body, where $\boldsymbol{x}_i$ is fixed in $\mathbb{R}_{o'}$, $\ddot{\boldsymbol{r}}_{o',i/o'}$ and $\boldsymbol{\omega}_{o'} \times \dot{\boldsymbol{r}}_{o',i/o'}$ are zero, the above simplifies to

$$\ddot{\boldsymbol{r}}_{o,i/o} = \ddot{\boldsymbol{r}}_{o,o'/o} + \dot{\boldsymbol{\omega}}_{o'/o} \times \boldsymbol{r}_{o',i} + \boldsymbol{\omega}_{o'/o} \times \left(\boldsymbol{\omega}_{o'/o} \times \boldsymbol{r}_{o',i}\right), \tag{6.5}$$

and describes the acceleration of a point $\boldsymbol{x}_i$ as seen by the inertial frame.

6.3 Conservation of Rigid Body Momentum

Since it is intended to express the fluid relative to the non-inertial frame, it is desirable to express the governing equations of the rigid body in the same frame. Further, for a more flexible numerical model it is favourable to express the equations away from the centre of mass, this allows for a rigid body to be 'constructed' from a number of sub-bodies, preventing a re-computation of position information due to a shifting centre of mass (and thereby a shifting local frame).

To begin, the conservation of linear and angular momentum of some rigid body is respectively expressed about its centre of mass by

$$\begin{aligned}\boldsymbol{G}_c &= \int_\Omega \rho(\boldsymbol{x}_i)\dot{\boldsymbol{r}}_{o,i/o}d\Omega = m\dot{\boldsymbol{r}}_{o,c/o},\\ \boldsymbol{H}_c &= \int_\Omega \rho(\boldsymbol{x}_i)\boldsymbol{r}_{c,i}\times(\boldsymbol{\omega}_{o'/o}\times\boldsymbol{r}_{c,i})d\Omega = \underline{\boldsymbol{I}}_c\boldsymbol{\omega}_{o'/o},\end{aligned}$$

where $\boldsymbol{x}_i$ is some point on the rigid body and $\boldsymbol{r}_{c,i}$ is the position vector from the centre of mass to the point. The velocity of the centre of mass, as seen by the inertial frame, is denoted $\dot{\boldsymbol{r}}_{o,c/o}$. $\underline{\boldsymbol{I}}_c$ is the 3×3 moment of inertia matrix about the centre of mass and has the form

$$\underline{\boldsymbol{I}}_c = -\sum_i m_i\boldsymbol{r}_{c,i}^\times\boldsymbol{r}_{c,i}^\times,$$

where the superscript '$\times$' denotes the skew-symmetric matrix cross product operator, for some $\boldsymbol{\phi}$ vector as

$$\boldsymbol{\phi}\times = \boldsymbol{\phi}^\times = \begin{bmatrix}0 & -\phi_z & \phi_y\\ \phi_z & 0 & -\phi_x\\ -\phi_y & \phi_x & 0\end{bmatrix}.$$

For the non-inertial frame, $\mathbb{R}_{o'}$, fixed to the rigid body, the above equations can be expressed about the frame's origin as

$$\begin{aligned}\boldsymbol{G}_{o'} &= \boldsymbol{G}_c = m\dot{\boldsymbol{r}}_{o,c/o},\\ \boldsymbol{H}_{o'} &= \boldsymbol{H}_c - \boldsymbol{r}_{o',c}\times\boldsymbol{G}_c = \underline{\boldsymbol{I}}_c\boldsymbol{\omega}_{o'/o} - \boldsymbol{r}_{o',c}\times m\dot{\boldsymbol{r}}_{o,c/o},\end{aligned}$$

using the *Parallel Axis Theorem*. The change in linear momentum of the body due to some force $\boldsymbol{f}$, assuming the solid mass is constant in time, is given by

$$\begin{aligned}\dot{\boldsymbol{G}}_{o'} = \int_\Omega d\boldsymbol{f} &= m\ddot{\boldsymbol{r}}_{o,c/o}\\ &= m\left(\ddot{\boldsymbol{r}}_{o,o'/o} + \dot{\boldsymbol{\omega}}_{o'/o}\times\boldsymbol{r}_{o',c} + \boldsymbol{\omega}_{o'/o}\times\left(\boldsymbol{\omega}_{o'/o}\times\boldsymbol{r}_{o',c}\right)\right)\\ &= m\ddot{\boldsymbol{r}}_{o,o'/o} + m\dot{\boldsymbol{\omega}}_{o'/o}\times\boldsymbol{r}_{o',c} + m\boldsymbol{\omega}_{o'/o}\times\left(\boldsymbol{\omega}_{o'/o}\times\boldsymbol{r}_{o',c}\right),\end{aligned}\tag{6.6}$$

using Equation (6.5), where $\ddot{\boldsymbol{r}}_{o,o'/o}$ is the acceleration of the origin of the non-inertial frame as seen by the inertial frame.

Assuming $\underline{\boldsymbol{I}}_c$ is constant in time, the change in angular momentum due to some torque about the centre of gravity, $\boldsymbol{\tau}_c$, is formulated as

$$\begin{aligned}\dot{\boldsymbol{H}}_{o'} = \int_\Omega d\boldsymbol{\tau}_{o'} &= \int_\Omega d\boldsymbol{\tau}_c - \boldsymbol{r}_{o',c}\times\int_\Omega d\boldsymbol{f}\\ &= \frac{d}{dt}\left(\underline{\boldsymbol{I}}_c\boldsymbol{\omega}_{o'/o}\right)_{/o} + \boldsymbol{r}_{o',c}\times\frac{d}{dt}\left(m\dot{\boldsymbol{r}}_{o,c/o}\right)_{/o}\\ &= \underline{\boldsymbol{I}}_c\dot{\boldsymbol{\omega}}_{o'/o} + \boldsymbol{\omega}_{o'/o}\times\underline{\boldsymbol{I}}_c\boldsymbol{\omega}_{o'/o} + \boldsymbol{r}_{o',c}\times m\ddot{\boldsymbol{r}}_{o,c/o}.\end{aligned}$$

Using Equation (6.5), the change in angular momentum is expanded further:

$$\begin{aligned}\int_\Omega d\boldsymbol{\tau}_{o'} =& \underline{\boldsymbol{I}}_c\dot{\boldsymbol{\omega}}_{o'/o} + \boldsymbol{\omega}_{o'/o} \times \underline{\boldsymbol{I}}_c\boldsymbol{\omega}_{o'/o} \\ &+ \boldsymbol{r}_{o',c} \times m\left(\ddot{\boldsymbol{r}}_{o',o/o} + \dot{\boldsymbol{\omega}}_{o'/o} \times \boldsymbol{r}_{o',c} + \boldsymbol{\omega}_{o'/o} \times \left(\boldsymbol{\omega}_{o'/o} \times \boldsymbol{r}_{o',c}\right)\right), \\ =& \underline{\boldsymbol{I}}_c\dot{\boldsymbol{\omega}}_{o'/o} + \boldsymbol{\omega}_{o'/o} \times \underline{\boldsymbol{I}}_c\boldsymbol{\omega}_{o'/o} + m\boldsymbol{r}_{o',c} \times \ddot{\boldsymbol{r}}_{o',o/o} + m\boldsymbol{r}_{o',c} \times \left(\dot{\boldsymbol{\omega}}_{o'/o} \times \boldsymbol{r}_{o',c}\right) \\ &+ m\boldsymbol{r}_{o',c} \times \left(\boldsymbol{\omega}_{o'/o} \times \left(\boldsymbol{\omega}_{o'/o} \times \boldsymbol{r}_{o',c}\right)\right). \end{aligned} \tag{6.7}$$

Noting that both Equations (6.6) and (6.7) contain linear and angular acceleration terms, and with some manipulation, the combined system can be expressed as,

$$\begin{bmatrix}\boldsymbol{f} \\ \boldsymbol{\tau}_{o'}\end{bmatrix} = \begin{bmatrix} m\underline{\boldsymbol{I}} & -m\boldsymbol{r}^\times_{o',c} \\ m\boldsymbol{r}^\times_{o',c} & \underline{\boldsymbol{I}}_c - m\boldsymbol{r}^\times_{o',c}\boldsymbol{r}^\times_{o',c}\end{bmatrix}\begin{bmatrix}\ddot{\boldsymbol{r}}_{o,o'/o} \\ \dot{\boldsymbol{\omega}}_{o'/o}\end{bmatrix} + \begin{bmatrix} m\boldsymbol{\omega}^\times_{o'/o}\boldsymbol{\omega}^\times_{o'/o}\boldsymbol{r}_{o',c} \\ \boldsymbol{\omega}^\times_{o'/o}\left(\underline{\boldsymbol{I}}_c - m\boldsymbol{r}^\times_{o',c}\boldsymbol{r}^\times_{o',c}\right)\boldsymbol{\omega}_{o'/o}\end{bmatrix}, \tag{6.8}$$

where $\underline{\boldsymbol{I}}$ is a 3×3 identity matrix. Finally rearranging such that the acceleration terms are on the left hand side the expression reads

$$\begin{bmatrix} m\underline{\boldsymbol{I}} & -m\boldsymbol{r}^\times_{o',c} \\ m\boldsymbol{r}^\times_{o',c} & \underline{\boldsymbol{I}}_c - m\boldsymbol{r}^\times_{o',c}\boldsymbol{r}^\times_{o',c}\end{bmatrix}\begin{bmatrix}\ddot{\boldsymbol{r}}_{o,o'/o} \\ \dot{\boldsymbol{\omega}}_{o'/o}\end{bmatrix} = \begin{bmatrix}\boldsymbol{f} \\ \boldsymbol{\tau}_{o'}\end{bmatrix} - \begin{bmatrix} m\boldsymbol{\omega}^\times_{o'/o}\boldsymbol{\omega}^\times_{o'/o}\boldsymbol{r}_{o',c} \\ \boldsymbol{\omega}^\times_{o'/o}\left(\underline{\boldsymbol{I}}_c - m\boldsymbol{r}^\times_{o',c}\boldsymbol{r}^\times_{o',c}\right)\boldsymbol{\omega}_{o'/o}\end{bmatrix},$$

or written more compactly as

$$\underline{\boldsymbol{I}}_{o'}\ddot{\boldsymbol{r}}_{o'} = \boldsymbol{f}_{o'}, \tag{6.9}$$

where $\boldsymbol{I}_{o'}$ is the system's inertial matrix, $\ddot{\boldsymbol{r}}_{o'}$ denotes a 6 entry column vector of linear and angular acceleration of the frame, and $\boldsymbol{f}_{o'}$ is the real and fictitious forces and moments around the frame's origin.

6.4 Temporal Discretisation

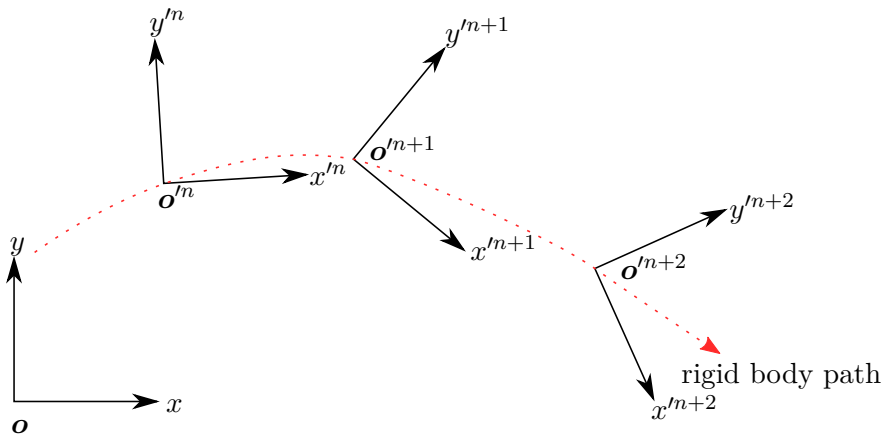


Figure 6.2: The rigid body's fixed frame's position over successive discrete time steps.

Equation (6.9) is rearranged,

$$\ddot{\boldsymbol{r}}_{o'} = \underline{\boldsymbol{I}}^{-1}_{o'}\boldsymbol{f}_{o'}, \tag{6.10}$$

to yield the inertial acceleration of the non-inertial frame origin. However, this expression provides only the instantaneous acceleration for the frame for some time step n as per Figure 6.2. The kinematic state of the rigid body is 'transferred' to the 'next frame' at $n+1$, using Equation (6.2), by

$$\begin{aligned}\frac{d}{dt}\left(\dot{\boldsymbol{r}}_{o,o'/o}\right)_{/o'^{n+1}} &= \frac{d}{dt}\left(\dot{\boldsymbol{r}}_{o,o'/o}\right)_{/o} + \boldsymbol{\omega}^n_{o/o'} \times \dot{\boldsymbol{r}}^n_{o,o'/o} \\ &= \ddot{\boldsymbol{r}}_{o,o'/o} - \boldsymbol{\omega}^n_{o'/o} \times \dot{\boldsymbol{r}}^n_{o,o'/o}.\end{aligned}$$

The displacement vector is updated similarly. Combining the above with Equation (6.10), the final semi-discrete numerical expression is given as

$$\ddot{\boldsymbol{r}}^{n+1}_{o'} = g_{\ddot{\boldsymbol{r}}_{o'}}(\boldsymbol{\omega}_{o'/o}, \boldsymbol{f}_{o'}(\boldsymbol{\omega}_{o'/o}, \boldsymbol{q})) = \underline{\boldsymbol{I}}^{-1}_{o'} \boldsymbol{f}_{o'}(\boldsymbol{\omega}_{o'/o}, \boldsymbol{q})\Big|^n - \begin{bmatrix} \boldsymbol{\omega}^{\times}_{o'/o} & 0 \\ 0 & 0 \end{bmatrix} \dot{\boldsymbol{r}}_{o'}\Bigg|^n,$$

where the bottom right entry is zero, since $\boldsymbol{\omega}_{o'/o} \times \boldsymbol{\omega}_{o'/o}$ is zero. $g_{\ddot{\boldsymbol{r}}_{o'}}$ is introduced as a function to represent the right hand side. This aids in compacting the Runge-Kutta formulae presented shortly. The displacement vector of the non-inertial frame is updated using

$$\dot{\boldsymbol{r}}^{n+1}_{o,o'/o} = g_{\dot{\boldsymbol{r}}_{o'}}(\boldsymbol{\omega}_{o'/o}, \boldsymbol{r}_{o,o'}) = \dot{\boldsymbol{r}}^n_{o,o'/o} - \boldsymbol{\omega}^n_{o'/o} \times \boldsymbol{r}^n_{o,o'},$$

and the attitude of the system is updated using the computed angular velocity and rearranging Equation (6.1),

$$\dot{\boldsymbol{q}}^{n+1} = g_{\dot{\boldsymbol{q}}}(\boldsymbol{q}, \boldsymbol{\omega}) = \frac{1}{2}\underline{\boldsymbol{W}}'_q(\boldsymbol{q}^n)^T \boldsymbol{\omega}^n_{o'/o}.$$

The full set of rigid body governing equations in continuous form are

$$\begin{bmatrix} \ddot{\boldsymbol{r}}_{o'} \\ \dot{\boldsymbol{r}}_{o'} \\ \dot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} g_{\ddot{\boldsymbol{r}}_{o'}}(\boldsymbol{\omega}_{o'/o}, \boldsymbol{f}_{o'}(\boldsymbol{\omega}_{o'/o}, \boldsymbol{q})) \\ g_{\dot{\boldsymbol{r}}_{o'}}(\boldsymbol{\omega}_{o'/o}, \boldsymbol{r}_{o,o'}) \\ g_{\dot{\boldsymbol{q}}}(\boldsymbol{q}, \boldsymbol{\omega}_{o'/o}), \end{bmatrix}$$

and written more compactly as

$$\dot{\boldsymbol{s}} = \boldsymbol{g}(\boldsymbol{\omega}_{o'/o}, \boldsymbol{q}, \boldsymbol{r}_{o,o'}, \boldsymbol{f}_{o'}) \tag{6.11}$$

The coupled system is solved with a 4th order Runge-Kutta method:

$$\begin{aligned}
\boldsymbol{k}_1 &= \Delta t \boldsymbol{g}(\boldsymbol{\omega}^n_{o'/o}, \boldsymbol{q}^n, \boldsymbol{r}^n_{o,o'}, \boldsymbol{f}^n_{o'}), \\
\boldsymbol{k}_2 &= \Delta t \boldsymbol{g}\left(\boldsymbol{\omega}^n_{o'/o} + \frac{\boldsymbol{k}_{1,\boldsymbol{\omega}_{o'/o}}}{2}, \boldsymbol{q}^n + \frac{\boldsymbol{k}_{1,\boldsymbol{q}}}{2}, \boldsymbol{r}^n_{o,o'} + \frac{\boldsymbol{k}_{1,\boldsymbol{r}_{o,o'}}}{2}, \boldsymbol{f}^n_{o'} + \frac{\boldsymbol{k}_{1,\boldsymbol{f}_{o'}}}{2}\right), \\
\boldsymbol{k}_3 &= \Delta t \boldsymbol{g}\left(\boldsymbol{\omega}^n_{o'/o} + \frac{\boldsymbol{k}_{2,\boldsymbol{\omega}_{o'/o}}}{2}, \boldsymbol{q}^n + \frac{\boldsymbol{k}_{2,\boldsymbol{q}}}{2}, \boldsymbol{r}^n_{o,o'} + \frac{\boldsymbol{k}_{2,\boldsymbol{r}_{o,o'}}}{2}, \boldsymbol{f}^n_{o'} + \frac{\boldsymbol{k}_{2,\boldsymbol{f}_{o'}}}{2}\right), \\
\boldsymbol{k}_4 &= \Delta t \boldsymbol{g}(\boldsymbol{\omega}^n_{o'/o} + \boldsymbol{k}_{3,\boldsymbol{\omega}_{o'/o}}, \boldsymbol{q}^n + \boldsymbol{k}_{3,\boldsymbol{q}}, \boldsymbol{r}^n_{o,o'} + \boldsymbol{k}_{3,\boldsymbol{r}_{o,o'}}, \boldsymbol{f}^n_{o'} + \boldsymbol{k}_{3,\boldsymbol{f}_{o'}}), \\
\boldsymbol{s}^{n+1} &= \frac{1}{6}\left(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4\right),
\end{aligned} \tag{6.12}$$

where the second subscripts to the stage vector $\boldsymbol{k}$ denotes the sub elements relating to the angular velocity, quaternion, position, and force and torque of the rigid body.

6.5 Results

6.5.1 Torque Free Rotating Body

The first test case demonstrates the capability of the scheme to model a body, with some arbitrary spin, subjected to a sinusoidal varying force. The latter is applied at its centre of gravity. For a disc, the moment of inertia about its centre of gravity is given by

$$\underline{\boldsymbol{I}}_c = \begin{bmatrix} \frac{mr_d^2}{4} + \frac{mh_d^2}{12} & 0 & 0 \\ 0 & \frac{mr_d^2}{4} + \frac{mh_d^2}{12} & 0 \\ 0 & 0 & \frac{mr_d^2}{2} \end{bmatrix}, \tag{6.13}$$

where m, r_d, and h_d are its mass, radius, and height. For this case a value of $m = 1\text{kg}$, $r_d = 0.1\text{m}$, and $h_d = 0.02\text{m}$ have been selected. The origin and initial attitude of the non-inertial frame is congruent with that of the inertial frame, as depicted in Figure 6.3. The centre of mass of the disc is displaced by the vector ${}^{o'}\boldsymbol{r}_{o',c} = [0.05, 0.1, 0.025]\text{m}$ from the origin. The initial linear velocity is ${}^{o'}\dot{\boldsymbol{r}}_{o',c/o} = [-1, 0.5, -0.5]\text{m.s}^{-1}$ and the rotational velocity set to ${}^{o'}\boldsymbol{\omega}_{o'/o} = [1\pi, 3\pi, 2\pi]\text{rad.s}^{-1}$. The applied force is varied such that the net imparted momentum over the time interval $[0, t_m]$ is zero,

$${}^{o}\boldsymbol{f}_c = [2.5, 1.25, 3.75] \sin\left(\frac{2\pi t}{t_m}\right) \text{N}.$$

Here the above vector is expressed with respect to the inertial frame's basis. For the purpose of the simulation, the force is transformed to the non-inertial frames' basis, thus testing the quaternion attitude dynamics. The simulation time was set to $t_m = 1\text{s}$.

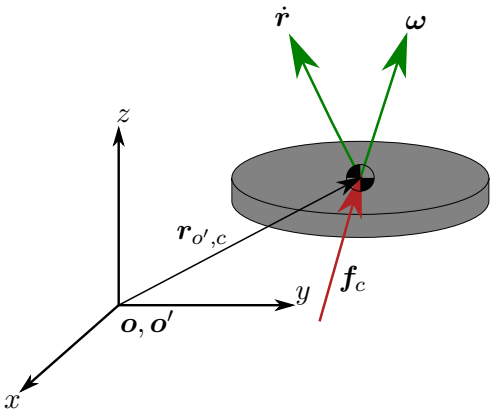


Figure 6.3: Schematic of the initial conditions for a rotating disc.

As a subtle note, a torque is present at the origin of the non-inertial frame, namely ${}^{o'}\boldsymbol{r}_{o',c} \times {}^{o'}\boldsymbol{f}_c$, which represents the transfer of the force from the centre of gravity of the body to the origin of the frame, however this is computed internally by the rigid body code/dynamics. However, as the test case name implies, this system is congruent to zero applied torque about the centre of gravity of the rigid body.

The resulting evolution of the linear and angular momenta for the body are plotted in Figure 6.4. As stated the net imparted momentum is zero, thus the error, ε, is determined via computing the change in momentum of the system at the end of the

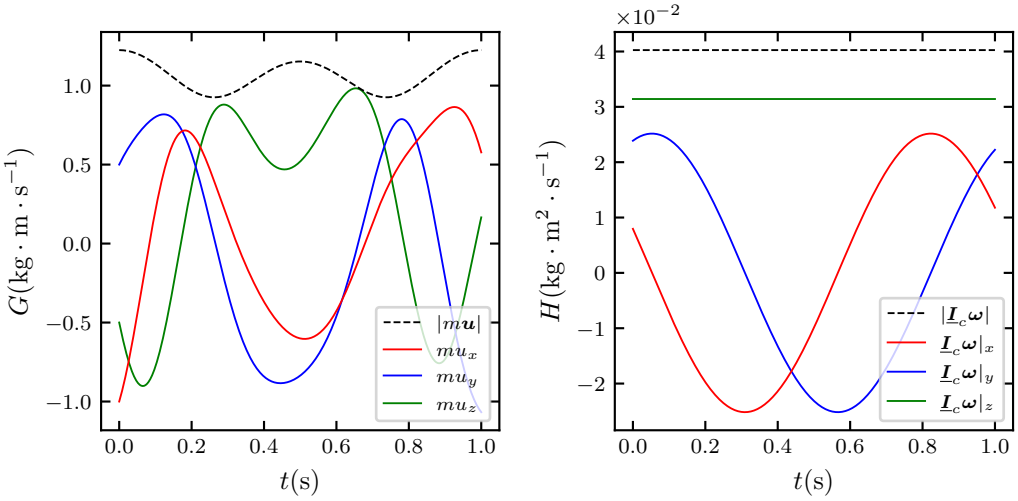


Figure 6.4: Time evolution of the linear(left) and angler(right) momenta of the centre of gravity for the disc.

simulation via

$$\varepsilon_{\boldsymbol{G}_c} = \left| \frac{|\boldsymbol{G}_c^{t_m}| - |\boldsymbol{G}_c^0|}{|\boldsymbol{G}_c^0|} \right|,$$
$$\varepsilon_{\boldsymbol{H}_c} = \left| \frac{|\boldsymbol{H}_c^{t_m}| - |\boldsymbol{H}_c^0|}{|\boldsymbol{H}_c^0|} \right|,$$

where the superscript 0 and t_m to the linear and angular momenta denote their values at the start and end of the simulation. To perform a time step size convergence study the simulation was repeated, where for every repeated simulation the time step size was reduce by an order of magnitude. The final time step resolution was $\Delta t = 1.0e^{-3}$s. The results are plotted in Figure 6.5, from which it is clear that the temporal scheme is 4th order accurate.

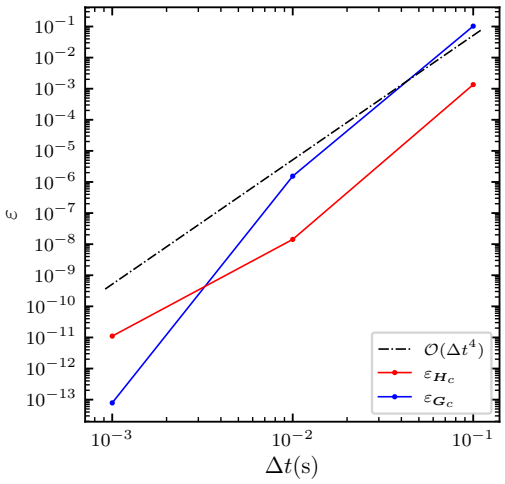


Figure 6.5: Convergence of the relative error of linear and angular momenta with respect to the time step size.

6.5.2 Lagrangian Top

To validate the angular momentum components the classic Lagrangian Top was simulated. The spinning top is modelled as a heavy axis-symmetric rigid body, fixed at the origin, but free to rotate, and under constant gravitational acceleration as shown in Figure 6.6. The top, due to the conservation of angular momentum and energy, will develop some precession and nutation rate around the inertial frame.

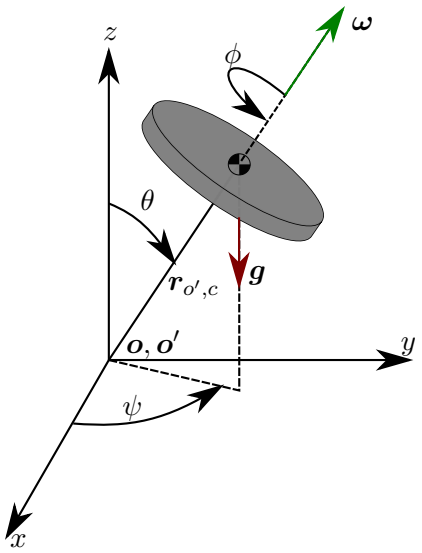


Figure 6.6: Diagram of a Lagrangian spinning top with Euler angles for precession, nutation, and spin drawn.

The analytical solution to the problem is derived and computed numerically using Euler Angles. The rates for the spin, precession, and nutation of the top are symbolised $\dot{\phi}, \dot{\psi}, \dot{\theta}$ respectively, are given by

$$\begin{aligned}
\dot{\phi} &= \frac{H_\phi}{I_{zz'}} - \frac{H_\psi - H_\phi \cos\theta}{I_{xx'} \sin^2\theta} \cos\theta, \\
\dot{\psi} &= \frac{H_\psi - H_\phi \cos\theta}{I_{xx'} \sin^2\theta}, \\
\dot{\vartheta} &= -\frac{E_p'(\theta)}{I_{xx'}}, \\
\dot{\theta} &= \vartheta,
\end{aligned}$$

where H_ϕ and H_ψ represent the angular momenta about the spin and precession axis. The nutation angle is given by θ and $I_{xx'}$ and $I_{zz'}$ symbolise the moment of inertia about the local x and z axes of the top (where the local z axis is the axis about which the top spins). Finally, E_p' is the derivative of the potential energy with respect to time. A derivation of the above analytical solution and further details regarding Euler Angles are provided in Appendix A.3.

The model of the top consists of a disc with a massless stork, and Equation (6.13) is used to obtain the second moment of inertia tensor for the body. The mass, radius,

and height of the top are calculated as

$$m = \frac{20}{g}\text{kg},$$
$$r_d = \sqrt{\frac{2}{m}}\text{m},$$
$$h_d = \sqrt{\frac{12(5 - m(1 + 0.25r_d^2)}{m}}\text{m}.$$

The centre of gravity for the top was placed at ${}^{o'}\boldsymbol{r}_{o',c} = [0, 0, 1]$m. The initial Euler Angle is set to $\boldsymbol{\theta}_{313} = [0.0, 20.0, 0.0]^\circ$ and their rates are initialised as $\dot{\boldsymbol{\theta}}_{313} = [50.0, 0.5, 0.0]\text{rad.s}^{-1}$. Finally the condition $\ddot{\boldsymbol{r}}_{o,o'/o} = 0\text{m.s}^{-2}$ is enforced throughout. Gravity is applied along the inertial z axis and at each time step transformed, using the quaternions, into the non-inertial basis.

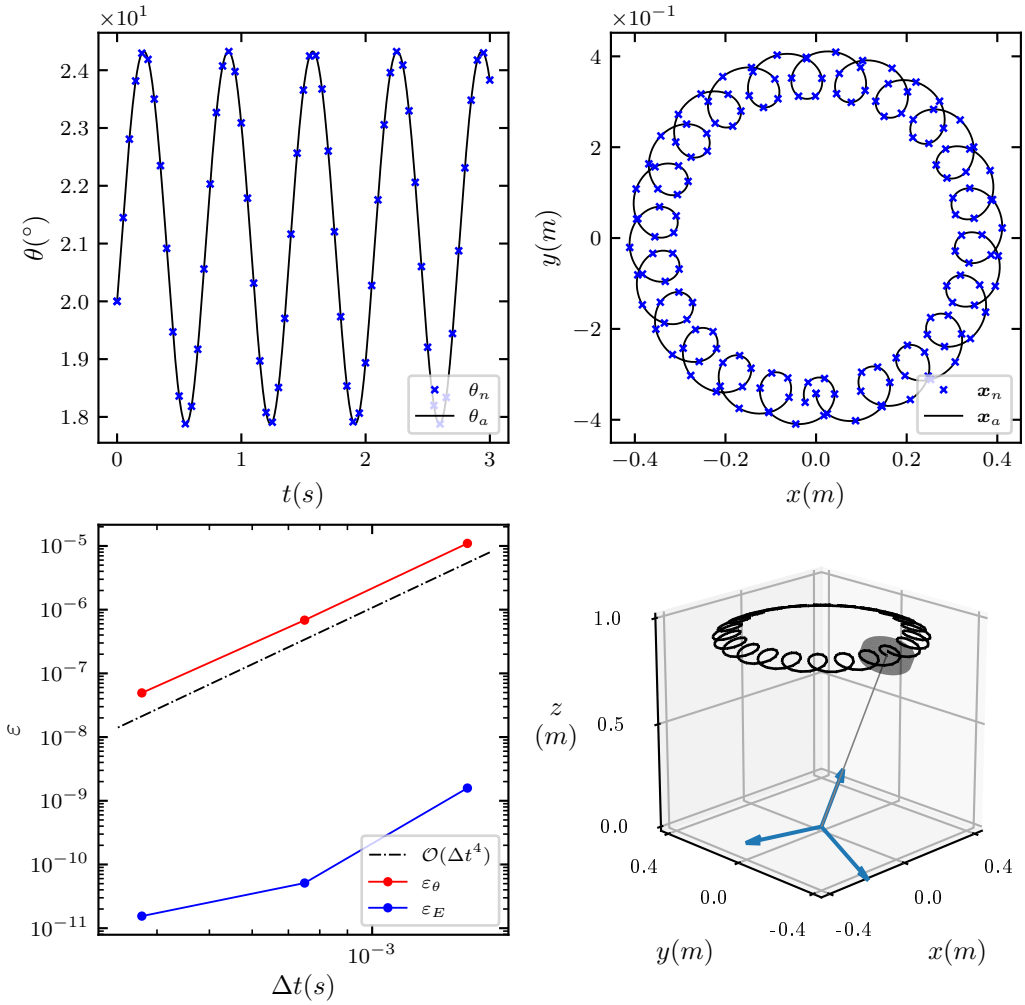


Figure 6.7: Graph of the evolution of the nutation angle, θ, with respect to time (top left) and the trace of the centre of gravity in global xy co-ordinates (top right). Subscripts a and n refer to analytical and numerical solutions respectively. The convergence graph of the relative energy error ε_E with a respect to the time step size (bottom left). A 3D plot of the spinning top and its centre of gravity traced in black (bottom right).

The evolution of the nutation angle as a function of time as well as the trace for the centre of gravity of the top, in the xy plane, are depicted in Figure 6.7. A complete precession takes approximately 15 seconds during which time the data for the xy trace is collected. The nutation plot is truncated to the first 3 seconds of the simulation. A further 3D plot is drawn in Figure 6.7, and shows the traced path the centre of mass makes as well as the orientation of the non-inertial frame at the end of the simulation.

Figure 6.7 shows agreement between the analytical and numerical solutions. However, to perform a more rigorous analysis, an order of accuracy study was conducted. Two metrics were used, the first is the final θ at t_m and the second is the loss of energy of the system, which is closed. The latter is computed by comparing the difference of the sum of the linear, rotational, and potential energy at the start and end of the simulation. The two relative error metrics are given by

$$\varepsilon_\theta = \left| \frac{\theta_n^{t_m} - \theta_a^{t_m}}{\theta_a^{t_m}} \right|,$$
$$\varepsilon_E = \left| \frac{E^{t_m} - E^0}{E^0} \right|,$$

where superscripts t_m and 0 denote the start and end quantities and the subscripts a and n denote analytical and numerical values respectively. The initial time step size was set to $1.5e^{-2}$s and each successive simulation reduced the step size by an order of magnitude. The collected data is plotted in Figure 6.7 and shows that both error metrics demonstrate that the scheme is 4th order with respect to time.

6.6 Conclusion

The chapter details the derivation of the Newton-Euler equations describing the dynamics of a rigid body. Additionally, the attitude of a rigid body was described using quaternions. The formulation was discretised in time using a fourth order Runge-Kutta approach resulting in a method to simulate the dynamics of a generic rigid body.

The model was used to simulate a disc with random initial momenta and a time varying force was applied. Where the formulated force was created such that the net imparted momentum remains zero. The second test case was a classic Lagrangian Top which, while having no linear components, has a convoluted angular component. Both required attitude tracking. For both test cases the model proved robust and demonstrated 4th order accuracy in time.

Chapter 7

Strong Coupling of Fluids and Rigid Bodies

7.1 Introduction

Numerical instability, due to the large liquid-to-mass ratios, must be overcome if a multi-physics Fluid-Structure Interaction (FSI) model of orbital craft is to be created. Stability can be obtained via a so called strongly coupled FSI approach. Gerrits and Veldman [23, 30] achieved this via an essentially monolithic approach which required extensive data transfer between fluid and solid solvers. This work aims to develop a strongly coupled fully partitioned method for spacecraft modelling by combining the components from previous chapters.

A partitioned approach is preferred as it allows easy coupling of generic black-box fluid and solid modelling codes, and ensures a flexible modelling tool into the future. An approach to create a tightly coupled model is the staggered Fixed-Point Iteration (FPI) method. To describe this method both the fluid and solid, Equations (1.7) and (1.6), are simplified to a root-finding formulation:

$$\begin{aligned} 0 &= \boldsymbol{k}_t^{n+1} - \mathcal{F}\left(\boldsymbol{k}_m^{n+1}\right), \\ 0 &= \boldsymbol{k}_m^{n+1} - \mathcal{S}\left(\boldsymbol{k}_t^{n+1}\right), \end{aligned}$$

where $\boldsymbol{k}_t$ represents a column vector of force and torque, and $\boldsymbol{k}_m$ is a vector containing the linear and angular accelerations as well as angular velocity. The fluid solver is denoted by $\mathcal{F}$ and the solid solver by $\mathcal{S}$. The resulting fully coupled non-linear objective function is then expressed for timestep $n+1$ as:

$$0 = \boldsymbol{k}_m^{n+1} - \mathcal{S}\left(\mathcal{F}\left(\boldsymbol{k}_m^{n+1}\right)\right). \tag{7.1}$$

By iterating, a value of $\boldsymbol{k}_m^{n+1}$ that satisfies the above coupled system can be found. It is well known that coupled systems, such as the afore mentioned, suffer from numerical instability caused by the so called added mass effect. This results, broadly, from a component of the fluid acting as a force rather than an inertial mass [27, 107–110].

It is possible to stabilise a FPI system by relaxing the change in $\boldsymbol{k}_m$ over successive iterations. However, the relaxation factor is not known for the general case, and further is likely to vary as the solution converges [107, 111]. Thus the need for a dynamic under-relaxation method; such as that of the Aitken's Δ^2 method [4]. The method has been extended for vector problems by Irons and Tuck [112] and has been used in number of FSI works [25, 109, 111]. The method uses extrapolation of the residual of the system during each iteration to determine an update to the under-relaxation parameter.

The focus of this chapter is to couple together the incompressible two phase viscous flow model and the six degree of freedom rigid body dynamics model. This is completed in a strongly coupled partitioned manner via the Aitken's Δ^2 FPI method. Note however to reduce simulation time, surface tension effects are neglected in this chapter.

7.2 Fluid Interface Considerations

7.2.1 Response Kinematic

To formulate the response kinetics generated by the fluid, the force and torque on the boundary are computed respectively as

$$\begin{aligned}\boldsymbol{f}_f &= \oint_{\partial\Omega} \left[p - \mu_i \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right)\right] \hat{\boldsymbol{n}}_{\partial\Omega} dA, \\ \boldsymbol{\tau}_{o',f} &= \oint_{\partial\Omega} \boldsymbol{r}_{o',i} \times \left[p - \mu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right)\right] \hat{\boldsymbol{n}}_{\partial\Omega} dA,\end{aligned}$$

where $\boldsymbol{r}_{o',i}$ is the position vector from the non-inertial frame origin to the boundary node i and is fixed to the fluid domain. Discretely the above results in

$$\begin{aligned}\boldsymbol{f}_f &= \sum_i \boldsymbol{f}_{f,i} = \sum_{f \in \partial\Omega \cap \partial\Omega_i} p_i A\hat{\boldsymbol{n}}|_f - \mu_i \left(\nabla \boldsymbol{u}_i + \nabla \boldsymbol{u}_i^T\right) A\hat{\boldsymbol{n}}|_f, \\ \boldsymbol{\tau}_{o'} &= \sum_i \boldsymbol{r}_{o',i} \times \boldsymbol{f}_{f,i}.\end{aligned}$$

The final expression for the 'fluid function' is given by

$$\mathcal{F}(\boldsymbol{k}_m^{n+1}) = \mathcal{F} \begin{pmatrix} \ddot{\boldsymbol{r}}_{o'/o}^{n+1} \\ \dot{\boldsymbol{\omega}}_{o'/o}^{n+1} \\ \boldsymbol{\omega}_{o'/o}^{n+1} \end{pmatrix} = \begin{bmatrix} \boldsymbol{f_f} \\ \boldsymbol{\tau}_{o',f} \end{bmatrix}^{n+1} = \boldsymbol{k}_{\boldsymbol{t}}^{n+1},$$

where $\mathcal{F}$ represents a full solve of the fluid governing equations to the subsequent time step in response to $\boldsymbol{k}_m^{n+1}$.

7.2.2 The No-Slip Paradox

For the work considered thus far the liquid-gas interface has remained off the boundary, and the transport of the contact line has not been considered. With more violent slosh expected in the test cases for this chapter the two phase interface will interact with the solid boundary. For a vertex centred mesh the no-slip boundary condition, $\boldsymbol{u} = 0$, results in the absence of a volume fraction transport velocity. To resolve this a Navier-Slip condition is applied to the boundary. The resulting velocity gradient at the boundary face, $\nabla \boldsymbol{u}_{ns}|_{\partial\Omega}$, is approximated discretely as

$$\nabla \boldsymbol{u}_{ns}|_{\partial\Omega} \approx \frac{\boldsymbol{u}_i - \boldsymbol{u}_w}{r_s} \otimes \hat{\boldsymbol{n}}_{\partial\Omega} \tag{7.2}$$

where r_s is the slip length, $\boldsymbol{u}_w$ symbolises the boundary wall velocity which is set to 0, and $\hat{\boldsymbol{n}}_{\partial\Omega}$ is the boundary normal. The nodal velocity at the boundary wall is denoted $\boldsymbol{u}_i$. In this work the slip length is set to the average cell size in the boundary normal direction as this approaches zero as the mesh size tends to zero. The viscous shear contribution, Equation (2.14), is updated to reflect the Navier-Slip condition

and is computed by

$$\nabla \cdot \mu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right)\big|_i \approx \frac{1}{V_i} \sum_{f \in \partial\Omega_i} \mu(\alpha_f) A_f \begin{cases} \left(\nabla \boldsymbol{u}_{ns} + \nabla \boldsymbol{u}_{ns}^T\right) \hat{\boldsymbol{n}}_f & \text{if } \partial\Omega_i \cap \partial\Omega \\ \left(\nabla \boldsymbol{u}_f + \nabla \boldsymbol{u}_f^T\right) \hat{\boldsymbol{n}}_f & \text{else} \end{cases},$$

where $\nabla \boldsymbol{u}_{ns}$ is computed as per Equation (7.2) and the internal faces using the compact derivative of Equation (2.13) for calculating $\nabla \boldsymbol{u}_f$.

7.3 Aitken's Δ^2 Method

The fluid-solid coupling approach is now presented. The rigid-body Equation (6.11) is reformulated as

$$\mathcal{S}(\boldsymbol{k}_t^{n+1}) = \mathcal{S}\begin{pmatrix} \boldsymbol{f}^{n+1} \\ \boldsymbol{\tau}_{o'}^{n+1} \end{pmatrix} = \begin{bmatrix} \ddot{\boldsymbol{r}}_{o'/o} \\ \dot{\boldsymbol{\omega}}_{o'/o} \\ \boldsymbol{\omega}_{o'/o} \end{bmatrix}^{n+1} = \boldsymbol{k}_m^{n+1},$$

where $\mathcal{S}$ both updates and extracts the appropriate state variables from the rigid body. The full FPI scheme can be expressed as

$$0 = \boldsymbol{k}_m^{n+1} - \mathcal{S}(\mathcal{F}(\boldsymbol{k}_m^{n+1}) + \boldsymbol{k}_{t,ext}), \tag{7.3}$$

where $\boldsymbol{k}_{t,ext}$ is some external kinetics to the system, for example the main rocket engine or manoeuvring thrusters. For brevity this will be omitted from further expressions. The residual for the FPI scheme is defined by

$$\boldsymbol{\mathcal{R}}^{\iota+1} = \boldsymbol{k}_m^{\iota+1} - \boldsymbol{k}_m^{\iota},$$

where $\boldsymbol{\mathcal{R}}^{\iota+1}$ is converged to a specific tolerance, and ι represents the iteration index. Aitken's Δ^2 method is employed to find the value of $\boldsymbol{k}_m^{\iota+1}$ by

$$\boldsymbol{k}_m^{\iota+1} = \left[\boldsymbol{k}_m + \varpi \left(\mathcal{H}(\boldsymbol{k}_m) - \boldsymbol{k}_m\right)\right]^{\iota},$$

In the above $\mathcal{H}(\cdot) = \mathcal{S}(\mathcal{F}(\cdot))$ and the dynamic relaxation parameter, ϖ, is updated using

$$\varpi^{\iota} = \varpi^{\iota-1} \frac{\boldsymbol{\mathcal{R}}^{\iota-1} \cdot \left(\boldsymbol{\mathcal{R}}^{\iota} - \boldsymbol{\mathcal{R}}^{\iota-1}\right)}{\|\boldsymbol{\mathcal{R}}^{\iota} - \boldsymbol{\mathcal{R}}^{\iota-1}\|_2^2}.$$

Once $\boldsymbol{\mathcal{R}}$ has been converged, the whole FSI system is advanced to the next time level. Perhaps it is worth noting that a traditional Aitken's Δ^2 updates ϖ every second iteration. Here the parameter is updated every iteration, increasing computational efficiency as demonstrated in [109, 111]. The starting value of ϖ is carried over from the previous time step.

7.4 Results

7.4.1 Hidden Mass

The first test case presented is that of a rigid system, and aims to demonstrate stable coupling with no loss in accuracy. The system contains two sold spheres with a mass

of 4kg and 2kg, labelled m_1 and m_2 respectively in Figure 7.1. The first sphere has a position vector $\boldsymbol{r}_{o',m_1} = [-2, 0, 0]$m and a radius of $r_1 = 1$m. The second sphere, with radius 0.5m, is located at $\boldsymbol{r}_{o',m_2} = [1, 0, 0]$m. The mass moment of inertia for each sphere is a 3×3 diagonal matrix, where the diagonal values are computed $I_{xx'} = I_{yy'} = I_{zz'} = 2/5mr^2$.

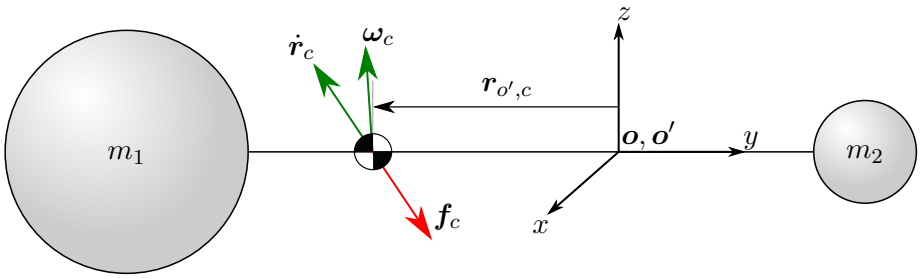


Figure 7.1: A translating and rotating system of two spherical masses connected by a massless rod, under a constant linear force.

The system is given an initial linear velocity of ${}^{o}\dot{\boldsymbol{r}}_{o',c} = [-2, 0.5, 1.0]$m.s^{-1} and an angular velocity of ${}^{o}\dot{\boldsymbol{\omega}}_{o',c} = [0.5\pi, -0.25\pi, 0.25\pi]$rad.s^{-1}. A constant force is applied, such that after one second the centre of gravity of the system should come to rest thus ${}^{o}\boldsymbol{f}_c = [12, -3, -6]$N.

Two approaches to modelling the system are conducted. The first approach is monolithic and used for validation purposes: i.e the full system is solved using the rigid body model laid out in the previous chapter. The second approach is partitioned, where each mass is modelled separately and coupled using the Aitken's Δ^2 method. The second mass, m_2, acts as the rigid body and is solved using the Newton-Euler equations, Equation (6.9). The first mass, m_1, provides a 'response' force and torque to the rigid body's acceleration. The Aitken's Δ^2 method is expected to provide stability to an otherwise unstable system (since $m_1 > m_2$).

The system was simulated using a time step size of $\Delta t = 0.01$s and for a total time of $t_m = 1.0$s. The L_2 norm of the difference in displacement and velocity at the conclusion of the simulation was recorded as $5.78e^{-14}$m and $7.51e^{-14}$m.s^{-1} respectively. The quaternion and angular velocity difference was reported at $2.08e^{-14}$ and $7.85e^{-15}$rad.s^{-1} respectively. Typically, 5 Aitken's Δ^2 iterations were required to converge the a time step. This demonstrates that the coupling algorithm is functioning correctly.

7.4.2 Flat Spin

The final test case presented is that of a fluid container undergoing a 'flat spin' similar to a case conducted by Gerrits and Veldman [23,30]. A partially filled container is put into an unstable equilibrium condition, see Figure 7.2. It is given an initial angular velocity about the axis with the smallest moment of inertia. The stable equilibrium condition dictates that the container must spin around the axis which corresponds to the lowest kinetic energy of the system, i.e, the axis with the highest moment of inertia.

The container, of dimension 1.2m $\times$ 0.8m $\times$ 0.2m (meshed with $\Delta x = 0.02$), is given a principle moment of inertia of $\boldsymbol{I} = \text{diag}[I_{xx}, I_{yy}, I_{zz}] = \text{diag}[1, 2, 4]$kg.m^2 and a mass of 12kg. The initial angular velocity is set to ${}^{o}\boldsymbol{\omega}_{o'/o} = [5, 0, 0]$rad.s^{-1}. The tank is filled such that the liquid occupies 65% of the volume, and using air and water at NTP the fluid mass is approximately 125kg. Finally, the field is initialised such that the gas forms a cylinder along the x axis.

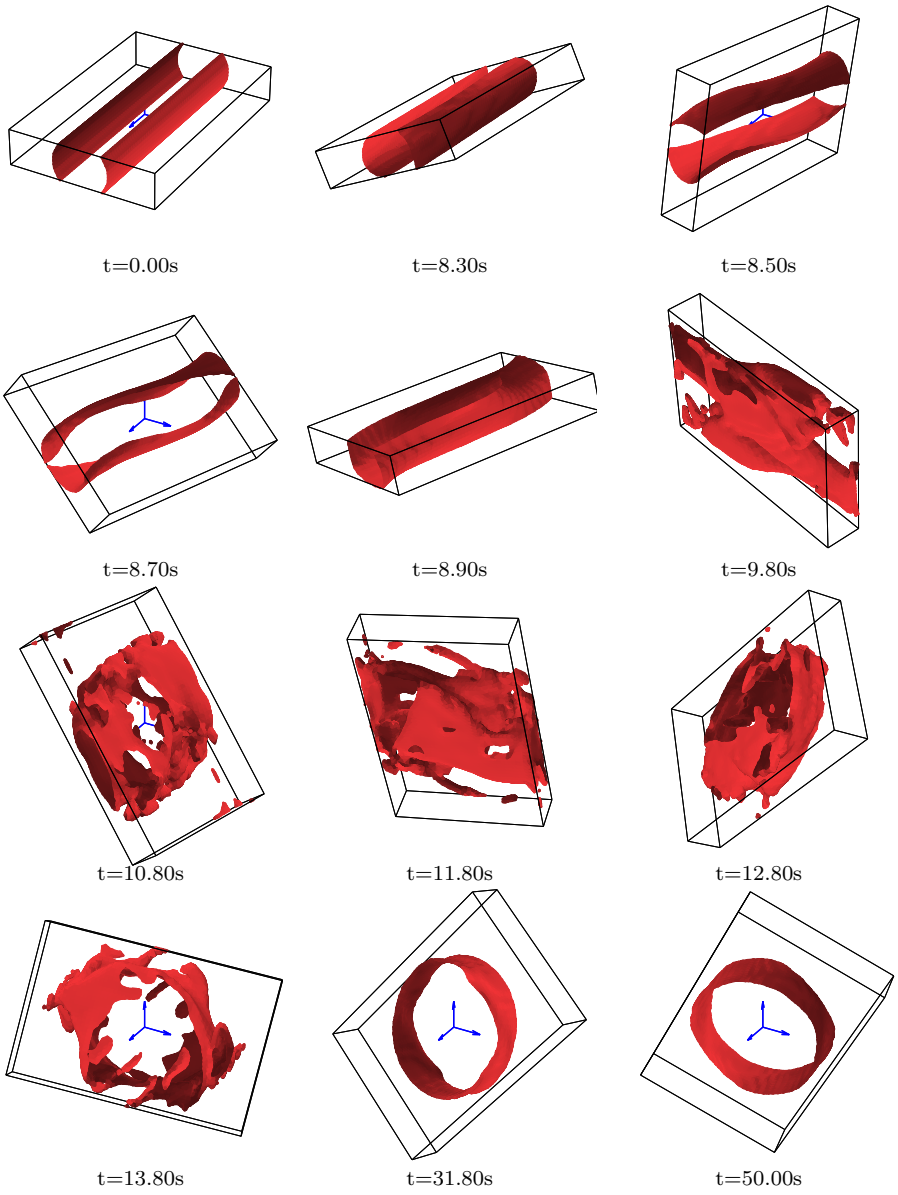


Figure 7.2: Evolution of the liquid-gas interface in a partially filled tank under going a flat spin (images not spaced evenly in time).

The evolution transition of the tank, from spinning about the local x- to z-axis, is depicted in Figure 7.2 with the liquid-gas interface illustrated. The initial phases of the instability are seen to occur during the 8^{th} second, where the interface begins to move but the tank's rotation remains largely around the local x-axis. The transition to a dominant rotation around the z-axis occurs relatively quickly as seen between the frames at 8.9s and 9.8s. The interface then starts to coalesce as a cylinder around the local z-axis and by 13.8s the interface has settled considerably. From this point on viscous effects continues to stabilise the interface and damp rotations about the local x- and y-axis.

Figure 7.3 depicts the evolution of the angular velocity, expressed in both the inertial and non-inertial basis. The flow 'trips' at *circa* 7s into the simulation and the container begins a transition to a rotation about the local z-axis. The 'trip'

occurs noticeable sooner when compared with [23, 113]. This is due to differences between the two cases, which include the absence of a gas in [23, 113] as well as surface tension. The stabilising manoeuvre and final equilibirum state are however comparable, demonstrating stable coupling. A convergence tolerance of $1.0e^{-6}$ was used, and typically 5 to 20 Aitken's iterations were need to converge a time step. Finally, the method reported machine precision accuracy with respect to mass (volume) conservation.

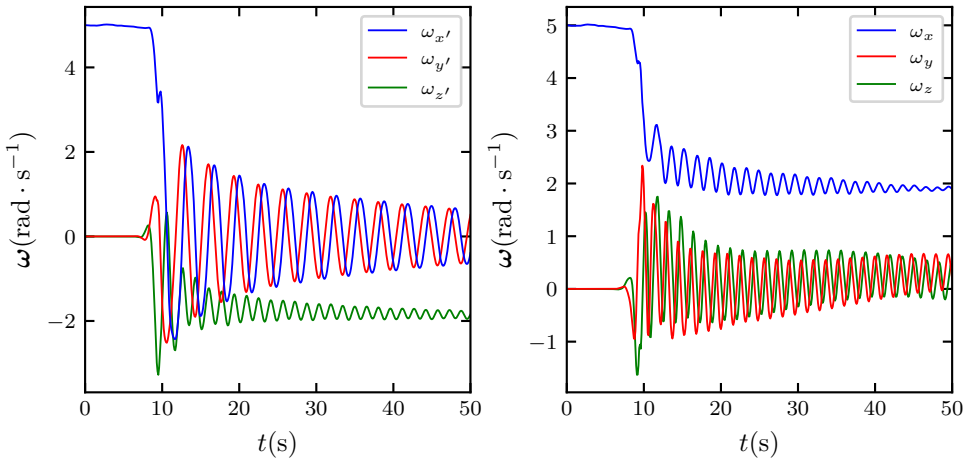


Figure 7.3: Evolution of the coupled system's angular velocity, expressed in the non-inertial (left) and inertial (right) basis.

7.5 Conclusion

The work presented here demonstrates a new partitioned strongly coupled analysis tool, capable of modelling the dynamics seen during spacecraft operation. To couple the system in a stable manner the Aitken's Δ^2 method was employed. The coupled algorithm was tested on two cases. The first demonstrated the working order of the coupled system by using a rigid body system with hidden masses. The second test case was that of a tank, with $m_l/m_s \approx 10$, initially spinning around the axis with the smallest moment of inertia. The coupling algorithm was shown to provide stability to the coupled problem and achieved the correct result; the tank rotating around the axis with the highest moment of inertia. Convergence of a time step was found to take up to 20 iterations thereby increasing the cost of simulation.

Chapter 8

Conclusion

8.1 Summary

This thesis details the expansion of the *Elemental* software, multi-physics CFD code, to enable the simulation of spacecraft. The added tools were created with specific focus to produce a numerical method suitable for high fidelity analysis of the two-phase flow in the propellent tanks of space vehicles under micro-gravity conditions. The following research contributions were made:

1. The first novel contribution involved a conservative initialisation scheme, named AGI, for the volume fraction field for arbitrary interface topologies on unstructured meshes. The developed method, through both polytope cutting and numerical quadrature, demonstrated that the field can be conservatively initialised to machine precision on unstructured grids in both 2 and 3D.

2. The transport of the volume fraction field was enhanced. Here the algebraic VoF methods of CICSAM and HiRAC were improved. A conservative HiRAC formulation was developed for the first time. Both CICSAM and HiRAC were reformulated to a 2nd order predictor-corrector method for temporal integration. Finally, CICSAM's method which corrects unbounding in the volume fraction field was reformulated for the new temporal approach, and for HiRAC a bounding formulation was introduced. Numerical studies showed that HiRAC does indeed outperform CICSAM, especially when considering numerical diffusivity in shear flow scenarios. An accuracy comparison was also conducted for the first time between the above algebraic methods and geometric VoF methods. On structured meshes the geometric methods were demonstrated to be superior. However, on unstructured meshes HiRAC compared well to recently developed geometric methods.

3. A CSF approach was formulated and implemented into the incompressible flow governing equation's of *Elemental* to add a surface tension modelling capability. The inclusion of the surface tension source term was completed in a manner which discretely yields a well balanced formation, while accounting for the density variations between the interfacing fluids. Due to the lack of a reliable 2nd order unstructured method for 2 and 3D curvature calculations the height function method was employed. Tests showed that the implemented scheme is indeed well-balanced. Numerical benchmark studies were again performed to compare the developed formulation against other methods. HiRAC was shown to suffer from significant added numerical noise which deteriorated surface tension accuracy on coarse meshes. At finer mesh resolutions the surface tension model compared well to existing models in literature. For HiRAC this represents the first surface tension work to demonstrated the well-balanced property.

4. Finally a 4th order in time rigid body model was constructed, and using quaternions was able to track the attitude of the local frame with respect to the global frame. The rigid body was strongly coupled to the fluid using the Aitken's Δ^2 method. The coupling was further completed in a partitioned and computationally robust manner, producing a versatile numerical simulation tool. The coupled system was finally demonstrated to be robust using a flat spin test case in 3D.

8.2 Further Research

In light of the motivations for this work, laid out at the opening of this thesis, the continued work of building a more accurate, efficient, and robust FSI models is recommended. A number of key areas are presented as possible avenues to improve aspects of spacecraft modelling:

1. Further investigation in regard to computational cost between algebraic and geometric VoF methods would be instructive. It is recommended to add the recent algebraic M-CICSAM [68] method to such a study. By implementing (or obtaining) the relevant methods and re-running the benchmarks tests of Chapter 4, but in a timed environment, a clearer picture of computational efficiency (cost vs accuracy) should emerge.

2. Emphasis was placed on unstructured methods, due to industrial applicability. However, the current lack of a unstructured robust 2D/3D interface curvature calculation method restricted cases to structured meshes. Work aimed at achieving a second order curvature capturing method on unstructured grids would seem prudent following this project, for which AGI could play a key role.

3. A natural extension from this project is to include surface tension effects at the contact line. This will enable the modelling of low We number flows. Methods to extend the height function algorithm for contact lines has been conducted by Afkhami et al. [114–116]. As a longer term goal a phase change model, as proposed by Malan [117] for example, could be incorporated into the fluid model to round out the spacecraft simulation tool.

4. For any iterative FSI problem there is clearly a sigficiant increase in computational cost, as time steps are repeated to convergence. Recent work by Mehl et al. [110], to create new coupling methods, should be incorporated to improve computation cost of coupled problems.

Bibliography

[1] B. W. Jones, A. G. Malan, N. A. Ilangakoon, The initialisation of volume fractions for unstructured grids using implicit surface definitions., Computers and Fluids 179 (2019) 194–205.
URL https://doi.org/10.1016/j.compfluid.2018.10.021

[2] O. Ubbink, R. Issa, A Method for Capturing Sharp Fluid Interfaces on Arbitrary Meshes, Journal of Computational Physics 153 (1) (1999) 26–50.
URL www.doi.org/10.1006/jcph.1999.6276

[3] J. A. Heyns, A. G. Malan, T. M. Harms, O. F. Oxtoby, Development of a compressive surface capturing formulation for modelling free-surface flow by using the volume-of-fluid approach, International Journal for Numerical Methods in Fluids 71 (6) (2013) 788–804.
URL https://doi.org/10.1002/fld.3694

[4] A. C. Aitken, XXV.—On Bernoulli's Numerical Solution of Algebraic Equations, Proceedings of the Royal Society of Edinburgh 46 (1927) 289–305.
URL https://doi.org/10.1017/S0370164600022070

[5] R. P. Fedkiw, T. Aslam, B. Merriman, S. Osher, A Non-oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (the Ghost Fluid Method), Journal of Computational Physics 152 (2) (1999) 457–492.
URL https://doi.org/10.1006/jcph.1999.6236

[6] M. Sussman, P. Smereka, S. Osher, A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow, Journal of Computational Physics 114 (1) (1994) 146–159.
URL https://doi.org/10.1006/jcph.1994.1155

[7] S. O. Unverdi, G. Tryggvason, A front-tracking method for viscous, incompressible, multi-fluid flows, Journal of Computational Physics 100 (1) (1992) 25–37.
URL https://doi.org/10.1016/0021-9991(92)90307-K

[8] C. W. Hirt, B. D. Nichols, Volume of Fluid (VOF) methods for the dynamics of free boundaries, Journal of Computational Physics 39 (1) (1981) 201–225.
URL https://doi.org/10.1016/0021-9991(81)90145-5

[9] S. Bnà, S. Manservisi, R. Scardovelli, P. Yecko, S. Zaleski, Numerical integration of implicit functions for the initialization of the VOF function, Computers and Fluids 113 (2015) 42–52.
URL https://doi.org/10.1016/j.compfluid.2014.04.010

[10] S. Bnà, S. Manservisi, R. Scardovelli, P. Yecko, S. Zaleski, Vofi - A library to initialize the volume fraction scalar field, Computer Physics Communications 200 (2016) 291–299.
URL http://dx.doi.org/10.1016/j.cpc.2015.10.026

[11] S. Strobl, A. Formella, T. Pöschel, Exact calculation of the overlap volume of spheres and mesh elements, Journal of Computational Physics 311 (2016) 158–172.
URL http://dx.doi.org/10.1016/j.jcp.2016.02.003

[12] B. Lafaurie, C. Nardone, R. Scardovelli, S. Zaleski, G. Zanetti, Modelling Merging and Fragmentation in Multiphase Flows with SURFER, Journal of Computational Physics 113 (1) (1994) 134–147.
URL https://doi.org/10.1006/jcph.1994.1123

[13] R. Scardovelli, S. Zaleski, Analytical Relations Connecting Linear Interfaces and Volume Fractions in Rectangular Grids, Journal of Computational Physics 164 (1) (2000) 228–237.
URL https://doi.org/10.1006/jcph.2000.6567

[14] S. Popinet, Gerris: A tree-based adaptive solver for the incompressible Euler equations in complex geometries, Journal of Computational Physics 190 (2) (2003) 572–600.
URL https://doi.org/10.1016/S0021-9991(03)00298-5

[15] J. B. Dupont, D. Legendre, Numerical simulation of static and sliding drop with contact angle hysteresis, Journal of Computational Physics 229 (7) (2010) 2453–2478.
URL http://www.doi.org/10.1016/j.jcp.2009.07.034

[16] G. Tryggvason, R. Scardovelli, S. Zaleski, Direct Numerical Simulations of Gas–Liquid Multiphase Flows, Cambridge University Press, 2011.
URL https://doi.org/10.1017/CBO9780511975264

[17] C. B. Ivey, P. Moin, Conservative and bounded volume-of-fluid advection on unstructured grids, Journal of Computational Physics 350 (2017) 387–419.
URL http://dx.doi.org/10.1016/j.jcp.2017.08.054

[18] T. Marić, H. Marschall, D. Bothe, An enhanced un-split face-vertex flux-based VoF method, Journal of Computational Physics 371 (2018) 967–993.
URL https://doi.org/10.1016/j.jcp.2018.03.048

[19] H. Scheufler, J. Roenby, Accurate and efficient surface reconstruction from volume fraction data on general meshes, Journal of Computational Physics 383 (2019) 1–23.
URL https://doi.org/10.1016/j.jcp.2019.01.009

[20] I. Newton, Philosophiæ Naturalis Principia Mathematica (Latin), Edmond Halley, Londini, 1687.

[21] L. Euler, Mechanica, Vol. 1, Euler Archive - All Works, 1736.
URL https://scholarlycommons.pacific.edu/euler-works/15

[22] L. Euler, Mechanica, Vol. 2, Euler Archive - All Works, 1736.
URL https://scholarlycommons.pacific.edu/euler-works/16

[23] J. Gerrits, A. E. P. Veldman, Dynamics of liquid-filled spacecraft, J. Eng. Math. 45 (1) (2003) 21–38.
URL https://doi.org/10.1023/A:1022055916067

[24] Y. Bazilevs, V. M. Calo, Y. Zhang, T. J. R. Hughes, Isogeometric Fluid–structure Interaction Analysis with Applications to Arterial Blood Flow, Computational Mechanics 38 (4-5) (2006) 310–322.
URL https://doi.org/10.1007/s00466-006-0084-3

[25] H. Jasak, I. Gatin, V. Vukčević, Monolithic coupling of the pressure and rigid body motion equations in computational marine hydrodynamics, Journal of Marine Science and Application 16 (4) (2017) 375–381.
URL https://doi.org/10.1007/s11804-017-1436-4

[26] O. F. Oxtoby, A. G. Malan, A matrix-free , implicit , incompressible fractional-step algorithm for fluid – structure interaction applications, J. Comput. Phys. 231 (16) (2012) 5389–5405.
URL http://dx.doi.org/10.1016/j.jcp.2012.04.037

[27] P. Causin, J. F. Gerbeau, F. Nobile, Added-mass effect in the design of partitioned algorithms for fluid-structure problems, Computer Methods in Applied Mechanics and Engineering 194 (42-44) (2005) 4506–4527.
URL https://doi.org/10.1016/j.cma.2004.12.005

[28] W. Press, S. Teukolsky, W. Vetterling, B. Flannery, Numerical Recipes: The Art of Scientific Computing, Technometrics 29 (4) (1987) 501.

[29] R. L. Burden, J. D. Faires, Numerical Analysis, 9th Edition, Cengage Learning, 2010.

[30] A. Veldman, J. Gerrits, R. Luppes, J. Helder, J. Vreeburg, The numerical simulation of liquid sloshing on board spacecraft, Journal of Computational Physics 224 (1) (2007) 82–99.
URL https://doi.org/10.1016/j.jcp.2006.12.020

[31] A. G. Malan, R. W. Lewis, P. Nithiarasu, An improved unsteady, unstructured, artificial compressibility, finite volume scheme for viscous incompressible flows: Part II. Application, International Journal for Numerical Methods in Engineering 54 (5) (2002) 715–729.
URL https://doi.org/10.1002/nme.443

[32] A. G. Malan, R. W. Lewis, P. Nithiarasu, An improved unsteady, unstructured, artificial compressibility, finite volume scheme for viscous incompressible flows: Part I. Theory and implementation, International Journal for Numerical Methods in Engineering 54 (5) (2002) 695–714.
URL https://doi.org/10.1002/nme.447

[33] J. A. Heyns, A. G. Malan, T. M. Harms, O. F. Oxtoby, A weakly compressible free-surface flow solver for liquid-gas systems using the volume-of-fluid approach, Journal of Computational Physics 240 (2013) 145–157.
URL http://dx.doi.org/10.1016/j.jcp.2013.01.022

[34] A. J. Chorin, A numerical method for solving incompressible viscous flow problems, Journal of Computational Physics 2 (1) (1967) 12–26.
URL https://doi.org/10.1016/0021-9991(67)90037-X

[35] A. J. Chorin, Numerical solution of the Navier-Stokes equations, Mathematics of Computation 22 (104) (1968) 745–745.
URL https://doi.org/10.1090/S0025-5718-1968-0242392-2

[36] A. J. Chorin, On the Convergence of Discrete Approximations to the Navier-Stokes Equations, Mathematics of Computation 23 (106) (1969) 341.
URL www.doi.org/10.2307/2004428

[37] O. Zienkiewicz, P. Nithiarasu, R. Codina, M. Vázquez, P. Ortiz, The characteristic-based-split procedure: an efficient and accurate algorithm for fluid problems, International Journal for Numerical Methods in Fluids 31 (1) (1999) 359–392.
URL http://doi.org/10.1002/(SICI)1097-0363(19990915)31:1{%}3C359::AID-FLD984{%}3E3.0.CO;2-7

[38] P. Nithiarasu, An efficient artificial compressibility (AC) scheme based on the characteristic based split (CBS) method for incompressible flows, International Journal for Numerical Methods in Engineering 56 (13) (2003) 1815–1845.
URL http://doi.org/10.1002/nme.712

[39] A. G. Malan, R. W. Lewis, An artificial compressibility CBS method for modelling heat transfer and fluid flow in heterogeneous porous materials, International Journal for Numerical Methods in Engineering 87 (1-5) (2011) 412–423.
URL http://doi.org/10.1002/nme.3125

[40] J. A. Heyns, Formulation of a weakly compressible two-fluid flow solver and the development of a compressive surface capturing scheme using the volume-of-fluid approach (December).
URL http://hdl.handle.net/10019.1/71934

[41] J. Brackbill, D. Kothe, C. Zemach, A continuum method for modeling surface tension, Journal of Computational Physics 100 (2) (1992) 335–354.
URL https://doi.org/10.1016/0021-9991(92)90240-Y

[42] Ansys, NEUTRAL FILE FORMAT-Gambit, Time (2006) 1–43.
URL https://web.stanford.edu/class/me469b/handouts/gambit{_}write.pdf

[43] P. I. Crumpton, P. Moinier, M. B. Giles, An Unstructured Algorithm for high Reynolds Number Flows on Highly-Stretched Grids, Numerical Methods in Laminar and Turbulent Flow (1997) 561–572.

[44] B. van Leer, Upwind-difference method for aerodynamic problems governed by the Euler equations, Large-Scale Computations in Fluid Mechanics (January 1985) (1985) 327–336.

[45] G. D. van Albada, B. van Leer, W. W. Roberts, A Comparative Study of Computational Methods in Cosmic Gas Dynamics, in: Upwind and High-Resolution Schemes, Vol. 108, Springer Berlin Heidelberg, Berlin, Heidelberg, 1997, pp. 95–103.
URL http://doi.org/10.1007/978-3-642-60543-7{_}6

[46] G. Karypis, V. Kumar, A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs, SIAM Journal on Scientific Computing 20 (1) (1998) 359–392.
URL https://doi.org/10.1137/S1064827595287997

[47] M. Sussman, A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles, Journal of Computational

Physics 187 (1) (2003) 110–136.
URL https://doi.org/10.1016/S0021-9991(03)00087-1

[48] S. J. Cummins, M. M. Francois, D. B. Kothe, Estimating curvature from volume fractions, Comput. Struct. 83 (6-7) (2005) 425–434.
URL https://doi.org/10.1016/j.compstruc.2004.08.017

[49] S. Popinet, An accurate adaptive solver for surface-tension-driven interfacial flows, Journal of Computational Physics 228 (16) (2009) 5838–5866.
URL http://dx.doi.org/10.1016/j.jcp.2009.04.042

[50] S. T. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids, Journal of Computational Physics 31 (3) (1979) 335–362.
URL https://doi.org/10.1016/0021-9991(79)90051-2

[51] R. P. Brent, Algorithms for minimization without derivatives, Vol. 19, 1974.

[52] E. Polak, G. Ribiére, Note sur la convergence de directions conjug{é}es, Rev. Francaise Informat Recherche Opertionelle 3e Ann{è}e (1969) 35–43.

[53] R. Scardovelli, S. Zaleski, DIRECT NUMERICAL SIMULATION OF FREE-SURFACE AND INTERFACIAL FLOW, Annual Review of Fluid Mechanics 31 (1) (1999) 567–603.
URL https://doi.org/10.1146/annurev.fluid.31.1.567

[54] B. j. Parker, D. L. Youngs, Two and three dimensional eulerian simulation of fluid flowwith material interfaces, Tech. rep., Aldermaston (1992).

[55] J. E. Pilliod, E. G. Puckett, Second-order accurate volume-of-fluid algorithms for tracking material interfaces, Journal of Computational Physics 199 (2) (2004) 465–502.
URL https://doi.org/10.1016/j.jcp.2003.12.023

[56] P. Liovic, M. Rudman, J.-L. Liow, D. Lakehal, D. Kothe, A 3D unsplit-advection volume tracking algorithm with planarity-preserving interface reconstruction, Computers & Fluids 35 (10) (2006) 1011–1032.
URL https://doi.org/10.1016/j.compfluid.2005.09.003

[57] L. Jofre, O. Lehmkuhl, J. Castro, A. Oliva, A 3-D Volume-of-Fluid advection method based on cell-vertex velocities for unstructured meshes, Computers & Fluids 94 (2014) 14–29.
URL https://doi.org/10.1016/j.compfluid.2014.02.001

[58] L. Jofre, R. Borrell, O. Lehmkuhl, A. Oliva, Parallel load balancing strategy for Volume-of-Fluid methods on 3-D unstructured meshes, Journal of Computational Physics 282 (2015) 269–288.
URL http://dx.doi.org/10.1016/j.jcp.2014.11.009

[59] J. Roenby, H. Bredmose, H. Jasak, A computational method for sharp interface advection, Royal Society Open Science 3 (11) (2016) 160405.
URL https://doi.org/10.1098/rsos.160405

[60] F. Xiao, Y. Honma, T. Kono, A simple algebraic interface capturing scheme using hyperbolic tangent function, International Journal for Numerical Methods in Fluids 48 (9) (2005) 1023–1040.
URL http://doi.org/10.1002/fld.975

[61] S. Muzaferija, M. Peric, P. Sames, T. Schellin, A Two-Fluid Navier-Stokes Solver to Simulate Water Entry, in: Naval hydrodynamics, National Academy Press, Washington, DC, 1999, pp. 638–651.

[62] M. Darwish, F. Moukalled, Convective Schemes for Capturing Interfaces of Free-Surface Flows on Unstructured Grids, Numerical Heat Transfer, Part B: Fundamentals 49 (1) (2006) 19–42.
URL http://doi.org/10.1080/10407790500272137

[63] Y.-y. Tsui, S.-w. Lin, T.-t. Cheng, T.-c. Wu, Flux-blending schemes for interface capture in two-fluid flows, International Journal of Heat and Mass Transfer 52 (23-24) (2009) 5547–5556.
URL http://doi.org/10.1016/j.ijheatmasstransfer.2009.06.026

[64] B. Leonard, H. Niknafs, Sharp monotonic resolution of discontinuities without clipping of narrow extrema, Computers & Fluids 19 (1) (1991) 141–154.
URL http://doi.org/10.1016/0045-7930(91)90011-6

[65] B. Leonard, The ULTIMATE conservative difference scheme applied to unsteady one-dimensional advection, Computer Methods in Applied Mechanics and Engineering 88 (1) (1991) 17–74.
URL http://doi.org/10.1016/0045-7825(91)90232-U

[66] M. Hoekstra, G. Vaz, B. Abeil, T. Bunnik, Free-surface flow modelling with interface capturing techniques, MARINE2007 2 (2007) 1–4.

[67] T. Wacławczyk, T. Koronowicz, Comparison of CICSAM and HRIC high-resolution schemes for interface capturing, Journal of Theoretical and Applied Mechanics 46 (2) (2008) 325–345.
URL http://www.ptmts.org.pl/Waclaw-Koron-2-08.pdf

[68] D. Zhang, C. Jiang, D. Liang, Z. Chen, Y. Yang, Y. Shi, A refined volume-of-fluid algorithm for capturing sharp fluid interfaces on arbitrary meshes, Journal of Computational Physics 274 (2014) 709–736.
URL http://dx.doi.org/10.1016/j.jcp.2014.06.043

[69] B. S. Mirjalili, S. S. Jain, M. S. Dodd, Interface-capturing methods for two-phase flows : An overview and recent developments (1) (2017) 117–135.

[70] O. Ubbink, Numerical prediction of two fluid systems with sharp interfaces, Ph.D. thesis (1997).
URL http://powerlab.fsb.hr/ped/kturbo/OpenFOAM/docs/OnnoUbbinkPhD.pdf

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

[72] D. J. Harvie, D. F. Fletcher, A New Volume of Fluid Advection Algorithm: The Stream Scheme, Journal of Computational Physics 162 (1) (2000) 1–32.

[73] W. J. Rider, D. B. Kothe, Reconstructing Volume Tracking, Journal of Computational Physics 141 (2) (1998) 112–152.
URL https://doi.org/10.1006/jcph.1998.5906

[74] M. Owkes, O. Desjardins, A computational framework for conservative, three-dimensional, unsplit, geometric transport with application to the volume-of-fluid (VOF) method, Journal of Computational Physics 270 (2014) 587–612.
URL http://dx.doi.org/10.1016/j.jcp.2014.04.022

[75] P. Cifani, W. Michalek, G. Priems, J. Kuerten, C. van der Geld, B. Geurts, A comparison between the surface compression method and an interface reconstruction method for the VOF approach, Computers & Fluids 136 (2016) 421–435.
URL http://dx.doi.org/10.1016/j.compfluid.2016.06.026

[76] R. J. LeVeque, High-Resolution Conservative Algorithms for Advection in Incompressible Flow, SIAM Journal on Numerical Analysis 33 (2) (1996) 627–665.
URL https://doi.org/10.1137/0733033

[77] J.-M. Ghidaglia, Capillary forces: A volume formulation, European Journal of Mechanics - B/Fluids 59 (2016) 86–89.
URL http://dx.doi.org/10.1016/j.euromechflu.2016.05.006

[78] T. Abadie, J. Aubin, D. Legendre, On the combined effects of surface tension force calculation and interface advection on spurious currents within Volume of Fluid and Level Set frameworks, Journal of Computational Physics 297 (2015) 611–636.
URL http://dx.doi.org/10.1016/j.jcp.2015.04.054

[79] N. W. Wiliams, D. B. Kothe, E. G. Puckett, Accuracy and Convergence of Continuum Surface Tension Models., Fluid Dynamics at Interfaces (1998) 294–305.

[80] F. Evrard, F. Denner, B. van Wachem, Estimation of curvature from volume fractions using parabolic reconstruction on two-dimensional unstructured meshes, Journal of Computational Physics 351 (2017) 271–294.
URL https://doi.org/10.1016/j.jcp.2017.09.034

[81] S. Popinet, Numerical models of surface tension, Annual Review of Fluid Mechanics 50 (0) (2018) 49–75.
URL https://doi.org/10.1146/annurev-fluid-122316-045034

[82] C. B. Ivey, P. Moin, Accurate interface normal and curvature estimates on three-dimensional unstructured non-convex polyhedral meshes, Journal of Computational Physics 300 (2015) 365–386.
URL http://dx.doi.org/10.1016/j.jcp.2015.07.055

[83] S. Popinet, S. Zaleski, A front-tracking algorithm for accurate representation of surface tension, International Journal for Numerical Methods in Fluids 30 (6) (1999) 775–793.

[84] D. Torres, J. Brackbill, The Point-Set Method: Front-Tracking without Connectivity, Journal of Computational Physics 165 (2) (2000) 620–644.
URL https://doi.org/10.1006/jcph.2000.6635

[85] Y. Renardy, M. Renardy, PROST: A Parabolic Reconstruction of Surface Tension for the Volume-of-Fluid Method, J. Comput. Phys. 183 (2) (2002) 400–421.
URL http://dx.doi.org/10.1006/jcph.2002.7190

[86] R. Scardovelli, S. Zaleski, Interface reconstruction with least-square fit and split Eulerian-Lagrangian advection, International Journal for Numerical Methods in Fluids 41 (3) (2003) 251–274.
URL https://doi.org/10.1002/fld.431

[87] Z. Jibben, N. Carlson, M. Francois, A paraboloid fitting technique for calculating curvature from piecewise-linear interface reconstructions on 3D unstructured meshes, Computers & Mathematics with Applications 78 (2) (2019) 643–653.
URL https://doi.org/10.1016/j.camwa.2018.09.009

[88] J. López, J. Hernández, On reducing interface curvature computation errors in the height function technique, Journal of Computational Physics 229 (13) (2010) 4855–4868.
URL http://dx.doi.org/10.1016/j.jcp.2010.03.032

[89] G. Bornia, A. Cervone, S. Manservisi, R. Scardovelli, S. Zaleski, On the properties and limitations of the height function method in two-dimensional Cartesian geometry, Journal of Computational Physics 230 (4) (2011) 851–862.
URL http://dx.doi.org/10.1016/j.jcp.2010.11.029

[90] F. Denner, B. G. M. van Wachem, Fully-Coupled Balanced-Force VOF Framework for Arbitrary Meshes with Least-Squares Curvature Evaluation from Volume Fractions, Numerical Heat Transfer, Part B: Fundamentals 65 (3) (2014) 218–255.
URL http://doi.org/10.1080/10407790.2013.849996

[91] M. M. Francois, S. J. Cummins, E. D. Dendy, D. B. Kothe, J. M. Sicilian, M. W. Williams, A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework, Journal of Computational Physics 213 (1) (2006) 141–173.
URL https://doi.org/10.1016/j.jcp.2005.08.004

[92] M. Herrmann, A balanced force refined level set grid method for two-phase flows on unstructured flow solver grids, Journal of Computational Physics 227 (4) (2008) 2674–2706.
URL https://doi.org/10.1016/j.jcp.2007.11.002

[93] R. Löhner, C. Yang, E. Oñate, On the simulation of flows with violent free surface motion, Computer Methods in Applied Mechanics and Engineering 195 (41-43) (2006) 5597–5620.
URL https://doi.org/10.1016/j.cma.2005.11.010

[94] R. Panahi, E. Jahanbakhsh, M. S. Seif, Development of a VoF-fractional step solver for floating body motion simulation, Applied Ocean Research 28 (3) (2006) 171–181.
URL https://doi.org/10.1016/j.apor.2006.08.004

[95] A. Malan, J. Meyer, R. Lewis, Modelling non-linear heat conduction via a fast matrix-free implicit unstructured-hybrid algorithm, Computer Methods in Applied Mechanics and Engineering 196 (45-48) (2007) 4495–4504.
URL https://doi.org/10.1016/j.cma.2007.05.012

[96] J. López, C. Zanzi, P. Gómez, R. Zamora, F. Faura, J. Hernández, An improved height function technique for computing interface curvature from volume fractions, Computer Methods in Applied Mechanics and Engineering 198 (33-36) (2009) 2555–2564.
URL http://dx.doi.org/10.1016/j.cma.2009.03.007

[97] S. Hysing, S. Turek, D. Kuzmin, N. Parolini, E. Burman, S. Ganesan, L. Tobiska, Quantitative benchmark computations of two-dimensional bubble dynamics, International Journal for Numerical Methods in Fluids 60 (11) (2009) 1259–1288.
URL http://doi.org/10.1002/fld.1934

[98] S. J. Osher, Fronts Propagating with Curvature Dependent Speed, Computational Physics 79 (1) (1988) 1–5.
URL https://doi.org/10.1016/0021-9991(88)90002-2

[99] S. Turek, Efficient solvers for incompressible flow problems: An algorithmic and computational approach, Computers & Mathematics with Applications 38 (11-12) (1999) 292.
URL https://doi.org/10.1016/S0898-1221(99)91285-3

[100] N. Parolini, Computational Fluid Dynamics for Naval Engineering Problems, Ph.D. thesis, Polytechnique F´ed´erale de Lausanne (2004).
URL https://doi.org/10.5075/epfl-thesis-3138

[101] N. Parolini, E. Burman, A finite element level set method for viscous free-surface flows, in: Applied and Industrial Mathematics in Italy, WORLD SCIENTIFIC, 2005, pp. 416–427.
URL https://doi.org/10.1142/9789812701817{_}0038

[102] V. John, G. Matthies, MooNMD – a program package based on mapped finite element methods, Computing and Visualization in Science 6 (2-3) (2004) 163–170.
URL http://doi.org/10.1007/s00791-003-0120-1

[103] M. A. Safi, N. Prasianakis, S. Turek, Benchmark computations for 3D two-phase flows: A coupled lattice Boltzmann-level set study, Computers & Mathematics with Applications 73 (3) (2017) 520–536.
URL http://doi.org/10.1016/j.camwa.2016.12.014

[104] FeatFlow Software (2016).
URL http://www.featflow.de/en/

[105] W. R. Hamilton, LXXVIII. On quaternions; or on a new system of imaginaries in Algebra, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 25 (169) (1844) 489–495.
URL https://doi.org/10.1080/14786444408645047

[106] J. Diebel, Representing attitude: Euler angles, unit quaternions, and rotation vectors, Matrix 58 (2006) 1–35.

[107] J. Vierendeels, K. Dumont, E. Dick, P. Verdonck, Analysis and Stabilization of Fluid-Structure Interaction Algorithm for Rigid-Body Motion, AIAA Journal 43 (12) (2005) 2549–2557.
URL https://doi.org/10.2514/1.3660

[108] C. Förster, W. A. Wall, E. Ramm, Artificial added mass instabilities in sequential staggered coupling of nonlinear structures and incompressible viscous flows, Computer Methods in Applied Mechanics and Engineering 196 (7) (2007) 1278–1293.
URL www.doi.org/10.1016/j.cma.2006.09.002

[109] J. Degroote, P. Bruggeman, R. Haelterman, J. Vierendeels, Stability of a coupling technique for partitioned solvers in FSI applications, Computers & Structures 86 (23-24) (2008) 2224–2234.
URL http://dx.doi.org/10.1016/j.compstruc.2008.05.005

[110] M. Mehl, B. Uekermann, H. Bijl, D. Blom, B. Gatzhammer, A. van Zuijlen, Parallel coupling numerics for partitioned fluid–structure interaction simulations, Computers & Mathematics with Applications 71 (4) (2016) 869–891.
URL http://doi.org/10.1016/j.camwa.2015.12.025

[111] U. Küttler, W. A. Wall, Fixed-point fluid–structure interaction solvers with dynamic relaxation, Computational Mechanics 43 (1) (2008) 61–72.
URL http://doi.org/10.1007/s00466-008-0255-5

[112] B. M. Irons, R. C. Tuck, A version of the Aitken accelerator for computer iteration, International Journal for Numerical Methods in Engineering 1 (3) (1969) 275–277.
URL http://doi.wiley.com/10.1002/nme.1620010306

[113] J. Gerrits, Dynamics of liquid-filled spacecraft: numerical simulation of coupled solid-liquid dynamics, Phd, University of Groningen (2001).

[114] S. Afkhami, M. Bussmann, Height functions for applying contact angles to 2D VOF simulations, International Journal for Numerical Methods in Fluids 57 (4) (2008) 453–472.
URL https://doi.org/10.1002/fld.1651http://doi.wiley.com/10.1002/fld.1651

[115] S. Afkhami, S. Zaleski, M. Bussmann, A mesh-dependent model for applying dynamic contact angles to VOF simulations, Journal of Computational Physics 228 (15) (2009) 5370–5389.
URL http://dx.doi.org/10.1016/j.jcp.2009.04.027

[116] S. Afkhami, M. Bussmann, Height functions for applying contact angles to 3D VOF simulations, International Journal for Numerical Methods in Fluids 61 (8) (2009) 827–847.
URL http://doi.wiley.com/10.1002/fld.1974

[117] L. Malan, Direct numerical simulation of free-surface and interfacial flow using the VOF method: cavitating bubble clouds and phase change, Ph.D. thesis, Pierre and Marie Curie University, Univercity of Cape Town (2017).

[118] G. Slade, Classical Symmetric Top in a Gravitational (2012).
URL https://www.researchgate.net/publication/265624866

Appendix A

Additional Derivations

A.1 2D Shear Flow Face Flux

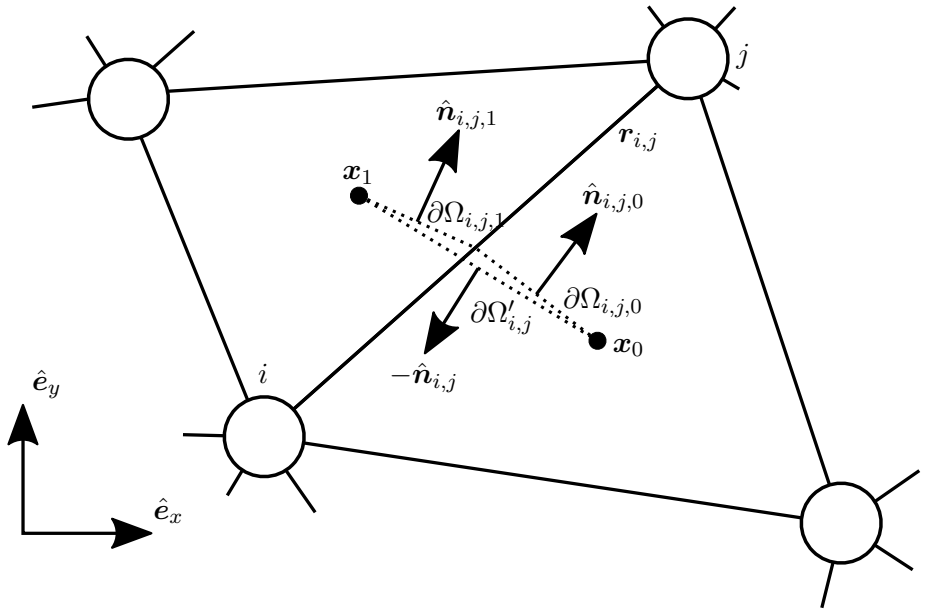


Figure A.1: Diagram of a single median duel cell face in 2D, with the sub-facets drawn.

Consider Figure A.1 with the face, $\partial\Omega_{i,j}$, the set consisting of the two sub-facets $\partial\Omega_{i,j,0}$ and $\partial\Omega_{i,j,1}$, resulting from the median duel cell construction. It is desired to compute the analytical face flux across $\partial\Omega_{i,j}$. A temporary control volume, Ω', can be constructed by connecting the two elements centres of the adjacent edge, $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$, forming the face $\partial\Omega'_{i,j}$. Given the divergence free velocity field expressed by

$$\boldsymbol{u}(x,y,t) = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} \sin(2\pi y)\sin^2(\pi x)\cos\left(\frac{\pi t}{t_m}\right) \\ -\sin(2\pi x)\sin^2(\pi y)\cos\left(\frac{\pi t}{t_m}\right) \end{bmatrix},$$

the face fluxes of the two sub-facets can be simplified using the divergence theorem,

$$\int_{\partial\Omega_{i,j}} \boldsymbol{u}\cdot\hat{\boldsymbol{n}}dA = \int_{\partial\Omega_{i,j,0}} \boldsymbol{u}\cdot\hat{\boldsymbol{n}}dA + \int_{\partial\Omega_{i,j,1}} \boldsymbol{u}\cdot\hat{\boldsymbol{n}}dA, \tag{A.1}$$

$$\int_{\partial\Omega_{i,j}} \boldsymbol{u}\cdot\hat{\boldsymbol{n}}dA + \int_{\partial\Omega'_{i,j}} \boldsymbol{u}\cdot\hat{\boldsymbol{n}}dA = \int_{\partial\Omega_{i,j,0}} \boldsymbol{u}\cdot\hat{\boldsymbol{n}}dA + \int_{\partial\Omega_{i,j,1}} \boldsymbol{u}\cdot\hat{\boldsymbol{n}}dA + \int_{\partial\Omega'_{i,j}} \boldsymbol{u}\cdot\hat{\boldsymbol{n}}dA,$$

$$\begin{aligned}\int_{\partial\Omega_{i,j}} \boldsymbol{u}\cdot\hat{\boldsymbol{n}}dA &= \int_{\Omega'} \nabla\cdot\boldsymbol{u}dV - \int_{\partial\Omega'_{i,j}} \boldsymbol{u}\cdot\hat{\boldsymbol{n}}dA,\\ &= -\int_{\partial\Omega'_{i,j}} \boldsymbol{u}\cdot\hat{\boldsymbol{n}}dA.\end{aligned}$$

The above surface integral can be converted to a 2D line integral,

$$-\int_{\partial\Omega'_{i,j}} \boldsymbol{u}\cdot\hat{\boldsymbol{n}}dA = \int_0^1 \boldsymbol{u}(\boldsymbol{r}(\lambda))\cdot\hat{\boldsymbol{n}}_{\partial\Omega i,j}\left|\boldsymbol{r}'(\lambda)\right|d\lambda, \tag{A.2}$$

and using the parametrisation

$$\boldsymbol{r}(\lambda) = \boldsymbol{x}_0 + \lambda(\boldsymbol{x}_1 - \boldsymbol{x}_0)\Big|\lambda\in[0,1]\,, \tag{A.3}$$

$$\boldsymbol{r}'(\lambda) = \boldsymbol{x}_1 - \boldsymbol{x}_0 = \Delta\boldsymbol{x} = \boldsymbol{r}', \tag{A.4}$$

where $\boldsymbol{x} = \{x, y\}$ and

$$\begin{aligned}c_t &= \cos\left(\frac{\pi t}{t_m}\right),\\ \Delta x &= x_1 - x_0,\\ \Delta y &= y_1 - y_0.\end{aligned}$$

Equation (A.2) is expanded where the velocity is parametrised by Equations (A.3) and (A.4):

$$\begin{aligned}\int_0^1 \boldsymbol{u}(\boldsymbol{r}(\lambda))\cdot\hat{\boldsymbol{n}}_{\partial\Omega i,j}\left|\boldsymbol{r}'(\lambda)\right|d\lambda = &\\ \left|\boldsymbol{r}'\right| c_t \int_0^1 &\sin^2\left(\pi\left(x_0+\Delta x\lambda\right)\right)\sin\left(2\pi\left(y_0+\Delta y\lambda\right)\right)n_x|_{\partial\Omega_{i,j}}\\ &-\sin^2\left(\pi\left(y_0+\Delta y\lambda\right)\right)\sin\left(2\pi\left(x_0+\Delta x\lambda\right)\right)n_y|_{\partial\Omega_{i,j}}d\lambda,\end{aligned} \tag{A.5}$$

and $\hat{\boldsymbol{n}}_{\partial\Omega_{i,j}} = \{n_x|_{\partial\Omega_{i,j}}, n_y|_{\partial\Omega_{i,j}}\}$. Expanding the first term of the right integrand as

$$= \left|\boldsymbol{r}'\right| c_t n_x|_{\partial\Omega_{i,j}} \int_0^1 \sin^2\left(\pi x_0 + \pi\Delta x\lambda\right)\sin\left(2\pi y_0 + 2\pi\Delta y\lambda\right)d\lambda.$$

Using trigonometric identities the integrand can be split into terms containing single trigonometric functions,

$$
\begin{aligned}
&= \sin^2\left(\pi x_0 + \pi\Delta x\lambda\right)\sin\left(2\pi y_0 + 2\pi\Delta y\lambda\right),\\
&= \left(1 - \cos^2\left(\pi x_0 + \pi\Delta x\lambda\right)\right)\sin\left(2\pi y_0 + 2\pi\Delta y\lambda\right),\\
&= \frac{1}{2}\left(1 - \cos\left(2\pi x_0 + 2\pi\Delta x\lambda\right)\right)\sin\left(2\pi y_0 + 2\pi\Delta y\lambda\right),\\
&= \frac{1}{2}\left(\sin\left(2\pi y_0 + 2\pi\Delta y\lambda\right) - \cos\left(2\pi x_0 + 2\pi\Delta x\lambda\right)\sin\left(2\pi y_0 + 2\pi\Delta y\lambda\right)\right),\\
&= \frac{1}{4}\big(2\sin\left(2\pi y_0 + 2\pi\Delta y\lambda\right) - \sin\left(2\pi x_0 + 2\pi y_0 + (2\pi\Delta x + 2\pi\Delta y)\lambda\right)\\
&\quad + \sin\left(2\pi x_0 - 2\pi y_0 + (2\pi\Delta x - 2\pi\Delta y)\lambda\right)\big),\\
&= \frac{1}{4}\big(2\sin\left(2\pi\left(y_0 + \Delta y\lambda\right)\right) - \sin\left(2\pi\left(x_0 + y_0 + (\Delta x + \Delta y)\lambda\right)\right)\\
&\quad + \sin\left(2\pi\left(x_0 - y_0 + (\Delta x - \Delta y)\lambda\right)\right)\big),
\end{aligned}
$$

and substituting back into the first term of the integrand of Equation (A.5) results in the evaluation of the integral as

$$
\begin{aligned}
&= \frac{|\boldsymbol{r}'|\, c_t n_x|_{\partial\Omega_{i,j}}}{4}\int_0^1 2\sin\left(2\pi\left(y_0 + \Delta y\lambda\right)\right) - \sin\left(2\pi\left(x_0 + y_0 + (\Delta x + \Delta y)\lambda\right)\right)\\
&\qquad + \sin\left(2\pi\left(x_0 - y_0 + (\Delta x - \Delta y)\lambda\right)\right) d\lambda,\\
&= \frac{|\boldsymbol{r}'|\, c_t n_x|_{\partial\Omega_{i,j}}}{4}\left[- \frac{2\cos(2\pi\left(y_0 + \Delta y\lambda\right)}{2\pi\Delta y} + \frac{\cos(2\pi\left(x_0 + y_0 + (\Delta x + \Delta y)\lambda\right))}{2\pi\left(\Delta x + \Delta y\right)}\right.\\
&\qquad \left. - \frac{\cos(2\pi\left(x_0 - y_0 + (\Delta x - \Delta y)\lambda\right))}{2\pi\left(\Delta x - \Delta y\right)}\right]_0^1,\\
&= \frac{|\boldsymbol{r}'|\, c_t n_x|_{\partial\Omega_{i,j}}}{8\pi}\left[\frac{2\cos(2\pi\left(r_x(\lambda) + r_y(\lambda)\right))}{\Delta x + \Delta y} - \frac{\cos(2\pi r_y(\lambda))}{\Delta y}\right.\\
&\qquad \left. - \frac{\cos(2\pi\left(r_x(\lambda) - r_y(\lambda)\right))}{\Delta x - \Delta y}\right]_0^1,
\end{aligned}
$$

where r_x and r_y denote the x and y components of $\boldsymbol{r}$. Finally, the above can be substituted back into Equation (A.1) which is then computed as

$$
\begin{aligned}
&\int_{\partial\Omega_{i,j}} \boldsymbol{u}\cdot\hat{\boldsymbol{n}}\, dA =\\
&\quad \frac{|\boldsymbol{r}'|\, c_t}{8\pi}\left[n_x|_{\partial\Omega_{i,j}}\left(\frac{2\cos(2\pi\left(r_x(\lambda) + r_y(\lambda)\right))}{\Delta x + \Delta y} - \frac{\cos(2\pi r_y(\lambda))}{\Delta y}\right.\right.\\
&\qquad \left.\left. - \frac{\cos(2\pi\left(r_x(\lambda) - r_y(\lambda)\right))}{\Delta x - \Delta y}\right)\right.\\
&\qquad \left. - n_y|_{\partial\Omega_{i,j}}\left(\frac{2\cos(2\pi\left(r_x(\lambda) + r_y(\lambda)\right))}{\Delta y + \Delta x} - \frac{\cos(2\pi r_x(\lambda))}{\Delta x}\right.\right.\\
&\qquad \left.\left. - \frac{\cos(2\pi\left(r_y(\lambda) - r_x(\lambda)\right))}{\Delta y - \Delta x}\right)\right]_0^1,
\end{aligned}
$$

where the second part of the above expression results from conducting a similar evaluation procedure for the second term of the integrand in Equation (A.5).

A.2 3D Deformation Flow Face Flux

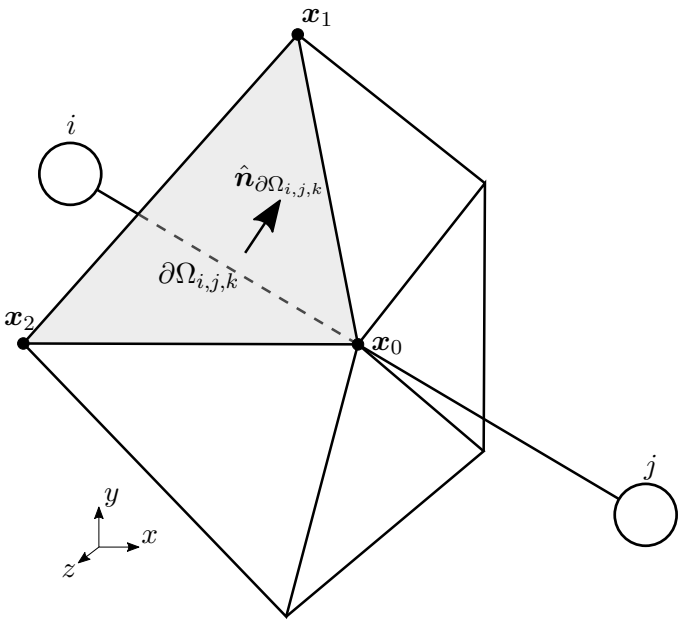


Figure A.2: Diagram of a single median duel cell face in 3D, with the sub-facets drawn and the k^{th} sub facet highlighted.

Consider the triangular surface $\partial\Omega_{i,j,k}$ depicted in Figure A.2. It is desired to compute the flux across the surface given the divergence free velocity field:

$$\boldsymbol{u}\left(x,y,z,t\right) = \begin{bmatrix} 2\sin^2\left(\pi x\right)\sin\left(2\pi y\right)\sin\left(2\pi z\right)\cos\left(\frac{\pi t}{t_m}\right) \\ -\sin\left(2\pi x\right)\sin^2(\pi y)\sin(2\pi z)\cos\left(\frac{\pi t}{t_m}\right) \\ -\sin\left(2\pi x\right)\sin(\pi y)\sin^2(\pi z)\cos\left(\frac{\pi t}{t_m}\right) \end{bmatrix},$$

where t and t_m are time and period respectively. Rather than expanding the integral. As in 2D, the face flux is computed directly here by Legendre-Gaussian quadrature. Further, since the surface patches are always triangular in nature the integration parametrisation developed for AGI in Chapter 3 can be utilised.

Essentially, the integrand of Equation (3.11) can be replaced with $\boldsymbol{u}(\boldsymbol{x}(\lambda),t)\cdot\boldsymbol{n}_{i,j,k}$. The surface integral, Equation (3.12) then can be used to compute the flux across the face. The modified single integral is formulated as

$$S'(\boldsymbol{u},\boldsymbol{X}_l,\boldsymbol{n},t) = \int_0^1 \boldsymbol{u}(\boldsymbol{a}(\lambda),t)\cdot\boldsymbol{n}|\boldsymbol{x}_1-\boldsymbol{x}_0|d\lambda,$$

where

$$\boldsymbol{a}(\lambda) = \left\{ \boldsymbol{x}_0 + \lambda(\boldsymbol{x}_1-\boldsymbol{x}_0) \middle| \lambda\in[0,1] \right\},$$

where $\boldsymbol{n}$ is a facet normal and $\boldsymbol{X}_l$ is a set containing two points,$\{\boldsymbol{x}_0,\boldsymbol{x}_1\}$, which bound the line integral. The region integral is modified to yield:

$$R'(\boldsymbol{u},\boldsymbol{X}_s,\boldsymbol{n},t) = \int_0^1 S'(\boldsymbol{u},\boldsymbol{X}_l,\boldsymbol{n},t)\left|(\boldsymbol{x}_1-\boldsymbol{x}_0)^{\perp\frac{(\boldsymbol{x}_2-\boldsymbol{x}_0)}{|\boldsymbol{x}_2-\boldsymbol{x}_0|}}\right| d\lambda,$$

where

$$\begin{aligned}\boldsymbol{X}_l &= \{\boldsymbol{b}_0(\lambda), \boldsymbol{b}_1(\lambda)\}, \\ \boldsymbol{b}_0(\lambda) &= \left\{ \boldsymbol{x}_0 + \lambda(\boldsymbol{x}_1 - \boldsymbol{x}_0) \middle| \lambda \in [0,1] \right\}, \\ \boldsymbol{b}_1(\lambda) &= \left\{ \boldsymbol{x}_2 + \lambda(\boldsymbol{x}_1 - \boldsymbol{x}_2) \middle| \lambda \in [0,1] \right\},\end{aligned}$$

where the set of 3 vertices denoted $\boldsymbol{x}_s$ define a 2-simplex, $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2\}$. The face flux over an edge is expressed as

$$\oint_{\partial\Omega_{i,j}} \boldsymbol{u}(\boldsymbol{x}, t) \cdot \boldsymbol{n}_{i,j} dA = \sum_{k}^{k_m} R'(\boldsymbol{u}, \boldsymbol{X}_{\partial\Omega_{i,j,k}}, \boldsymbol{n}_{i,j,k}, t), \tag{A.6}$$

where k_m is the number of facets attached to the i, j edge and $\boldsymbol{X}_{\partial\Omega_{i,j,k}}$ is the set of edge, face, and element centroids. Numerically, the above equation is evaluated similar to Equations (3.14) and (3.15), with $m_e = m_i = 20$.

A.3 Euler Angle 313

The Euler angle method is used to obtain the analytical expressions for a heavy Lagrangian Top. The method provides 3D attitude information for a given rigid body. Three angles, the Euler angles, are used in sequence to provide 3 rotations around a co-ordinate axis. The selected sequence of rotations is first a precession rotation by an angle ψ around the non-inertial z axis, followed by a nutation rotation by θ about the non-inertial x axis, and finally a spin rotation, ϕ, about the non-inertial z axis, see Figure 6.6. This order is commonly referred to a 3-1-3 sequence of Euler angles, where the angles are expressed in vector form as

$$\boldsymbol{\theta}_{313} = [\phi, \theta, \psi].$$

The transformation of some vector, $\boldsymbol{r}$, from the inertial to the non-inertial basis and *visa-versa* is expressed, using $\boldsymbol{\theta}_{313}$, as

$$\begin{aligned}{}^{o'}\boldsymbol{r} &= \underline{\boldsymbol{R}}_e(\boldsymbol{\theta}_{313})\,{}^{o}\boldsymbol{r}, \\ {}^{o}\boldsymbol{r} &= \underline{\boldsymbol{R}}_e(\boldsymbol{\theta}_{313})^T\,{}^{o'}\boldsymbol{r},\end{aligned}$$

where $\underline{\boldsymbol{R}}_e$ is a product of the three rotations and is given by

$$\underline{\boldsymbol{R}}_e(\boldsymbol{u}) = \begin{bmatrix} \cos\phi\cos\psi - \sin\phi\cos\theta\sin\psi & \cos\phi\sin\psi + \sin\phi\cos\theta\cos\psi & \sin\phi\sin\theta \\ -\sin\phi\cos\psi - \cos\phi\cos\theta\sin\psi & -\sin\phi\sin\psi + \cos\phi\cos\theta\cos\psi & \cos\phi\sin\theta \\ \sin\theta\sin\psi & -\sin\theta\sin\psi & \cos\theta. \end{bmatrix}$$

The rotational velocity of the body can be determined via the Euler angle and the Euler angle rates, $\dot{\boldsymbol{\theta}}_{313}$, as

$$\begin{aligned}{}^{o}\boldsymbol{\omega}_{o'/o} &= \underline{\boldsymbol{W}}_e(\boldsymbol{\theta}_{313})\dot{\boldsymbol{\theta}}_{313}, \\ {}^{o'}\boldsymbol{\omega}_{o'/o} &= \underline{\boldsymbol{W}}'_e(\boldsymbol{\theta}_{313})\dot{\boldsymbol{\theta}}_{313},\end{aligned}$$

where

$$\underline{\boldsymbol{W}}_e = \begin{bmatrix} \sin\theta\sin\psi & \cos\psi & 0 \\ -\sin\theta\cos\psi & \sin\psi & 0 \\ \cos\theta & 0 & 1 \end{bmatrix} \text{ and } \underline{\boldsymbol{W}}'_e = \begin{bmatrix} 0 & \cos\phi & \sin\phi\sin\theta \\ 0 & -\sin\phi & \cos\phi\sin\theta \\ 1 & 0 & \cos\theta \end{bmatrix}.$$

The spinning top is now modelled using the above representation. The angular momenta are given for the spin and precession as

$$\begin{aligned} H_\phi &= I_{zz'}\omega_{z'} = I_{zz'}(\dot{\phi} + \dot{\psi}\cos\theta), && \text{(A.7)} \\ H_\psi &= I_{xx'}\omega_{x'}\left(\sin\theta\sin\phi\right) + I_{yy'}\omega_{y'}\left(\sin\theta\cos\phi\right) + I_{zz'}\omega_{z'}\cos\theta, \\ &= I_{xx'}\dot{\psi}\sin^2\theta + I_{zz'}\left(\dot{\phi} + \dot{\psi}\cos\theta\right)\cos\theta, && \text{(A.8)} \end{aligned}$$

and the total energy of the top is expressed by

$$\begin{aligned} E &= \frac{1}{2}I_{xx'}\omega_{x'}^2 + \frac{1}{2}I_{yy'}\omega_{y'}^2 + \frac{1}{2}I_{zz'}\omega_{z'}^2 + mgl\cos\theta, \\ &= \frac{1}{2}I_{xx'}\left(\dot{\theta}^2 + \dot{\psi}^2\sin\theta\right) + \frac{1}{2}I_{zz'}(\dot{\phi} + \dot{\psi}\cos\theta)^2 + mgl\cos\theta, \\ &= \frac{\left(H_\psi - H_\phi\cos\theta\right)^2}{2I_{xx'}\sin^2\theta} + \frac{H_\phi^2}{2I_{zz'}} + \frac{1}{2}I_{xx'}\dot{\theta}^2 + mgl\cos\theta. \end{aligned}$$

The total energy can be split into a kinetic and potential component (Lagrangian form), E_k and E_p, formulated as

$$\begin{aligned} E &= E_k + E_p(\theta). \\ \frac{1}{2}I_{xx'}\dot{\theta}^2 &= E - E_p(\theta), && \text{(A.9)} \end{aligned}$$

where only the potential energy is a function of θ and has the form

$$E_p(\theta) = \frac{\left(H_\psi - H_\phi\cos\theta\right)^2}{2I_{xx'}\sin^2\theta} + \frac{H_\phi^2}{2I_{zz'}} + mgl\cos\theta.$$

Noting that E is constant, Equation (A.9) is differentiated with respect to time to obtain the nutation acceleration,

$$\begin{aligned} I_{xx'}\dot{\theta}\ddot{\theta} &= -E'_p(\theta)\dot{\theta}, \\ \ddot{\theta} &= -\frac{E'_p(\theta)}{I_{xx'}}, && \text{(A.10)} \end{aligned}$$

where E'_p is given,

$$E'_p(\theta) = \frac{\left(H_\psi - H_\phi\cos\theta\right)H_\phi}{I_{xx'}\sin\theta} - \frac{\left(H_\psi - H_\phi\cos\theta\right)^2\cos\theta}{I_{xx'}\sin^3\theta} + mgl\sin\theta.$$

Finally, the set of rate expressions for each Euler angle can be determined using Equations (A.7), (A.8), and (A.10):

$$\begin{aligned}
\dot{\phi} &= \frac{H_\phi}{I_{zz'}} - \frac{H_\psi - H_\phi \cos\theta}{I_{xx'} \sin^2\theta} \cos\theta, \\
\dot{\psi} &= \frac{H_\psi - H_\phi \cos\theta}{I_{xx'} \sin^2\theta}, \\
\dot{\vartheta} &= -\frac{E_p'(\theta)}{I_{xx'}}, \\
\dot{\theta} &= \vartheta.
\end{aligned}$$

Further details on Lagrangian Tops can be found in work by Slade [118].