A Higher-Order VOF Interface Reconstruction Scheme for Non-Orthogonal Structured Grids

with Application to Surface Tension Modelling

Author:
Niran A. Ilangakoon

Supervisor:
Prof. Arnaud G. Malan

Thesis presented for the degree of
Doctor of Philosophy
in the
Department of Mechanical Engineering,
University of Cape Town

October, 2021

Declaration

I, Niran Ilangakoon, hereby declare that the work on which this thesis is based is my original work (except where acknowledgements indicate otherwise) and that neither the whole work nor any part of it has been, is being, or is to be submitted for another degree in this or any other university.

I authorise the University of Cape Town to reproduce, for the purpose of research, either the whole or any portion of the contents in any manner whatsoever.

Signed by candidate

Date: 26th January 2021

To my parents Gihan and Devinka, and brother Kivan

"Whatever you can do, or dream you can, begin it.
Boldness has genius, power, and magic in it."

– Johann Wolfgang von Goethe

Abstract

The volume-of-fluid (VOF) method [24] is widely used to track the interface for the purpose of simulating liquid-gas interfacial flows numerically. The key strength of VOF is its mass conserving property. However, interface reconstruction is required when geometric properties such as curvature need to be accurately computed. For surface tension modelling in particular, computing the interface curvature accurately is crucial to avoiding so-called *spurious* or *parasitic* currents. Of the existing VOF-based schemes, the height-function (HF) method [10, 16, 18, 42, 46, 53] allows accurate interface representation on Cartesian grids. No work has hitherto been done to extend the HF philosophy to non-orthogonal structured grids.

To this end, this work proposes a higher-order accurate VOF interface reconstruction method for non-orthogonal structured grids. Higher-order in the context of this work denotes up to 4^{th}-order. The scheme generalises the interface reconstruction component of the HF method. Columns of control volumes that straddle the interface are identified, and piecewise-linear interface constructions (PLIC) are computed in a volume-conservative manner in each column. To ensure efficiency, this procedure is executed by a novel sweep-plane algorithm based on the convex decomposition of the control volumes in each column. The PLIC representation of the interface is then smoothed by iteratively refining the PLIC facet normals. Rapid convergence of the latter is achieved *via* a novel *spring-based* acceleration procedure. The interface is then reconstructed by fitting higher-order polynomial curves/surfaces to local stencils of PLIC facets in a least squares manner [29]. Volume conservation is optimised for at the central column.

The accuracy of the interface reconstruction procedure is evaluated *via* grid convergence studies in terms of volume conservation and curvature errors. The scheme is shown to achieve arbitrary-order accuracy on Cartesian grids and up to fourth-order accuracy on non-orthogonal structured grids. The curvature computation scheme is finally applied in a balanced-force continuum-surface-force (CSF) [4] surface tension scheme for variable-density flows on non-orthogonal structured grids in 2D. Up to fourth-order accuracy is demonstrated for the Laplace pressure jump in the simulation of a 2D stationary bubble with a high liquid-gas density ratio. A significant reduction in parasitic currents is demonstrated. Lastly, second-order accuracy is achieved when computing the frequency of a 2D inviscid oscillating droplet in zero gravity. The above tools were implemented and evaluated using the *Elemental*® multi-physics code and using a vertex-centred finite volume framework. For the purpose of VOF advection the algebraic CICSAM scheme (available in *Elemental*®) was employed.

Acknowledgements

I'd like to express my heartfelt gratitude to my supervisor, Prof. Arnaud Malan — thank you for all your years of guidance, patience, unwavering support, and endless encouragement without which this thesis would not have been brought to completion. Working with you, being a part of the InCFD Research Group, and contributing to the *Elemental*® software have impressed upon me a passion for CFD that I will always cherish.

I am grateful for the financial support provided in part by the National Research Foundation of South Africa (grant number 89916) and also by ArianeGroup (AG). I would like to thank Dr Philipp Behruzi (AG) and Mr Francesco De Rose (AG) for their key insights that contributed to the developmental stages of this research.

To my fellow InCFD teammates — Bevan, Leon, Roy, Tomas, Yusuf, Alaa, Prince, Michael, Elrich, and Darrian — thank you for contributing to the creative and intellectually stimulating yet light-hearted atmosphere in InCFD that has made it such a joy to be a part of. I would particularly like to thank Dr Bevan Jones for the many insightful discussions that shaped aspects of my research. Thank you also for your immense contributions to the *Elemental*® code that streamlined my coding experience — your developments in AGI, the incompressible flow solver, and the VOF module were pivotal to some of the key results in this work.

I'd like to acknowledge Simon Chili, Yanga Bavuma, and Brandon du Preez — thank you for sharing with me your passion for mathematics and for introducing me to a level of rigour that undoubtedly influenced many derivations in this work.

To my dear friends in Cape Town — Neha, Michael, Céline, Anroe, Sarig, Lara, Litha, Mick, Shoko, and Adir — I am so grateful to each of you for your endless support, for keeping me humble, and for being a constant source of joy and laughter throughout this journey.

Lastly, I'd like to express my deepest love and gratitude to my parents Gihan and Devinka, and brother Kivan — this journey would neither have been possible nor worthwhile without your endless love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Immiscible liquid-gas interfacial phenomena are ubiquitous in nature and play a significant role in a variety of industrial applications involving two-phase fluid flow. The volume-of-fluid (VOF) method [24] is widely used to track the interface when simulating such flows numerically. The key strength of VOF, in contrast to other interface tracking/capturing methods, is its mass conserving property. However, unlike level-set (LS) methods [2, 6, 12, 21, 40, 41, 58, 60] for instance, where the interface is represented by the zero contour of a smooth and differentiable level-set function, the major challenge with VOF lies in accurately computing interface-related geometric properties such as position, normal, and curvature. In addressing this challenge, accurate reconstruction of the VOF interface plays an important role in the modelling of interfacial phenomena such as phase change [37, 54, 57, 66] and surface tension [1, 4, 17, 46, 48, 51, 55]. In the case of the latter, which is the ultimate application of the present work, accurately computing the geometric curvature of the interface is crucial for avoiding so-called *spurious* or *parasitic* currents [1, 48, 55].

Of the existing VOF-based approaches, the height-function (HF) method has been widely used [10, 16, 18, 42, 46, 53] for accurate interface representation on Cartesian grids. The HF method reconstructs the VOF interface by computing discrete *heights* of the interface in *columns* of control volumes that overlap the interface, each of which is constructed in the direction nearest to the interface normal. Central difference formulae are then applied to *stencils* of heights to approximate the first- and second-order derivatives required for interface curvature computation.

The HF method was employed by Popinet [46] for surface tension computations on adaptively refined Cartesian grids, with second-order accurate curvature computation. The extension of HF to higher-order accuracy was first investigated by Sussman and Ohta [59], who achieved fourth-order accurate curvature estimates on uniform Cartesian grids in $\mathbb{R}^3$. Francois and Schwartz later [18] generalised HF for fourth-order accurate curvature estimates on non-uniform Cartesian grids in $\mathbb{R}^2$. Recently (2020), Evrard *et al.* [16] generalised HF for arbitrary-order curvature computations on both uniform and non-uniform Cartesian grids. A particular challenge with HF is the computation of curvature on under-resolved interfaces [42, 46], which can lead to scenarios where well-defined heights cannot be computed, thus resulting in insufficient heights in some stencils. To address this, Owkes and Desjardins [42] recently introduced a second-order accurate mesh-decoupled HF method which performs better than the standard HF method on coarser grids. This was achieved by enabling the construction of columns in directions parallel to the interface normal, and thus misaligned with the coordinate axes.

An alternate strategy for computing interface curvature is to explicitly represent the VOF interface by a parabola/paraboloid in the region of interest. Pilliod and Pucket [45] employed this strategy in the second-order accurate *efficient least-squares VOF interface reconstruction algorithm* (ELVIRA) on Cartesian grids. Renardy and Renardy [51] developed the *parabolic reconstruction of surface tension* (PROST) method on Cartesian grids, which computes the interface curvature from a weighted parabolic fit in a manner that locally minimises the error in the volume fraction field. This is achieved by an iterative procedure, where the volume fraction of each control volume intersected by the parabola is approximately evaluated at each iteration until an objective function is minimised.

Recent developments in surface tension modelling have also included unstructured grids [27–29] and non-orthogonal structured grids [6, 7, 25, 65]. Ito *et al.* [27] adapted the HF concept directly on unstructured grids in $\mathbb{R}^2$, but achieved less than first-order accuracy of the interface curvature. Ivey and Moin [28] later introduced the embedded height-function (EHF) method for convex polyhedral grids, which conservatively interpolates the VOF field onto a superimposed Cartesian grid on which the standard HF procedure is then carried out. Similar to the ELVIRA and PROST methods on Cartesian grids, Evrard *et al.* [15] recently proposed a method to compute local parabolic reconstructions of the interface on 2D unstructured grids, and demonstrated second-order accuracy of the computed curvature. Similarly, Jibben *et al.* [29] developed a 3D method for arbitrary polyhedral grids where a paraboloid is fitted to piecewise linear interface constructions (PLIC) [3, 11, 13, 14, 28, 33–35, 52, 56, 58] in an approximately volume-conservative manner. Their method was shown to yield between first- and second-order accuracy of the computed curvature. The author of this thesis recently generalised this method to obtain second- and fourth-order accurate curvature estimates on non-orthogonal structured grids in $\mathbb{R}^2$ [26]. A further component of the present work is its generalisation to non-orthogonal structured grids in $\mathbb{R}^3$.

Non-orthogonal structured grids have numerous applications in CFD and are particularly well suited for generating body-fitted grids to curved geometries such as satellite fuel tanks. Huang *et al.* [25] used an LS approach to model air/water turbulent flows around ship hulls on overset curvilinear grids. Wang *et al.* [65] developed a VOF method that reconstructs a distance function on a computational domain, onto which a curvilinear physical domain is mapped. More recently, Cao *et al.* [6, 7] developed a coupled LS-VOF method that directly reconstructs the LS distance function on the physical domain. However, the above methods achieved less than first-order accuracy in curvature, which was computed as a function of the second-derivative of a second-order accurate reconstructed distance function.

Unlike unstructured grids, the regularity of the vertex connectivity[1] of structured grids naturally allows for the development of robust geometric algorithms for VOF interface reconstruction. This is largely owing to the constant widths/cross-sections of interface columns. However, to the best of the author's knowledge, no work has hitherto been done in extending the philosophy of the HF method (as generalised by Evrard *et al.* [16]) to non-orthogonal structured grids. The latter results in interface columns with varying widths/cross-sections. Therefore, this research aims to develop VOF-based higher-order interface reconstruction capabilities on non-orthogonal structured grids for vertex-centred finite volume applications. To this end, the proposed scheme extends the following aspects to non-orthogonal structured grids — (i) the interface reconstruction component of the HF method [10, 16, 46, 53], and (ii) the notion of

[1]Here, the term *regularity* refers to each interior vertex being connected to $2d$ other vertices in the d-dimensional case.

volume-conservative polynomial fitting [15, 16, 29, 51] for accurate interface representation. The efficacy of the developed scheme shall be demonstrated by adopting it for interface curvature computations in surface-tension-driven flow problems. All non-orthogonal structured grids shall be stretched, anisotropic, block-structured grids similar to those shown in Fig. 1.1. Grid stretching (approx. 1.1) and anisotropy (max. 1:2) is included in the interest of industrial applicability. Note that this work employs a vertex-centred finite volume framework, but is equally relevant to cell-centred methods.

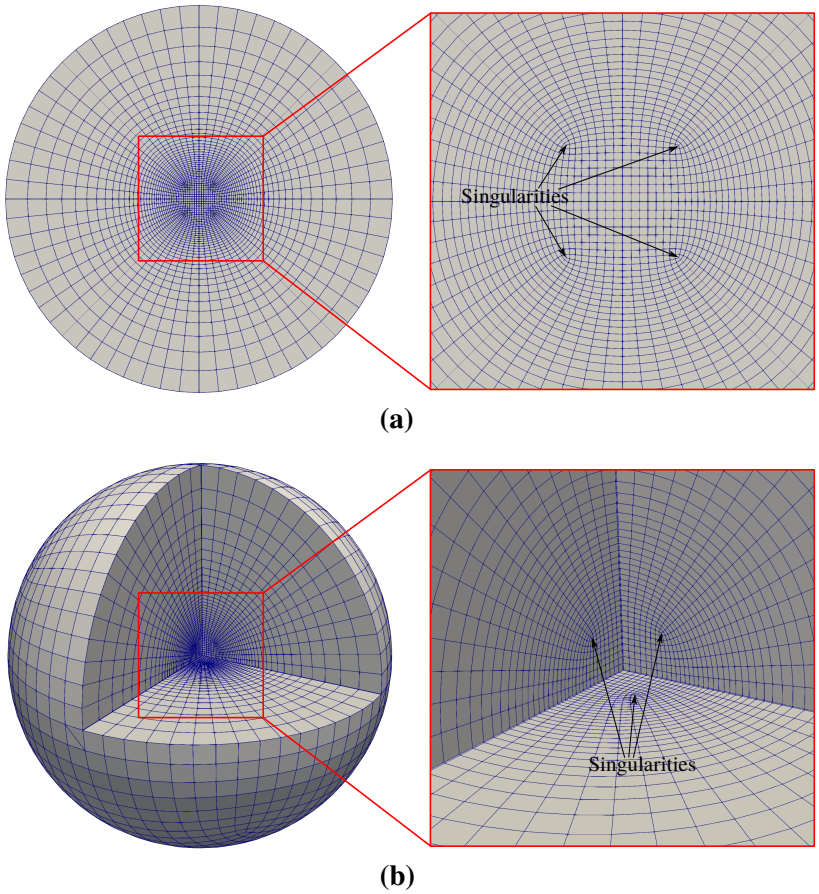


Figure 1.1: Typical non-orthogonal structured grids used in this work with close-ups of the inner structured regions. Block-structured grids of—(a) a circular domain and (b) a spherical domain.

The proposed scheme was implemented in the *Elemental*® multi-physics code [8, 32, 36, 43]. The first step in demonstrating higher-order accuracy of VOF interface reconstruction is the VOF field initialisation. For this purpose, the arbitrary grid initialiser (AGI) tool in *Elemental*®, developed by Jones *et al.* [30], is employed to initialise the VOF field to machine-precision accuracy.

1.2 Scope of Work and Novel Research Contributions

In achieving the aforementioned, the core research activities explored and the scope of each activity are summarised as follows:

- Identify a set of analytical functions describing interfaces in $\mathbb{R}^2/\mathbb{R}^3$. In the case of non-orthogonal grids, circular/spherical flow domains represented by block-structured grids are employed (see Fig. 1.1). Only internal (non-boundary) reconstruction is considered, and thus the investigation of contact-line phenomena (interaction of the interface with the solid-wall boundary) is beyond the scope of this work.

- Develop a means of constructing interface columns overlapping the interface, with careful consideration of how the interface curvature is to be computed in regions where the directions of columns switch. The developed method works generically in regions of the grid where the vertex connectivity is the same as that of an orthogonal structured grid. The handling of cases where the interface passes through vertices labelled as *singularities* in Fig. 1.1 requires the specialisation of the developed algorithms[2]. Therefore, the handling of such cases is left as future work.

- Extend the HF interface reconstruction philosophy to non-orthogonal structured grids with underlying vertex-centred finite volume discretisation (although the developed scheme is directly applicable to cell-centred methods as well). This entailed the development of the geometric tools required for PLIC reconstruction of the interface in each column. Note that this work employs an algebraic VOF scheme. The premise is therefore that no existing geometric interface representation is available.

- Formulate a method of refining the normals of the computed PLIC facets to a sufficient degree to enable higher-order interface representation through stencils of PLIC facets (see Fig. 1.2). Although interfaces with a change in the sign of curvature (negative Gaussian curvature in $\mathbb{R}^3$) are not investigated in this work, the developed PLIC normal refinement scheme applies generically to such cases.

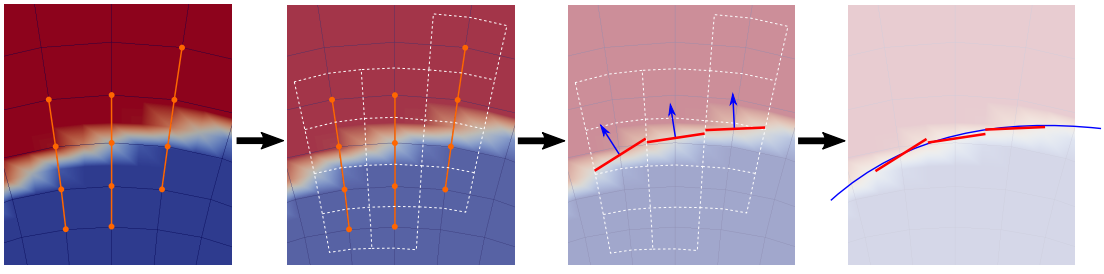


Figure 1.2: Procedure for polynomial ($n = 2$) representation of the VOF interface employing a stencil of three PLIC facets in $\mathbb{R}^2$ — (a) three interface columns, (b) the vertex-centred median dual-cells of the vertices in each column, (c) conservative PLIC reconstruction (red line segments) corresponding to the PLIC normals (blue arrows), and (d) the fitted polynomial.

[2]Note that in $\mathbb{R}^3$ (see Fig. 1.1(b)), these *singularities* lie along the twelve edges of the inner cubic structured region of the grid, and along the edges emanating outward from the eight corners of the same. When the interface intersects these regions, column construction is currently only feasible along the direction of these edges, and not in transverse directions.

- Identify an appropriate polynomial fitting scheme for interface representation *via* polynomials of order n. In this work, $n \in \{2, 4, 6, 8\}$ and $n \in \{2, 4\}$ are investigated on Cartesian and non-orthogonal structured grids, respectively. This activity involved the comparison of the methods by Evrard *et al.* [16] and Jibben *et al.* [29] on Cartesian grids.

- Employ a continuum surface force (CSF) [4] representation of the surface tension source term to model surface tension effects. The flow solver is tested for gas-liquid density ratios of up to $1:1000$ only in 2D and in the absence of gravity. The corresponding 3D flow cases could not be completed due to time constraints of the project, and are thus left for future work.

- Temporal discretisation of the governing equations using a standard semi-implicit fractional step solution procedure [4, 36, 43, 47] and an edge-based spatial discretisation of the same. A consistent and well-balanced discretisation of the pressure and surface tension source terms is employed, with careful consideration of the non-orthogonality of the underlying grid with respect to control volume faces.

The following is a summary of the novel research contributions which were required to develop an accurate VOF interface reconstruction method for non-orthogonal structured grids:

- A column construction algorithm was developed for non-orthogonal structured grids that—(i) accounts for the arbitrary directions of the grid edges with respect to the interface normals, and (ii) ensures sufficient overlap in regions of the interface where column directions switch. The latter property is introduced to allow for a *gradual* and *bounded* transition of the computed interface curvature from one region of contiguous columns to another.

- A novel PLIC construction scheme was developed, based on the convex-decomposition of the control volumes in a given interface column. The scheme is initialised with the so-called bracketing procedure, which is followed by an *analytical* evaluation of the position of the PLIC facet that — (i) is oriented with respect to a pre-computed normal, and (ii) conserves a given reference fluid volume in each column.

- A localised PLIC facet normal refinement procedure was introduced that relies solely on the centroids of the PLIC facets in a given stencil. This method was found to result in slow convergence of the overall refinement procedure. It was thus enhanced by a novel *spring-based* normal refinement acceleration procedure that both *locally* and *globally* reduces the errors of the PLIC normals across the interface in a single refinement iteration.

- The 3D least squares paraboloid fitting scheme of Jibben *et al.* [29] was chosen as a suitable polynomial fitting procedure to be adapted to non-orthogonal structured grids. The mathematical formulation of this scheme was generalised for fitting polynomials of order n in both $\mathbb{R}^2$ and $\mathbb{R}^3$. However, as mentioned above, the scheme was only employed for the cases $n \in \{2, 4\}$ on non-orthogonal structured grids.

1.3 Publications

The following journal publications resulted from the above novel contributions:

N. A. Ilangakoon, A. G. Malan, and B. W. Jones, *A higher-order accurate surface tension modelling volume-of-fluid scheme for 2D curvilinear meshes*, Journal of Computational Physics, 420 (2020), 109717.

N. A. Ilangakoon and A. G. Malan, *A higher-order accurate VOF interface curvature computation scheme for 3D non-orthogonal structured grids*. Submitted for review to Journal of Computational Physics.

1.4 Thesis Outline

This thesis document is subdivided into ten chapters including the introduction and conclusion. The following list provides a synopsis of each chapter:

- ***Chapter 1: Introduction.*** This chapter introduced the background of the project and provided an overview of the project scope, the novel contributions in this work, and now an outline of the chapters in this document.
- ***Chapter 2: The Height Function Method.*** The philosophy behind height-function (HF) method and its recent extension to arbitrary order accuracy by Evrard *et al.* [16] on Cartesian grids are briefly detailed. This is to set the stage for the impending generalisation of the interface reconstruction component of the HF method to non-orthogonal structured grids.
- ***Chapter 3: Height Based Least Squares Polynomial Fitting.*** The least squares paraboloid fitting technique of Jibben *et al.* [29] is here generalised to arbitrary order polynomial fits in $\mathbb{R}^2$ and $\mathbb{R}^3$ for interface representation on Cartesian grids. This chapter motivates the choice of a least squares polynomial fitting approach (as opposed to the exact polynomial fitting approach of Evrard *et al.* [16]) in enabling the formulation of a higher-order interface reconstruction scheme for non-orthogonal structured grids. The chapter is concluded with the validation of the polynomial fitting scheme on Cartesian grids.
- ***Chapter 4: Interface Column Construction.*** As with the HF method, the scheme proposed in this work commences with the construction of columns overlapping the liquid-gas interface. This chapter details the procedure for interface column construction. Particular consideration is given to the creation of an overlapping region of columns where the column directions switch.
- ***Chapter 5: Conservative PLIC Reconstruction.*** After the construction of the interface columns on non-orthogonal grids, the next step is to obtain a PLIC representation of the VOF interface. Each PLIC facet is computed in a manner that conserves the volume of a reference fluid in each interface column. The chapter closes with the procedure for constructing appropriate PLIC facet stencils that can be further utilised for computing the normal and curvature of the interface.
- ***Chapter 6: Refinement of PLIC Facet Normals.*** To achieve higher-order accuracy of the polynomial fits, this work adopts an iterative PLIC facet normal refinement procedure. The

standard refinement procedure leads to slow convergence of the normals and is thus enhanced by a novel *spring-based* refinement acceleration procedure which is developed in this chapter.

- ***Chapter 7: PLIC Facet Based Least Squares Polynomial Fitting.*** The least squares polynomial fitting procedure introduced in Chapter 3 is here extended to non-orthogonal structured grids. Polynomials are fitted through stencils of PLIC facets as opposed to stencils of heights. While the mathematical formulation of the scheme generically applies to fitting polynomials of order n, only $n \in \{2, 4\}$ is considered in this work.

- ***Chapter 8: Surface Tension Modelling Mathematical Formulation.*** For the purpose of curvature computation, the higher-order interface reconstruction tools developed in the previous chapters are here integrated into a numerical flow solver. The latter is based on the incompressible Navier-Stokes equations with surface tension for variable-density two-phase flow. A key consideration in this chapter is the consistent and well-balanced spatial discretisation of the pressure term and the surface tension source term.

- ***Chapter 9: Results.*** A variety of test cases are used to evaluate the capabilities of the higher-order interface reconstruction tools in terms of spatial accuracy and computational cost. The spatial accuracy of the scheme is tested by means of the convergence characteristics of the RMS and maximum error of interface curvature and those of volume conservation of the reference fluid for a circular/spherical and elliptical/ellipsoidal interfaces. It is also tested by means of a 2D inviscid oscillating droplet test case for which an analytical solution for the oscillation frequency is available. The efficacy of the well-balanced spatial discretisation of the governing equations is evaluated *via* a 2D static bubble test. A breakdown of the computational cost of the various components of the higher-order interface reconstruction scheme is provided. Lastly, the computational cost of the overall scheme is compared to that of the incompressible flow solver.

- ***Chapter 10: Conclusions and Future Work.*** This chapter provides a summary of the novel research contributions in this work and offers recommendations for furthering the research.

Fig. 1.3 depicts the overall flow diagram of the proposed scheme and the relationships between the content presented in each chapter.

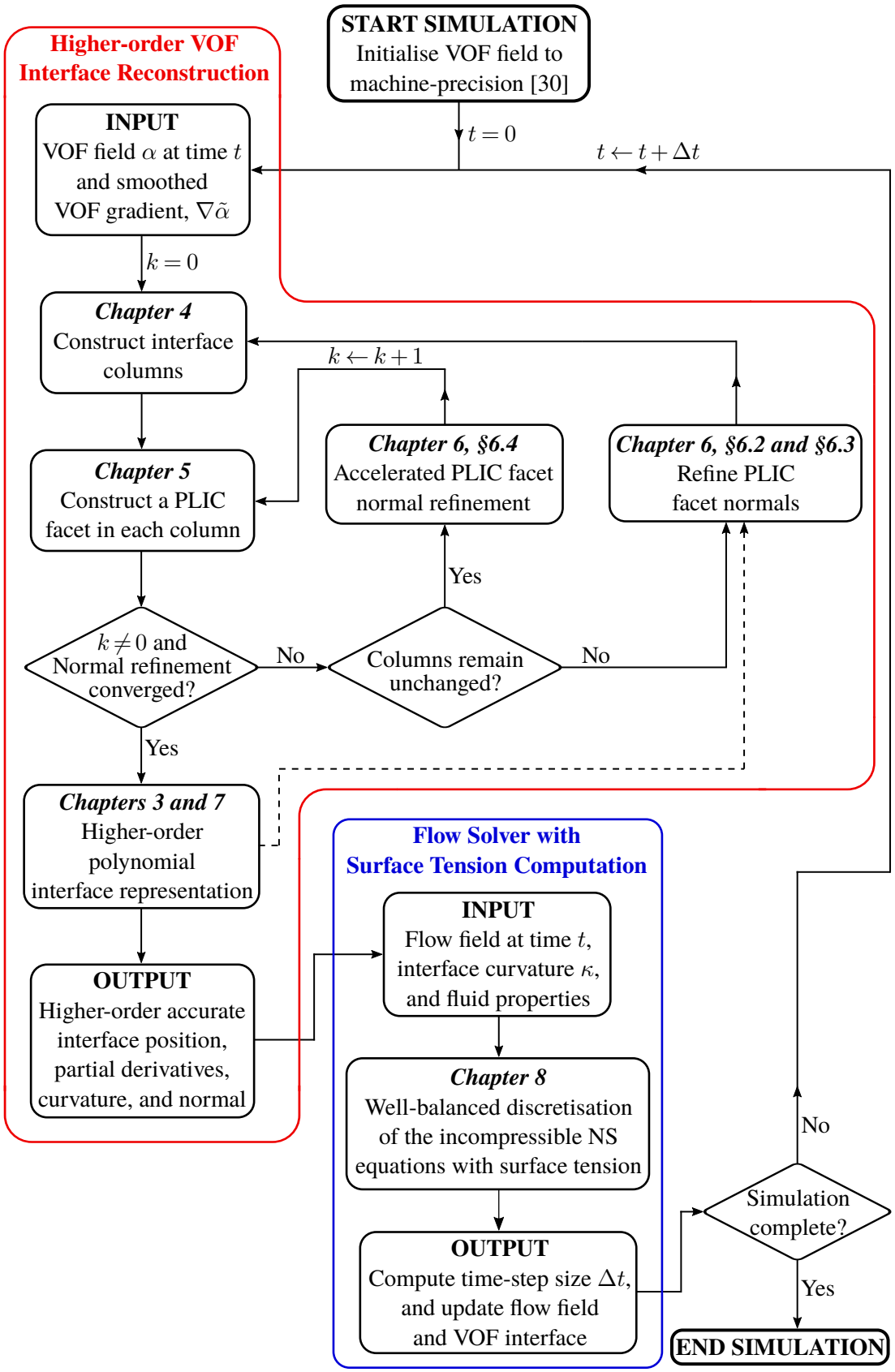


Figure 1.3: Flow diagram of the overall higher-order VOF interface reconstruction algorithm applied to surface tension modelling. The dashed arrow indicates the dependence of some of the procedures detailed in Chapter 6 on those detailed in Chapter 7.

Chapter 2

The Height Function Method

2.1 Background

This chapter describes the basic height-function (HF) method [10, 16–18, 46, 53] on Cartesian grids. This is for the dual purpose of establishing the nomenclature as well as highlighting the key concepts that must be generalised for application to non-orthogonal structured grids. The HF method is grounded on the simple observation that a local coordinate system can be defined in which the interface is described as the graph of a function [48]. Consider a function, $y=f(x)$, that describes a portion of a two-dimensional interface, as shown in Fig. 2.1.

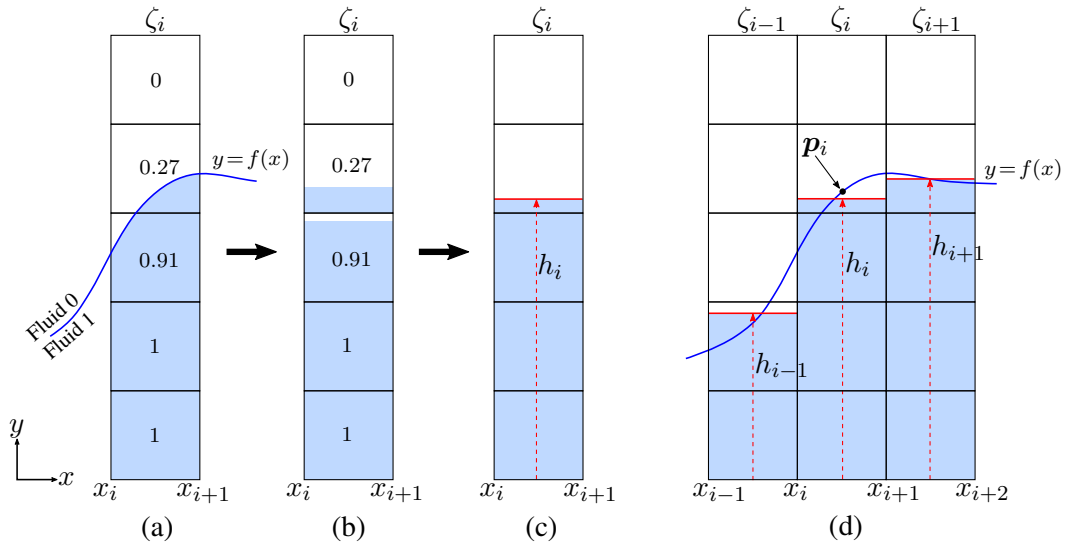


Figure 2.1: Construction of a stencil of heights from the volume fraction field associated to an interface (blue curve) described by $y=f(x)$. (a) The analytical distribution of two fluids in a height column ζ_i. (b) The associated volume fraction field. (c) The computed height, h_i, from the volume fraction field in ζ_i. (d) A three-column stencil.

Suppose we seek to compute the unit normal $\boldsymbol{n}$ and curvature κ of $y = f(x)$ at the point $\boldsymbol{p}_i = (x_{i+1/2},\ f(x_{i+1/2}))$, where $x_{i+1/2} = (x_i + x_{i+1})/2$. These can be *analytically* expressed

by the formulae

$$\boldsymbol{n} = \frac{1}{\sqrt{1+f_x^2}}\begin{bmatrix}-f_x\\1\end{bmatrix}, \tag{2.1}$$

$$\kappa = \frac{f_{xx}}{(1+f_x^2)^{3/2}}. \tag{2.2}$$

This requires the evaluation of the first (f_x) and second (f_{xx}) derivatives of $f(x)$ at $x = x_{i+1/2}$. However, since $y = f(x)$ is not known beforehand in VOF, the derivatives of $f(x)$ must be *numerically* approximated by reconstructing the interface from the volume fraction data. The HF method reconstructs the interface by compacting the volume fraction field to obtain discrete *heights* of the interface with respect to some datum. Each height is computed in a single *column* of control volumes that straddles the interface, and is constructed in the direction nearest to the interface normal (*e.g.* the gradient of the smoothed volume fraction field). Figs. 2.1(a) – (c) illustrate this procedure for a column, ζ_i, which straddles a portion of the interface described by the function $y = f(x)$.

Assuming that the x-axis is the datum (as per Fig. 2.1), Eq. (2.3) demonstrates how the height, h_i, is computed in column ζ_i. This is done by enforcing the volume conservation of fluid 1 in the column as follows:

$$\Delta x_i h_i = \int_{x_i}^{x_{i+1}} f(x)\,dx \quad\Longleftrightarrow\quad h_i = \frac{1}{\Delta x_i}\int_{x_i}^{x_{i+1}} f(x)dx = \frac{1}{\Delta x_i}\sum_{k\in\zeta_i}\alpha_k\mathcal{V}_k, \tag{2.3}$$

where $\Delta x_i = x_{i+1} - x_i$, and $\mathcal{V}_k$ and α_k are the volume and volume fraction of control volume $k \in \zeta_i$, respectively. The derivatives of $f(x)$ can then be approximated by applying central difference formulae to a *stencil* of three heights, $\{h_{i-1}, h_i, h_{i+1}\}$, as depicted in Fig. 2.1(d). In the special case of uniform Cartesian grids, where $\Delta x_{i-1} = \Delta x_i = \Delta x_{i+1} = \Delta x$, the derivatives of $f(x)$ can be approximated to second-order accuracy at $x = x_{i+1/2}$ by

$$\begin{aligned} f_x &= \frac{h_{i+1} - h_{i-1}}{2\Delta x} + \mathcal{O}(\Delta x^2),\\ f_{xx} &= \frac{h_{i+1} - 2h_i + h_{i-1}}{\Delta x^2} + \mathcal{O}(\Delta x^2). \end{aligned}$$

Similarly in $\mathbb{R}^3$, the equivalent of Fig. 2.1(d) is a 3×3 stencil of columns centred around column $\zeta_{i,j}$, with the interface described by a function $z = f(x, y)$. A projection of this stencil onto the xy-plane is depicted in Fig. 2.2. Suppose that the unit normal $\boldsymbol{n}$ and mean curvature κ of $z = f(x, y)$ were to be computed at the point $\boldsymbol{p}_i = (x_{i+1/2},\ y_{j+1/2},\ f(x_{i+1/2}, y_{j+1/2}))$, where $y_{j+1/2} = (y_j + y_{j+1})/2$ and $x_{i+1/2}$ is as defined previously. These can respectively be analytically expressed as

$$\boldsymbol{n} = \frac{1}{\sqrt{1+f_x^2+f_y^2}}\begin{bmatrix}-f_x\\-f_y\\1\end{bmatrix}, \tag{2.4}$$

$$\kappa = \frac{f_{xx}(1+f_y^2) - 2f_x f_y f_{xy} + f_{yy}(1+f_x^2)}{(1+f_x^2+f_y^2)^{3/2}}, \tag{2.5}$$

which require the evaluation of the first- and second-order partial derivatives of $f(x,y)$ at $x = x_{i+1/2}$ and $y = y_{j+1/2}$. Note that in $\mathbb{R}^3$, the definition of mean curvature employed in this work[3] is $\kappa = \kappa_1 + \kappa_2$, where κ_1 and κ_2 are the principal curvatures.

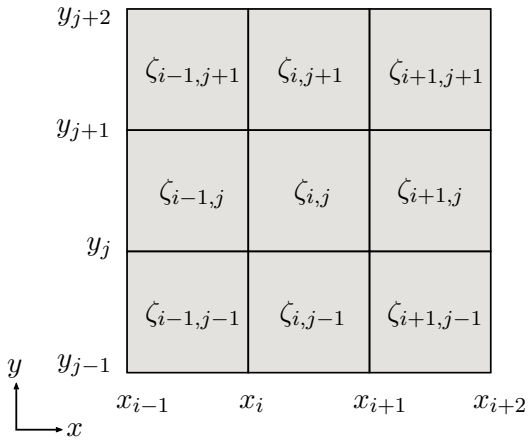


Figure 2.2: A 3×3 stencil of columns centred around column $\zeta_{i,j}$, projected onto the xy-plane. The columns are constructed in the direction of the z-axis, which points out of the page.

As in the 2D case, an interface height must be computed in each column of the stencil to approximate the partial derivatives of $f(x,y)$. The volume-conservative approach in Eq. (2.3) is again adopted to compute the height $h_{i,j}$ in the column $\zeta_{i,j}$ as

$$h_{i,j} = \frac{1}{\Delta x_i \Delta y_j} \int_{y_j}^{y_{j+1}} \int_{x_i}^{x_{i+1}} f(x,y)\, dxdy = \frac{1}{\Delta x_i \Delta y_j} \sum_{k \in \zeta_{i,j}} \alpha_k \mathcal{V}_k, \tag{2.6}$$

where $\Delta y_j = y_{j+1} - y_j$, and $\Delta x_i = x_{i+1} - x_i$. For uniform Cartesian grids, where each column has a constant cross-section of $\Delta x \times \Delta y$, the first- and second-order partial derivatives of $f(x,y)$ at $(x_{i+1/2}, y_{i+1/2})$ can be approximated to second-order accuracy by

$$\begin{aligned}
f_x &= \frac{h_{i+1,j} - h_{i-1,j}}{2\Delta x} + \mathcal{O}(\Delta x^2), \\
f_y &= \frac{h_{i,j+1} - h_{i,j-1}}{2\Delta y} + \mathcal{O}(\Delta y^2), \\
f_{xx} &= \frac{h_{i+1,j} - 2h_{i,j} + h_{i-1,j}}{\Delta x^2} + \mathcal{O}(\Delta x^2), \\
f_{yy} &= \frac{h_{i,j+1} - 2h_{i,j} + h_{i,j-1}}{\Delta y^2} + \mathcal{O}(\Delta y^2), \\
f_{xy} &= \frac{h_{i+1,j+1} - h_{i+1,j-1} - h_{i-1,j+1} + h_{i-1,j-1}}{4\Delta x \Delta y} + \mathcal{O}(\Delta x^2, \Delta y^2).
\end{aligned}$$

In summary, for uniform Cartesian grids, the procedures outlined thus far produce second-order accurate estimates of the partial derivatives with respect to Δx (and Δy in $\mathbb{R}^3$). In general,

[3]This definition differs from the differential geometric definition of mean curvature as $\kappa_m = (\kappa_1 + \kappa_2)/2$. This is preferred as the conventional fluid mechanics definition of the mean curvature used to compute the Laplace pressure jump across a liquid-gas interface is $\kappa = 2\kappa_m$.

however, a drop of one order of accuracy can be expected for non-uniform Cartesian grids [16, 18, 59]. Over the last decade, several strategies have been proposed, to increase the order of accuracy of the HF method by enlarging the stencil of heights employed. The next section details the most recent of these strategies [16], that allows for arbitrary-order accuracy.

2.2 Recent Extension to Arbitrary Order of Accuracy

The extension to higher-order accuracy, of the traditional HF method outlined in the previous section, was first investigated by Sussman and Ohta [59]. They achieved fourth-order accurate curvature estimates on 3D uniform Cartesian grids. Francois and Schwartz later [18] introduced a strategy for estimating curvature to fourth-order accuracy on 2D non-uniform Cartesian grids. Zhang [67] then extended HF to arbitrary-order accuracy on 2D uniform Cartesian grids. Recently, Evrard *et al.* [16] extended HF to arbitrary-order accuracy on non-uniform Cartesian grids in both 2D and 3D. Due to its importance and relevance to this work, the method of Evrard *et al.* is briefly outlined next.

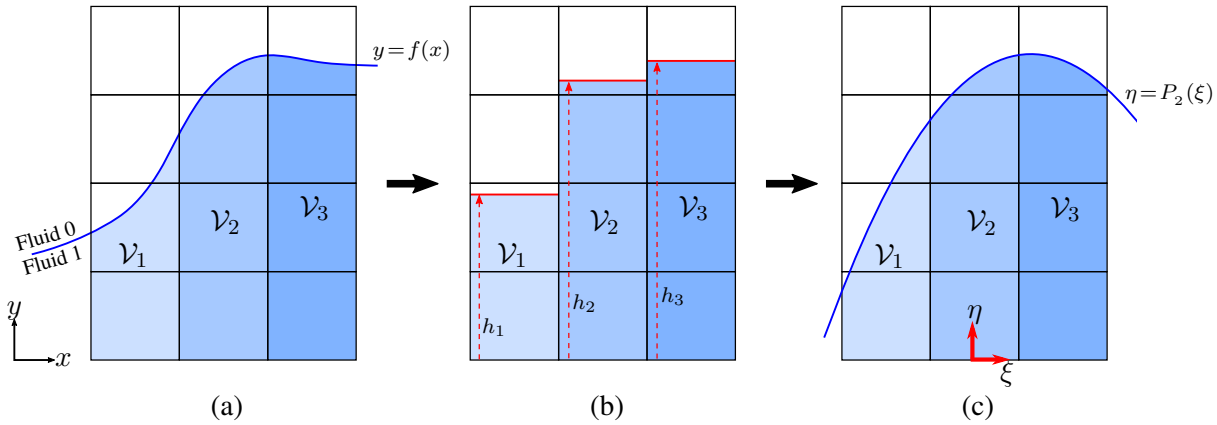


Figure 2.3: Construction of a parabola in a stencil of three heights that conserves the volume of fluid 1 in each column.

2.2.1 2D Method

To achieve arbitrary-order accuracy, the method of Evrard *et al.* [16] relies on reconstructing the interface *via* an n^{th}-order polynomial fit. The polynomial is fitted to a stencil of $n+1$ heights centred around the column for which it is required. The polynomial $P_n(\xi)$ is defined in a local $\xi\eta$-coordinate system with the column centre at $\xi = 0$ (see Fig. 2.3):

$$P_n(\xi) = \sum_{r=1}^{n+1} c_r \xi^{r-1}, \tag{2.7}$$

where the unknown coefficients $\boldsymbol{c} = [c_1\ c_2\ ...\ c_{n+1}]^T$ are to be determined. The column indexing convention in Section 2.1 is now replaced for convenience — the index p shall refer to an arbitrary height, h_p, computed above the column integration interval $\Omega_p = [\xi_{p_{min}},\ \xi_{p_{max}}]$, where the column width is $\Delta\xi_p = \xi_{p_{max}} - \xi_{p_{min}}$. Having pre-computed the heights h_p for all $p \in \{1, 2, ..., (n+1)\}$ using Eq. (2.3), the unknown coefficients $\boldsymbol{c}$ are computed such that the

volume of fluid 1 is conserved in each column (see Fig. 2.3). This equates to

$$\begin{aligned} h_p &= \frac{1}{\Delta\xi_p}\int_{\Omega_p} P_n(\xi)\, d\xi \\ &= \frac{1}{\Delta\xi_p}\sum_{r=1}^{n+1} c_r \int_{\xi_{p_{min}}}^{\xi_{p_{max}}} \xi^{r-1}\, d\xi \\ &= \frac{1}{\Delta\xi_p}\sum_{r=1}^{n+1} c_r \left(\frac{\xi_{p_{max}}^r - \xi_{p_{min}}^r}{r}\right). \end{aligned}$$

Note that the above equation is to be satisfied for all $n+1$ columns. Therefore, given a vector of $n+1$ heights, $\boldsymbol{h}$, $\boldsymbol{c}$ is solved for from the linear system

$$\boldsymbol{A}\boldsymbol{c} = \boldsymbol{h}, \tag{2.8}$$

where the row (p) and column (r) entries of $\boldsymbol{A}$ are given by

$$A_{pr} = \frac{\xi_{p_{max}}^r - \xi_{p_{min}}^r}{r\Delta\xi_p}, \quad p,r \in \{1,2,...,(n+1)\}.$$

The derivatives associated to the interface at $\xi = 0$ can then be straightforwardly computed from Eq. (2.7). This method was demonstrated to achieve up to n^{th}-order accuracy and $(n-1)^{\text{th}}$-order accuracy on uniform and non-uniform Cartesian grids, respectively [16].

2.2.2 3D Method

The above 2D procedure was extended by Evrard *et al.* [16] to 3D. Here, they fit an n^{th}-order bivariate polynomial surface through a stencil of $N=(n+1)^2$ heights. The polynomial is defined with respect to a local $\xi\eta\psi$-coordinate system such that the partial derivatives of the interface are approximated at $\xi = \eta = 0$. Given the unknown coefficient vector $\boldsymbol{c} = [c_1\ c_2\ ...\ c_N]^T$, the bivariate polynomial is defined as

$$P_n(\xi,\eta) = \sum_{s=1}^{n+1}\sum_{r=1}^{n+1} c_i \xi^{r-1}\eta^{s-1}, \quad i = r + (s-1)(n+1) \in \{1,2,...,N\}. \tag{2.9}$$

Adopting the indexing convention from the previous section, p shall represent an arbitrary height, h_p, computed above the integration domain $\Omega_p = [\xi_{p_{min}}, \xi_{p_{max}}] \times [\eta_{p_{min}}, \eta_{p_{max}}]$. Next, by similarly defining $\Delta\xi_p = \xi_{p_{max}} - \xi_{p_{min}}$ and $\Delta\eta_p = \eta_{p_{max}} - \eta_{p_{min}}$, conserving the volume of fluid 1 in each column of the stencil reduces to equating

$$\begin{aligned} h_p &= \frac{1}{\Delta\xi_p\Delta\eta_p}\int_{\Omega_p} P_n(\xi,\eta)\, d\xi d\eta \\ &= \frac{1}{\Delta\xi_p\Delta\eta_p}\sum_{s=1}^{n+1}\sum_{r=1}^{n+1} c_i \int_{\eta_{p_{min}}}^{\eta_{p_{max}}}\int_{\xi_{p_{min}}}^{\xi_{p_{max}}} \xi^{r-1}\eta^{s-1}\, d\xi d\eta \\ &= \frac{1}{\Delta\xi_p\Delta\eta_p}\sum_{s=1}^{n+1}\sum_{r=1}^{n+1} c_i \left(\frac{\xi_{p_{max}}^r - \xi_{p_{min}}^r}{r}\right)\left(\frac{\eta_{p_{max}}^s - \eta_{p_{min}}^s}{s}\right), \end{aligned} \tag{2.10}$$

which must hold for all N columns. Therefore, given a vector of N heights $\boldsymbol{h}$ the above again yields the linear system

$$\boldsymbol{A}\boldsymbol{c} = \boldsymbol{h}$$

from which $\boldsymbol{c}$ is solved and the row (p) and column (q) entries of matrix $\boldsymbol{A}$ are

$$A_{pq} = \frac{(\xi^r_{p_{max}} - \xi^r_{p_{min}})(\eta^s_{p_{max}} - \eta^s_{p_{min}})}{rs\Delta\xi_p\Delta\eta_p}, \tag{2.11}$$
$$p, q \in \{1, 2, ..., N\}, \quad s = 1 + \lfloor (q-1)/(n+1) \rfloor, \quad \text{and} \;\; r = q - (s-1)(n+1),$$

and $\lfloor \cdot \rfloor$ denotes the floor function. Once again, the partial derivatives associated to the interface at $\xi = \eta = 0$ can be straightforwardly computed from Eq. (2.9). As per the 2D method, the above 3D method was shown to achieve up to n^{th}-order accuracy and $(n-1)^{\text{th}}$-order accuracy on uniform and non-uniform Cartesian grids, respectively [16].

2.3 Closure

This chapter outlined the standard height-function method as well as its recent extension [16] to arbitrary-order accuracy on non-uniform Cartesian grids. The interface reconstruction employed by the cited authors can be considered an *exact* n^{th}-order polynomial fitting procedure. This is because the number of heights must always equal the number of polynomial coefficients, so as to enforce mass conservation in each column of the stencil as per Eqs. (2.8) and (2.10). However, this method is strictly limited to Cartesian grids as will be explained in the next chapter. A more general polynomial fitting approach for non-orthogonal structured grids will be employed in this work and compared to the cited work in the context of Cartesian grids.

Chapter 3

Height Based Least Squares Polynomial Fitting

3.1 Background

This chapter lays the foundation for the polynomial curve/surface fitting procedure that is adopted in this work. Unlike the method of Evrard *et al.* [16], the method employed here is based on that of Jibben *et al.* [29]. The latter involves a polynomial of order n being fitted in a least squares sense through *at least* $(n{+}1)$ heights in $\mathbb{R}^2$ and $(n{+}1)(n{+}2)/2$ heights in $\mathbb{R}^3$. Fig. 3.1 illustrates the case of a parabola fitted through a stencil of three heights in $\mathbb{R}^2$. Note that considerations regarding changing the direction of height columns are excluded from this chapter as the focus is on HF-based polynomial fitting in stencils of contiguous columns.

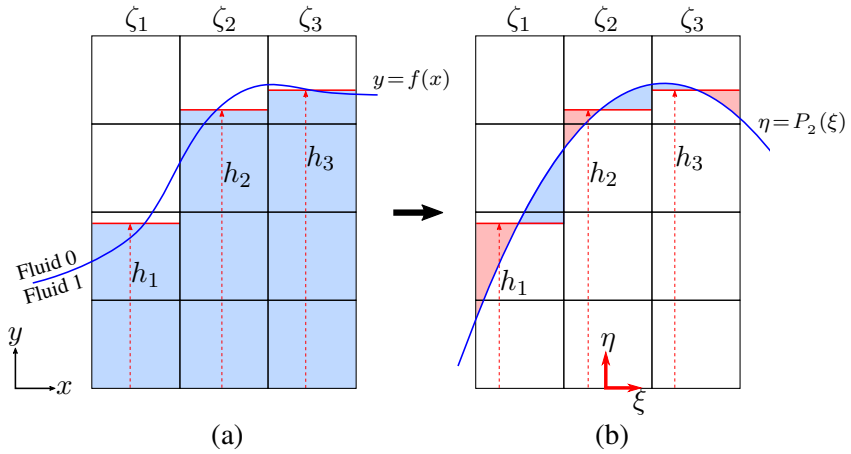


Figure 3.1: Fitting a parabola to a stencil of three heights associated to an interface (blue curve) defined by $y = f(x)$. (a) Construction of the heights, and (b) least squares parabolic fit through the height stencil.

A least squares approach (as opposed to the exact fitting approach of Evrard *et al.* [16], and thus the standard HF method) is adopted here for two reasons. Firstly, it is more versatile in terms of the number of heights through which the polynomial can be fitted. This becomes an important characteristic in the context of block-structured grids which contain regions where a *symmetric* stencil of *exactly* $(n+1)$ and $(n+1)^2$ columns cannot be constructed in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively. Secondly, in $\mathbb{R}^3$, the exact fitting method [16] results in a polynomial whose mean curvature at $\xi = \eta = 0$ is dependent on the orientation of the (ξ, η, ψ) local coordinate system

about the ψ-axis (direction of height construction). However, there is no universally agreed upon orientation of the local coordinate system in the case of non-orthogonal structured grids (the ψ-axis is aligned with the interface normal in this work). As per Eq. (2.9), this orientation-dependence is due to the strict requirement of expressing an n^{th}-order polynomial as a linear combination of the polynomial basis functions in the set

$$\mathcal{B}_1 = \big\{\xi^r \eta^s |\ r, s \in \{0, 1, ..., n\}\big\},$$

which contains terms of order up to $2n$. The consequence of this requirement can be demonstrated as follows. Consider the following coordinate transformation resulting from a clockwise rotation of the local coordinate system about the ψ-axis by an angle θ:

$$\begin{bmatrix} \xi' \\ \eta' \\ \psi' \end{bmatrix} = \boldsymbol{R}(\theta) \begin{bmatrix} \xi \\ \eta \\ \psi \end{bmatrix}, \qquad \boldsymbol{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.1}$$

A polynomial, $P_n(\xi', \eta')$, defined in the basis $\mathcal{B}_1$ in the $\xi'\eta'\psi'$-coordinate system is expressed as

$$P_n(\xi', \eta') = \sum_{s=0}^{n} \sum_{r=0}^{n} c'_{rs} (\xi')^r (\eta')^s. \tag{3.2}$$

Using Eq. (3.1), P_n can likewise be expressed with respect to the $\xi\eta\psi$-coordinate system as

$$\begin{aligned} P_n(\xi, \eta) &= \sum_{s=0}^{n} \sum_{r=0}^{n} c'_{rs} (\xi\cos\theta - \eta\sin\theta)^r (\xi\sin\theta + \eta\cos\theta)^s \\ &= \sum_{s=0}^{2n} \sum_{r=0}^{2n-s} c_{rs} \xi^r \eta^s, \end{aligned} \tag{3.3}$$

where $c_{rs} \equiv c_{rs}\big(\theta,\ \{c'_{ij} \mid i, j \in \{0, 1, ..., n\}\}\big)$. The above demonstrates that, to account for arbitrary coordinate system rotations described by Eq. (3.1) in the method of Evrard *et al.* [16] (such that the mean curvature at $\xi = \eta = 0$ is independent of the orientation), one would require a different number of heights. This is because the following set of polynomial basis functions is required instead:

$$\mathcal{B}'_1 = \big\{\xi^r \eta^s |\ r, s \in \{0, 1, ..., 2n\},\ r + s \leq 2n\big\}.$$

Fig. 3.2 shows that the required polynomial basis functions before and after rotation for the case $n = 2$ can be visualised in Pascal's triangle. This demonstrates that $\mathcal{B}_1$ has insufficient polynomial basis functions to account for rotations described by $\boldsymbol{R}(\theta)$. Evrard *et al.* [16] do not encounter this issue as the local coordinate axes can always be made parallel to the edges of Cartesian grids, thus ensuring polynomial fits that are consistent over the entire liquid-gas interface. However, since this strategy cannot be enforced on non-orthogonal structured grids, a polynomial fitting procedure that is invariant with respect to rotations about the ψ-axis is needed.

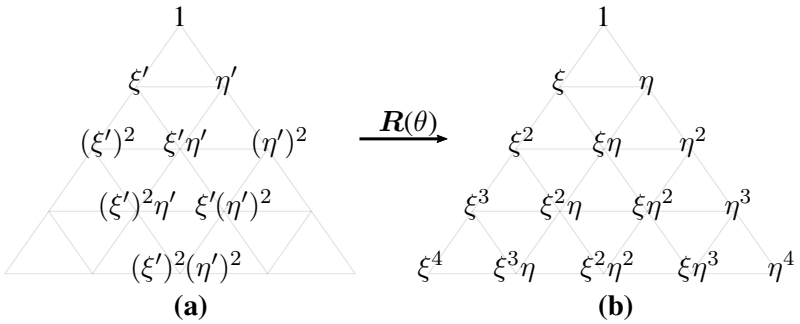


Figure 3.2: The visualisation of the polynomial basis functions in Pascal's triangle for the case $n = 2$ (a) before and (b) after applying $\boldsymbol{R}(\theta)$ to a polynomial in the scheme of Evrard *et al.* [16].

This leads to the method of Jibben *et al.* [29] which, when generalised (see sections to follow), expresses a polynomial of order n as a linear combination of the elements in the set

$$\mathcal{B}_2 = \big\{\xi^r\eta^s |\; r,s \in \{0,1,...,n\},\; r+s \leq n\big\}.$$

Unlike $\mathcal{B}_1$, the set of all bivariate polynomials that can be expressed as a linear combination of the elements in $\mathcal{B}_2$ is closed under rotations described by $\boldsymbol{R}(\theta)$. This can be demonstrated by applying $\boldsymbol{R}(\theta)$ and assuming the same approach as that in Eqs. (3.2) and (3.3):

$$P_n(\xi',\eta') = \sum_{s=0}^{n}\sum_{r=0}^{n-s} c'_{rs}(\xi')^r(\eta')^s \iff P_n(\xi,\eta) = \sum_{s=0}^{n}\sum_{r=0}^{n-s} c_{rs}\xi^r\eta^s. \tag{3.4}$$

where $c_{rs} \equiv c_{rs}\big(\theta,\; \big\{c'_{ij} \mid i \in \{0,1,...,n\},\; j \in \{0,1,...,n-i\}\big\}\big)$. The associated visualisation of the polynomial basis functions in Pascal's triangle for the case $n=2$ is illustrated below.

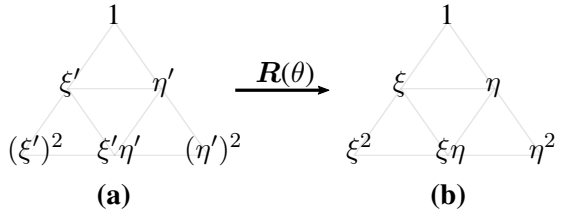


Figure 3.3: The visualisation of the polynomial basis functions in Pascal's triangle for the case $n = 2$ (a) before and (b) after applying $\boldsymbol{R}(\theta)$ to a polynomial in the scheme of Jibben *et al.* [29].

The above demonstrates that pure rotations about the ψ-axis of a polynomial in $\mathcal{B}_2$ can be described by some other polynomial in $\mathcal{B}_2$. This, in turn, implies that a fixed set of data entities (heights, in this case) can be fitted by a polynomial in $\mathcal{B}_2$ such that the resulting mean curvature at $\xi = \eta = 0$ is invariant with respect to rotations of the local coordinate system about the ψ-axis.

It is instructive to reiterate here that the above-mentioned property primarily has utility in the context of non-orthogonal structured grids in $\mathbb{R}^3$. Therefore, since this chapter specifically considers Cartesian grids, Section 3.3 (which details the least squares fitting procedure in $\mathbb{R}^3$) will still specialise to the case where the local coordinate axes are parallel to the edges of the Cartesian grid. The generalised non-orthogonal case, which involves a significant increase in complexity, will be considered in Chapter 7.

In terms of volume conservation, the core difference between the methods of Evrard *et al.* [16] and Jibben *et al.* [29] is that the former enforces volume conservation in each height

column, while the latter enforces volume conservation over the entire stencil of columns (*i.e.* as a collective). Nonetheless, the latter can be adapted to enforce volume conservation in any given column in the stencil by appropriately adjusting the position of the fitted polynomial in the η-direction in $\mathbb{R}^2$ and the ψ-direction in $\mathbb{R}^3$ (as detailed in the sections to follow). This will, of course, have no impact on the partial derivatives of the polynomial, and thus the normal and curvature of the same.

Sections 3.2 and 3.3 respectively introduce the philosophy of the polynomial fitting procedure in $\mathbb{R}^2$ and $\mathbb{R}^3$ by specialising the work of Jibben *et al.* [29] to uniform and non-uniform Cartesian grids. This procedure is later extended to non-orthogonal structured grids in Chapter 7.

3.2 2D Method

This section specialises to 2D, the 3D paraboloid fitting technique proposed by Jibben *et al.* [29], and extends the technique to arbitrary-order polynomial curves. To aid in the understanding of the least squares procedure, the notion of a *height* is now enhanced to that of a *PLIC facet* - the horizontal line segment (red lines in Fig. 3.1) in each column, ζ_p, that lies at a height h_p from the ξ-axis[4]. An n^{th}-order polynomial is fitted in a manner that minimises the signed area bounded between the polynomial itself and at least $(n+1)$ PLIC facets, with respect to the η direction. As depicted in Fig. 3.1(b), this is equivalent to ensuring that the sum of the areas shaded in light red is equal to that of the areas shaded in light blue.

For the remainder of this section, $N = (n+1)$ shall denote the number of polynomial coefficients and $M \geq N$, the number of PLIC facets in the stencil. Consider the n^{th}-order polynomial with unknown coefficients $\boldsymbol{c}=[c_1\ c_2\ ...\ c_N]^T$, defined by

$$P_n(\xi) = \sum_{i=1}^{N} c_i \xi^{i-1}. \tag{3.5}$$

As in the previous chapter, the local $\xi\eta$-coordinate system is located such that the required partial derivatives can be evaluated at $\xi = 0$. The signed area between the PLIC facets and the polynomial is then minimised by the following functional:

$$E(\boldsymbol{c}) = \sum_{p=1}^{M} \left(\int_{\Omega_p} \Big(P_n(\xi) - h_p \Big)\, d\xi \right)^2, \tag{3.6}$$

where $\Omega_p = [\xi_{p_{min}}, \xi_{p_{max}}]$ is the integration interval for PLIC facet p. Minimising Eq. (3.6) with respect to the polynomial coefficients in $\boldsymbol{c}$ gives

$$\sum_{p=1}^{M} \left(\int_{\Omega_p} \Big(P_n(\xi) - h_p \Big)\, d\xi \right) \left(\int_{\Omega_p} \phi\, d\xi \right) = 0, \tag{3.7}$$

[4]Note that the horizontal line segments depicted in Fig. 3.1 are in fact Simple Line Interface Calculations (SLIC). This terminology is reserved for interface reconstructions whose normals are aligned with one of the coordinate axes. PLIC methods, on the other hand, account for interface normals oriented in arbitrary directions. However, since the cases in Fig. 3.1 are special cases of PLIC, and since this chapter seeks to establish the terminology for the generalised case, the phrase *PLIC facet* shall collectively refer to both PLIC and SLIC in this work.

which needs to be solved for each $\phi \in \left\{\xi^{i-1} |\ i \in \{1,2,...,N\}\right\}$. The integrals $\int_{\Omega_p} \phi\ d\xi$ are denoted as

$$s_{pi} \equiv \int_{\Omega_p} \xi^{i-1}\, d\xi \ = \ \frac{\xi^i_{p_{max}} - \xi^i_{p_{min}}}{i}, \quad i \in \{1,2,...,N\}. \tag{3.8}$$

The above yields the solution of $\boldsymbol{c}$ from the linear system

$$\boldsymbol{A}\boldsymbol{c} = \boldsymbol{b}, \tag{3.9}$$

$$A_{ij} = \sum_{p=1}^{M} s_{pi} s_{pj}, \quad b_i = \sum_{p=1}^{M} s_{pi} s_{p1} h_p, \quad i,j \in \{1,2,...,N\}.$$

The above system can be straightforwardly shown to be equivalent (*i.e.* on Cartesian grids) to the system of Evrard *et al.* [16] (see Eq. (2.8)) when $M = N$.

As mentioned previously, this method enforces volume conservation over the entire stencil, and not in every column. For some PLIC facet $q \in \{1,2,...,M\}$, suppose that ζ_q is the column (the central column in this work, *i.e.* $q = 2$ in Fig. 3.1) in which volume conservation is to be enforced[5]. To do so, the coefficient, $\overline{c}_1$, of the vertically offset polynomial

$$\overline{P}_n(\xi) = P_n(\xi) - c_1 + \overline{c}_1 \tag{3.10}$$

must be computed such that

$$\begin{aligned}
&\int_{\Omega_q} \overline{P}_n(\xi) d\xi \ = \int_{\Omega_q} h_q\, d\xi \\
\Rightarrow\ &\int_{\Omega_q} \big(P_n(\xi) - c_1 + \overline{c}_1\big) d\xi \ = \int_{\Omega_q} h_q\, d\xi \\
\Rightarrow\ &\overline{c}_1 s_{q1} + \sum_{i=2}^{N} c_i s_{qi} = h_q s_{q1} \\
\Rightarrow\ &\overline{c}_1 = h_q - \frac{1}{s_{q1}} \sum_{i=2}^{N} c_i s_{qi}.
\end{aligned} \tag{3.11}$$

The first and second partial derivatives of $\overline{P}_n(\xi)$ can then be computed at $\xi = 0$ as

$$\frac{\partial \overline{P}_n}{\partial \xi} = c_2, \ \frac{\partial^2 \overline{P}_n}{\partial \xi^2} = 2c_3.$$

Using Eqs. (2.1) and (2.2), the above expressions lead to the following equations for the unit normal and curvature of $\overline{P}_n(\xi)$ at $\xi = 0$:

$$\begin{aligned}
\boldsymbol{n} &= \frac{1}{\sqrt{1+c_2^2}} \begin{bmatrix} -c_2 \\ 1 \end{bmatrix} \\
\kappa &= \frac{2c_3}{(1+c_2^2)^{3/2}}.
\end{aligned} \tag{3.12}$$

[5] As mentioned previously, this would have no bearing on the derivatives (and thus the curvature) of the interface.

Similar to Evrard *et al.* [16], the above equations can be approximated to n^{th}-order accuracy (when evaluated at the central column) on uniform and $(n-1)^{\text{th}}$-order accuracy on non-uniform Cartesian grids, respectively (see results in Section 3.4).

3.3 3D Method

This section extends the method of Jibben *et al.* [29] to arbitrary-order bivariate polynomial surfaces in $\mathbb{R}^3$. As before, a local $\xi\eta\psi$-coordinate system is located such that the required partial derivatives can be evaluated at $\xi = \eta = 0$. For 3D Cartesian grids, the *PLIC facets* are the horizontal rectangles bounded by each column, ζ_p, that lie at a height h_p from the $\xi\eta$-plane (see Fig. 3.1 for 2D analogy).

For the remainder of this section, $N = (n+1)(n+2)/2$, shall denote the number of polynomial coefficients and, $M \geq N$, the number of PLIC facets in the stencil (thus not equivalent to Evrard *et al.* [16]). Consider the n^{th}-order bivariate polynomial with unknown coefficients $\boldsymbol{c} = [c_1\ c_2\ ...\ c_N]^T$:

$$P_n(\xi,\eta) = \sum_{s=0}^{n}\sum_{r=0}^{n-s} c_i \xi^r \eta^s, \tag{3.13}$$

$$i = s + 1 + \frac{1}{2}(r+s)(r+s+1) \in \{1,2,...,N\}. \tag{3.14}$$

The signed volume between the PLIC facets and the polynomial can be minimised by the following functional:

$$E(\boldsymbol{c}) = \sum_{p=1}^{M}\left(\int_{\Omega_p}\Big(P_n(\xi,\eta) - h_p\Big)\, d\xi d\eta\right)^2. \tag{3.15}$$

Here, $\Omega_p = [\xi_{p_{min}}, \xi_{p_{max}}]\times[\eta_{p_{min}}, \eta_{p_{max}}]$ is the integration domain for PLIC facet p. Minimising the above with respect to the polynomial coefficients $\boldsymbol{c}$ gives

$$\sum_{p=1}^{M}\left(\int_{\Omega_p}\Big(P_n(\xi) - h_p\Big)\, d\xi d\eta\right)\left(\int_{\Omega_p}\phi\, d\xi d\eta\right) = 0, \tag{3.16}$$

which needs to be solved for each $\phi \in \big\{\xi^r\eta^s |\ r,s \in \{0,1,...,n\},\ r+s \leq n\big\}$ to find $\boldsymbol{c}$. The double integrals $\int_{\Omega_p}\phi\, d\xi d\eta$ may be denoted as

$$s_{pi} \equiv \int_{\Omega_p}\xi^r\eta^s\, d\xi d\eta \ = \ \frac{(\xi_{p_{max}}^{r+1} - \xi_{p_{min}}^{r+1})(\eta_{p_{max}}^{s+1} - \eta_{p_{min}}^{s+1})}{(r+1)(s+1)}, \tag{3.17}$$

where i is computed as per Eq. (3.14). Similar to Eq. (3.9), this leads to the following linear system to solve for the coefficients $\boldsymbol{c}$:

$$\boldsymbol{A}\boldsymbol{c} = \boldsymbol{b}, \tag{3.18}$$

$$A_{ij} = \sum_{p=1}^{M} s_{pi}s_{pj}, \quad b_i = \sum_{p=1}^{M} s_{pi}s_{p1}h_p, \quad i,j \in \{1,2,...,N\}.$$

Lastly, for some PLIC facet $q \in \{1, 2, ..., M\}$ in the stencil, volume conservation can again be enforced in the column ζ_q if required, by computing the vertically offset polynomial

$$\overline{P}_n(\xi, \eta) = P_n(\xi, \eta) - c_1 + \bar{c}_1. \tag{3.19}$$

The coefficient $\bar{c}_1$ is computed by applying the same procedure as that in Eq. (3.11), which gives

$$\int_{\Omega_q} \overline{P}_n(\xi, \eta)\, d\xi d\eta \;=\; \int_{\Omega_q} h_q\, d\xi d\eta \quad \Longleftrightarrow \quad \bar{c}_1 = h_q - \frac{1}{s_{q1}} \sum_{i=2}^{N} c_i s_{qi}. \tag{3.20}$$

The first- and second-order partial derivatives of $\overline{P}_n(\xi, \eta)$ can then be computed at $\xi = \eta = 0$ as

$$\frac{\partial \overline{P}_n}{\partial \xi} = c_{_2}, \;\; \frac{\partial \overline{P}_n}{\partial \eta} = c_{_3},$$
$$\frac{\partial^2 \overline{P}_n}{\partial \xi^2} = 2c_{_4}, \;\; \frac{\partial^2 \overline{P}_n}{\partial \xi \partial \eta} = c_{_5}, \;\; \frac{\partial^2 \overline{P}_n}{\partial \eta^2} = 2c_{_6}.$$

Using Eqs. (2.4) and (2.5), the above expressions lead to the following equations for the unit normal and mean curvature of $\overline{P}_n(\xi)$ at $\xi = \eta = 0$:

$$\begin{aligned} \boldsymbol{n} \;&=\; \frac{1}{\sqrt{1 + c_{_2}^2 + c_{_3}^2}} \begin{bmatrix} -c_{_2} \\ -c_{_3} \\ 1 \end{bmatrix}, \\ \kappa \;&=\; \frac{2\big(c_{_4}(1 + c_{_3}^2) - c_{_2} c_{_3} c_{_5} + c_{_6}(1 + c_{_2}^2)\big)}{(1 + c_{_2}^2 + c_{_3}^2)^{3/2}}. \end{aligned} \tag{3.21}$$

Once again, the above equations can be approximated to n^{th}-order accuracy (when evaluated at the central column) on uniform and $(n-1)^{\text{th}}$-order accuracy on non-uniform Cartesian grids, respectively.

It is worth briefly digressing to mention that matrix $\boldsymbol{A}$ in Eqs. (3.9) and (3.18) can be constructed from Vandermonde matrices [5], which are notoriously susceptible to the early onset (as the grid is refined) of ill-conditioning. This is exacerbated as the order n of the polynomial fit is increased, as demonstrated by Waidyaratne [64]. In this work, this imposes a lower bound of 3.0e-3 on the grid spacing used in all test cases (Evrard *et al.* [16] enable further grid refinement *via* quad-precision floating-point accuracy). A steady increase in the computed curvature error was observed with further refinement of this lower bound. Note, however, that this was not considered prohibitive as all curvature results obtained (for the selected ranges of grid spacing values in this work) were in the asymptotic region of convergence. Nonetheless, a tractable solution to address the ill-conditioning of $\boldsymbol{A}$ ought to be of future interest. Brubeck *et al.* [5] recently proposed a solution for normalising Vandermonde matrices arising in coordinate-based least squares problems by employing an iterative Arnoldi orthogonalisation procedure. While the applicability of their solution to the present work was not investigated, this is left for future work. End of digression.

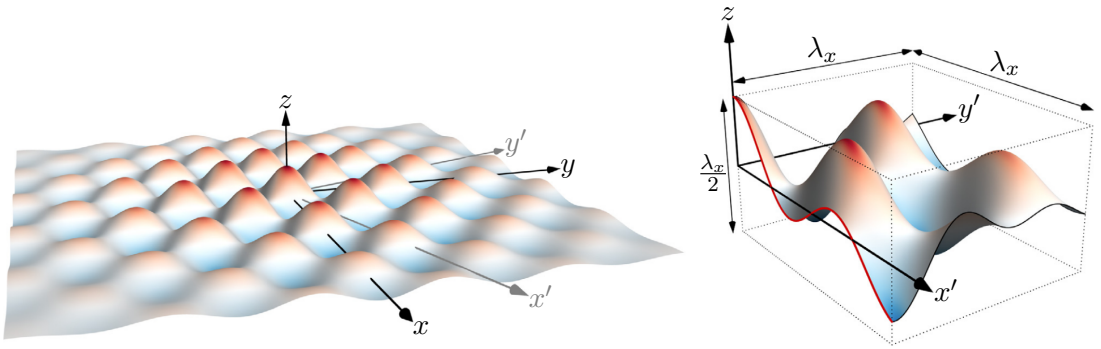
The previously detailed least squares reconstruction methods in $\mathbb{R}^2$ and $\mathbb{R}^3$ are next evaluated numerically in terms of curvature accuracy.

3.4 Results

The following equation of the test surface from Evrard *et al.* [16] is employed to validate the implementation of the least squares polynomial fitting scheme detailed in this chapter.

$$z = \frac{\lambda_x}{4}\cos\left(\frac{2\pi x}{\lambda_x}\right)\cos\left(\frac{2\pi y}{\lambda_y}\right)\exp\left(-\frac{x^2}{4\lambda_x^2}\right)\exp\left(-\frac{y^2}{4\lambda_y^2}\right). \tag{3.22}$$

Fig. 3.4(a) illustrates this surface for the case where $\lambda_x = \lambda_y$. The surface colouring corresponds to the local mean curvature normalised by the maximum mean curvature, κ_{max} (which occurs at $x = y = 0$). The test case is performed in a $x'y'z$ reference frame, resulting from the rotation of the xyz reference frame about the z-axis by a random angle, $\theta \in [0, 2\pi]$. Similar to Evrard *et al.* [16], the parameters in this test case are set as $\lambda_x = 1$ and $\lambda_y = 2/3$, and the two- and three-dimensional domains considered are $\mathcal{D}_2^\theta = \{x' \in \mathbb{R} \mid 0 \leq x' \leq \lambda_x\}$ and $\mathcal{D}_3^\theta = \{(x', y') \in \mathbb{R}^2 \mid 0 \leq x' \leq \lambda_x,\ 0 \leq y' \leq \lambda_x\}$, respectively. Therefore in $\mathbb{R}^2$, the test curve is the intersection of the surface with the $x'z$-plane. Fig. 3.4(b) depicts an example test curve and surface for $\theta = \pi/10$, with the curve shown in red.



(a) (b)

Figure 3.4: (a) The surface described by Eq. (3.22) for the case of $\lambda_x = \lambda_y$, with the colouring representing the mean curvature. (b) An example test curve (red line) and surface in the domains $\mathcal{D}_2^\theta$ and $\mathcal{D}_3^\theta$, with $\lambda_x = 1$, $\lambda_y = 2/3$, and $\theta = \pi/10$. (This is an adaptation of a figure from Evrard *et al.* [16])

The least squares scheme is here tested solely in terms of curvature computation, due to its relevance to this thesis. Additionally, this is only performed in the central column of uniform and non-uniform stencils. Note that Evrard *et al.* [16] additionally account for curvature computation in — (i) columns on stencil boundaries, and in (ii) stencils comprising non-adjacent columns. However, this is beyond the scope of the present work. In the results to follow, volume may be conserved to machine-precision in the central column by applying the offsetting procedures in Eqs. (3.11) and (3.20).

Similar to Evrard *et al.* [16], the curvature is numerically evaluated at $1{\times}10^6$ points resulting from the randomisation of the angle θ, followed by that of points in the domains $\mathcal{D}_2^\theta$ and $\mathcal{D}_3^\theta$. For each random point, $\boldsymbol{x}_r \in \{\boldsymbol{x} \in \mathcal{D}_d^\theta \mid \theta \in [0, 2\pi]\}$ in the d-dimensional case, the central column of the stencil is centred on $\boldsymbol{x}_r$, as depicted in Figs. 3.5 and 3.7. The required heights in each column are computed using Eqs. (2.3) and (2.6), with the integrals evaluated using fifth-order Gauss-Legendre quadrature formulae applied to Eq. (3.22). The numerically computed

curvature is then compared to the analytical curvature evaluated at $\boldsymbol{x}_r$. The latter is computed *via* the procedure detailed in Appendix A for the 2D case, and using Eq. (2.5) for the 3D case.

Next, the curvature convergence properties of the least squares fitting scheme are analysed in Section 3.4.1 for uniform and non-uniform grids in $\mathbb{R}^2$ and $\mathbb{R}^3$. Section 3.4.2 then analyses the sensitivity of the convergence of curvature in relation to grid non-uniformity.

3.4.1 Curvature Grid Convergence Study

2D Case

The two-dimensional test case considers two stencil configurations—(i) a uniform stencil, and (ii) a non-uniform stencil with grid stretching factor $\beta_x = 1.2$. Polynomial curves of order $n \in \{2, 4, 6, 8\}$ are fitted to $(n + 1)$ columns. The column widths are sized in relation to the central column width, Δx, and the stencil is repeatedly refined such that $\Delta x \in [0.003, 0.1]$. The projections of the aforementioned stencils onto the x'-axis are illustrated in Fig. 3.5 for the case $n = 4$, with the central column highlighted in red.

Figure 3.5: Projection of a five-column stencil onto the x'-axis with the central column highlighted in red—(a) uniform stencil, (b) non-uniform stencil with stretch factor $\beta_x = 1.2$.

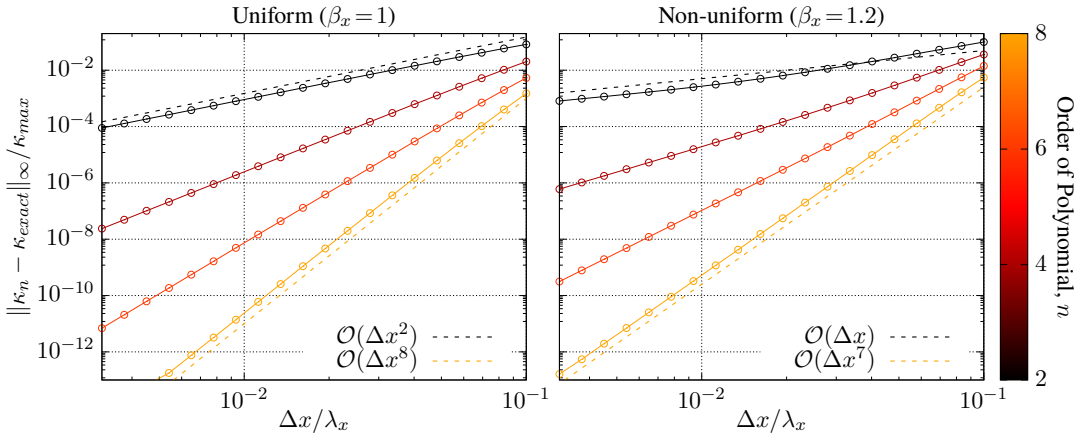


Figure 3.6: The 2D curvature error convergence plots for the selected uniform and non-uniform stencil configurations.

Fig. 3.6 plots the maximum absolute error of curvature for the two stencil configurations, normalised by κ_{max}. For a polynomial fit of order n, the order of accuracy is $\mathcal{O}(\Delta x^n)$ on the uniform stencil and tends to $\mathcal{O}(\Delta x^{n-1})$ on the non-uniform stencil. As was pointed out in the previous chapter, when an n^{th}-order curve is fitted to exactly $(n + 1)$ columns, the least squares scheme is mathematically equivalent to the scheme of Evrard *et al.* [16]. Next, the associated 3D test case is analysed.

3D Case

The three-dimensional test case considers — (i) a uniform stencil whose columns have square cross-sections, and (ii) a non-uniform stencil with stretch factors $\beta_x = 1.1$ and $\beta_y = 1.2$ in the x'- and y'-directions, respectively. For the latter, the length and width of the central column are Δx and $\Delta y = \Delta x/2$, respectively. The range of Δx values considered is the same as that in the 2D case. Polynomial surfaces of order $n \in \{2, 4, 6, 8\}$ are fitted to stencils of $(n+1)\times(n+1)$ columns. The projections of the two stencil configurations onto the $x'y'$-plane are illustrated in Fig. 3.7 for the case $n = 4$, with the central column highlighted in red.

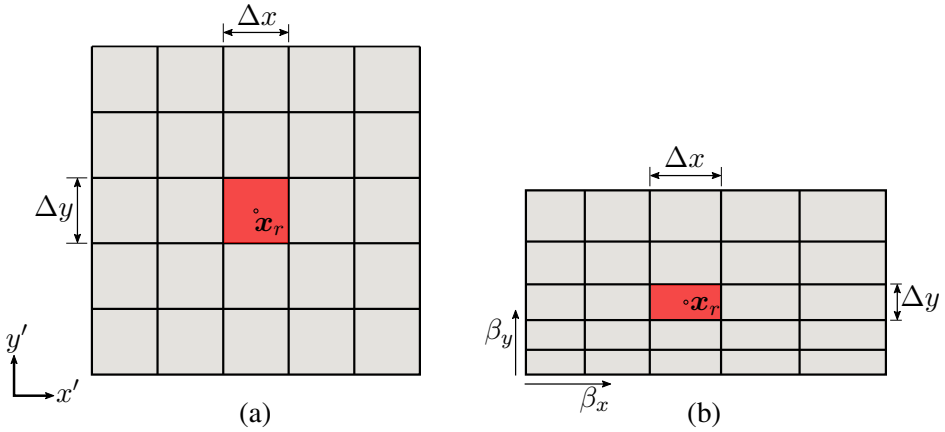


Figure 3.7: Projection of a 5×5-column stencil onto the $x'y'$-plane, with the central column highlighted in red — (a) uniform stencil with $\Delta y = \Delta x$, (b) non-uniform stencil with $\Delta y = \Delta x/2$ and stretch factors $\beta_x = 1.1$ and $\beta_y = 1.2$ in the x'- and y'-directions, respectively.

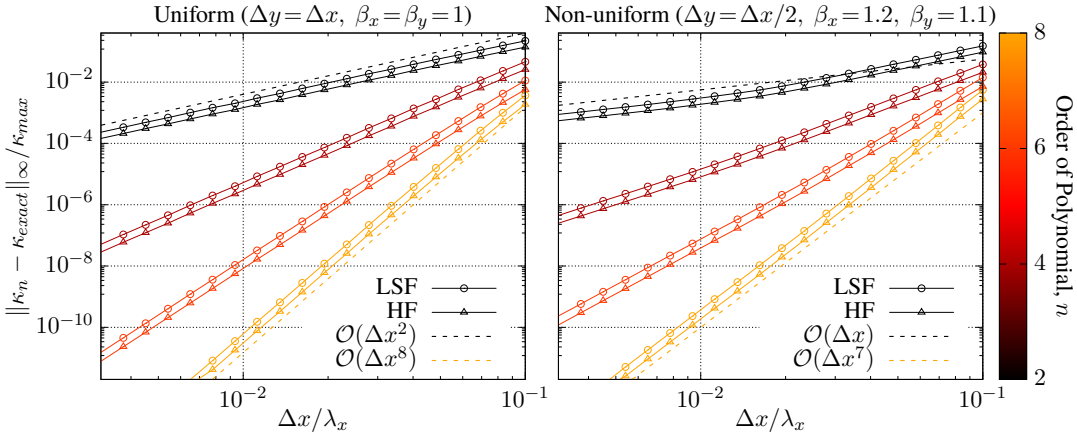


Figure 3.8: The 3D curvature error convergence plots for the selected uniform (left) and non-uniform (right) stencil configurations.

Fig. 3.8 shows the comparison between the least squares fitting scheme (denoted as LSF) and the HF method of Evrard *et al.* [16], in terms of the maximum absolute error of curvature normalised by κ_{max}. As in the 2D case, the order of accuracy is $\mathcal{O}(\Delta x^n)$ on the uniform stencil and tends to $\mathcal{O}(\Delta x^{n-1})$ on the non-uniform stencil. The reduction in curvature accuracy of LSF compared to HF (as a percentage of κ_{max}) is plotted in Fig. 3.9 for both stencil configurations. This slight reduction in accuracy would be expected and is not seen as prohibitive, given that the LSF method is well suited to non-orthogonal grids (see Chapter 7).

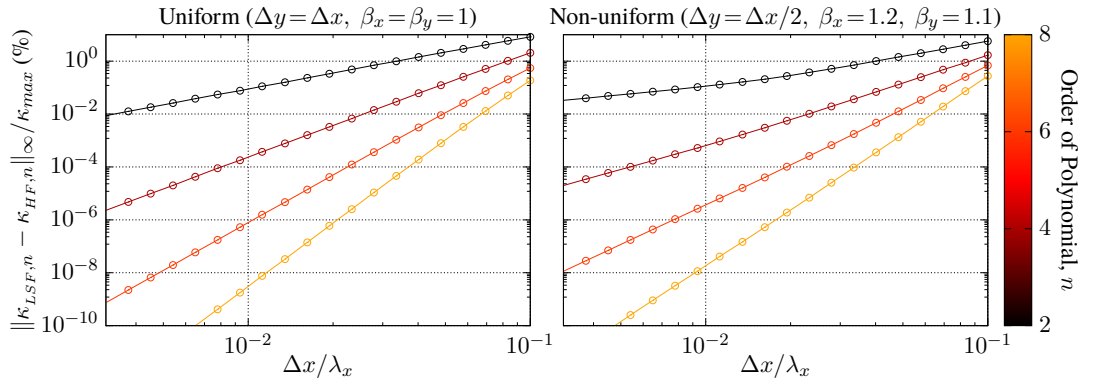


Figure 3.9: The reduction in 3D curvature accuracy of the least squares fitting (LSF) scheme in comparison to the HF method of Evrard *et al.* [16], as a percentage of κ_{max}.

3.4.2 Grid Non-uniformity *vs* Order of Accuracy

It is clear from Figs. 3.6(b) and 3.8(b) that grid non-uniformity causes a loss in accuracy of one order. Considering the cases $n \in \{2, 4\}$, Fig. 3.10 plots the order of accuracy (as computed by the gradient of the above plots) against grid size for a variety of grid stretching factors, $\beta_x \in [1.0, 1.2]$, and $\beta_y = \beta_x$ in the case of 3D. As shown, the order of accuracy initially increases as Δx decreases and then asymptotes to $\mathcal{O}(\Delta x^{n-1})$.

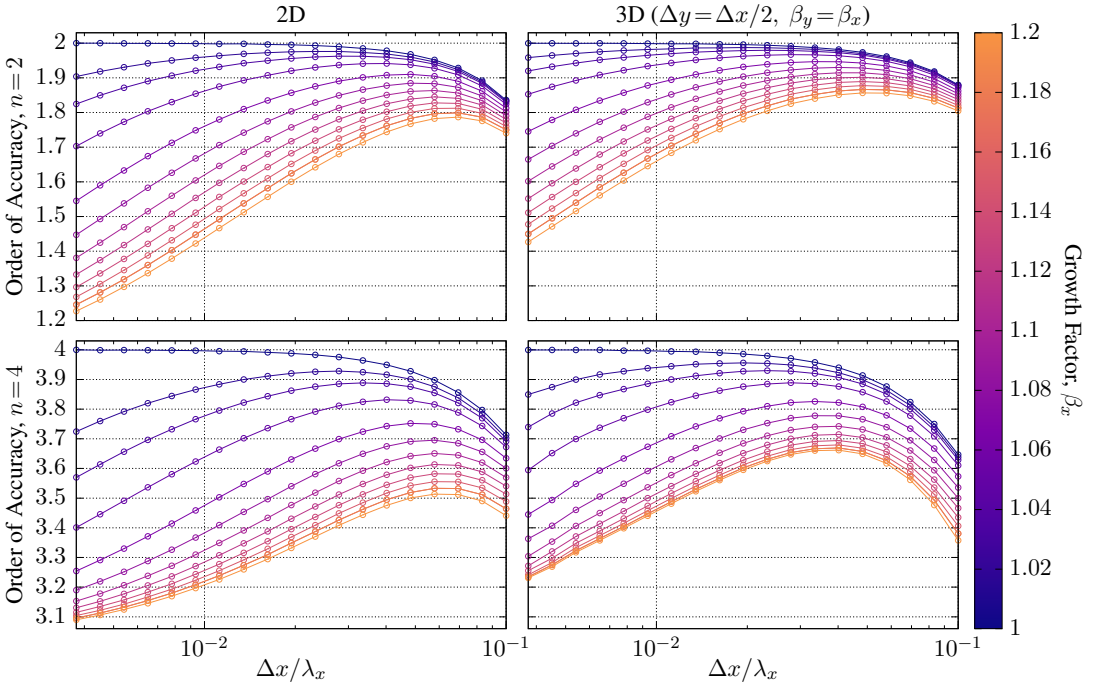


Figure 3.10: The apparent order of accuracy of curvature as a function of varying stretch factors, $\beta_x \in [1.0, 1.2]$, for cases $n \in \{2, 4\}$ in 2D and 3D.

3.5 Closure

This chapter focussed on the least squares polynomial fitting procedure of Jibben *et al.* [29] that is employed in this work. The scheme was generalised to polynomials of arbitrary order and specialised to Cartesian grids. Although a slight reduction in accuracy was observed relative to the exact fitting procedure in $\mathbb{R}^3$, the least squares fitting procedure was selected as it provides more versatility in the context of non-orthogonal structured grids. Further, in $\mathbb{R}^3$, the least squares procedure was demonstrated to yield a consistent mean curvature regardless of the orientation of the local coordinate system about the vertical axis. This is a key requirement when generalising the scheme to non-orthogonal structured grids. During this generalisation, a key consideration will be the orientation of the PLIC facets in each column.

Chapter 4

Interface Column Construction

4.1 Background

As is the case with HF on Cartesian grids, it is challenging to compute curvature in regions of the interface where the directions of the interface columns switch. On non-orthogonal structured grids, this challenge is exacerbated by the fact that the grid spacing in one direction may vary significantly from that of another (anisotropic regions). This results in the interface being represented more accurately by PLIC facets constructed in interface columns oriented in the direction with smaller column widths/cross-sections. This discrepancy in accuracy is in turn reflected in an abrupt change in the computed curvature, which may cause spurious surface-tension-induced velocities. Therefore, on non-orthogonal structured grids, a region of overlapping interface columns (see Figs. 4.1 and 4.2) is required. This is to enable curvature to be appropriately interpolated (see Section 8.5) when the interface transitions between two (or three in $\mathbb{R}^3$) contiguous regions of columns. The coarsest grids in this work were chosen to ensure at least two overlapping columns in this region.

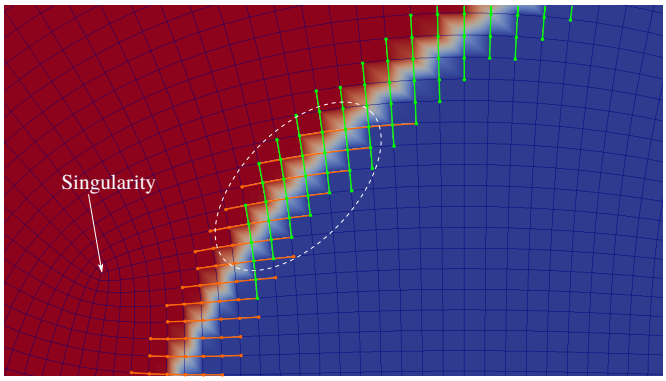
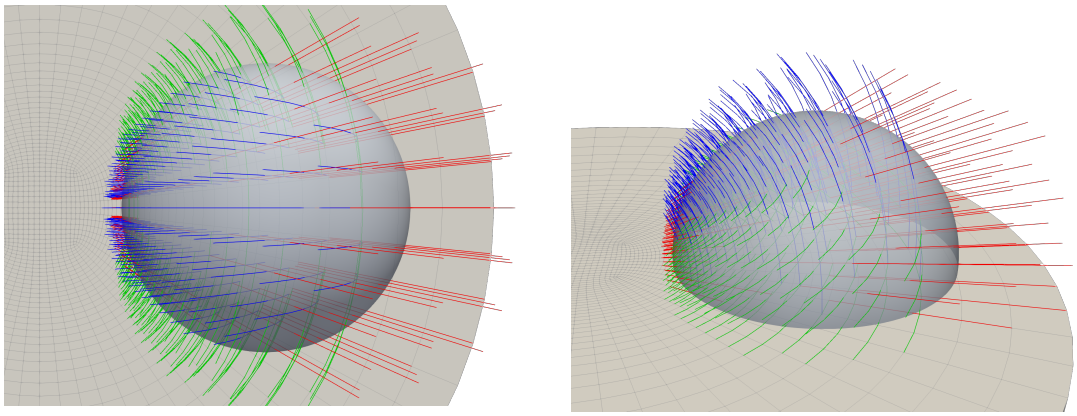


Figure 4.1: Interface columns constructed over a portion of a circular interface. The white dashed line demarcates a region of overlapping columns.

The first step in the proposed interface column construction procedure is to identify suitable directions for column construction at each interface vertex. Note that, given a volume fraction field α, a vertex i is considered to be an *interface vertex* if $\alpha_i < 0.5$ and it is connected to at least one other vertex j with $\alpha_j \geq 0.5$. Considering that each interior vertex of a general d-dimensional structured grid is connected to $2d$ other vertices, at least 1 and at most d columns can be constructed at an interface vertex. The exception here is at *singularity* vertices (see Figs.

1.1 and 4.1), which are connected to less than $2d$ vertices in the grids employed in this work. The handling of such singularities is beyond the scope of this work.



(a) **(b)**

Figure 4.2: A portion of a spherical interface showing the constructed columns, with regions of overlap. The cutaway of the underlying 3D grid is shown with the gridlines displayed in light grey.

The direction in which a column is constructed depends on the computed interface normal at a given interface vertex. For this purpose, a suitably accurate representation of the interface normal is necessary. In this work, the normal at an interface vertex i is initially approximated by

$$\boldsymbol{n}_i = \frac{\nabla \tilde{\alpha}_i}{\|\nabla \tilde{\alpha}_i\|}, \tag{4.1}$$

where $\tilde{\alpha}$ is the smoothed volume fraction field computed as per Heyns [23] by evolving the following equation in pseudo-time:

$$\begin{aligned} \tilde{\alpha}_i^0 &= \alpha_i^m \\ \tilde{\alpha}_i^{\tau+1} &= \tilde{\alpha}_i^\tau + \Delta\tau \nabla\cdot\nabla\tilde{\alpha}_i^\tau, \end{aligned}$$

where α_i^m is the volume fraction at time-step m, and the above equation is iterated twice in this work. The spatial discretisation of the last term above is detailed in Section 8.2. On non-orthogonal grids, Eq. (4.1) leads to insufficient accuracy of the interface normals to ensure regions of contiguous columns. Therefore, in addition to improving the overall accuracy of the curvature computation, the PLIC-based iterative normal refinement procedure in this work also serves to obtain a more accurate interface normal for inferring directions for column construction. As a result, the present chapter details the procedure for constructing columns at the k^{th} iteration of the iterative normal refinement procedure (see Chapter 6) based on the interface normals, $\boldsymbol{n}_i^k$. The normals, $\boldsymbol{n}_i^0$, are seeded by the smoothed volume fraction gradient (as per Eq. (4.1)). A generalised definition for $\boldsymbol{n}_i^k$ is introduced shortly.

To date, schemes that have employed the HF method (see Chapter 2) have either directly [10, 16–18, 46, 53] or indirectly [28, 42, 46] relied on Cartesian grids, where each column is aligned with one of the primary coordinate axes. However, in the general case of non-orthogonal structured grids, interface columns lie along arbitrary directions. In the d-dimensional case, this requires having to compute angles between each interface normal $\boldsymbol{n}_i^k$ and the d edges between

which it is bounded. Figs. 4.3 and 4.4 illustrate these cases, with the bounding edges denoted by $\overline{ij_r}, r \in [1, d]$. It is instructive to point out that at iteration $k = 0$ of the interface normal refinement procedure, there are as many interface normals as there are interface vertices. However, for $k > 0$, there are as many interface normals as there are columns constructed after iteration $(k-1)$. This is because, as will be shown in the next chapter, each normal is used to conservatively construct a PLIC facet in its associated interface column. Each interface vertex i simply inherits the set of iteratively refined normals associated to all columns connected to it. Therefore, in the d-dimensional case, the generalised definition of the interface normal employed in this work can be summarised as

$$\boldsymbol{n}_i^k = \begin{cases} \dfrac{\nabla \tilde{\alpha}_i}{\|\nabla \tilde{\alpha}_i\|}, & \text{if } k = 0 \\ -\boldsymbol{n}, \text{ where } \boldsymbol{n} \in \{\boldsymbol{n}_{p_r}^k \mid r \in \{1, 2, ..., M_i\},\ M_i \leq d\}, & \text{otherwise.} \end{cases} \tag{4.2}$$

Here, $\boldsymbol{n}_{p_r}^k$ is the normal of the PLIC facet p_r associated to column ζ_r, which is one of M_i columns ($M_i > 1$ when i lies in the overlapping region) constructed at vertex i after iteration $(k-1)$. Note that when $k > 0$, each of the M_i definitions of $\boldsymbol{n}_i^k$ are separately interrogated to flag the appropriate directions for column construction, based on the procedures detailed in Sections 4.2 and 4.3 for $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively. The final set of directions for column construction is then taken to be the union of all flagged directions for each definition of $\boldsymbol{n}_i^k$.

4.2 2D Column Construction

Fig. 4.3 depicts the general scenario for the case in $\mathbb{R}^2$. The position of the interface normal within the region bounded between edges $\overline{ij_1}$ and $\overline{ij_2}$ determines whether a column is to be constructed in the direction of either one or both of the edges.

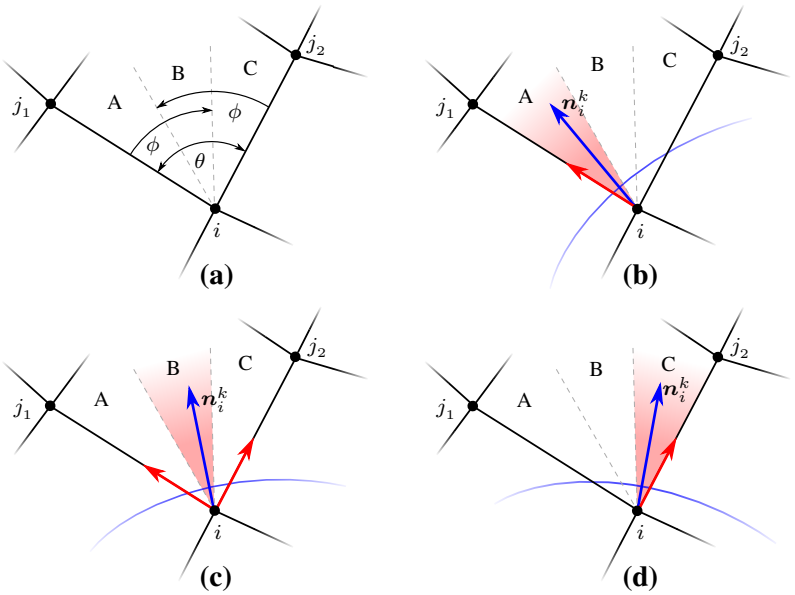


Figure 4.3: Sector divisions between the two edges bounding an interface normal, for determining the directions for column construction at an interface vertex i in $\mathbb{R}^2$. (a) Angles associated with each sector. The normal lies in — (b) sector A resulting in a single column, (c) sector B resulting in two overlapping columns, or (d) sector C resulting in single a column.

To ensure a region of overlapping columns where required (as seen in Fig. 4.1(a)), the angle between the bounding edges is divided into three sectors (labelled as A, B, and C in Fig. 4.3). An interface vertex lies in an overlapping region if its normal lies in sector B. Two columns are constructed at all vertices in an overlapping region, and only one otherwise (the red arrows in the figure signify the directions in which columns are constructed). Fig. 4.3(a) illustrates that the sectors are defined by an angle $\phi = \frac{1}{2}(\theta + \theta_{_B})$, where $\theta_{_B} \in [0, \frac{\pi}{4}]$ is a user-defined angle prescribed for sector B in the figure. In this work, the prescribed angle is $\theta_{_B} = \frac{\pi}{6}$ (to ensure a minimum of two overlapping columns for the coarsest grid). For a circular interface on a Cartesian grid, $\theta_{_B}$ is the angle subtended by the overlapping region and the centre of the circle. Algorithm 1 outlines the pseudo-code for the above procedure.

Algorithm 1 Column construction at an interface vertex for a given interface normal in $\mathbb{R}^2$.

Input: Interface vertex i, normal $\boldsymbol{n}_i^k$, column overlap angle $\theta_{_B}$, volume fraction tolerance α_{tol}
Output: The array of columns, Z, that are to be constructed at interface vertex i.

1: **procedure** ConstructColumns2D($i, \boldsymbol{n}_i^k, \theta_{_B}, \alpha_{tol}$)
2: **set** $E \leftarrow$ the set of two edges connected to vertex i between which $\boldsymbol{n}_i^k$ is bounded. $E[0]$ is the edge CCW from $\boldsymbol{n}_i$, and $E[1]$ is CW from $\boldsymbol{n}_i^k$.
3: **set** $\theta \leftarrow$ the angle between the two edges in E.
4: **set** $\theta_i \leftarrow$ the angle between the edge $E[0]$ and $\boldsymbol{n}_i^k$.
5: **set** $\phi \leftarrow \frac{1}{2}(\theta + \theta_{_B})$.
6: $\boldsymbol{D} = []$ // The array of edge directions in which columns are to be constructed.
7: **if** $\theta_i < \phi$ **then**
8: **add** $\boldsymbol{D} \leftarrow$ outward-pointing direction vector (w.r.t. i) of edge $E[0]$.
9: **end**
10: **if** $(\theta - \theta_i) < \phi$ **then**
11: **add** $\boldsymbol{D} \leftarrow$ outward-pointing direction vector (w.r.t. i) of edge $E[1]$.
12: **end**
13: $Z = []$ // The array of columns that are to be constructed at vertex i.
14: **for all** $\boldsymbol{d} \in \boldsymbol{D}$ **do**
15: $\zeta = []$ // A column, defined by the array of vertices that it contains.
16: **set** $j \leftarrow i$
17: **while** $\alpha_j < (1 - \alpha_{tol})$ **do** // Traverse the mesh into Fluid 1, starting from vertex i
18: **set** $j \leftarrow$ the vertex connected to j whose outward-pointing direction vector (w.r.t. j) is nearest to $\boldsymbol{d}$.
19: **end**
20: **while** $\alpha_j > \alpha_{tol}$ **do** // Traverse the mesh into Fluid 0, starting from vertex j
21: **add** $\zeta \leftarrow j$
22: **set** $j \leftarrow$ the vertex connected to j whose outward-pointing direction vector (w.r.t. j) is nearest to $-\boldsymbol{d}$.
23: **end**
24: **add** $Z \leftarrow \zeta$
25: **end**
26: **return** Z

Lines 14 - 25 in Algorithm 1 detail the column construction procedure, after identifying the

appropriate column directions. Each column is constructed such that the control volumes on both ends of the column are fully immersed in Fluid 0 or Fluid 1 (to a given volume fraction tolerance, α_{tol}), *i.e.* the associated volume fractions on each end are $\alpha < \alpha_{tol}$ and $\alpha > (1-\alpha_{tol})$, respectively. This tolerance is necessary to account for the smearing of the VOF interface, as a consequence of employing a CICSAM-based [63] VOF advection scheme (see Chapter 8). The prescribed volume fraction tolerance used in this work is $\alpha_{tol} = 1\times10^{-10}$. This typically leads to each column being composed of 3 – 8 control volumes across the interface in all test cases in this work.

4.3 3D Column Construction

Next, consider the equivalent case in $\mathbb{R}^3$, as depicted in Fig. 4.4. The interface normal $\boldsymbol{n}_i^k$ is now bounded in a region defined by the edges $\overline{ij_1}$, $\overline{ij_2}$, $\overline{ij_3}$ (the vector $\boldsymbol{n}_i^k$ has been omitted in the figure to avoid clutter). Figs. 4.4(a) – (c) demonstrate that the three-dimensional extension of the two-dimensional sector-division approach (see Fig. 4.3) yields nine wedge-like divisions (three sets of three wedges). Each set of wedges is centred on the edge, $\overline{ij_r}$ where $r \in \{1,2,3\}$, and is bounded by the three planes described by the edge pairs $(\overline{ij_r},\ \overline{ij_s})$, $(\overline{ij_r},\ \overline{ij_t})$, and $(\overline{ij_s},\ \overline{ij_t})$, where $s,t \in \{1,2,3\}\backslash\{r\}$ and $s \neq t$.

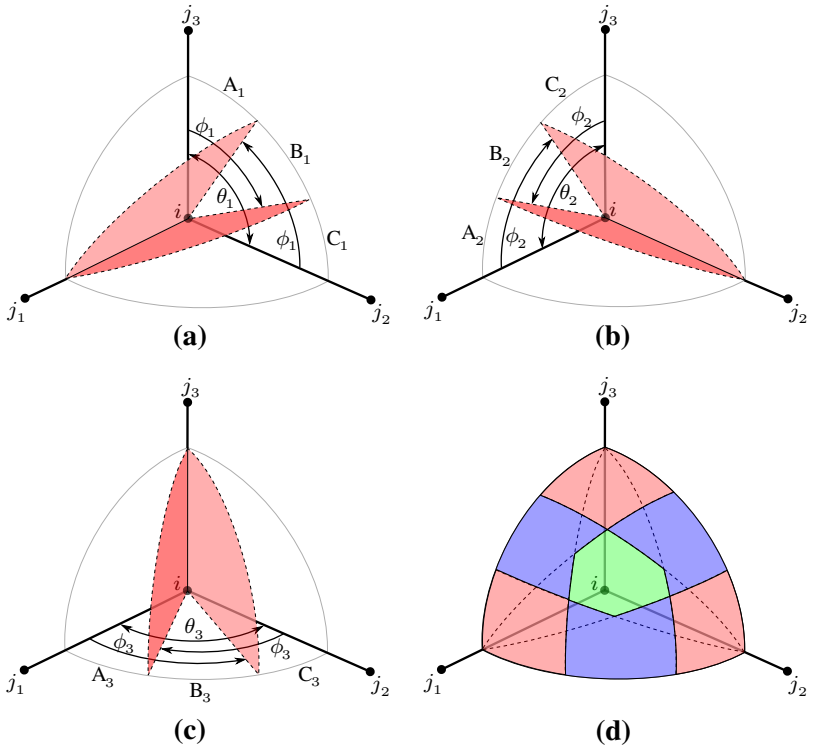


Figure 4.4: Divisions of the region defined by three edges bounding an interface normal at a vertex i in $\mathbb{R}^3$. The wedge divisions between the planes described by the pairs of edges (a) $(\overline{ij_1}, \overline{ij_2})$ and $(\overline{ij_1}, \overline{ij_3})$, (b) $(\overline{ij_1}, \overline{ij_2})$ and $(\overline{ij_2}, \overline{ij_3})$, and (c) $(\overline{ij_1}, \overline{ij_3})$ and $(\overline{ij_2}, \overline{ij_3})$. (d) Locus of all possible points traced by the interface normal in the bounded region, divided into seven possible sub-regions defined by the wedge divisions.

Similar to the case in $\mathbb{R}^2$, the region between the planes $(\overline{ij_r},\ \overline{ij_s})$ and $(\overline{ij_r},\ \overline{ij_t})$ is divided into the wedges A_r, B_r, and C_r. The total angle between the planes is denoted as θ_r, and the associated wedge angles are denoted as $\phi_r = \frac{1}{2}(\theta_r + \theta_{B_r})$, where again $\theta_{B_r} = \frac{\pi}{6}$. Applying the same criteria as in Figs. 4.3(b) – (d), which of the wedges A_r, B_r, or C_r encloses $\boldsymbol{n}_i^k$ determines whether the edges $\overline{ij_s}$ and $\overline{ij_t}$ are to be flagged as potential directions for column construction. Each case for $r \in \{1, 2, 3\}$ can be reduced to an equivalent two-dimensional case (as in Fig. 4.3) by projecting $\boldsymbol{n}_i^k$ onto the $(\overline{ij_s},\ \overline{ij_t})$ plane. Therefore, Algorithm 1 can be straightforwardly adapted to *recursively* perform all three cases in Figs. 4.3(b) – (d). A column is then constructed at vertex i in the direction of each edge that is flagged for column construction exactly twice, for a given definition of $\boldsymbol{n}_p^k$ (as per Eq. (4.2)).

As depicted in Fig. 4.4(d), the outcome of the aforementioned recursive procedure can be visualised as seven possible regions in which $\boldsymbol{n}_i^k$ could be contained. If $\boldsymbol{n}_i^k$ is contained in one of the three light red regions, the edge adjacent to the region is flagged for column construction. If $\boldsymbol{n}_i^k$ is contained in one of the three light blue regions, the two edges adjacent to the region are flagged. Lastly, if $\boldsymbol{n}_i^k$ is contained in the light green region, all three bounding edges are flagged. The columns are once again constructed as per lines 14–25 in Algorithm 1.

4.4 Column Construction Across Parallel Domain Boundaries

In the case of parallel simulations, columns may need to be constructed across the boundaries of parallel domains. For this purpose, the standard master-slave paradigm is adopted to communicate the volume fraction field α and the interface normals $\boldsymbol{n}_i^k$. In this work, each parallel domain is enlarged by six layers of slave vertices, each of which inherits the required data from its associated master vertex in the surrounding encroached domains. This allows for columns to be constructed in full at all non-slave (*i.e.* either master or non-parallel) vertices in a parallel domain, while satisfying the α_{tol} criteria in Algorithm 1. As will be seen in Section 8.5, this serves the additional purpose of guaranteeing a consistent *band* across the interface from which the interface curvature can be queried for discretisation purposes.

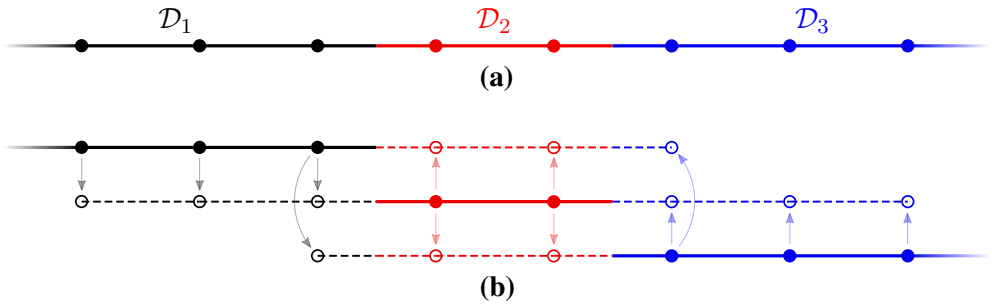


Figure 4.5: One-dimensional schematic of the parallel pathways for communicating α and $\boldsymbol{n}_i^k$ across— (a) three parallel domains demarcated in black, red, and blue, where (b) demonstrates the master-slave communication between each domain.

A one-dimensional analogy of this approach is illustrated in Fig. 4.5, which depicts three parallel domains which are enlarged by three slave vertex layers. The master and slave vertices are depicted by the filled-in and empty circles, respectively.

4.5 Closure

This chapter detailed the interface column construction strategy employed in this work. The adopted strategy generalises the interface-normal-based column construction procedure in Cartesian grids to non-orthogonal structured grids. Since the smoothed volume fraction gradient provides insufficient accuracy for determining the column directions, the procedures detailed here pre-emptively utilise the normals from the iterative normal refinement procedure. Two important considerations in this chapter were the creation of — (i) contiguous regions of columns aligned in the same general direction, and (ii) regions of sufficient overlap, to enable interpolation of the interface curvature where columns switch directions. The next step in the higher-order interface reconstruction procedure is to obtain a PLIC representation of the interface by conservatively reconstructing the VOF field in each constructed column. This is detailed in the next chapter.

Chapter 5

Conservative PLIC Reconstruction

5.1 Background

It was previously shown that the HF method (see Chapter 2) utilises a volume-conservative reconstruction of the interface to obtain a height in each column. The notion of a *height* was later extended in Chapter 3 to that of a *PLIC facet* for the purpose of introducing the least squares polynomial fitting procedure. In the case of Cartesian grids, the orientation (direction of the unit normal) of the PLIC facet is inconsequential, provided it does not intersect the top or bottom edges of the column. This is because a PLIC facet p can be pivoted about its centroid (which remains at height h_p) while conserving the volume, $\mathcal{V}_p$, of the reference fluid (fluid 1 for the remainder of this chapter) below it. Key to this is the constant width/cross-section of the column. A simple geometric proof of this property in $\mathbb{R}^2$ is illustrated in Fig. 5.1. A similar geometric proof can be constructed in $\mathbb{R}^3$ for a column with a rectangular base, with the PLIC facet pivoted about its area centroid (see Chapter 7 for an algebraic proof for columns that are right-angled prisms with arbitrary polygonal bases).

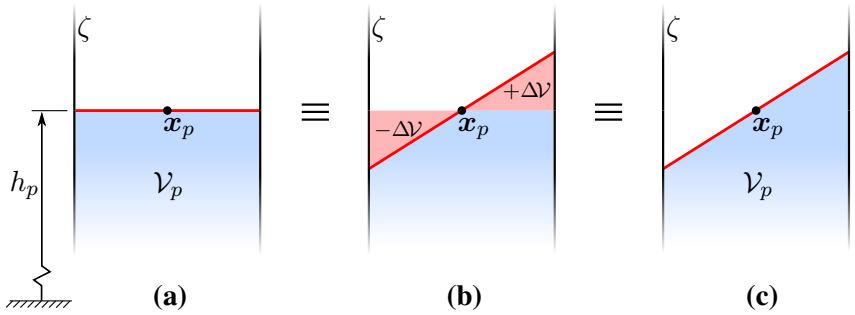


Figure 5.1: Invariance of the conserved volume, $\mathcal{V}_p$, in column ζ when the PLIC facet (red line segment) is arbitrarily pivoted about its centroid, $\boldsymbol{x}_p$. (a) Horizontal PLIC facet at height h_p from a datum. (b) Adding and subtracting equal triangular areas to and from $\mathcal{V}_p$. (c) An arbitrarily oriented PLIC facet with the volume $\mathcal{V}_p$ still conserved.

In the general case of non-orthogonal structured grids (which implies columns of varying cross-sections), the centroid of the PLIC facet is affected by the prescribed interface normal[6]. The reader is here reminded that the PLIC facet normals are therefore iteratively refined (as will be detailed in Chapter 6) to achieve higher-order accuracy of the partial derivatives (and thus the curvature) of the fitted polynomials. As a result, the PLIC facets themselves need to be reconstructed as part of each iteration. For iterations $k > 0$, $\boldsymbol{n}_p^k$ is the normal computed

[6] A PLIC facet, p, is constructed in each column, ζ, based on the reference volume $\mathcal{V}_p$ and the unit normal, $\boldsymbol{n}_p^k$.

after iteration $(k-1)$, and $\boldsymbol{n}_p^0 = -\nabla\tilde{\alpha}_i/\|\nabla\tilde{\alpha}_i\|$ where i is the interface vertex at which ζ was constructed. The total reference fluid volume in ζ that must be conserved when constructing the PLIC facet p is

$$\mathcal{V}_p = \sum_{i\in\zeta} \alpha_i \mathcal{V}_i, \tag{5.1}$$

where α_i and $\mathcal{V}_i$ respectively, are the volume fraction and volume of the median dual-cell of a grid vertex $i \in \zeta$.

The development of robust and efficient tools, for constructing the unique PLIC facet p that conserves $\mathcal{V}_p$ and is orthogonal to the normal $\boldsymbol{n}_p^k$, has received much attention from the PLIC-VOF community in recent years [3, 11, 13, 14, 28, 33–35, 38, 52, 56, 58]. The existing literature for conservative PLIC reconstruction can be broadly categorised into three approaches, based on the geometry of the polytope (general term for polygon/polyhedron) within which the PLIC facet is constructed. These are — (i) convex polytopes [11, 13, 14, 33, 34], (ii) arbitrary polytopes with convex decomposition (CD) (*i.e.* the splitting of a non-convex polytope into triangles/tetrahedra) [3, 28, 58], and (iii) arbitrary polytopes without CD [35]. The common feature in all these approaches is the use of a sweep-plane approach to *sweep* through the polytope/s along a prescribed direction until the plane that exactly conserves $\mathcal{V}_p$ is determined. (For brevity, note that the term *plane* shall generically refer to both *lines* ($\mathbb{R}^2$) and *planes* ($\mathbb{R}^3$) unless the context requires otherwise.) This sweep-plane procedure requires the sorting of all vertices in the polytope/s in the direction of the prescribed normal.

A CD approach is employed in the current work, where the median dual-cells of the grid vertices in ζ are decomposed into triangles/tetrahedra. This is achieved by connecting the grid vertices with the centroids of the surrounding edges and elements (and element faces in $\mathbb{R}^3$). This results in the triangulation/tetrahedronalisation, T_ζ, of column ζ. An example in $\mathbb{R}^2$, of a column and its corresponding triangulation, is depicted in Figs. 5.2(a) and 5.2(b). To sort the vertices of T_ζ, a signed distance, $\eta : \mathbb{R}^d \to \mathbb{R}$, is assigned to each vertex in T_ζ. The signed-distance function is defined as

$$\eta_{\boldsymbol{x}} \equiv \eta(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{x}_{ref,\zeta})\cdot\boldsymbol{n}_p^k, \quad \boldsymbol{x}\in\mathbb{R}^d, \tag{5.2}$$

where $\boldsymbol{x}_{ref,\zeta}$ is some fixed reference vertex in T_ζ. Figs. 5.2(c) and (d) depict the sorting of the vertices of T_ζ and the corresponding iso-η planes passing through each vertex. In the d-dimensional case, $\mathcal{V}_\zeta : \mathbb{R} \to \mathbb{R}$, shall denote the total volume contained in T_ζ that lies below a given iso-η plane, and is defined as

$$\mathcal{V}_\zeta(\eta) = \int_{\Omega_\eta} d\mathcal{V}, \quad \Omega_\eta = T_\zeta \cap \{\boldsymbol{x}\in\mathbb{R}^d \mid \eta(\boldsymbol{x}) \le \eta\}. \tag{5.3}$$

The PLIC reconstruction problem can now be concisely restated as — computing η_p such that $\mathcal{V}_\zeta(\eta_p) = \mathcal{V}_p$, where η_p is the η-value of the line/plane containing PLIC facet p.

In the interest of computational efficiency, the PLIC reconstruction commences with the so-called *bracketing* procedure, which determines the η-values, η_{lo} and η_{up} (illustrated in Figs. 5.2(c) and (d)), such that the restriction of $\mathcal{V}_\zeta(\eta)$ to the domain $\eta \in [\eta_{lo}, \eta_{up}]$ can be defined in $\mathbb{R}^d$ by a unique polynomial function of order at most d. This is followed by a novel CD-based approach for solving η_p *analytically*, using the aforementioned polynomial function. To the best of the author's knowledge, existing methods based on CD compute η_p *via* the use of iterative approaches like the stabilised secant/bisection method used by Ahn and Shashkov [3], Brent's method by Ivey and Moin [28], and the Muller-Newton method by Skarysz *et al.* [58].

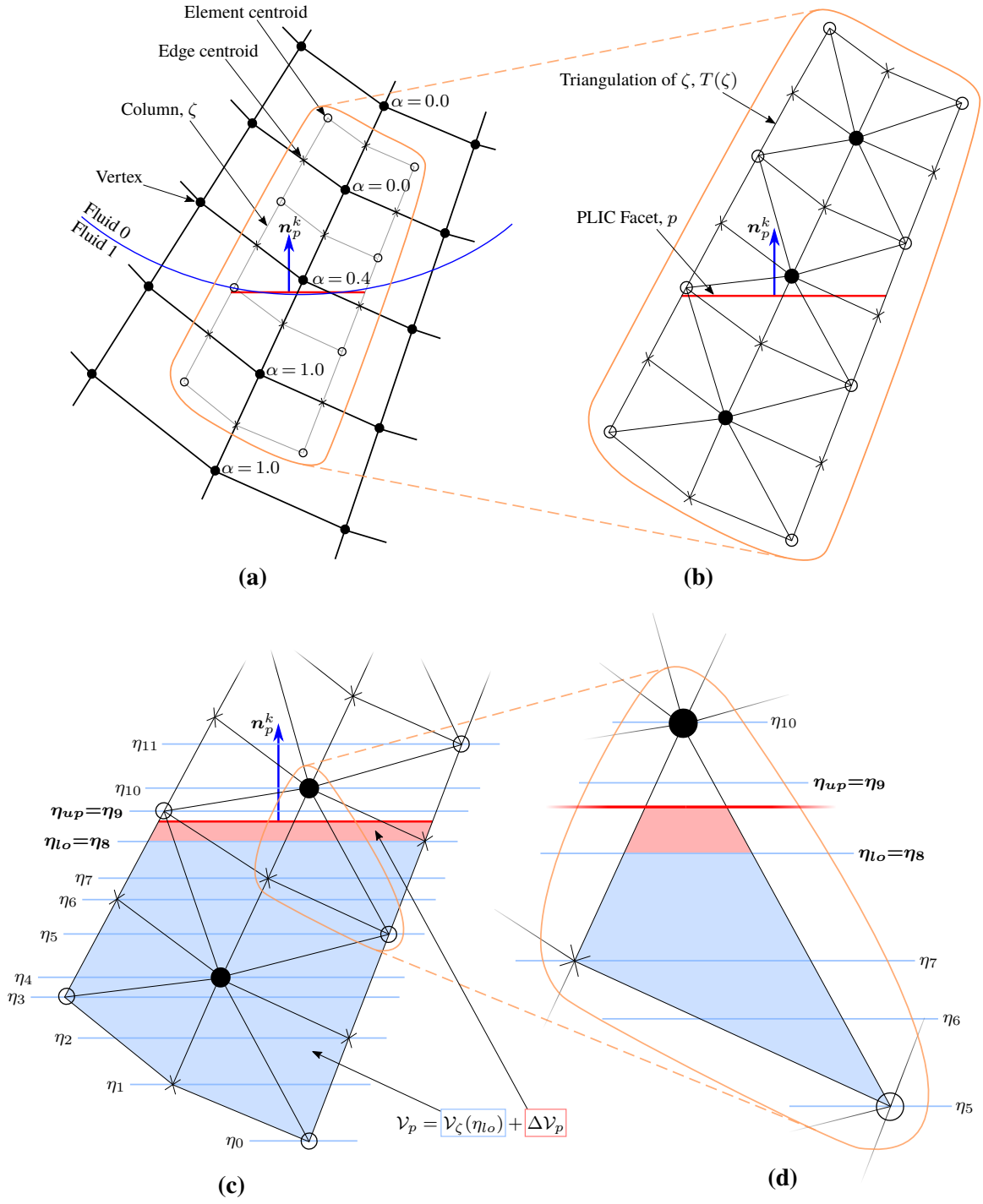


Figure 5.2: PLIC reconstruction procedure in $\mathbb{R}^2$ — (a) A dual-cell column ζ, (b) decomposition of ζ into its associated triangulation T_ζ, (c) sorting of the vertices of T_ζ by the signed distance function $\eta(\boldsymbol{x})$, and (d) a close-up of a triangle in T_ζ. The blue curve is the interface, the blue arrow is the PLIC facet normal $\boldsymbol{n}_p^k$, and the red line is the PLIC facet p which conserves the reference fluid volume $\mathcal{V}_p$ in ζ.

We digress to note that similar to the proposed method, López *et al.* [35] recently proposed a scheme that combines bracketing with an analytical solution procedure, but with no requirement for CD. They employ a vertex connectivity table for storing the face information of each polytope. This allows — (i) application to non-convex polyhedra and (ii) the use of the divergence theorem for volume calculations. Their scheme is proposed to be more efficient than CD approaches [38]. However, the results in Section 9.6 detail why this would not be the case in the context of the vertex-centred median dual-cell grids employed in this work. This is attributed to the median dual-cells in each interface column having to be constructed *ad hoc* to avoid significant increases in memory costs. Additionally, CD significantly simplifies the solution bracketing procedure and the overall implementation of the proposed method, as will be seen shortly. End of digression.

It is worth noting that although the current application of the proposed method is to construct a PLIC facet in a column ζ, the procedures detailed in this chapter are directly applicable to PLIC reconstruction in arbitrary (*i.e.* either convex or non-convex) polytopes as well. The following sections describe the bracketing procedure and the novel analytical solution procedure for computing η_p in T_ζ (or any triangulation/tetrahedronalisation of an arbitrary polytope).

5.2 Bracketing the Solution

For a general column in $\mathbb{R}^d$, the function $\mathcal{V}_\zeta(\eta)$ (see Eq. (5.3)) is monotonically increasing with respect to η, and is composed of piecewise-polynomial functions of order at most d, with C^0 continuity at each $\eta_{\boldsymbol{x}}$ iso-plane passing through each vertex $\boldsymbol{x} \in T_\zeta$. Therefore, bracketing the solution involves computing

$$\eta_{lo} = \max\big\{\eta_{\boldsymbol{x}} \,|\, \boldsymbol{x} \in T_\zeta,\ \mathcal{V}_\zeta(\eta_{\boldsymbol{x}}) \leq \mathcal{V}_p\big\}, \tag{5.4}$$

$$\begin{aligned}\eta_{up} &= \min\big\{\eta_{\boldsymbol{x}} \,|\, \boldsymbol{x} \in T_\zeta,\ \mathcal{V}_\zeta(\eta_{\boldsymbol{x}}) > \mathcal{V}_p\big\} \\ &= \min\big\{\eta_{\boldsymbol{x}} \,|\, \boldsymbol{x} \in T_\zeta,\ \eta_{\boldsymbol{x}} > \eta_{lo}\big\},\end{aligned} \tag{5.5}$$

and $\mathcal{V}_\zeta(\eta_{lo})$, as depicted by the light blue region in Fig. 5.2(c). As mentioned previously, this allows the restriction of $\mathcal{V}_\zeta(\eta)$ in $\eta \in [\eta_{lo}, \eta_{up}]$ to be described by a single polynomial equation.

Solving Eq. (5.4) requires explicitly computing the cut-off volume $\mathcal{V}_\zeta(\eta_{\boldsymbol{x}})$ iteratively for a subset of the vertices $\boldsymbol{x} \in T_\zeta$. This commences with an initial guess for the starting vertex, and proceeds until the required condition for η_{lo} is satisfied. As an initial guess, the $\eta_{\boldsymbol{x}}$ nearest to the η-value of a vertex that is interpolated to $\alpha = 0.5$ is used in this work. For an edge $\overline{ij} \in \zeta$ where $0.5 \in [\min(\alpha_i, \alpha_j), \max(\alpha_i, \alpha_j)]$, the aforementioned vertex is computed *via* linear interpolation of the mesh vertices $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. With this initial guess, Eq. (5.4) is solved in at most three iterations for all test cases performed in Chapter 9.

Considering Eq. (5.3), the computation of $\mathcal{V}_\zeta(\eta_{\boldsymbol{x}})$ in each iteration involves computing the net volume intersected by the region below the $\eta_{\boldsymbol{x}}$ iso-plane and each triangle/tetrahedron, $\tau \in T_\zeta$. Whether τ is below, above, or intersected by the $\eta_{\boldsymbol{x}}$ iso-plane is determined by how many of its vertices have η-values greater than $\eta_{\boldsymbol{x}}$. In $\mathbb{R}^d$, this results in a total of $(d+2)$ possible cases. Firstly, if all vertices have $\eta \leq \eta_{\boldsymbol{x}}$, τ is below the $\eta_{\boldsymbol{x}}$ iso-plane and contributes its whole area/volume. (For improved computational efficiency, the area/volume contributions in such cases are stored for reuse, should they remain un-intersected in subsequent iterations). Secondly, if all vertices have $\eta > \eta_{\boldsymbol{x}}$, τ is above the $\eta_{\boldsymbol{x}}$ iso-plane and contributes zero area/volume. Any one of the d remaining cases indicates that τ is intersected by the $\eta_{\boldsymbol{x}}$ iso-plane.

Considering $\mathbb{R}^2$ first, the two remaining cases are — either one or two vertices of the triangle τ have $\eta \leq \eta_{\boldsymbol{x}}$. The intersection points between the $\eta_{\boldsymbol{x}}$ iso-plane and an edge $\overline{mn}$ of τ can be computed *via* Eq. (5.6), and the resulting submerged area of τ can be straightforwardly computed thereafter.

$$\boldsymbol{x}_{\cap}(\eta_{\boldsymbol{x}}, \boldsymbol{x}_m, \boldsymbol{x}_n) = r_{\cap}\boldsymbol{x}_n + (1 - r_{\cap})\boldsymbol{x}_m, \tag{5.6}$$
$$\eta_{\boldsymbol{x}} \in \big[\min(\eta(\boldsymbol{x}_m),\ \eta(\boldsymbol{x}_n)),\ \max(\eta(\boldsymbol{x}_m),\ \eta(\boldsymbol{x}_n))\big],\quad r_{\cap} = \frac{\eta_{\boldsymbol{x}} - \eta(\boldsymbol{x}_m)}{\eta(\boldsymbol{x}_n) - \eta(\boldsymbol{x}_m)}.$$

Considering $\mathbb{R}^3$ next, the three remaining cases are — (i) only one vertex of τ has $\eta \leq \eta_{\boldsymbol{x}}$, (ii) three vertices have $\eta \leq \eta_{\boldsymbol{x}}$, or (iii) two vertices have $\eta \leq \eta_{\boldsymbol{x}}$. Cases (i) and (ii), as depicted in Figs. 5.3(a) and (b), require the computation of three intersection points between the edges of τ and the $\eta_{\boldsymbol{x}}$ iso-plane. Case (iii), as depicted in Fig. 5.3(c), requires the computation of four intersection points. Eq. (5.6) is again employed for this purpose.

Computing the volume contribution for case (i) involves computing the volume of the blue tetrahedron in Fig. 5.3(a). For this purpose, the volume of an arbitrary tetrahedron with vertices $\boldsymbol{x}_0$, $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_3$ is computed *via*

$$\mathcal{V}(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = \frac{1}{6}\left\|(\boldsymbol{x}_3 - \boldsymbol{x}_0)\cdot\big((\boldsymbol{x}_2 - \boldsymbol{x}_0)\times(\boldsymbol{x}_1 - \boldsymbol{x}_0)\big)\right\|. \tag{5.7}$$

Likewise, the volume contribution for case (ii) is computed in the manner depicted on the right-hand side of Fig. 5.3(b), which requires the computation of two tetrahedral volumes using Eq. (5.7).

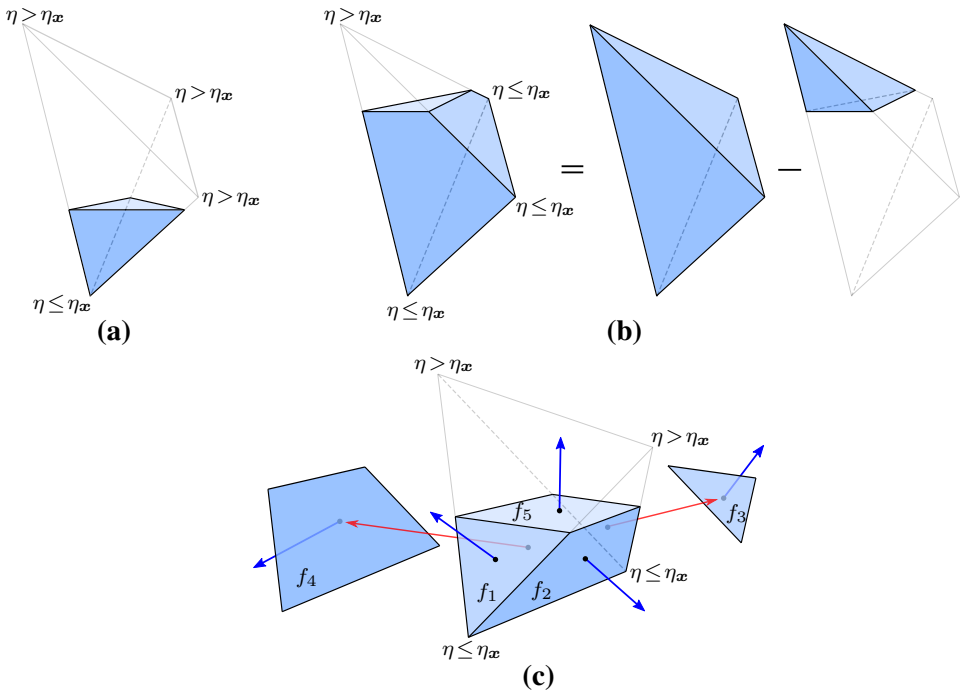


Figure 5.3: Computing the cut-off volume for a single tetrahedron $\tau \in T_{\zeta}$ — (a) volume of the blue submerged tetrahedron, (b) subtracting the volume of the upper tetrahedron from that of the parent tetrahedron, and (c) indirect volume calculation *via* face coefficients of the submerged polyhedron (for visibility, the two hidden faces are pushed out along the red arrows).

Unlike cases (i) and (ii), the volume contribution for case (iii) is computed indirectly *via* the application of divergence theorem to the five faces of the blue polyhedron that is depicted in Fig. 5.3(c). For each face f, given the face area $\mathcal{A}_f$, the outward-pointing unit normal $\boldsymbol{n}_f$, the face coefficient vector $\boldsymbol{c}_f = \mathcal{A}_f \boldsymbol{n}_f$, and an arbitrary vertex $\boldsymbol{x}_f$ in the plane of f, the volume of this polyhedron can be computed *via*

$$\mathcal{V} = \frac{1}{3}\sum_{f=1}^{5} \boldsymbol{c}_f \cdot \boldsymbol{x}_f.$$

The face coefficients of faces $1-4$ are straightforwardly computed *via* cross products. Fig. 5.4 illustrates a simple procedure for computing the face coefficients of quadrilateral submerged faces (such as f_2 and f_4 in Fig. 5.3(c)).

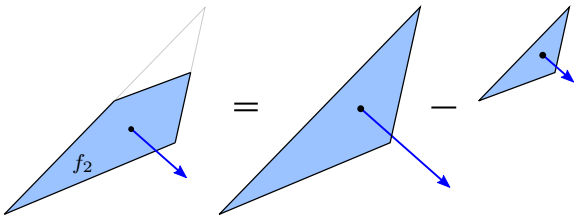


Figure 5.4: The procedure for computing the face coefficient vector of a quadrilateral submerged face of τ. The example shown here is face f_2 from Fig. 5.3(c).

The face coefficient vector $\boldsymbol{c}_5$, however, is indirectly computed by noting that, for the submerged polyhedron shown in Fig. 5.3(c),

$$\sum_{f=1}^{5} \boldsymbol{c}_f = 0 \iff \boldsymbol{c}_5 = -\sum_{f=1}^{4} \boldsymbol{c}_f.$$

This concludes the development of the tools required for the solution bracketing procedure. It should now be clear that the convex decomposition of ζ makes the bracketing procedure a lot more straightforward than it would be if ζ were not decomposed into triangles/tetrahedra. The novel analytical solution procedure in the bracketed interval is detailed next.

5.3 Polynomial Equation in the Bracketed Interval

This section details the formulation of the polynomial equation that describes the restriction of $\mathcal{V}_\zeta(\eta)$ (see Eq. (5.3)) to the domain $\eta \in [\eta_{lo}, \eta_{up}]$. This polynomial is used to compute η_p (the η-value of the plane containing PLIC facet p) that — (i) conserves the reference fluid volume $\mathcal{V}_p$ (see Eq. (5.1)), and (ii) is orthogonal to the pre-computed PLIC normal, $\boldsymbol{n}_p^k$. Property (i) is enforced by accounting for the additional volume

$$\Delta\mathcal{V}_p = \mathcal{V}_p - \mathcal{V}_\zeta(\eta_{lo}),$$

where $\mathcal{V}_\zeta(\eta_{lo})$ is an output of the bracketing procedure in the previous section. The light red region in Fig. 5.2(c) depicts $\Delta\mathcal{V}_p$ in $\mathbb{R}^2$. Property (ii) has already been enforced by virtue of the orientation of the sweep-plane. To compute the η-value increment, $\Delta\eta_p$, that corresponds

to $\Delta\mathcal{V}_p$, an incremental volume function $\Delta\mathcal{V}_\zeta(\Delta\eta)$ in column ζ is introduced for $\mathbb{R}^2$ and $\mathbb{R}^3$ in Sections 5.3.1 and 5.3.2, respectively.

5.3.1 2D Incremental Volume Equation

As shown in Figs. 5.2(c) and (d), determining the incremental volume function, $\Delta\mathcal{V}_\zeta(\Delta\eta)$, in $\mathbb{R}^2$ involves accounting for the individual volume increments, $\Delta\mathcal{V}_\tau(\Delta\eta)$, in each triangle $\tau \in \mathcal{T}_\zeta$ that is intersected by the η_{lo} iso-line. This sub-triangulation shall be denoted as $\mathcal{T}_{\zeta,lo} \subset \mathcal{T}_\zeta$. For illustrative purposes, the triangle in Fig. 5.2(d) shall be used to represent the general case of a triangle $\tau \in \mathcal{T}_{\zeta,lo}$. Fig. 5.5 demonstrates that $\Delta\mathcal{V}_\tau(\Delta\eta)$ must be defined for two distinct sections of τ, *i.e.* s_0 and s_1. These shall be referred to as *sweep-sections*.

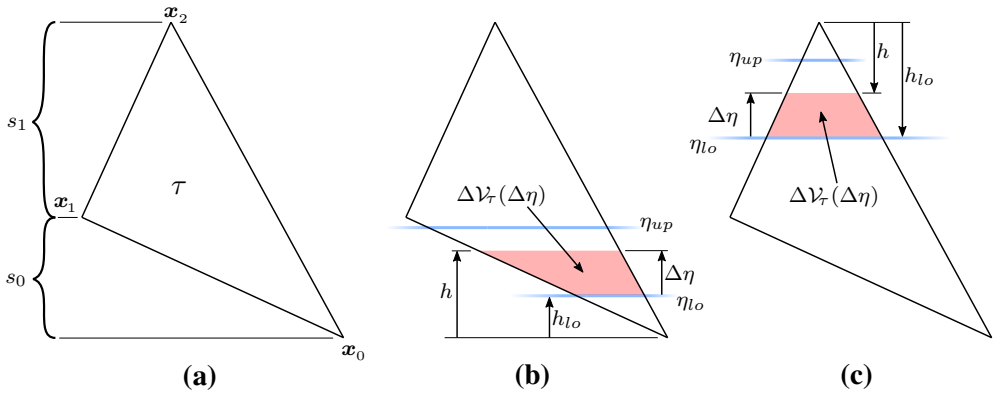


Figure 5.5: Incremental volume function for a triangle τ — (a) sweep-sections of τ, the incremental volume in sweep-section (b) s_0, and (c) s_1.

Each sweep-section has a unique quadratic equation describing $\Delta\mathcal{V}_\tau(\Delta\eta)$, whose coefficients are determined by the pair of edges that bound the respective sweep-section. To this end, we first define an arbitrary normalised (with respect to the PLIC facet normal $\boldsymbol{n}_p^k$) tangent vector, $\hat{\boldsymbol{t}}_{mn}$, of an edge that joins vertices $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$ as

$$\hat{\boldsymbol{t}}_{mn} = \begin{cases} \dfrac{\boldsymbol{t}_{mn}}{\boldsymbol{t}_{mn}\cdot\boldsymbol{n}_p^k} & \text{if } \boldsymbol{t}_{mn}\cdot\boldsymbol{n}_p^k \neq 0 \\ \boldsymbol{0} & \text{otherwise} \end{cases}, \tag{5.8}$$

where

$$\boldsymbol{t}_{mn} = \boldsymbol{x}_n - \boldsymbol{x}_m, \quad \eta(\boldsymbol{x}_n) \geq \eta(\boldsymbol{x}_m), \tag{5.9}$$

and $m, n \in \{0, 1, 2\}$ are local vertex indices of τ, as per Fig. 5.5(a). The second condition in Eq. (5.8) accounts for degenerate cases, which occur when a sweep-section collapses as a result of one of its bounding edges being parallel to the sweep-line. Such sweep-sections have zero volume contribution.

It is here instructive to point out that since $\eta(\boldsymbol{x})$ is the signed distance between the iso-η lines passing through $\boldsymbol{x}$ and $\boldsymbol{x}_{ref,\zeta}$ (see Eq. (5.2)), the difference between two η-values is the signed distance between their associated iso-η lines. With this in mind, the quantities h and h_{lo} shown in Figs. 5.5(b) and 5.5(c) are computed from

$$h \equiv h(\eta, \boldsymbol{x}_{ref,\tau}) = \eta - \eta(\boldsymbol{x}_{ref,\tau}), \tag{5.10}$$

where $\boldsymbol{x}_{ref,\tau}$ is a reference vertex in τ, and is defined as per Fig. 5.5(a) as

$$\boldsymbol{x}_{ref,\tau} = \begin{cases} \boldsymbol{x}_0, & \text{for } s_0 \\ \boldsymbol{x}_2, & \text{for } s_1 \end{cases}. \tag{5.11}$$

Using the η_{lo} iso-line as the datum, a sweep-line increment, $\Delta\eta$, is defined as

$$\Delta\eta \;=\; (\eta - \eta_{lo}) \;=\; (h - h_{lo}), \quad \Delta\eta \in \big[0, (\eta_{up} - \eta_{lo})\big]. \tag{5.12}$$

Next, the following terms are defined:

$$\hat{\boldsymbol{t}}_s = \begin{cases} \hat{\boldsymbol{t}}_{01}, & \text{for sweep-section } s_0 \\ \hat{\boldsymbol{t}}_{12}, & \text{for sweep-section } s_1 \end{cases} \quad \text{and} \quad \varsigma_s = \begin{cases} 1, & \text{for sweep-section } s_0 \\ -1, & \text{for sweep-section } s_1 \end{cases}, \tag{5.13}$$

where ς_s ensures that $\Delta\mathcal{V}_\tau(\Delta\eta) \geq 0$, since $h < 0$ for sweep-section s_1 (see Fig. 5.5(c)). Combining Eqs. (5.8) - (5.13), and referring to Fig. 5.5, the incremental volume function in τ can be expressed as

$$\begin{aligned} \Delta\mathcal{V}_\tau(\Delta\eta) \;&=\; \varsigma_s \left[\frac{1}{2} \left\| (h\hat{\boldsymbol{t}}_{02}) \times (h\hat{\boldsymbol{t}}_s) \right\| - \frac{1}{2} \left\| (h_{lo}\hat{\boldsymbol{t}}_{02}) \times (h_{lo}\hat{\boldsymbol{t}}_s) \right\| \right] \\ &=\; \frac{\varsigma_s}{2} \left(h^2 - h_{lo}^2 \right) \left\| \hat{\boldsymbol{t}}_{02} \times \hat{\boldsymbol{t}}_s \right\| \\ &=\; \frac{\varsigma_s}{2} \left((h_{lo} + \Delta\eta)^2 - h_{lo}^2 \right) \left\| \hat{\boldsymbol{t}}_{02} \times \hat{\boldsymbol{t}}_s \right\| \\ &=\; \frac{\varsigma_s}{2} \left(2h_{lo}\Delta\eta + \Delta\eta^2 \right) \left\| \hat{\boldsymbol{t}}_{02} \times \hat{\boldsymbol{t}}_s \right\| \\ &=\; \sum_{r=1}^{2} c_{r,\tau} \Delta\eta^r, \end{aligned} \tag{5.14}$$

where

$$c_{1,\tau} = \varsigma_s h_{lo} \left\| \hat{\boldsymbol{t}}_{02} \times \hat{\boldsymbol{t}}_s \right\| \;\text{ and }\; c_{2,\tau} = \frac{\varsigma_s}{2} \left\| \hat{\boldsymbol{t}}_{02} \times \hat{\boldsymbol{t}}_s \right\|$$

are the quadratic coefficients that must be computed for each triangle $\tau \in T_{\zeta,lo}$. Using Eq. (5.14), the incremental volume function of column ζ can finally be expressed as the sum

$$\Delta\mathcal{V}_\zeta(\Delta\eta) \;=\; \sum_{\tau \in T_{\zeta,lo}} \Delta\mathcal{V}_\tau(\Delta\eta) \;=\; \sum_{r=1}^{2} C_r \Delta\eta^r, \tag{5.15}$$

where

$$C_r = \sum_{\tau \in T_{\zeta,lo}} c_{r,\tau}, \quad r \in \{1, 2\}.$$

Having developed the function that describes the incremental volume of ζ in the domain $[\eta_{lo}, \eta_{up}]$, what remains to be computed is $\Delta\eta_p \in \big[0, (\eta_{up} - \eta_{lo})\big]$ such that $\Delta\mathcal{V}_\zeta(\Delta\eta_p) = \Delta\mathcal{V}_p$, where $\Delta\eta_p$ is the increment from η_{lo} to the iso-η line containing PLIC facet p. This will be detailed generically for $\mathbb{R}^2$ and $\mathbb{R}^3$ in Section 5.4. The next section extends the above-detailed procedure to $\mathbb{R}^3$.

5.3.2 3D Incremental Volume Equation

Determining $\Delta\mathcal{V}_\zeta(\Delta\eta)$ in $\mathbb{R}^3$ involves accounting for the individual volume increments, $\Delta\mathcal{V}_\tau(\Delta\eta)$, in each tetrahedron $\tau \in T_\zeta$ that is intersected by the η_{lo} iso-plane (Figs. 5.2(c) and (d) depict the corresponding analogy in $\mathbb{R}^2$). As before, this subset of tetrahedra shall be denoted as $T_{\zeta,lo} \subset T_\zeta$. Fig. 5.6(a) demonstrates that $\Delta\mathcal{V}_\tau(\Delta\eta)$ must be defined for three distinct *sweep-sections* of τ.

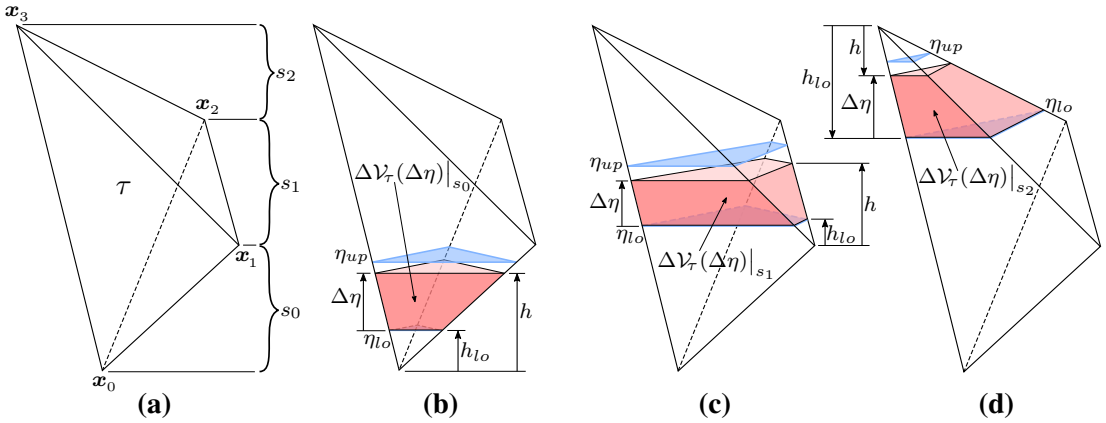


Figure 5.6: Incremental volume function of a tetrahedron τ - (a) sweep-sections of τ, the incremental volume in sweep-section (b) s_0, (c) s_1, and (d) s_2.

Each sweep-section has a unique cubic equation describing $\Delta\mathcal{V}_\tau(\Delta\eta)$, whose coefficients are again determined by the set of edges enclosing the respective sweep-section. For this purpose, the definitions of Eqs. (5.8) – (5.10), and (5.12) are re-used in this section, as they are generic to $\mathbb{R}^2$ and $\mathbb{R}^3$. The reference vertices required for Eq. (5.11) are here redefined in the context of the sweep-sections in Fig. 5.6(a) as

$$\boldsymbol{x}_{ref,\tau} = \begin{cases} \boldsymbol{x}_0, & \text{for } s_0 \\ \boldsymbol{x}_1, & \text{for } s_1 \\ \boldsymbol{x}_3, & \text{for } s_2 \end{cases}. \tag{5.16}$$

Once again, the second condition in Eq. (5.8) accounts for degenerate cases, which occur when a sweep-section collapses as a result of at least one of its enclosing edges being parallel to the sweep-plane. Such sweep-sections have zero volume contribution. For a general tetrahedron τ, at most two sweep-sections could be degenerate — one degeneracy when only one edge of τ is parallel to the sweep-plane, and two degeneracies when two (which necessarily implies three) edges of τ are parallel to the sweep-plane. The following subsections detail the derivation of $\Delta\mathcal{V}_\tau(\Delta\eta)$ for each sweep-section of τ.

Sweep-sections s_0 and s_2

In order to develop the equations for $\Delta\mathcal{V}_\tau(\Delta\eta)$ in sweep-sections s_0 and s_2, note that the common feature of these sweep-sections is that they are both enclosed by three edges, as shown in Figs. 5.6(b) and (d). Using Eq. (5.8), the normalised tangent of the common edge between the two sweep-sections is $\hat{\boldsymbol{t}}_{03}$. Likewise, the normalised tangents for the two remaining edges in each sweep-section are denoted by

$$\hat{\boldsymbol{t}}_{ab}, \quad \text{where } (a,b) = \begin{cases} (0,1) & \text{for } s_0 \\ (2,3) & \text{for } s_2 \end{cases}, \tag{5.17}$$

and

$$\hat{\boldsymbol{t}}_{cd}, \quad \text{where } (c,d) = \begin{cases} (0,2) & \text{for } s_0 \\ (1,3) & \text{for } s_2 \end{cases}. \tag{5.18}$$

Using Eqs. (5.7), (5.17), and (5.18), the incremental volume in sweep-sections s_0 and s_2 can now be expressed as

$$\Delta\mathcal{V}_\tau(\Delta\eta)\big|_{s_0/s_2} = \frac{1}{6}\|\hat{\boldsymbol{t}}_{03}\cdot(\hat{\boldsymbol{t}}_{ab}\times\hat{\boldsymbol{t}}_{cd})\|\left(3h_{lo}^2\Delta\eta + 3h_{lo}\Delta\eta^2 + \Delta\eta^3\right). \tag{5.19}$$

The above equation is derived in Appendix B.1. Next, the equation for $\Delta\mathcal{V}_\tau(\Delta\eta)$ in sweep-section s_1 is developed.

Sweep-section s_1

Sweep-section s_1 is enclosed by four edges that emanate from a triangular base that is coincident with the iso-η plane passing through vertex $\boldsymbol{x}_1$ of τ (see Fig. 5.6(a)). Before determining the equation for $\Delta\mathcal{V}_\tau(\Delta\eta)|_{s_1}$, we first characterise the volume, $\mathcal{V}_\tau(h)|_{s_1}$, bounded between the base of s_1 and a plane that lies at a height $h > 0$ above the base (see Fig. 5.7(a)). The required incremental volume function is then simply $\Delta\mathcal{V}_\tau(\Delta\eta)|_{s_1} = \mathcal{V}_\tau(h)|_{s_1} - \mathcal{V}_\tau(h_{lo})|_{s_1}$.

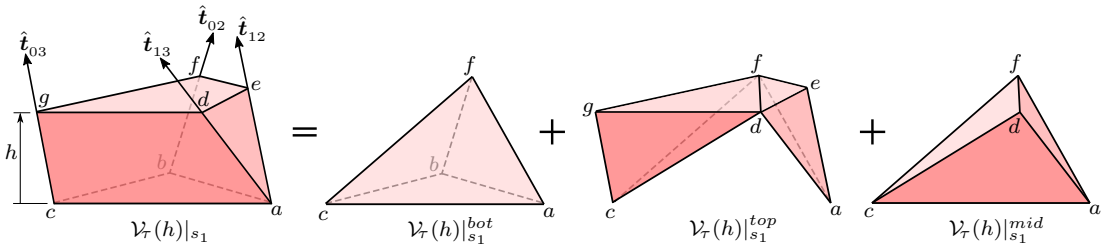


Figure 5.7: (a) The volume, $\mathcal{V}_\tau(h)|_{s_1}$, bounded between the base of sweep-section s_1 and a plane at a height h above the base. This volume is decomposed into three components — (b) the tetrahedron with volume $\mathcal{V}_\tau(h)|^{bot}_{s_1}$, (c) the two tetrahedra with total volume $\mathcal{V}_\tau(h)|^{top}_{s_1}$, and (d) the tetrahedron with volume $\mathcal{V}_\tau(h)|^{mid}_{s_1}$.

Fig. 5.7 demonstrates that $\mathcal{V}_\tau(h)|_{s_1}$ can be decomposed into three components, $\mathcal{V}_\tau(h)|^{bot}_{s_1}$, $\mathcal{V}_\tau(h)|^{top}_{s_1}$, and $\mathcal{V}_\tau(h)|^{mid}_{s_1}$, whose individual volumes can be computed more straightforwardly. To this end, Eqs. (5.8) and (5.9) are employed to compute the set of vertices $\{\boldsymbol{x}_a, \ldots, \boldsymbol{x}_g\}$ (as per Fig. 5.7(a)) as

$$\boldsymbol{x}_a \equiv \boldsymbol{x}_1, \tag{5.20}$$

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + \frac{\eta(\boldsymbol{x}_1) - \eta(\boldsymbol{x}_0)}{\eta(\boldsymbol{x}_n) - \eta(\boldsymbol{x}_0)}\boldsymbol{t}_{0n}, \quad (m,n) \in \big\{(b,2),(c,3)\big\}, \tag{5.21}$$

and

$$\boldsymbol{x}_p \equiv \boldsymbol{x}_p(h) = \boldsymbol{x}_q + h\hat{\boldsymbol{t}}_{rs}, \tag{5.22}$$
$$(p,q,r,s) \in \big\{(d,a,1,3),\ (e,a,1,2),\ (f,b,0,2),\ (g,c,0,3)\big\}.$$

Using Eqs. (5.7) – (5.9) and (5.20) – (5.22), and by defining $\hat{\boldsymbol{t}}_A = (\hat{\boldsymbol{t}}_{02} - \hat{\boldsymbol{t}}_{13})$ and $\hat{\boldsymbol{t}}_B = (\hat{\boldsymbol{t}}_{03} - \hat{\boldsymbol{t}}_{12})$, the volumes of the three components in Fig. 5.7 can be expressed as

$$\mathcal{V}_\tau(h)|_{s_1}^{bot} \;=\; \frac{1}{6}\big(h\boldsymbol{n}_p^k\big)\cdot\big(\boldsymbol{t}_{ab}\times\boldsymbol{t}_{ac}\big) \;=\; \frac{h}{6}\boldsymbol{n}_p^k\cdot\big(\boldsymbol{t}_{ab}\times\boldsymbol{t}_{ac}\big), \tag{5.23}$$

$$\mathcal{V}_\tau(h)|_{s_1}^{top} \;=\; \frac{h}{6}\boldsymbol{n}_p^k\cdot\big(h^2\hat{\boldsymbol{t}}_A\times\hat{\boldsymbol{t}}_B + h(\boldsymbol{t}_{ab}\times\hat{\boldsymbol{t}}_B - \boldsymbol{t}_{ac}\times\hat{\boldsymbol{t}}_A) + \boldsymbol{t}_{ab}\times\boldsymbol{t}_{ac}\big), \tag{5.24}$$

and

$$\mathcal{V}_\tau(h)|_{s_1}^{mid} \;=\; \frac{h}{6}\big(\boldsymbol{t}_{ac}\times\hat{\boldsymbol{t}}_{13}\big)\cdot\big(\boldsymbol{t}_{ab} + h\hat{\boldsymbol{t}}_{02}\big). \tag{5.25}$$

Eqs. (5.24) and (5.25) are derived in Appendix B.2. The volume $\mathcal{V}_\tau(h)|_{s_1}$ can now be expressed by adding the like terms in Eqs. (5.23) - (5.25), which gives

$$\begin{aligned}\mathcal{V}_\tau(h)|_{s_1} \;&=\; \mathcal{V}_\tau(h)|_{s_1}^{bot} + \mathcal{V}_\tau(h)|_{s_1}^{top} + \mathcal{V}_\tau(h)|_{s_1}^{mid}\\ &=\; \sum_{r=1}^{3} m_r h^r,\end{aligned} \tag{5.26}$$

where

$$\begin{aligned} m_1 \;&=\; \frac{1}{6}\boldsymbol{t}_{ab}\cdot\big(\boldsymbol{t}_{ac}\times(2\boldsymbol{n}_p^k + \hat{\boldsymbol{t}}_{13})\big),\\ m_2 \;&=\; \frac{1}{6}\big(\boldsymbol{n}_p^k\cdot(\boldsymbol{t}_{ab}\times\hat{\boldsymbol{t}}_B - \boldsymbol{t}_{ac}\times\hat{\boldsymbol{t}}_A) + \boldsymbol{t}_{ac}\cdot(\hat{\boldsymbol{t}}_{13}\times\hat{\boldsymbol{t}}_{02})\big),\\ m_3 \;&=\; \frac{1}{6}\boldsymbol{n}_p^k\cdot\big(\hat{\boldsymbol{t}}_A\times\hat{\boldsymbol{t}}_B\big).\end{aligned}$$

Lastly, using Eq. (5.26), the incremental volume $\Delta\mathcal{V}_\tau(\Delta\eta)|_{s_1}$ (see Fig. 5.6(c)) is expressed as

$$\begin{aligned}\Delta\mathcal{V}_\tau(\Delta\eta)|_{s_1} &= \mathcal{V}_\tau(h)|_{s_1} - \mathcal{V}_\tau(h_{lo})|_{s_1}\\ &= \big(3m_3h_{lo}^2 + 2m_2h_{lo} + m_1\big)\Delta\eta + \big(3m_3h_{lo} + m_2\big)\Delta\eta^2 + m_3\Delta\eta^3.\end{aligned} \tag{5.27}$$

The derivations for m_1 and Eq. (5.27) are also in Appendix B.2. The next subsection summarises the terms that compose $\Delta\mathcal{V}_\tau(\Delta\eta)$ in $\mathbb{R}^3$ and finalises the formulation of the incremental volume function $\Delta\mathcal{V}_\zeta(\Delta\eta)$ in column ζ.

Summary of the Terms in $\Delta\mathcal{V}_\zeta(\Delta\eta)$

Considering Eqs. (5.19) and (5.27), the incremental volume function in a general tetrahedron $\tau \in T_{\zeta,lo}$ can be expressed as the cubic equation

$$\Delta\mathcal{V}_\tau(\Delta\eta) = \sum_{r=1}^{3} c_{r,\tau}\Delta\eta^r, \tag{5.28}$$

where the coefficients $c_{r,\tau}$ are tabulated for the three sweep-sections of τ in Table 5.1. The relevant terms are defined in the two preceding subsections, and $\aleph = \|\hat{\boldsymbol{t}}_{03}\cdot(\hat{\boldsymbol{t}}_{ab}\times\hat{\boldsymbol{t}}_{cd})\|$.

Table 5.1: Coefficients of $\Delta\mathcal{V}_\tau(\Delta\eta)$ for the sweep-sections in τ.

Coefficient	Sweep-section in τ	
	s_0/s_2	s_1
$c_{1,\tau}$	$h_{lo}^2\aleph/2$	$3m_3h_{lo}^2+2m_2h_{lo}+m_1$
$c_{2,\tau}$	$h_{lo}\aleph/2$	$3m_3h_{lo}+m_2$
$c_{3,\tau}$	$\aleph/6$	m_3

Employing Eq. (5.28), the incremental volume function for a column ζ in $\mathbb{R}^3$ can finally be expressed as

$$\Delta\mathcal{V}_\zeta(\Delta\eta) \;=\; \sum_{\tau\in T_{\zeta,lo}} \Delta\mathcal{V}_\tau(\Delta\eta) \;=\; \sum_{r=1}^{3} C_r\Delta\eta^r, \tag{5.29}$$

where

$$C_r = \sum_{\tau\in T_{\zeta,lo}} c_{r,\tau}\,, \quad r\in\{1,2,3\}.$$

The next section utilises the definitions for $\Delta\mathcal{V}_\zeta(\Delta\eta)$ in $\mathbb{R}^2$ and $\mathbb{R}^3$, and generically details the procedure for positioning PLIC facet p in column ζ.

5.4 PLIC Facet Positioning Procedure

Section 5.3 formulated two polynomial functions that describe the behaviour of the restriction of $\mathcal{V}_\zeta(\eta)$ (see Eq. (5.3)) to the domain $[\eta_{lo},\eta_{up}]$. That is, $\mathcal{V}_\zeta(\eta)$ is described in this domain by the equation

$$\mathcal{V}_\zeta(\eta) \;=\; \mathcal{V}_\zeta(\eta_{lo})+\Delta\mathcal{V}_\zeta(\Delta\eta) \;=\; \mathcal{V}_\zeta(\eta_{lo})+\Delta\mathcal{V}_\zeta(\eta-\eta_{lo}), \quad \eta\in[\eta_{lo},\eta_{up}],$$

where, for the d-dimensional case,

$$\Delta\mathcal{V}_\zeta(\Delta\eta) = \sum_{r=1}^{d} C_r\Delta\eta^r, \tag{5.30}$$

and the coefficients C_r are defined as per Eqs. (5.15) and (5.29) for $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively. As mentioned in Section 5.1, computing PLIC facet p in column ζ requires solving for $\eta_p\in[\eta_{lo},\eta_{up}]$ such that $\mathcal{V}_\zeta(\eta_p)=\mathcal{V}_p$. To this end, η_p is determined *via* Eq. (5.30), by first solving for the root, $\Delta\eta_p\in\big[0,(\eta_{up}-\eta_{lo})\big]$, of the following equation:

$$\Delta\mathcal{V}_p - \sum_{r=1}^{d} C_r\Delta\eta^r = 0, \tag{5.31}$$

where $\Delta\mathcal{V}_p=\mathcal{V}_p-\mathcal{V}_\zeta(\eta_{lo})$. The significance of the relevant terms is again illustrated in Fig. 5.8 for convenience. Note that Eq. (5.31) has exactly one root in $\big[0,(\eta_{up}-\eta_{lo})\big]$ because $\Delta\mathcal{V}_p\geq 0$, $\Delta\mathcal{V}_\zeta(0)=0$, and $\Delta\mathcal{V}_\zeta(\Delta\eta)$ is monotonically increasing. The root is solved for analytically *via* the quadratic formula in $\mathbb{R}^2$ and Cardano's cubic formula in $\mathbb{R}^3$.

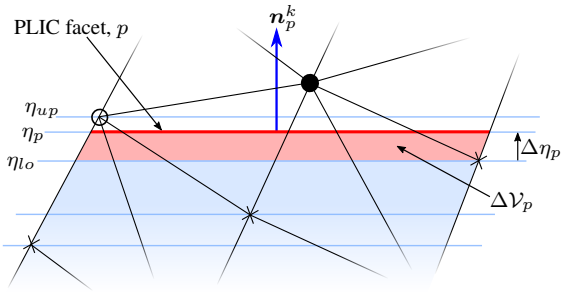


Figure 5.8: Close-up of Fig. 5.2(c), showing the reference fluid volume increment $\Delta\mathcal{V}_p$ (and corresponding sweep-plane increment $\Delta\eta_p$) to be added from the η_{lo} iso-η plane up to the PLIC facet.

Given $\Delta\eta_p$, as per Fig. 5.8, the solution for η_p is then

$$\eta_p = \eta_{lo} + \Delta\eta_p.$$

The vertices of PLIC facet p are defined by the intersections of the η_p iso-plane and the edges on the boundary of T_ζ. Once these bounding edges are identified, the vertices are computed using Eq. (5.6). In $\mathbb{R}^3$, the vertices are sorted counter-clockwise with respect to $\boldsymbol{n}_p^k$ (required for the polynomial fitting procedure detailed in Chapter 7).

5.5 PLIC Facet Stencil Construction

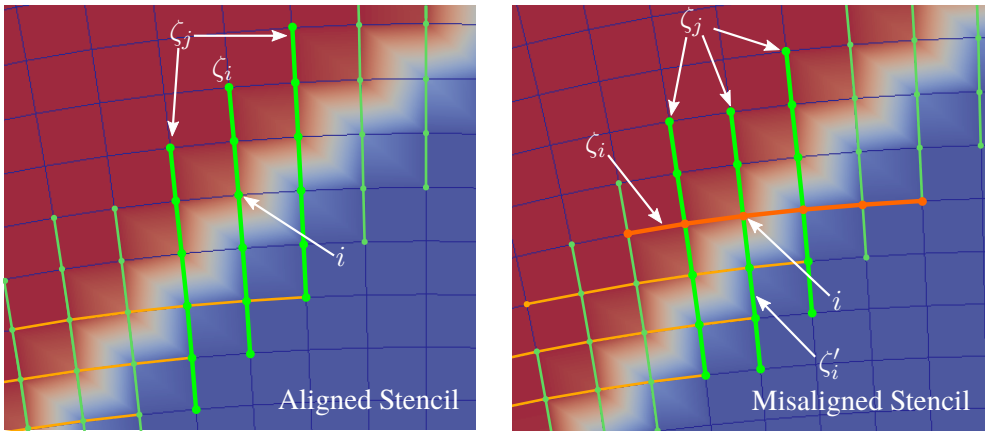
The procedures detailed thus far allow for the interface to be represented by a set of PLIC facets. Just as the HF method (see Chapter 2) utilises *stencils of heights* to locally approximate the partial derivatives of a function describing the interface on Cartesian grids, *stencils of PLIC facets* shall be used for the same purpose on non-orthogonal structured grids. The latter shall also be used to refine the PLIC facet normals in the iterative normal refinement procedure detailed in Chapter 6.

The PLIC facet stencils are determined by first identifying appropriate column stencils, based on the direction in which they were constructed[7]. The number of columns required for a given stencil is characterised by the order of the polynomial that is to be fitted (using the least squares procedure detailed in Chapter 7). That is, a stencil of order n comprises at least $(n+1)$ and $(n+1)(n+2)/2$ columns in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively. The reader is reminded that this work only considers $n \in \{2, 4\}$ for non-orthogonal structured grids, and does not consider boundary stencils, except when symmetry boundary conditions are imposed. The stencil construction procedure at interior vertices and symmetry boundaries is detailed in the sections to follow.

5.5.1 Stencil Construction at Interior Vertices

Three types of stencils are considered in this work, which will be referred to as *aligned*, *misaligned*, and *mixed* stencils. Fig. 5.9 depicts examples of an aligned and a misaligned stencil in $\mathbb{R}^2$. Each stencil is constructed around a column, ζ_i, where the required attribute (*e.g.* partial derivatives, PLIC facet normal, curvature, *etc.*) is to be computed.

[7]For this reason, the phrases "PLIC facet stencil" and "column stencil" shall be used analogously, for the remainder of this work.



(a) **(b)**

Figure 5.9: The 2D analogy of an (a) aligned and (b) misaligned stencil associated to a column ζ_i, constructed at the interface vertex i, with neighbouring columns ζ_j. Each sub-figure is a close-up of Fig. 4.1

Considering an interface vertex i, an *aligned* stencil occurs when a column ζ_i that passes through i is surrounded by neighbouring columns ζ_j such that, in the cases of $n \in \{2, 4\}$ — (i) each ζ_j is at most $(n/2)$ vertices/elements away[8] from ζ_i, and (ii) all ζ_j lie within 45° of ζ_i. Fig. 5.10 depicts the cross-sections of the possible topologies of aligned stencils that could occur in $\mathbb{R}^3$, on the block-structured grid that was shown in Fig. 1.1(b). The red circles represent the central column ζ_i in each stencil, and the black and grey circles represent columns ζ_j that are one and two vertices/elements away from ζ_i, respectively. (Figs. 5.10(b) and 5.10(d) – (f) depict cases where ζ_i lies along edges that emanate outward from the eight corners of the innermost structured region of the grid in Fig. 1.1(b)).

A *misaligned* stencil occurs when at least one of the aforementioned conditions for an aligned stencil is not satisfied for a column ζ_i, but both conditions are satisfied for another column, ζ_i', that also passes through interface vertex i (see Fig. 5.9(b) for an example in $\mathbb{R}^2$ and Fig. 6.4 for the PLIC facets associated to a misaligned stencil). Note that this implies that each misaligned stencil contains a sub-stencil that qualifies as an aligned stencil. In each misaligned stencil, the required attribute is always computed with respect to the PLIC facets associated to this *aligned sub-stencil*. Misaligned stencils typically occur at the fringes of contiguous regions of columns, while overlapping a different region (see Figs. 4.1, 4.2, and 5.9).

A *mixed* stencil occurs if neither an aligned nor a misaligned stencil can be constructed at column ζ_i. Such stencils comprise all columns passing through vertices that are at most one element away from interface vertex i. Mixed stencils typically occur in the first 1–2 iterations of the PLIC facet normal refinement procedure (see details in the next chapter). This is due to the refinement of the normals eventually resulting in contiguous regions of columns overlapping the liquid-gas interface. Therefore, mixed stencils are only utilised for refining PLIC facet normals, and not for polynomial fitting purposes.

[8]Two columns are considered *one element away* from each other if they are both connected to a common element; and are *two elements away* from each other if there exists a common vertex that is one element away from both columns; and so on.

Algorithm 2 Computation of a stencil of order n, centred around a column ζ_i in $\mathbb{R}^2$.

Input: Central column ζ_i, and the order of the stencil n.
Output: Stencil of columns Z_i centred around ζ_i, and the resulting stencil type.

procedure ComputeColumnStencil2D(ζ_i, n)
  set $i \leftarrow$ the index of the mesh vertex at which ζ_i was constructed.
  set $\boldsymbol{d}_i \leftarrow$ the unit direction vector of ζ_i (red arrows in Fig. 4.3).
  set $Z \leftarrow$ the set of all columns connected to any vertex that is at most $(n/2)$ vertices/elements away from i.
  $Z_0 = []$ // The subset of columns in Z whose direction vectors are within 45° of $\boldsymbol{d}_i$.
  $Z_1 = []$ // The subset of columns in Z whose direction vectors aren't within 45° of $\boldsymbol{d}_i$.
  for all $\zeta_j \in Z$ **do** // Separate columns that are aligned/misaligned with $\boldsymbol{d}_i$.
    if $\zeta_j \neq \zeta_i$ **then**
      set $\boldsymbol{d}_j \leftarrow$ the unit direction vector of ζ_j (red arrows in Fig. 4.3).
      if $\boldsymbol{d}_j \cdot \boldsymbol{d}_i > \frac{1}{\sqrt{2}}$ **then**
        add $Z_0 \leftarrow \zeta_j$ // Within 45° of $\boldsymbol{d}_i$.
      else
        add $Z_1 \leftarrow \zeta_j$ // Not within 45° of $\boldsymbol{d}_i$.
      end
    end
  end
  $stenType$ // Stencil type: 'Aligned' or 'Misaligned'
  set $\mathcal{N}_i \leftarrow$ the array of all vertices that are at most $(n/2)$ vertices/elements away from i.
  for all $j \in \mathcal{N}_i$ **do**
    if $\exists! \zeta_j \in Z_0,\ j \in \zeta_j$, **then**
      $stenType =$ 'Aligned'
    else if $\exists! \zeta_j \in Z_1,\ j \in \zeta_j$, **then**
      $stenType =$ 'Misaligned'
    end
  end
  $Z_i = []$ // The resulting stencil of columns at ζ_i, where ζ_i is always the first column.
  add $Z_i \leftarrow \zeta_i$
  if $stenType =$ 'Aligned', **then**
    append $Z_i \leftarrow Z_0$
  else
    append $Z_i \leftarrow Z_1$
  end
  return Z_i, $stenType$

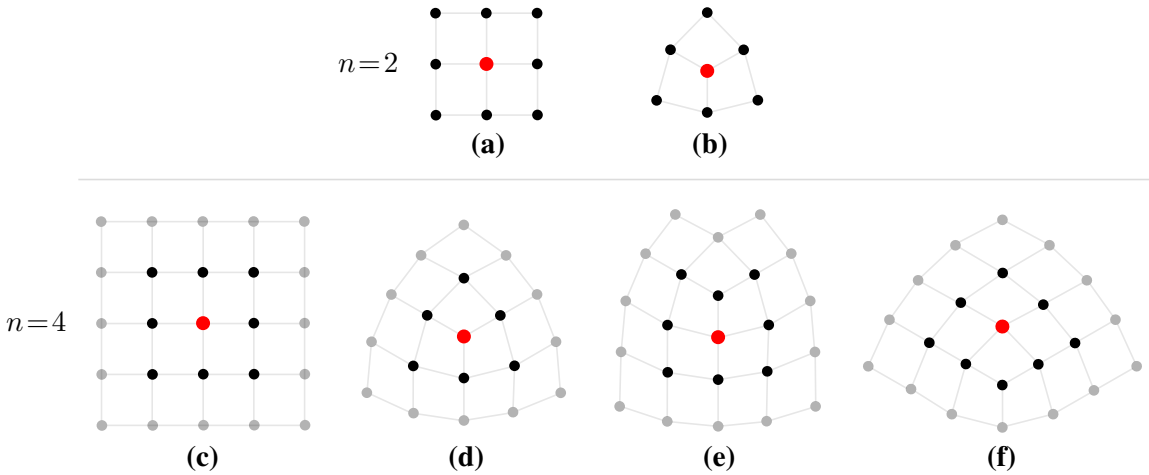


Figure 5.10: Cross-sections of the possible topologies of aligned stencils in the block-structured grid depicted in Fig. 1.1(b), for polynomial fits of order $n \in \{2, 4\}$ in $\mathbb{R}^3$. The circles represent the columns, with the central column ζ_i in red, and the neighbouring columns in black/grey. (a) and (b) are the topologies for $n=2$, and (c) – (f) are the topologies for $n=4$.

Algorithm 2 broadly outlines the procedure for constructing aligned or misaligned stencil types in $\mathbb{R}^2$. It is instructive to point out that this algorithm can be straightforwardly extended to $\mathbb{R}^3$ by accounting for the fact that a column ζ_i could have up to two misaligned stencils constructed around it. This is because interfaces in three-dimensional structured grids may consist of regions where up to three separate contiguous sections of columns overlap each other (as shown in Fig. 4.2). In the case of two misaligned stencils being constructed at the column ζ_i, the manner in which the two stencils are utilised is dependent on the specific attribute (*i.e.* interface normal, curvature, *etc.*) that is to be computed in ζ_i. This will be further elaborated in Chapter 6 and Section 8.5.

The next section details the treatment of PLIC facets in column stencils adjacent to symmetry boundaries.

5.5.2 Stencil Construction at Symmetry Boundaries

This work employs symmetry boundary conditions to expedite the simulation of the static bubble and oscillating droplet test cases in Chapter 9. To this end, this section details the aspects to be considered when constructing a stencil of PLIC facets adjacent to at most $(d-1)$ neighbouring symmetry boundaries[9] in $\mathbb{R}^d$.

Since a vertex-centred approach is used in this work, each PLIC facet constructed in a column that lies adjacent to at least one symmetry boundary must be appropriately reflected about each boundary. Fig. 5.11 depicts a portion of a PLIC facet stencil adjacent to a symmetry boundary in $\mathbb{R}^2$. Since the reflected portion(s) of each boundary PLIC facet must be co-linear/co-planar with the original PLIC facet, the normals of these PLIC facets must be orthogonal to the boundary normal. This is done by — (i) computing the PLIC facet normal using the standard

[9]When constructing stencils adjacent to two neighbouring symmetry boundaries in $\mathbb{R}^3$, this work only considers cases where the two boundaries are mutually orthogonal.

procedure[10] that applies to interior PLIC facets (see Chapter 6 for details), followed by (ii) removing the component of the computed normal that is parallel to the normal of each adjacent boundary. Additionally, the centroid $\boldsymbol{x}_p$ of the subsequently constructed boundary PLIC facet must be consecutively projected onto each adjacent boundary, as depicted by the red dot in Fig. 5.11. The projected centroid is, in turn, utilised in the iterative normal refinement procedure.

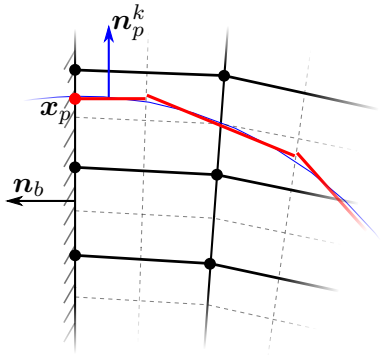


Figure 5.11: Portion of a stencil of PLIC facets adjacent to a symmetry boundary in $\mathbb{R}^2$. Each PLIC facet p associated to a column that lies on the symmetry boundary is constructed with a normal, $\boldsymbol{n}_p^k$, that is orthogonal to the boundary normal, $\boldsymbol{n}_b$. The red dot depicts the projection of of the boundary PLIC facet centroid onto the boundary. The dashed lines demarcate the vertex-centred median dual-cells.

Fig. 5.12 illustrates all possible configurations in $\mathbb{R}^2$, of stencils of order $n \in \{2, 4\}$ that are adjacent to a symmetry boundary. The PLIC facets in the stencil that are reflected across the boundary are depicted as dashed lines, and the red dots indicate the central PLIC facet in each stencil.

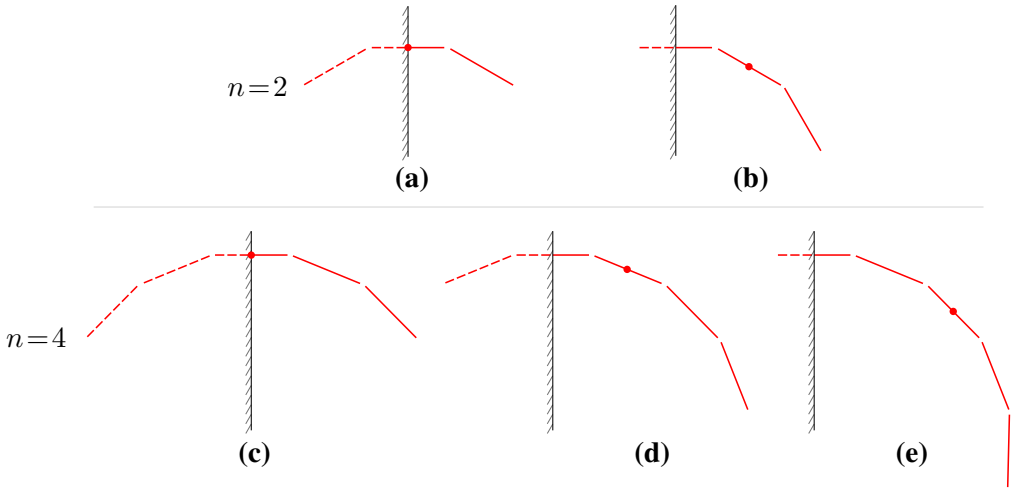


Figure 5.12: All possible configurations of symmetry boundary stencils of order $n \in \{2, 4\}$ in $\mathbb{R}^2$. The procedure for PLIC facet reflection is also analogously applicable to creating $(n+1) \times (n+1)$ stencils in $\mathbb{R}^3$.

[10] It is instructive to note that this only influences the normals of PLIC facets in $\mathbb{R}^3$ that are adjacent to a single symmetry boundary. This is because the computed normal still has a degree of freedom in the plane of the boundary. In the two remaining cases (*i.e.* when the PLIC facet is adjacent to one symmetry boundary in $\mathbb{R}^2$ or two symmetry boundaries in $\mathbb{R}^3$) there is only one possible direction in which the computed normal can be oriented.

In $\mathbb{R}^3$, the only stencil types that are encountered at symmetry boundaries in this work, are the two $(n+1)\times(n+1)$ stencils depicted in Figs. 5.10(a) and (c) (*i.e.* after reflection of PLIC facets). The configurations depicted in Fig. 5.12 can be considered as the lateral views of these stencils, and therefore these configurations analogously apply to $\mathbb{R}^3$ too. Fig. 5.13(a) illustrates an example stencil of order $n=4$ in $\mathbb{R}^3$, that is adjacent to two symmetry boundaries with normals $\boldsymbol{n}_{b_1}$ and $\boldsymbol{n}_{b_2}$, respectively. Once again, the red dot depicts the central PLIC facet. Figs. 5.13(b) and (c) highlight in light red, the PLIC facets in the stencil that are consecutively reflected across the symmetry boundary. These two reflections are analogous to the configurations shown in Figs. 5.12(e) and (d), respectively.

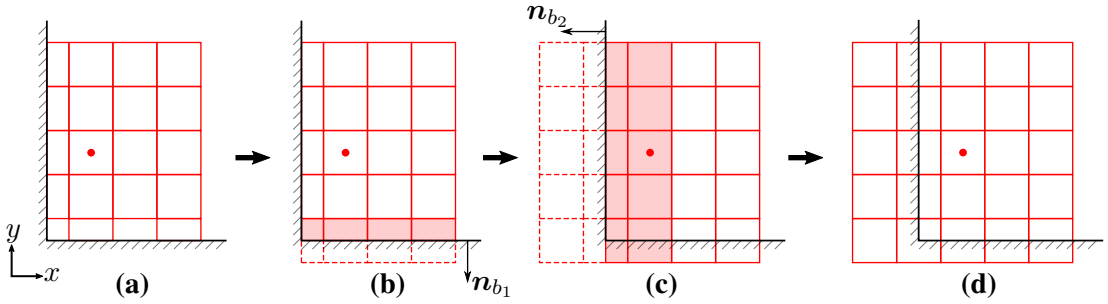


Figure 5.13: Plan view of a stencil, illustrating the procedure for consecutively reflecting the appropriate PLIC facets to form a stencil of order $n=4$ in $\mathbb{R}^3$. The PLIC facets shaded in light red are those that are to be reflected, analogous to Fig. 5.12(d) and (e). The red dot indicates the central PLIC facet in the stencil.

After symmetrising a boundary stencil, it is treated in the same manner as an interior stencil. That is, the higher-order interface reconstruction procedure detailed in Chapter 7 and the PLIC facet normal refinement procedure in Chapter 6 can be directly applied to the symmetrised version of a boundary stencil.

5.6 Closure

This chapter detailed the procedure for — (i) computing a PLIC facet in a column ζ, given the volume of the reference fluid to be conserved and a pre-computed unit normal, and (ii) constructing stencils of PLIC facets for the dual purpose of interface normal refinement and higher-order interface reconstruction. A novel approach based on the convex decomposition of the median dual-cells in each column was introduced, to address (i). The PLIC reconstruction procedure detailed in this chapter is fully volume conservative, which will prove to be a crucial aspect for accurately extending the least squares polynomial fitting strategy to non-orthogonal structured in grids in Chapter 7.

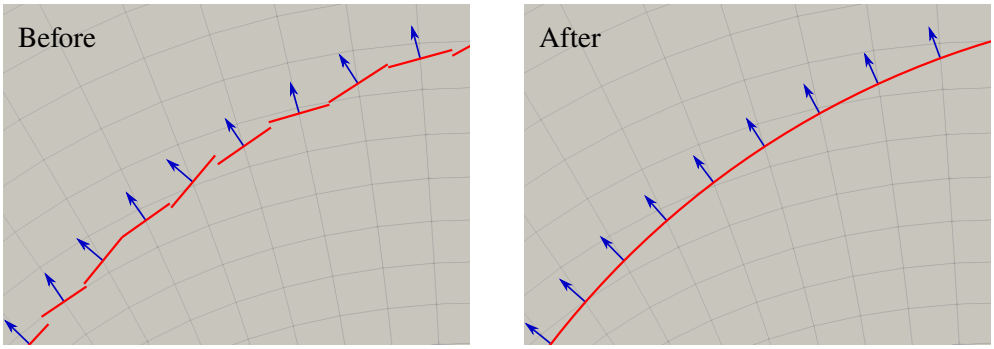
Given the resulting PLIC representation of the interface, the next step is to utilise stencils of PLIC facets to iteratively refine the normals of each facet. This is detailed next.

Chapter 6

Refinement of PLIC Facet Normals

6.1 Background

In Chapter 4, it was stated that interface normals are initialised using $\nabla\tilde{\alpha}/\|\nabla\tilde{\alpha}\|$ (from the smoothed VOF field) for interface column construction and PLIC reconstruction purposes. This was found to result in—(i) an excessive proportion of non-contiguous columns overlapping the interface, and (ii) a staggering of the constructed PLIC facets (see Figs. 6.1(a) and 6.2(a)). The latter *staggering* significantly impairs the accuracy of the polynomials fitted through the PLIC facets (see Chapter 7) during higher-order interface reconstruction. This, in turn, adversely impacts the accuracy of the resulting interface curvature computation used in surface tension modelling. These challenges are here addressed by adopting an iterative PLIC facet normal refinement procedure which, in each iteration k, utilises stencils composed of PLIC facets constructed in iteration $(k-1)$. The interface normals at $k=0$ are seeded by $\nabla\tilde{\alpha}/\|\nabla\tilde{\alpha}\|$. The resulting refined normals (see Figs. 6.1(b) and 6.2(b)) significantly improves the accuracy of the higher-order (up to 4^{th}-order in this work) interface reconstruction, and thereby that of the interface curvature computation. The accuracy and cost metrics associated to this procedure are quantified in Sections 9.2 and 9.6, respectively.



(a) **(b)**

Figure 6.1: PLIC facets representing a portion of a circular interface (a) before (using $\nabla\tilde{\alpha}/\|\nabla\tilde{\alpha}\|$), and (b) after, PLIC facet normal refinement.

Changes in PLIC facet normals result in a change in both orientation and position of the PLIC facets. Therefore, the iterative approach invariably requires that the PLIC facets be conservatively reconstructed at each iteration (as per Chapter 5). Moreover, in the first few iterations (typically 1–2 in this work), the interface columns may also need to be reconstructed, as the directions for interface column construction may vary between iterations (see Chapter 4). For this

reason, using the iterative procedure without some form of convergence acceleration is significantly more expensive. To address this, a novel *spring-based* normal refinement acceleration procedure is proposed in Section 6.4.

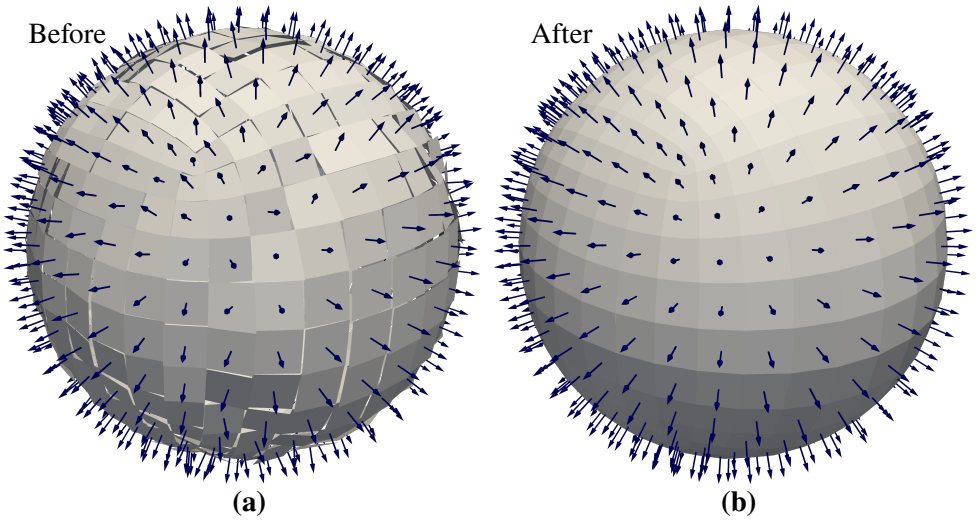


Figure 6.2: PLIC facets representing a spherical interface (a) before (normals inherited from $\nabla\tilde{\alpha}/\|\nabla\tilde{\alpha}\|$), and (b) after PLIC facet normal refinement.

The PLIC normal refinement procedure is carried out in two phases—*non-accelerated* and *accelerated*. The sub-routines composing the non-accelerated phase are respectively detailed in Sections 6.2 and 6.3 for aligned and mixed stencils, and misaligned stencils. The sub-routines in the accelerated phase are then introduced in Section 6.4. Lastly, Section 6.5 summarises all sub-routines in the iterative normal refinement scheme.

6.2 PLIC Normal Refinement on Aligned and Mixed Stencils

The procedures detailed in this section apply generically to both *aligned* and *mixed* stencils (see Section 5.5). With reference to Fig. 6.3, the refined normal, $\boldsymbol{n}_p^{k+1}$, is to be computed for the PLIC facet p which denotes the PLIC facet associated to the interface column around which the stencil was constructed (central PLIC facet in the case of aligned stencils). The refined normal is computed as an inverse-distance-weighted sum of the inferred normals $\boldsymbol{n}_{pq}^k$ (blue arrows):

$$\boldsymbol{n}_p^{k+1} = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}, \quad \boldsymbol{v} = \sum_{q\in\mathfrak{N}_p} \frac{\boldsymbol{n}_{pq}^k}{\|\boldsymbol{t}_{pq}^k\|}, \tag{6.1}$$

where $\mathfrak{N}_p$ is the set of indices of the PLIC facets q neighbouring p in the stencil (see Section 4 for neighbour identification procedure), and

$$\boldsymbol{t}_{pq}^k = \boldsymbol{x}_q^k - \boldsymbol{x}_p^k.$$

Here, $\boldsymbol{x}_p^k$ and $\boldsymbol{x}_q^k$ are the centroids of the respective PLIC facets p and q, as constructed in iteration k. Consequently, $1/\|\boldsymbol{t}_{pq}^k\|$ imposes an inverse-distance weighting on each $\boldsymbol{n}_{pq}^k$, which

is computed as

$$\boldsymbol{n}_{pq}^{k} = \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}, \quad \boldsymbol{w} = \boldsymbol{t}_{pq}^{k} \times \left(\boldsymbol{n}_{p}^{k} \times \boldsymbol{t}_{pq}^{k}\right), \quad \forall q \in \mathfrak{N}_{p}.$$

In Fig. 6.3, $\boldsymbol{n}_{pq}^{k}$ is the normal of the dashed line. In $\mathbb{R}^3$, it is the normal contained in the plane defined by $\boldsymbol{t}_{pq}^{k}$ and the normal of p in the current iteration, $\boldsymbol{n}_{p}^{k}$. The reason for the use of $\boldsymbol{n}_{pq}^{k}$ (as opposed to $\boldsymbol{n}_{q}^{k}$) is faster convergence as a result of $\boldsymbol{n}_{pq}^{k}$ solely being a function of the PLIC centroids in the stencil.

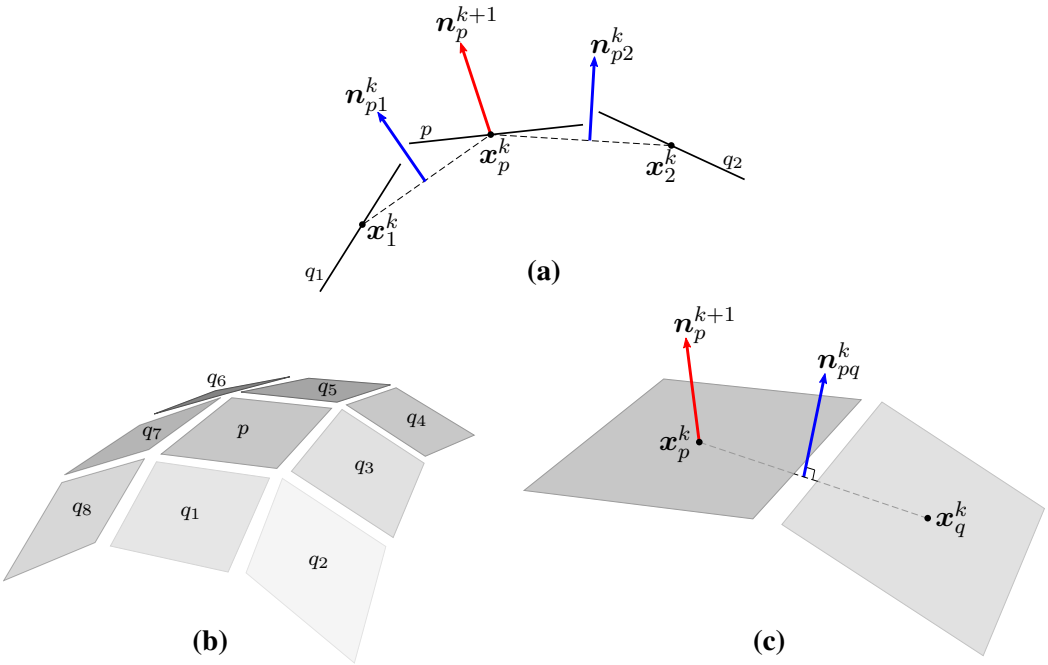


Figure 6.3: The PLIC facet normal computation procedure for aligned and mixed stencils.

It was, however, found that applying the above procedure for refining the normals of PLIC facets associated to *misaligned* stencils prevented higher-order grid convergence of the maximum error of the interface curvature. Therefore, the alternative procedure that was adopted is detailed next.

6.3 PLIC Normal Refinement on Misaligned Stencils

The computation of the refined normal $\boldsymbol{n}_{p}^{k+1}$ of the PLIC facet p (that which is associated to the column about which the stencil was constructed) of *misaligned* stencils is aided by the least squares polynomial fitting procedure that will be detailed[11] in Chapter 7. This is done by fitting polynomials of order $n=2$ to the *aligned sub-stencil* (see definition in Section 5.5.1) associated to each misaligned stencil. The refined normal is then inferred from the analytically computed normal of the fitted polynomial, at the point nearest to the centroid of p. This procedure is now detailed for $\mathbb{R}^2$ and $\mathbb{R}^3$.

[11] For the purposes of this section, the required polynomial is assumed to be suitably fitted, as per the procedures yet to be detailed in Chapter 7. Therefore, only the polynomial definitions in Eqs. (7.1) and (7.4) for $n=2$ are here utilised.

6.3.1 2D Method

Fig. 6.4 illustrates the procedure for computing the interface normal for a misaligned stencil in $\mathbb{R}^2$. The three PLIC facets demarcated by the green dashed line in Fig. 6.4(a) constitute the aligned sub-stencil of the misaligned stencil, through which a parabola is to be fitted. The origin of the local coordinate system of the parabola is located at the centroid of the central PLIC facet of the aligned sub-stencil (labelled as $\boldsymbol{x}_{q_2}^k$ in Fig. 6.4(a)). Further, the η-axis is aligned with the refined normal of the same (labelled as $\boldsymbol{n}_{q_2}^{k+1}$ Fig. 6.4(a)). Using $\boldsymbol{n}_{q_2}^{k+1}$ for this purpose, as opposed to $\boldsymbol{n}_{q_2}^{k}$, was found to result in faster convergence of the overall normal refinement scheme. However, this requires that the normal of the central PLIC facet in the aligned sub-stencil of each misaligned stencil be computed prior to the computation of $\boldsymbol{n}_{p}^{k+1}$. This is achieved by refining the normals of all central PLIC facets of aligned stencils before visiting misaligned stencils.

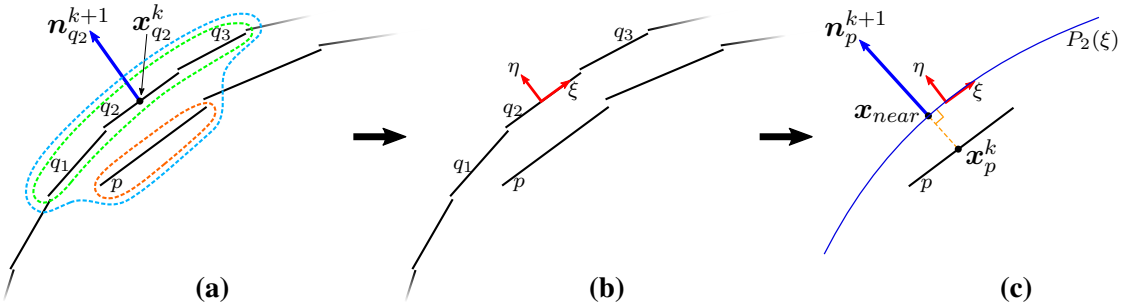


Figure 6.4: The PLIC facet normal computation procedure for a misaligned stencil in $\mathbb{R}^2$ — (a) demarcation of the misaligned stencil (blue dashed line) comprising an aligned sub-stencil (green dashed line) and PLIC facet p (red dashed line), (b) the local coordinate system of the parabolic fit, and (c) the procedure for computing the refined normal, $\boldsymbol{n}_{p}^{k+1}$. For clarity, the interface columns are not shown.

Consider the following parabola that is fitted through the aligned sub-stencil of the misaligned stencil, and whose coefficients are computed as per Section 7.1:

$$P_2(\xi) = \sum_{i=1}^{3} c_i \xi^{i-1}.$$

Let an arbitrary point on this parabola be defined with respect to the $\xi\eta$-coordinate system as $\boldsymbol{x}_{P_2} = \big(\xi, P_2(\xi)\big)$. Further, let $\boldsymbol{x}_p^k = (\xi_p^k, \eta_p^k)$ be the definition of the centroid of PLIC facet p in the $\xi\eta$-coordinate system. The nearest point to $\boldsymbol{x}_p^k$ on the parabola can be determined by first defining the quadrance (square of the distance) between $\boldsymbol{x}_p^k$ and $P_2(\xi)$ as

$$Q(\xi) = \|\boldsymbol{x}_p^k - \boldsymbol{x}_{P_2}\|^2 = (\xi_p^k - \xi)^2 + \left(\eta_p^k - \sum_{i=1}^{3} c_i \xi^{i-1}\right)^2. \tag{6.2}$$

Minimising the above with respect to ξ results in the following cubic equation (see Appendix C.1 for derivation) whose roots are to be computed:

$$2\big(c_2(c_1 - \eta_p^k) - \xi_p^k\big) + 2\big(c_2^2 + 2c_3(c_1 - \eta_p^k) + 1\big)\xi + 6c_3c_2\xi^2 + 4c_3^2\xi^3 \;=\; 0.$$

This is solved analytically using Cardano's formula for cubic equations. Since the solution to the above equation could yield at most three real roots, the nearest point $\boldsymbol{x}_{near}$ corresponds to the root, ξ_{near}, that yields the lowest value of $Q(\xi)$. The refined normal $\boldsymbol{n}_p^{k+1}$ is thus computed as

$$\boldsymbol{n}_p^{k+1} = \text{sgn}\big(\boldsymbol{n}_p^k \cdot \boldsymbol{n}_{near}\big)\frac{\boldsymbol{n}_{near}}{\|\boldsymbol{n}_{near}\|}, \quad \boldsymbol{n}_{near} = \mathbf{T}_{L2G}\begin{bmatrix}\frac{\partial P_2}{\partial \xi}\\ -1\end{bmatrix}. \tag{6.3}$$

Here, $\partial P_2/\partial\xi$ is evaluated at ξ_{near}, $\boldsymbol{n}_{near}$ is the normal of the parabola at the point $\boldsymbol{x}_{near}$ (see Fig. 6.4(c)), and $\mathbf{T}_{L2G}$ is the transformation matrix from the local (ξ, η) coordinate system to the global (x, y) coordinate system. The above procedure is extended to $\mathbb{R}^3$ next.

6.3.2 3D Method

In extending the 2D procedure to $\mathbb{R}^3$, the overall strategy for computing $\boldsymbol{n}_p^{k+1}$ for PLIC facet p remains the same. The only nuance, however (mentioned in Section 5.5.1), is that a column in $\mathbb{R}^3$ can have up to two misaligned stencils constructed around it. In such cases, the procedure that follows is to be applied consecutively to each misaligned stencil. The final normal of PLIC facet p is then simply taken to be the mean of the normals computed in each stencil. Therefore, the remainder of this section will only consider the normal refinement procedure for a single misaligned stencil.

Consider the following paraboloid that is fitted through the aligned sub-stencil of the misaligned stencil, and whose coefficients c_i are computed as per Section 7.2:

$$P_2(\xi, \eta) = \sum_{s=0}^{2}\sum_{r=0}^{2-s} c_i \xi^r \eta^s, \tag{6.4}$$

$$i = s + 1 + \frac{1}{2}(r + s)(r + s + 1) \in [1, 6]. \tag{6.5}$$

Consider an arbitrary point, $\boldsymbol{x}_{P_2} = \big(\xi, \eta, P_2(\xi, \eta)\big)$, on this paraboloid and let $\boldsymbol{x}_p^k = (\xi_p^k, \eta_p^k, \psi_p^k)$ be the definition of the centroid of the central PLIC facet p in the local coordinate system. To determine the nearest point to $\boldsymbol{x}_p^k$ on the paraboloid, the quadrance between $\boldsymbol{x}_p^k$ and $P_2(\xi, \eta)$ is similarly defined as

$$Q(\xi, \eta) = \|\boldsymbol{x}_p^k - \boldsymbol{x}_{P_2}\|^2 = (\xi_p^k - \xi)^2 + (\eta_p^k - \eta)^2 + \left(\psi_p^k - \sum_{s=0}^{2}\sum_{r=0}^{2-s} c_i \xi^r \eta^s\right)^2, \tag{6.6}$$

where again, i is computed as per Eq. (6.5).

Eq. (6.6) is minimised using a multivariate Newton's method, for which the first- and second-order partial derivatives of Q in the gradient and Hessian,

$$\nabla Q \equiv \nabla Q(\xi, \eta) = \begin{bmatrix} Q_\xi \\ Q_\eta \end{bmatrix} \text{ and } \mathbf{H} \equiv \mathbf{H}(\xi, \eta) \equiv \nabla \otimes \nabla Q = \begin{bmatrix} Q_{\xi\xi} & Q_{\xi\eta} \\ Q_{\eta\xi} & Q_{\eta\eta} \end{bmatrix},$$

are defined as follows (see Appendix C.1 for derivations):

$$Q_\xi = 2\big(c_2 d_1 - \xi_p^k + d_2\xi + d_3\eta + 3c_2c_5\xi^2 + 2d_4\xi\eta + d_5\eta^2 + 2c_5^2\xi^3 + 3c_4c_5\xi^2\eta + d_6\xi\eta^2 + c_4c_6\eta^3\big),$$

$$Q_\eta = 2\big(c_3d_1 - \xi_p^k + d_3\xi + d_7\eta + d_4\xi^2 + 2d_5\xi\eta + 3c_3c_6\eta^2 + c_4c_5\xi^3 + d_6\xi^2\eta + 3c_4c_6\xi\eta^2 + 2c_6^2\eta^3\big),$$

$$Q_{\xi\xi} = 2\big(d_2 + 6c_2c_5\xi + 2d_4\eta + 6c_5^2\xi^2 + 6c_4c_5\xi\eta + d_6\eta^2\big),$$

$$Q_{\eta\eta} = 2\big(d_7 + 2d_5\xi + 6c_3c_6\eta + d_6\xi^2 + 6c_4c_6\xi\eta + 6c_6^2\eta^2\big),$$

$$Q_{\xi\eta} = Q_{\eta\xi} = 2\big(d_3 + 2d_4\xi + 2d_5\eta + 3c_4c_5\xi^2 + 2d_6\xi\eta + 3c_4c_6\eta^2\big).$$

Here, the coefficients d_i are defined as

$$d_1 = c_1 - \psi_p^k, \quad d_2 = c_2^2 + 2c_4d_1 + 1, \quad d_3 = c_5d_1 + c_2c_3, \quad d_4 = c_2c_5 + c_3c_4,$$

$$d_5 = c_2c_6 + c_3c_5, \quad d_6 = c_5^2 + 2c_4c_6, \text{ and } \; d_7 = c_3^2 + 2c_6d_1 + 1.$$

Lastly, given the vector of unknowns, $\boldsymbol{e} = [\xi \;\; \eta]^T$, and by setting $\boldsymbol{e}_0 = \mathbf{0}$, Eq. (6.6) can be minimised iteratively using the following routine:

$$\boldsymbol{e}_{\tau+1} = \boldsymbol{e}_\tau - \mathbf{H}^{-1}\nabla Q,$$

where $\mathbf{H} \equiv \mathbf{H}(\boldsymbol{e}_\tau)$ and $\nabla Q \equiv \nabla Q(\boldsymbol{e}_\tau)$. The routine is deemed converged when the residual $\|\boldsymbol{e}_{\tau+1} - \boldsymbol{e}_\tau\|$ falls below a user-defined tolerance (1×10^{-5} in this work). The refined PLIC facet normal $\boldsymbol{n}_p^{k+1}$ is then similarly computed as

$$\boldsymbol{n}_p^{k+1} = \text{sgn}\big(\boldsymbol{n}_p^k \cdot \boldsymbol{n}_{near}\big)\frac{\boldsymbol{n}_{near}}{\|\boldsymbol{n}_{near}\|}, \quad \boldsymbol{n}_{near} = \mathbf{T}_{L2G}\begin{bmatrix}\frac{\partial P_2}{\partial \xi}\\ \frac{\partial P_2}{\partial \eta}\\ -1\end{bmatrix}, \tag{6.7}$$

where $\partial P_2/\partial\xi$ and $\partial P_2/\partial\eta$ are evaluated at $\boldsymbol{e}_{\tau+1}$, $\boldsymbol{n}_{near}$ is the normal of the paraboloid at the nearest point $\boldsymbol{x}_{near}$, and $\mathbf{T}_{L2G}$ is the transformation matrix from the local (ξ, η, ψ) coordinate system to the global (x, y, z) coordinate system.

As noted previously, the outlined PLIC normal refinement procedure requires several iterations to converge. To address this, a novel PLIC normal refinement acceleration procedure is proposed next.

6.4 PLIC Normal Refinement Acceleration Procedure

As mentioned previously, the non-accelerated PLIC normal refinement procedures detailed in the previous two sections may suffer from slow convergence (see Section 9.2). Although this work proposes a solution acceleration strategy, a pre-requisite for its use is the absence of *mixed* PLIC stencils. A minimum number of iterations is therefore required, where the PLIC facet normals must invariably be refined without acceleration. This enables the refinement of the PLIC normals to a sufficient degree of accuracy such that no new interface columns are constructed in subsequent iterations. The iteration at which this condition is first satisfied is denoted as k_{min}, *i.e.* the interface columns constructed at iterations k_{min} and $(k_{min}-1)$ are the same[12]. Thereafter, for all subsequent iterations $k \geq k_{min}$, the presently detailed procedures replace the preceding non-accelerated refinement procedures.

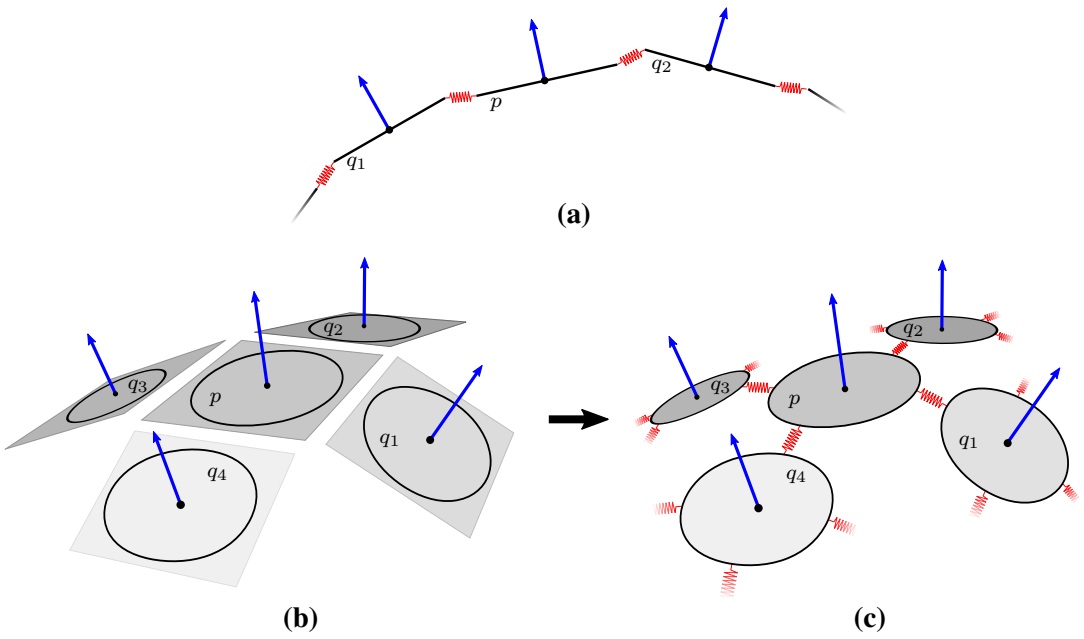


Figure 6.5: The localised setup of the spring system between PLIC facet p and its neighbours—(a) the 2D spring setup in an aligned stencil, (b) the subset of PLIC facets used in a 3D aligned stencil, and (c) the simplification of the 3D PLIC facets to disks in the same plane, with springs attached to the boundaries of each disk. (Note that the gaps between PLIC facets have been exaggerated for visual clarity).

Consider a closed liquid-gas interface[13] that can be described by a continuous function. A significant improvement in accuracy of the normals of the PLIC facets representing such an interface can be achieved by minimising the *gaps* between the boundaries of adjacent PLIC facets, over the entire interface. This is visually apparent from Figs. 6.1 and 6.2. Underpinning this observation is the mass conservative nature of the PLIC reconstruction (see Chapter 5). To minimise these gaps in an accelerated manner, a *spring-based* methodology is employed, where

[12] It is instructive to point out that k_{min} is not a fixed number of iterations, as it may vary depending on the shape/topology of the liquid-gas interface. However, for all interfaces tested in this work, k_{min} is either 1 or 2.

[13] The reader is here reminded that the interaction of the liquid-gas interface with solid-wall domain boundaries is beyond the scope of this work.

tension springs (of zero unstretched length) are attached between the boundaries of neighbouring PLIC facets (represented as disks in $\mathbb{R}^3$ for simplicity), as illustrated in Fig. 6.5. Thereafter, provided that each PLIC facet is allowed to pivot freely about its centroid, the global minimum energy state of the resulting spring system is sought. In doing so, both the *local* and *global* errors of the PLIC normals are instantaneously reduced, within a single refinement iteration k. This is in contrast to the purely *local* error reduction of the non-accelerated refinement procedures. This spring analogy emulates the notion of surface tension pulling the PLIC facets taut over the interface[14].

As a result of the geometrically non-linear nature of the spring system, the acceleration procedure involves sub-iterations. Therefore, in the k^{th} iteration of the PLIC refinement procedure, the total energy of the spring system is iteratively minimised with each sub-iteration, λ. For this purpose, the total spring energy is expressed as a function of the unit normals of the PLIC facets representing the interface. Each unit normal is described with respect to a local polar/spherical coordinate system in $\mathbb{R}^2/\mathbb{R}^3$, where the origin is located at the centroid of the PLIC facet. In the d-dimensional case, this allows for the evolution of each unit normal to be described as a function of $(d-1)$ variables. For each central PLIC facet in an *aligned* stencil, the refined unit normal is computed using a modified Newton's method [39]. However, the refined unit normals of PLIC facets associated to *misaligned* stencils are computed in a non-accelerated manner (as per Section 6.3), and subsequently set as Dirichlet boundary conditions in the linearised system that is solved in each sub-iteration.

The following two sections detail the derivations of the normal refinement acceleration procedure in $\mathbb{R}^2$ and $\mathbb{R}^3$.

6.4.1 2D Derivation

Fig. 6.6 depicts the geometry associated to a single spring connection between the boundaries of two adjacent PLIC facets p and q. The spring attachment points, $\boldsymbol{x}_{pq}$ and $\boldsymbol{x}_{qp}$, are to be determined as a function of θ_p and θ_q, respectively. For the λ^{th} sub-iteration of the k^{th} iteration of the normal refinement procedure in $\mathbb{R}^2$, the unit normal, $\boldsymbol{n}_p^{k,\lambda}$, of each PLIC facet p is given by Eq. (6.8), where $\mathcal{M}$ is the total number of PLIC facets representing the liquid-gas interface[15].

$$\boldsymbol{n}_p^{k,\lambda} \equiv \boldsymbol{n}_p^{k,\lambda}(\theta_p^\lambda) = \begin{bmatrix} \cos(\theta_p^\lambda) \\ \sin(\theta_p^\lambda) \end{bmatrix}, \quad \forall p \in \{1, 2, ..., \mathcal{M}\}. \tag{6.8}$$

For clarity, noting that all variables to be computed in the derivations to follow are defined at the λ^{th} sub-iteration of the k^{th} iteration, the k and λ superscripts are suppressed for the remainder of this section, unless the specific context requires otherwise.

As shown in the figure, the definition of the unit normal in Eq. (6.8) employs a local polar coordinate system whose origin is located at the centroid, $\boldsymbol{x}_p^k$, of PLIC facet p, and whose angle is computed relative to an axis that is parallel to the x-axis in the global coordinate system (see dashed lines). For each iteration $k \geq k_{min}$, the acceleration procedure is initialised (at $\lambda = 0$) by computing the corresponding polar coordinates of the unit normals computed in iteration $(k-1)$.

[14] This implies that the acceleration procedure may damp out the highest frequency capillary waves to an extent. This is to be investigated in future work.

[15] Although cases with multiple *disconnected* interfaces are not investigated in this work, the proposed acceleration scheme applies generically to all such cases.

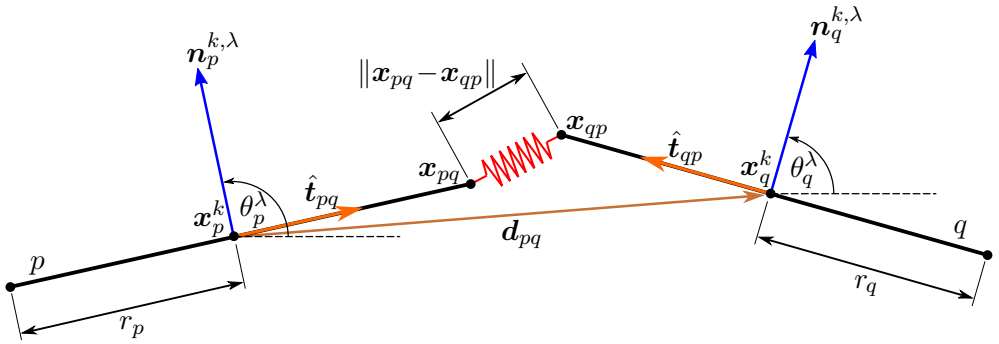


Figure 6.6: Close-up of the geometry of a single spring connection between neighbouring PLIC facets p and q in $\mathbb{R}^2$.

Considering the computation of the point $\boldsymbol{x}_{pq}$ first, the direction vector $\boldsymbol{d}_{pq}$ is first defined as

$$\boldsymbol{d}_{pq} = \boldsymbol{x}_q^k - \boldsymbol{x}_p^k, \tag{6.9}$$

where $\boldsymbol{x}_p^k$ and $\boldsymbol{x}_q^k$ respectively, are the centroids of PLIC facets p and its neighbouring PLIC facet q, as constructed at iteration k (note that the PLIC facet centroids remain invariant during the acceleration procedure). Next, the unit vector $\hat{\boldsymbol{t}}_{pq}$, from the centroid $\boldsymbol{x}_p^k$ to the spring attachment point $\boldsymbol{x}_{pq}$, is defined using Eq. (6.8) as

$$\hat{\boldsymbol{t}}_{pq} \equiv \hat{\boldsymbol{t}}_{pq}(\theta_p, \boldsymbol{d}_{pq}) = \boldsymbol{n}_p(\theta_p + \varsigma_{pq}\tfrac{\pi}{2}) = \begin{bmatrix} \cos(\theta_p + \varsigma_{pq}\frac{\pi}{2}) \\ \sin(\theta_p + \varsigma_{pq}\frac{\pi}{2}) \end{bmatrix}, \tag{6.10}$$

where ς_{pq} takes on the appropriate sign such that $\hat{\boldsymbol{t}}_{pq}$ can be computed by simply rotating $\boldsymbol{n}_p$ by $\pi/2$ radians towards PLIC facet q (ς_{pq} would be negative in Fig. 6.6). It is thus defined as

$$\varsigma_{pq} = \text{sgn}(\boldsymbol{n}_p \times \boldsymbol{d}_{pq}) = \text{sgn}(n_{p,x}d_{pq,y} - n_{p,y}d_{pq,x}), \tag{6.11}$$

given the respective x- and y-components of $\boldsymbol{n}_p$ and $\boldsymbol{d}_{pq}$. Next, given the half-length, r_p, of PLIC facet p (see Fig. 6.6), the spring attachment point $\boldsymbol{x}_{pq}$ can be computed as

$$\boldsymbol{x}_{pq} \equiv \boldsymbol{x}_{pq}(\theta_p, \boldsymbol{d}_{pq}, r_p) = \boldsymbol{x}_p^k + r_p\hat{\boldsymbol{t}}_{pq}. \tag{6.12}$$

Similarly, the neighbouring spring attachment point $\boldsymbol{x}_{qp}$ can be straightforwardly computed by swapping the p and q subscripts in Eqs. (6.9) – (6.12).

The total spring energy in the resulting system can be assumed to be proportional to the sum of the quadrances (squared distances) between adjacent spring attachment points. For the purpose of defining this total spring energy, the sets $\mathfrak{N}_p$ and $\mathfrak{N}_p^+$ are first defined for each PLIC facet $p \in \{1, 2, ..., \mathcal{M}\}$, such that[16]

$$\mathfrak{N}_p = \big\{q \in \{1, 2, ..., \mathcal{M}\} \backslash \{p\} \mid q \text{ is a neighbour of } p \text{ in the aligned stencil}\big\}, \tag{6.13}$$

[16] See Section 4 for the procedure for identifying the neighbours in $\mathfrak{N}_p$. Further, note that the given definition of $\mathfrak{N}_p^+$ allows for $\mathfrak{N}_p^+ = \mathfrak{N}_p$, $\mathfrak{N}_p^+ \subset \mathfrak{N}_p$, or $\mathfrak{N}_p^+ = \varnothing$.

$$\mathfrak{N}_p^+ = \big\{q \in \mathfrak{N}_p \mid q > p\big\}. \tag{6.14}$$

An example of $\mathfrak{N}_p$ is the set $\{q_1, q_2\}$ in Fig. 6.5(a). The set of all pairs of neighbouring PLIC facets associated to each spring connection is defined by

$$\mathfrak{P} = \big\{(p,q) \mid p \in \{1,2,...,\mathcal{M}\},\ q \in \mathfrak{N}_p^+\big\}. \tag{6.15}$$

The total spring energy, $Q : \mathbb{R}^{\mathcal{M}} \to \mathbb{R}$, can now be defined as a function of the vector of angles, $\boldsymbol{\theta} = [\theta_1\ \theta_2\ ...\ \theta_{\mathcal{M}}]^T$, as follows:

$$\begin{aligned} Q(\boldsymbol{\theta}) &= \sum_{(p,q)\in\mathfrak{P}} \|\boldsymbol{x}_{pq} - \boldsymbol{x}_{qp}\|^2 \\ &= \sum_{(p,q)\in\mathfrak{P}} \big(r_p^2 \hat{\boldsymbol{t}}_{pq}\cdot\hat{\boldsymbol{t}}_{pq} + r_q^2 \hat{\boldsymbol{t}}_{qp}\cdot\hat{\boldsymbol{t}}_{qp} - 2r_p r_q \hat{\boldsymbol{t}}_{pq}\cdot\hat{\boldsymbol{t}}_{qp} + (\boldsymbol{d}_{pq} + 2r_q\hat{\boldsymbol{t}}_{qp} - 2r_p\hat{\boldsymbol{t}}_{pq})\cdot\boldsymbol{d}_{pq}\big), \end{aligned} \tag{6.16}$$

where r_q is the half-length of PLIC facet q (see Fig. 6.6) and $\hat{\boldsymbol{t}}_{qp}$ is computed by swapping the p and q subscripts in Eq. (6.10). The derivations for Eq. (6.16) and the equations to follow can be found in Appendix C.2.

Eq. (6.16) is minimised using a modified Newton's method, which requires the gradient and Hessian of Q with respect to $\boldsymbol{\theta}$. The gradient of Q is expressed as

$$\nabla Q \equiv \nabla Q(\boldsymbol{\theta}) = [\partial Q/\partial\theta_1\ ...\ \partial Q/\partial\theta_{\mathcal{M}}]^T.$$

It is important to point out that, given the symmetry between the p and q terms in Eq. (6.16), and given that $(p,q) \in \mathfrak{P} \Rightarrow (q,p) \notin \mathfrak{P}$, the partial derivatives of Q with respect to θ_p must account for the contributions from all $q \in \mathfrak{N}_p$ (instead of only from all $q \in \mathfrak{N}_p^+$, as per Eq. (6.15)). Therefore, the first-order partial derivatives of Q are of the form

$$\frac{\partial Q}{\partial\theta_p} = 2r_p \sum_{q\in\mathfrak{N}_p} \frac{\partial\hat{\boldsymbol{t}}_{pq}}{\partial\theta_p}\cdot\boldsymbol{s}_{pq}, \quad \forall p \in \{1,2,...,\mathcal{M}\}, \tag{6.17}$$

where

$$\boldsymbol{s}_{pq} = r_p\hat{\boldsymbol{t}}_{pq} - r_q\hat{\boldsymbol{t}}_{qp} - \boldsymbol{d}_{pq}. \tag{6.18}$$

Further, using Eq. (6.10), the first-order partial derivatives of $\hat{\boldsymbol{t}}_{pq}$ in Eq. (6.17) are computed as

$$\frac{\partial\hat{\boldsymbol{t}}_{pq}}{\partial\theta_p} = \begin{bmatrix} -\sin(\theta_p + \varsigma_{pq}\frac{\pi}{2}) \\ \cos(\theta_p + \varsigma_{pq}\frac{\pi}{2}) \end{bmatrix}, \tag{6.19}$$

where ς_{pq} is defined as per Eq. (6.11). Next, using Eq. (6.16), the second-order partial derivatives of Q in the $\mathcal{M}\times\mathcal{M}$ Hessian matrix, $\mathbf{H} \equiv \nabla\otimes\nabla Q$, are expressed as

$$H_{pp} = \frac{\partial^2 Q}{\partial\theta_p^2} = 2r_p \sum_{q\in\mathfrak{N}_p} \left(\frac{\partial^2\hat{\boldsymbol{t}}_{pq}}{\partial\theta_p^2}\cdot\boldsymbol{s}_{pq} + r_p\left\|\frac{\partial\hat{\boldsymbol{t}}_{pq}}{\partial\theta_p}\right\|^2 \right), \quad \forall p \in \{1,2,...,\mathcal{M}\},$$

and

$$H_{pq} = H_{qp} = \frac{\partial^2 Q}{\partial\theta_p\partial\theta_q} = \frac{\partial^2 Q}{\partial\theta_q\partial\theta_p} = -2r_p r_q \left(\frac{\partial\hat{\boldsymbol{t}}_{pq}}{\partial\theta_p}\cdot\frac{\partial\hat{\boldsymbol{t}}_{qp}}{\partial\theta_q} \right), \quad \forall p \in \{1,2,...,\mathcal{M}\},\ \forall q \in \mathfrak{N}_p.$$

Using Eq. (6.19), the second-order partial derivatives of $\hat{\boldsymbol{t}}_{pq}$ in the above equations are

$$\frac{\partial^2 \hat{\boldsymbol{t}}_{pq}}{\partial \theta_p^2} = -\begin{bmatrix} \cos(\theta_p + \varsigma_{pq}\frac{\pi}{2}) \\ \sin(\theta_p + \varsigma_{pq}\frac{\pi}{2}) \end{bmatrix},$$

where ς_{pq} is defined as per Eq. (6.11). The iterative scheme, based on a modified Newton's method, for minimising Eq. (6.16) is detailed for $\mathbb{R}^2$ and $\mathbb{R}^3$ in Section 6.4.3. The above acceleration procedure is extended to $\mathbb{R}^3$ next.

6.4.2 3D Derivation

For the λ^{th} sub-iteration of the k^{th} iteration of the normal refinement procedure in $\mathbb{R}^3$, the unit normal, $\boldsymbol{n}_p^{k,\lambda}$, of each PLIC facet p is given by Eq. (6.20), where $\mathcal{M}$ is again the total number of PLIC facets representing the liquid-gas interface[17].

$$\boldsymbol{n}_p^{k,\lambda} \equiv \boldsymbol{n}_p^{k,\lambda}(\theta_p^\lambda, \phi_p^\lambda) = \begin{bmatrix} \cos(\theta_p^\lambda)\sin(\phi_p^\lambda) \\ \sin(\theta_p^\lambda)\sin(\phi_p^\lambda) \\ \cos(\phi_p^\lambda) \end{bmatrix}, \quad \forall p \in \{1, 2, ..., \mathcal{M}\}. \tag{6.20}$$

The above definition employs a local spherical coordinate system whose origin is located at the centroid $\boldsymbol{x}_p^k$ of p, and whose axes (with respect to which θ_p^λ and ϕ_p^λ are measured) are parallel to those of the global (x, y, z) coordinate system. Once again, for each iteration $k \geq k_{min}$, the acceleration procedure is initialised (at $\lambda = 0$) by computing the corresponding spherical coordinates of the PLIC facet normals computed in iteration $(k-1)$. The k and λ superscripts shall again be suppressed for the remainder of this section, unless otherwise specified.

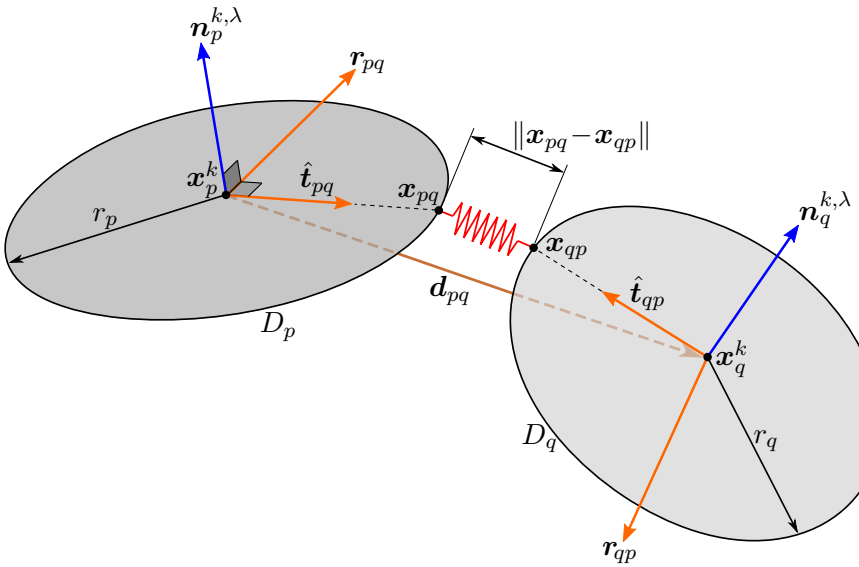


Figure 6.7: Geometry of a single spring connection between the disk representations of neighbouring PLIC facets p and q in $\mathbb{R}^3$.

[17] Although interfaces with negative Gaussian curvature are not investigated in this work, the proposed acceleration scheme applies generically to such cases.

Next, let PLIC facet p be represented by the disk[18] D_p (see Fig. 6.7) which lies in the plane of p, is centred at the area centroid $\boldsymbol{x}_p^k$, and has a radius of

$$r_p = \beta\sqrt{\frac{A_p}{\pi}}. \tag{6.21}$$

Here, A_p is the surface area of PLIC facet p, and the global constant $\beta \in (0,1)$ is a heuristic scaling factor that ensures that r_p is small enough such that, for all p, D_p lies in the interior of p ($\beta = 0.5$ is used in this work). This avoids the additional computational cost in determining the largest inscribed circle in each PLIC facet explicitly.

At face value it would seem reasonable to attach each spring to the nearest points between neighbouring disks. However, for each pair of neighbouring disks, two additional independent variables would need to be introduced (in addition to θ_p and ϕ_p) to parametrise the boundaries of each disk in the pair. This would significantly increase the number of variables in the system. To avoid this, the spring attachment point, $\boldsymbol{x}_{pq}$, is defined as the intersection between the boundary of disk D_p and the plane containing — (i) the normal $\boldsymbol{n}_p^{k,\lambda}$ and (ii) the direction vector $\boldsymbol{d}_{pq}$.

Equivalent to Eq. (6.10) in $\mathbb{R}^2$, the unit vector $\hat{\boldsymbol{t}}_{pq}$ from the centroid $\boldsymbol{x}_p^k$ to the spring attachment point $\boldsymbol{x}_{pq}$ is here computed as

$$\hat{\boldsymbol{t}}_{pq} \equiv \hat{\boldsymbol{t}}_{pq}(\theta_p, \phi_p, \boldsymbol{d}_{pq}) \;=\; \frac{\boldsymbol{r}_{pq}\times\boldsymbol{n}_p}{\|\boldsymbol{r}_{pq}\times\boldsymbol{n}_p\|} \;=\; \frac{\boldsymbol{r}_{pq}\times\boldsymbol{n}_p}{\|\boldsymbol{r}_{pq}\|\|\boldsymbol{n}_p\|\sin\left(\frac{\pi}{2}\right)} \;=\; \frac{\boldsymbol{r}_{pq}\times\boldsymbol{n}_p}{\|\boldsymbol{r}_{pq}\|}, \tag{6.22}$$

where

$$\boldsymbol{r}_{pq} = \boldsymbol{n}_p \times \boldsymbol{d}_{pq}, \tag{6.23}$$

and $\boldsymbol{d}_{pq}$ is computed as per Eq. (6.9). This allows for $\boldsymbol{x}_{pq}$ to be computed as

$$\boldsymbol{x}_{pq} \equiv \boldsymbol{x}_{pq}(\theta_p, \phi_p, \boldsymbol{d}_{pq}, r_p) = \boldsymbol{x}_p^k + r_p\hat{\boldsymbol{t}}_{pq}. \tag{6.24}$$

Likewise, $\boldsymbol{x}_{qp}$ can be computed by swapping the p and q subscripts in Eqs. (6.22) – (6.24). As was the case in $\mathbb{R}^2$, the total spring energy in the system is defined as the sum of the squared distances between adjacent attachment points. First, the definition of $\mathfrak{N}_p$ in Eq. (6.13) is modified slightly to give

$$\mathfrak{N}_p = \big\{q \in \{1,2,...,\mathcal{M}\}\backslash\{p\} \mid q \text{ is a lateral neighbour of } p \text{ in the aligned stencil}\big\},$$

where *lateral* neighbours are in contrast to *diagonal* neighbours[19], which would otherwise include the corner PLIC facets in each aligned stencil as well (*e.g.* the PLIC facets q_2, q_4, q_6, and q_8 in Fig. 6.3(b)). An example of $\mathfrak{N}_p$ is the set $\{q_1, q_2, q_3, q_4\}$ in Fig. 6.5(a). The definitions of $\mathfrak{N}_p^+$ and $\mathfrak{P}$ remain the same as those in Eqs. (6.14) and (6.15), respectively.

By stacking the vectors of angles, $\boldsymbol{\theta} = [\theta_1\ \theta_2\ ...\ \theta_{\mathcal{M}}]^T$ and $\boldsymbol{\phi} = [\phi_1\ \phi_2\ ...\ \phi_{\mathcal{M}}]^T$, to form a single vector $\boldsymbol{\gamma} = [\boldsymbol{\theta}^T\ \boldsymbol{\phi}^T]^T$, the function, $Q : \mathbb{R}^{2\mathcal{M}} \to \mathbb{R}$, describing the total spring energy in

[18] In all 3D cases performed in Chapter 9, the largest PLIC facet aspect ratio observed was $1:1.386$. The applicability of the disk representation to larger aspect ratios is to be investigated in future work.

[19] A *lateral* neighbour of PLIC facet p in the aligned stencil is associated to an interface column whose elements share *faces* with the elements of the interface column to which p is associated. Likewise, a *diagonal* neighbour of PLIC facet p is associated to an interface column whose elements share *edges* with the elements of p's interface column.

the system can be defined as

$$Q(\boldsymbol{\gamma}) = \sum_{(p,q)\in\mathfrak{P}} \|\boldsymbol{x}_{pq} - \boldsymbol{x}_{qp}\|^2, \tag{6.25}$$

whose expansion is symbolically equivalent to Eq. (6.16). Once again, the derivations for the above equation and the equations to follow can be found in Appendix C.2.

Similar to the case in $\mathbb{R}^2$, Eq. (6.25) is minimised using a modified Newton's method, which requires the gradient and Hessian of Q. The gradient of Q is here of the form

$$\nabla Q = [\partial Q/\partial\theta_1 \; ... \; \partial Q/\partial\theta_{\mathcal{M}} \; \partial Q/\partial\phi_1 \; ... \; \partial Q/\partial\phi_{\mathcal{M}}]^T,$$

where the first-order partial derivatives[20] of Q with respect to all θ_p and ϕ_p are expressed as

$$\frac{\partial Q}{\partial\gamma_p} = 2r_p \sum_{q\in\mathfrak{N}_p} \frac{\partial\hat{\boldsymbol{t}}_{pq}}{\partial\gamma_p}\cdot\boldsymbol{s}_{pq}, \quad \gamma\in\{\theta,\phi\}, \; \forall p\in\{1,2,...,\mathcal{M}\}, \tag{6.26}$$

where $\boldsymbol{s}_{pq}$ is computed as per Eq. (6.18), The first-order partial derivatives of $\hat{\boldsymbol{t}}_{pq}$ in the above equation are expressed as

$$\frac{\partial\hat{\boldsymbol{t}}_{pq}}{\partial\gamma_p} = \frac{\|\boldsymbol{r}_{pq}\|\big(\boldsymbol{v}^{\gamma}_{pq}\times\boldsymbol{n}_p + \boldsymbol{r}_{pq}\times\frac{\partial\boldsymbol{n}_p}{\partial\gamma_p}\big) - (\boldsymbol{r}_{pq}\cdot\boldsymbol{v}^{\gamma}_{pq})\hat{\boldsymbol{t}}_{pq}}{\|\boldsymbol{r}_{pq}\|^2}, \quad \gamma\in\{\theta,\phi\}, \tag{6.27}$$

where

$$\boldsymbol{v}^{\gamma}_{pq} = \frac{\partial\boldsymbol{r}_{pq}}{\partial\gamma_p} = \frac{\partial\boldsymbol{n}_p}{\partial\gamma_p}\times\boldsymbol{d}_{pq}, \tag{6.28}$$

and the first-order partial derivatives of $\boldsymbol{n}_p$ are

$$\frac{\partial\boldsymbol{n}_p}{\partial\theta_p} = \begin{bmatrix} -\sin\theta_p\sin\phi_p \\ \cos\theta_p\sin\phi_p \\ 0 \end{bmatrix} \text{ and } \frac{\partial\boldsymbol{n}_p}{\partial\phi_p} = \begin{bmatrix} \cos\theta_p\cos\phi_p \\ \sin\theta_p\cos\phi_p \\ -\sin\phi_p \end{bmatrix}.$$

Next, using Eq. (6.26), the second-order partial derivatives of Q in the $(2\mathcal{M})\times(2\mathcal{M})$ Hessian matrix, $\mathbf{H}\equiv\nabla\otimes\nabla Q$, are to be computed. To this end, by first defining a row/column offset index $\mathcal{O}$ as

$$\mathcal{O} = \begin{cases} 0 & \text{if } \gamma=\theta \\ \mathcal{M} & \text{otherwise} \end{cases},$$

the second-order partial derivatives and their corresponding positions in $\mathbf{H}$ can be expressed for all $p\in\{1,2,...,\mathcal{M}\}$, $q\in\mathfrak{N}_p$, and $\gamma\in\{\theta,\phi\}$ as follows:

[20] The reader is here reminded that, given the symmetry between the p and q terms in Eq. (6.25), the partial derivatives of Q in Eq. (6.26) are defined such that the contributions from all $q\in\mathfrak{N}_p$ are accounted for.

$$H_{p+\mathcal{O},p+\mathcal{O}} = \frac{\partial^2 Q}{\partial \gamma_p^2} = 2r_p \sum_{q\in\mathfrak{N}_p} \left(\frac{\partial^2 \hat{\boldsymbol{t}}_{pq}}{\partial \gamma_p^2} \cdot \boldsymbol{s}_{pq} + r_p \left\| \frac{\partial \hat{\boldsymbol{t}}_{pq}}{\partial \gamma_p} \right\|^2 \right),$$

$$H_{p+\mathcal{O},q+\mathcal{O}} = H_{q+\mathcal{O},p+\mathcal{O}} = \frac{\partial^2 Q}{\partial \gamma_p \partial \gamma_q} = \frac{\partial^2 Q}{\partial \gamma_q \partial \gamma_p} = -2r_p r_q \left(\frac{\partial \hat{\boldsymbol{t}}_{pq}}{\partial \gamma_p} \cdot \frac{\partial \hat{\boldsymbol{t}}_{qp}}{\partial \gamma_q} \right),$$

$$H_{p,p+\mathcal{M}} = H_{p+\mathcal{M},p} = \frac{\partial^2 Q}{\partial \theta_p \partial \phi_p} = \frac{\partial^2 Q}{\partial \phi_p \partial \theta_p} = 2r_p \sum_{q\in\mathfrak{N}_p} \left(\frac{\partial^2 \hat{\boldsymbol{t}}_{pq}}{\partial \theta_p \partial \phi_p} \cdot \boldsymbol{s}_{pq} + r_p \frac{\partial \hat{\boldsymbol{t}}_{pq}}{\partial \theta_p} \cdot \frac{\partial \hat{\boldsymbol{t}}_{pq}}{\partial \phi_p} \right),$$

$$H_{p,q+\mathcal{M}} = H_{q+\mathcal{M},p} = \frac{\partial^2 Q}{\partial \theta_p \partial \phi_q} = \frac{\partial^2 Q}{\partial \phi_q \partial \theta_p} = -2r_p r_q \left(\frac{\partial \hat{\boldsymbol{t}}_{pq}}{\partial \theta_p} \cdot \frac{\partial \hat{\boldsymbol{t}}_{qp}}{\partial \phi_q} \right),$$

and

$$H_{p+\mathcal{M},q} = H_{q,p+\mathcal{M}} = \frac{\partial^2 Q}{\partial \phi_p \partial \theta_q} = \frac{\partial^2 Q}{\partial \theta_q \partial \phi_p} = -2r_p r_q \left(\frac{\partial \hat{\boldsymbol{t}}_{pq}}{\partial \phi_p} \cdot \frac{\partial \hat{\boldsymbol{t}}_{qp}}{\partial \theta_q} \right),$$

where the first-order partial derivatives of $\hat{\boldsymbol{t}}_{qp}$ are computed by swapping the p and q subscripts in Eq. (6.27). Additionally, by first suppressing the p and pq subscripts for brevity (unless otherwise specified), the second-order partial derivatives of $\hat{\boldsymbol{t}}_{pq}$ in the above equations can be computed as

$$\frac{\partial^2 \hat{\boldsymbol{t}}_{pq}}{\partial \gamma_p^2} = \frac{1}{\|\boldsymbol{r}\|^2} \Big[\|\boldsymbol{r}\| \big(\boldsymbol{r} \times \boldsymbol{n}_{\gamma\gamma} + 2\boldsymbol{v}^\gamma \times \boldsymbol{n}_\gamma + \boldsymbol{w}^\gamma \times \boldsymbol{n} \big) - \big(\boldsymbol{r} \cdot \boldsymbol{w}^\gamma + \|\boldsymbol{v}^\gamma\|^2 \big) \hat{\boldsymbol{t}} - \big(\boldsymbol{r} \cdot \boldsymbol{v}^\gamma \big) \hat{\boldsymbol{t}}_\gamma \Big] + \frac{\boldsymbol{r} \cdot \boldsymbol{v}^\gamma}{\|\boldsymbol{r}\|^4} \Big[2\big(\boldsymbol{r} \cdot \boldsymbol{v}^\gamma \big) \hat{\boldsymbol{t}} - \|\boldsymbol{r}\| \big(\boldsymbol{r} \times \boldsymbol{n}_\gamma + \boldsymbol{v}^\gamma \times \boldsymbol{n} \big) \Big],$$

$$\frac{\partial^2 \hat{\boldsymbol{t}}_{pq}}{\partial \theta_p \partial \phi_p} = \frac{\partial^2 \hat{\boldsymbol{t}}_{pq}}{\partial \phi_p \partial \theta_p} = \frac{1}{\|\boldsymbol{r}\|^2} \Big[\|\boldsymbol{r}\| \big(\boldsymbol{r} \times \boldsymbol{n}_{\theta\phi} + \boldsymbol{v}^\theta \times \boldsymbol{n}_\phi + \boldsymbol{v}^\phi \times \boldsymbol{n}_\theta + (\boldsymbol{n}_{\theta\phi} \times \boldsymbol{d}) \times \boldsymbol{n} \big) - \big((\boldsymbol{n}_{\theta\phi} \times \boldsymbol{d}) \cdot \boldsymbol{r} + \boldsymbol{v}^\theta \cdot \boldsymbol{v}^\phi \big) \hat{\boldsymbol{t}} - \big(\boldsymbol{r} \cdot \boldsymbol{v}^\theta \big) \hat{\boldsymbol{t}}_\phi \Big] + \frac{\boldsymbol{r} \cdot \boldsymbol{v}^\phi}{\|\boldsymbol{r}\|^4} \Big[2\big(\boldsymbol{r} \cdot \boldsymbol{v}^\theta \big) \hat{\boldsymbol{t}} - \|\boldsymbol{r}\| \big(\boldsymbol{r} \times \boldsymbol{n}_\theta + \boldsymbol{v}^\theta \times \boldsymbol{n} \big) \Big],$$

where

$$\boldsymbol{w}^\gamma \equiv \boldsymbol{w}_{pq}^\gamma = \frac{\partial \boldsymbol{v}_{pq}}{\partial \gamma_p} = \frac{\partial^2 \boldsymbol{n}_p}{\partial \gamma_p^2} \times \boldsymbol{d}_{pq}, \tag{6.29}$$

and

$$\boldsymbol{n}_\gamma \equiv \frac{\partial \boldsymbol{n}_p}{\partial \gamma_p}, \quad \boldsymbol{n}_{\gamma\gamma} \equiv \frac{\partial^2 \boldsymbol{n}_p}{\partial \gamma_p^2}, \quad \boldsymbol{n}_{\theta\phi} \equiv \frac{\partial^2 \boldsymbol{n}_p}{\partial \theta_p \partial \phi_p}, \text{ and } \hat{\boldsymbol{t}}_\gamma \equiv \frac{\partial \hat{\boldsymbol{t}}_{pq}}{\partial \gamma_p}.$$

Further, the second-order partial derivatives of $\boldsymbol{n}_p$ are computed as

$$\frac{\partial^2 \boldsymbol{n}_p}{\partial \theta_p^2} = \begin{bmatrix} -\cos\theta_p \sin\phi_p \\ -\sin\theta_p \sin\phi_p \\ 0 \end{bmatrix}, \quad \frac{\partial^2 \boldsymbol{n}_p}{\partial \phi_p^2} = \begin{bmatrix} -\cos\theta_p \sin\phi_p \\ -\sin\theta_p \sin\phi_p \\ -\cos\phi_p \end{bmatrix},$$

and

$$\frac{\partial^2 \boldsymbol{n}_p}{\partial\theta_p\phi_p} = \frac{\partial^2 \boldsymbol{n}_p}{\partial\phi_p\theta_p} = \begin{bmatrix} -\sin\theta_p\cos\phi_p \\ \cos\theta_p\cos\phi_p \\ 0 \end{bmatrix}.$$

The iterative spring energy minimisation routine that is generic to $\mathbb{R}^2$ and $\mathbb{R}^3$ is detailed next.

6.4.3 Total Spring Energy Minimisation Routine

Referring to the definitions of the vectors $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$ in the previous two sections, a vector of variables that is generic to $\mathbb{R}^2$ and $\mathbb{R}^3$ is defined as

$$\boldsymbol{\Gamma} \equiv \begin{cases} \boldsymbol{\theta} & \text{in } \mathbb{R}^2 \\ \boldsymbol{\gamma} & \text{in } \mathbb{R}^3 \end{cases},$$

Eqs. (6.16) and (6.25) can be minimised iteratively, using the following routine:

$$\boldsymbol{\Gamma}_{\lambda+1} = \boldsymbol{\Gamma}_\lambda + \Delta\boldsymbol{\Gamma}_\lambda, \tag{6.30}$$

where $\Delta\boldsymbol{\Gamma}_\lambda$ is the Newton descent direction, which is solved from the following linear system:

$$\mathbf{H}^+\Delta\boldsymbol{\Gamma}_\lambda = -\nabla Q. \tag{6.31}$$

In the above, $\nabla Q \equiv \nabla Q(\boldsymbol{\Gamma}_\lambda)$, and $\mathbf{H}^+ \equiv \mathbf{H}^+(\boldsymbol{\Gamma}_\lambda)$ is a positive-definite modification of the Hessian matrix $\mathbf{H}$ (see previous two sections), employed to guarantee that $\Delta\boldsymbol{\Gamma}_\lambda$ is a descent direction [39], and is defined as

$$\mathbf{H}^+ = \mathbf{H} + \delta\mathbf{I},$$

where $\mathbf{I}$ is the identity matrix, and δ is a central diagonal offset parameter defined as

$$\delta = \begin{cases} |\Delta H_{min}| + \epsilon_{min} & \text{if } \Delta H_{min} < \epsilon_{min} \\ 0 & \text{otherwise} \end{cases},$$

where $\epsilon_{min} = 1\times10^{-10}$ in this work, and in the d-dimensional case,

$$\Delta H_{min} = \min\big\{\Delta H_p \mid p \in \{1, 2, ..., (d-1)\mathcal{M}\}\big\}, \quad \Delta H_p = H_{pp} - \sum_{\forall q\neq p} H_{pq}.$$

Here, as previously, $\mathcal{M}$ is the total number of PLIC facets representing the liquid-gas interface. Eq. (6.31) is solved using a biconjugate gradient (BiCG) sparse matrix solver. As mentioned previously, it is important to note that the only variables that are updated in Eq. (6.30) are those associated to central PLIC facets of *aligned stencils*. The variables of PLIC facets associated to *misaligned stencils* are first computed in a non-accelerated manner (as per Section 6.3), and then set as Dirichlet boundary conditions in Eq. (6.31) (*i.e.* by zeroing the corresponding entries in ∇Q and the corresponding rows in $\mathbf{H}^+$ with the central diagonal entries set to 1). The iterative routine in Eq. (6.30) is terminated when $\nabla Q \cdot \Delta\boldsymbol{\Gamma}_\lambda/2$ falls below a user-defined tolerance (1×10^{-8} in this work).

6.5 Summary of the Iterative Normal Refinement Procedure

This section summarises the application of all procedures detailed since Chapter 4. Algorithm 3 outlines the overall iterative PLIC facet reconstruction procedure in $\mathbb{R}^2$, where the PLIC facet normal refinement procedure in each iteration (see line 19) is outlined in Algorithm 4. Both algorithms can be straightforwardly generalised to $\mathbb{R}^3$ by extending Algorithms 1 and 2 to $\mathbb{R}^3$, as per the procedures elaborated in Chapters 4 and 5, respectively. The iterative procedure terminates when the residuals of the components of each PLIC facet normal fall below a prescribed tolerance $\mathcal{R}_{tol}$.

Algorithm 3 Iterative PLIC facet reconstruction procedure in $\mathbb{R}^2$.

Input: The residual tolerance, $\mathcal{R}_{tol}$, for the components of the PLIC facet normals.
Output: Iteratively reconstruct PLIC facets and refine their normals in each iteration.

1: **procedure** IterativePlicFacetReconstruct2D($\mathcal{R}_{tol}$)
2: **set** $\mathcal{R}_k \leftarrow \mathcal{R}_{tol}+1$ // The residual (L_2-norm) of the PLIC facet normals.
3: **set** $k \leftarrow 0$ // Iteration counter.
4: **set** $k_{min} \leftarrow -1$ // Minimum number of iterations before acceleration procedure.
5: **while** $\mathcal{R}_k > \mathcal{R}_{tol}$ **do**
6: **if** $k_{min} = -1$, **then**
7: **for all** interface vertices i, **do**
8: **set** $\boldsymbol{n}_i \leftarrow$ Compute vertex normal for current iteration, as per Eq. (4.2).
9: ConstructColumns2D($i, \boldsymbol{n}_i, \theta_B, \alpha_{tol}$) (see Algorithm 1).
10: **end**
11: **if** $k > 0$ **and** columns in iterations k and $(k-1)$ are the same, **then**
12: **set** $k_{min} \leftarrow k$
13: **end**
14: **end**
15: **set** $Z \leftarrow$ The array of all columns constructed in iteration k.
16: **for all** $\zeta_p \in Z$ **do**
17: Construct PLIC facet p in column ζ_p as per the procedures in Chapter 5.
18: **end**
19: **set** $\mathcal{R}_k \leftarrow$ PlicFacetNormalRefine2D(Z, k, k_{min})
20: **set** $k \leftarrow k+1$
21: **end**

Algorithm 4 PLIC facet normal refinement in a single iteration k of Algorithm 3.

Input: Array of interface columns Z, current refinement iteration k, acceleration scheme threshold iteration k_{min}.

Output: Residual of the refined normals of all PLIC facets constructed in iteration k.

1: **procedure** PLICFACETNORMALREFINE2D(Z, k, k_{min})
2: **set** $\mathcal{R}_k \leftarrow 0$
3: **if** $k_{min} = -1$, **then** // Standard normal refinement procedure
4: **for all** $iter \in \{1, 2\}$, **do** // Visit aligned stencils first, then misaligned stencils.
5: **for all** $\zeta_p \in Z$ **do** // Refine the normal of the PLIC facet p, associated to each ζ_p.
6: **set** Z_p, $stenType \leftarrow$ COMPUTECOLUMNSTENCIL2D(ζ_p)
7: **if** ($iter = 1$ **and** $stenType =$ 'Aligned') **or** $stenType =$ 'Mixed', **then**
8: **set** $\boldsymbol{n}_p^{k+1} \leftarrow$ Refine normal as per Eq. (6.1).
9: **else if** $iter = 2$ **and** $stenType =$ 'Misaligned', **then**
10: **set** $\boldsymbol{n}_p^{k+1} \leftarrow$ Refine normal as per Eq. (6.3) or Eq. (6.7).
11: **end**
12: **set** $\mathcal{R}_k \leftarrow \mathcal{R}_k + \|\boldsymbol{n}_p^{k+1} - \boldsymbol{n}_p^k\|^2$
13: **end**
14: **end**
15: **else** // Normal refinement acceleration procedure
16: **set** $\lambda \leftarrow 0$ // Acceleration procedure iteration counter.
17: **set** $\mathcal{R}_\lambda \leftarrow 1$
18: **while** $\mathcal{R}_\lambda > 1 \times 10^{-8}$ **do**
19: **do** Compute ∇Q, $\mathbf{H}^+$, and $\Delta\boldsymbol{\Gamma}_\lambda$ (seeEq. (6.31)).
20: **for all** $iter \in \{1, 2\}$, **do** // Visit aligned stencils first, then misaligned stencils.
21: **for all** $\zeta_p \in Z$ **do**
22: **set** Z_p, $stenType \leftarrow$ COMPUTECOLUMNSTENCIL2D(ζ_p)
23: **if** ($iter = 1$ **and** $stenType =$ 'Aligned'), **then**
24: **set** $\boldsymbol{n}_p^{k,\lambda+1} \leftarrow$ Refine normal as per Eq. (6.30) in Section 6.4.
25: **else if** $iter = 2$ **and** $stenType =$ 'Misaligned', **then**
26: **set** $\boldsymbol{n}_p^{k,\lambda+1} \leftarrow$ Refine normal as per Section 6.3.
27: **end**
28: **end**
29: **end**
30: **set** $\mathcal{R}_\lambda \leftarrow \frac{1}{2}\nabla Q \cdot \Delta\boldsymbol{\Gamma}_\lambda$
31: **set** $\lambda \leftarrow \lambda + 1$
32: **end**
33: **for all** $\zeta_p \in Z$ **do**
34: **set** $\boldsymbol{n}_p^{k+1} \leftarrow \boldsymbol{n}_p^{k,\lambda+1}$
35: **set** $\mathcal{R}_k \leftarrow \mathcal{R}_k + \|\boldsymbol{n}_p^{k+1} - \boldsymbol{n}_p^k\|^2$
36: **end**
37: **end**
38: **set** $\mathcal{R}_k \leftarrow \sqrt{\frac{\mathcal{R}_k}{size(Z)}}$

The reader is here reminded that, for all iterations $k < k_{min}$, it is necessary to update the normals of all interface vertices as per Eq. (4.2) at the beginning of each iteration (see line 8 in Algorithm 3). This is required because the normals computed at the interface vertices (which,

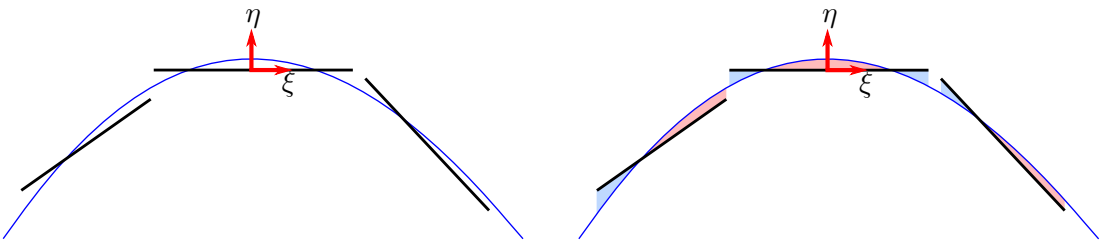
in the case of multiple overlapping columns, is different to those of the PLIC facets) inform the direction in which interface columns are to be constructed in the current iteration.

The purpose of all components of the interface reconstruction scheme detailed thus far is to prepare the PLIC representation of the interface for higher-order interface reconstruction, which is detailed in the next chapter.

Chapter 7

PLIC Facet Based Least Squares Polynomial Fitting

This chapter generalises to non-orthogonal structured grids, the least squares polynomial fitting procedure [29] that was introduced in Chapter 3. The procedures detailed thus far constitute the tools required for this generalisation. To this end, polynomials of order $n \in \{2, 4\}$ shall be fitted through stencils of at least $(n+1)$ and $(n+1)(n+2)/2$ PLIC facets in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively. This is for the dual purpose of—(i) obtaining a higher-order interface reconstruction from PLIC facets associated to aligned stencils, and (ii) computing the refined normal, $\boldsymbol{n}_p^{k+1}$, of each PLIC facet associated to the central column of a misaligned stencil. In the latter, the polynomial is fitted through the PLIC facets in the associated aligned sub-stencil (see Section 6.3).



(a) **(b)**

Figure 7.1: (a) A parabolic curve (in blue) fitted through three PLIC facets. (b) Volume conserving property of the polynomial fit – the sum of the areas shaded in light blue equals that of the areas shaded in light red.

In both $\mathbb{R}^2$ and $\mathbb{R}^3$, the local coordinate system of the polynomial fit is positioned at the centroid of the central PLIC facet in the stencil. Further, the vertical axis (η-axis in $\mathbb{R}^2$ and ψ-axis in $\mathbb{R}^3$) is oriented in the direction of the normal of the central PLIC facet (see Fig. 7.1(a)).

Similar to Chapter 3, each polynomial is fitted in a manner that minimises the signed area/volume bounded in the direction of the vertical axis, between the polynomial itself and the PLIC facets in the stencil. Fig. 7.1(b) illustrates this property in $\mathbb{R}^2$, where the parabola is fitted such that the sum of the areas of the light red regions equals that of the light blue regions. It is important to point out that, on non-orthogonal structured grids, the fitted polynomial is not strictly volume conservative over the entire stencil, as is the case on Cartesian grids. This is because, in the local coordinate system, the computed area/volume is bounded by lines/planes enclosing each PLIC facet that are not parallel to the vertical axis (η-axis in $\mathbb{R}^2$ and ψ-axis in $\mathbb{R}^3$). Therefore, the variation in the cross-section of each column with respect to the vertical direction is unaccounted for. Accounting for this variation is left for future work.

The only departure from Chapter 3 is that the fitting procedure must here account for the orientation of each PLIC facet. Despite this fact, as will be seen shortly, the fitting procedure can be simplified to the point of being orientation invariant, and thereby equivalent to the procedures detailed in Chapter 3. Sections 7.1 and 7.2 detail the aforementioned generalisation of the polynomial fitting procedure in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively.

7.1 2D Method

Similar to Section 3.2, consider a polynomial of order n, with the $N=(n+1)$ unknown coefficients $\boldsymbol{c}=[c_1\ c_2\ ...\ c_N]^T$, which is here restated as per Eq. (3.5) as

$$P_n(\xi) = \sum_{i=1}^{N} c_i\, \xi^{i-1}. \tag{7.1}$$

Letting $M \geq N$ denote the number of PLIC facets in the stencil, each PLIC facet p can be described in the local coordinate system by a linear equation of the form[21]

$$l_p(\xi) = m_p(\xi - \xi_p) + \eta_p, \quad \xi \in \Omega_p,\ p \in \{1, 2, ..., M\}, \tag{7.2}$$

where $\Omega_p=[\xi_{p_{min}}, \xi_{p_{max}}]$ is the domain of p in the local coordinate system, m_p is its slope, and (ξ_p, η_p) is an arbitrary point on the line containing p. The signed area between the PLIC facets and the polynomial can be minimised by the following functional:

$$E(\boldsymbol{c}) = \sum_{p=1}^{M} \Bigg(\int_{\Omega_p} \Big(P_n(\xi) - l_p(\xi) \Big)\, d\xi \Bigg)^2. \tag{7.3}$$

The above is a restatement of Eq. (3.6), but with the PLIC facets being described by Eq. (7.2) instead. However, when (ξ_p, η_p) is the centroid of PLIC facet p, Eqs. (3.6) and (7.3) can be straightforwardly shown to be equivalent by equating

$$\int_{\Omega_p} l_p(\xi)\, d\xi = \int_{\Omega_p} \eta_p\, d\xi \iff \xi_p = \frac{\int_{\Omega_p} \xi\, d\xi}{\int_{\Omega_p} d\xi} = \frac{\xi_{p_{min}} + \xi_{p_{max}}}{2}.$$

A simple geometric proof of the above result was illustrated in Fig. 5.1. Given the above simplification, and by setting $h_p \equiv \eta_p$ as the height of the centroid of p from the ξ-axis, the remaining equations in this section turn out to be equivalent to Eqs. (3.7) – (3.12). The linear system to be solved for the coefficients $\boldsymbol{c}$ is here restated for convenience:

$$\boldsymbol{A}\boldsymbol{c} = \boldsymbol{b},$$

$$A_{ij} = \sum_{p=1}^{M} s_{pi} s_{pj}, \quad b_i = \sum_{p=1}^{M} s_{pi} s_{p1} h_p, \quad s_{pi} = \frac{\xi_{p_{max}}^i - \xi_{p_{min}}^i}{i}, \quad i, j \in \{1, 2, ..., N\}.$$

[21] Note that this *slope-intercept* form of the linear equation pre-supposes that no PLIC facet in the stencil is parallel to the vertical axis of the local coordinate system. Therefore, the current scheme does not account for such cases.

In Section 3.2, an offset polynomial $\overline{P}_n(\xi)$ was computed *via* Eq. (3.11), that conserves the volume of fluid 1 in the central column of the stencil. This procedure is repeated above, except that the volume is conserved within the associated vertical bounds of the central PLIC facet in the local coordinate system. Therefore, the conservation in the central column is only approximately captured. The errors associated to this approximation are quantified *via* numerical experiments in Section 9.3. The generalisation of the above procedure to $\mathbb{R}^3$ is detailed next.

7.2 3D Method

Similar to Section 3.3, consider a bivariate polynomial of order n, with $N=(n+1)(n+2)/2$ unknown coefficients $\boldsymbol{c}=[c_{_1}\ c_{_2}\ ...\ c_{_N}]^T$, which is here restated as per Eq. (3.13) as

$$P_n(\xi,\eta)=\sum_{s=0}^{n}\sum_{r=0}^{n-s}c_i\xi^r\eta^s, \tag{7.4}$$

$$i=s+1+\frac{1}{2}(r+s)(r+s+1)\in\{1,2,...,N\}. \tag{7.5}$$

Once again, letting $M\geq N$ denote the number of PLIC facets in the stencil, each PLIC facet p can be described in the local coordinate system by a linear equation of the form

$$l_p(\xi,\eta)=m_{p,\xi}(\xi-\xi_p)+m_{p,\eta}(\eta-\eta_p)+\psi_p,\quad (\xi,\eta)\in\Omega_p,\ p\in\{1,2,...,M\}, \tag{7.6}$$

where Ω_p is the arbitrary polygonal domain of p in the local coordinate system, (ξ_p,η_p,ψ_p) is an arbitrary point in the plane containing p expressed in local coordinates, and $m_{p,\xi}$ and $m_{p,\eta}$ are the slopes of p along the ξ- and η-directions, respectively. The signed volume between the PLIC facets and the polynomial is minimised by the following functional:

$$E(\boldsymbol{c})=\sum_{p=1}^{M}\Bigg(\int_{\Omega_p}\Big(P_n(\xi,\eta)-l_p(\xi,\eta)\Big)d\mathcal{A}\Bigg)^2, \tag{7.7}$$

where $d\mathcal{A}\equiv d\xi d\eta$. Once again, as was shown in the previous section, the equivalence between the present method and the associated Cartesian method in Section 3.3 can be demonstrated by equating

$$\int_{\Omega_p}l_p(\xi,\eta)\ d\mathcal{A}\ =\int_{\Omega_p}\psi_p\ d\mathcal{A}, \tag{7.8}$$

which gives

$$\int_{\Omega_p}m_{p,\xi}(\xi-\xi_p)\ d\mathcal{A}+\int_{\Omega_p}m_{p,\eta}(\eta-\eta_p)\ d\mathcal{A}=0. \tag{7.9}$$

Eq. (7.9) has infinitely many solutions for ξ_p and η_p, all of which lie along the horizontal line that is parallel to PLIC facet p and passes through the point $(\xi_p,\eta_p,0)$ such that

$$\xi_p=\frac{\int_{\Omega_p}\xi\ d\mathcal{A}}{\int_{\Omega_p}d\mathcal{A}},\quad \eta_p=\frac{\int_{\Omega_p}\eta\ d\mathcal{A}}{\int_{\Omega_p}d\mathcal{A}},\quad \forall p\in\{1,2,...,M\},$$

which is the definition of the area centroid of Ω_p. Therefore, since Ω_p is the projection of PLIC facet p onto the $\xi\eta$-plane, Eq. (7.8) is satisfied when (ξ_p, η_p, ψ_p) is the area centroid of p. Given the resulting equivalence between Eqs. (3.15) and (7.7), minimising Eq. (7.7) with respect to the polynomial coefficients $\boldsymbol{c}$ gives Eq. (3.16), which is here restated as

$$\sum_{p=1}^{M} \Big(\int_{\Omega_p} \Big(P_n(\xi,\eta) - h_p \Big)\, d\mathcal{A} \Big) \Big(\int_{\Omega_p} \phi\, d\mathcal{A} \Big) = 0.$$

Here, $h_p \equiv \psi_p$ is the height of the area centroid of p above the $\xi\eta$-plane. The above equation must be solved for each $\phi \in \{\xi^r \eta^s |\ r, s \in [0, n],\ r+s \leq n\}$.

In Section 3.3, the double integrals $\int_{\Omega_p} \phi\, d\mathcal{A}$ were straightforward to compute, since the bounds of each integration domain Ω_p are always parallel to either the ξ- or η-axis on Cartesian grids. However, on non-orthogonal structured grids, Ω_p is an arbitrary polygon. Therefore, the double integrals are here evaluated *via* Green's theorem. To this end, let $N_{v,p}$ be the number of vertices bounding PLIC facet p whose vertices v are ordered counter-clockwise with respect to its normal. Further, defining $\Delta\xi_{v,p} = (\xi_{v+1,p} - \xi_{v,p})$ and $\Delta\eta_{v,p} = (\eta_{v+1,p} - \eta_{v,p})$, the required integrals for polynomials of order $n \in \{2, 4\}$ can be evaluated as shown below (the subscript p is suppressed for brevity):

$$s_{p1} = \int_{\Omega_p} d\mathcal{A} = \frac{1}{2} \sum_{v=1}^{N_{v,p}} \big(\xi_v \eta_{v+1} - \xi_{v+1} \eta_v \big),$$

$$s_{p2} = \int_{\Omega_p} \xi\, d\mathcal{A} = \frac{1}{6} \sum_{v=1}^{N_{v,p}} \big(\xi_v + \xi_{v+1} \big) \big(\xi_v \eta_{v+1} - \xi_{v+1} \eta_v \big),$$

$$s_{p3} = \int_{\Omega_p} \eta\, d\mathcal{A} = \frac{1}{6} \sum_{v=1}^{N_{v,p}} \big(\eta_v + \eta_{v+1} \big) \big(\xi_v \eta_{v+1} - \xi_{v+1} \eta_v \big),$$

$$s_{p4} = \int_{\Omega_p} \xi^2\, d\mathcal{A} = \frac{1}{6} \sum_{v=1}^{N_{v,p}} \big(\xi_v + \xi_{v+1} \big) \big(\xi_v^2 + \xi_{v+1}^2 \big) \big(\eta_v - \eta_{v+1} \big),$$

$$s_{p5} = \int_{\Omega_p} \xi\eta\, d\mathcal{A} = \frac{1}{6} \sum_{v=1}^{N_{v,p}} \big(\xi_v \eta_{v+1} - \xi_{v+1} \eta_v \big) \big(2\xi_v \eta_v + \xi_v \eta_{v+1} + \xi_{v+1} \eta_v + 2\xi_{v+1} \eta_{v+1} \big),$$

$$s_{p6} = \int_{\Omega_p} \eta^2\, d\mathcal{A} = \frac{1}{6} \sum_{v=1}^{N_{v,p}} \big(\eta_v + \eta_{v+1} \big) \big(\eta_v^2 + \eta_{v+1}^2 \big) \big(\xi_{v+1} - \xi_v \big),$$

$$s_{p7} = \int_{\Omega_p} \xi^3\, d\mathcal{A} =$$

$$\frac{1}{80} \sum_{v=1}^{N_{v,p}} \big(\eta_{v+1} - \eta_v \big) \Big(2\xi_{v+1}^2 + (1 - \sqrt{5}) \xi_v \xi_{v+1} + 2\xi_v^2 \Big) \Big(2\xi_{v+1}^2 + (1 + \sqrt{5}) \xi_v \xi_{v+1} + 2\xi_v^2 \Big),$$

$$s_{p8} = \int_{\Omega_p} \xi^2 \eta\, d\mathcal{A} = \frac{1}{60} \sum_{v=1}^{N_{v,p}} (\eta_{v+1} - \eta_v) \big(10\xi_v^3 \big(\eta_v + \eta_{v+1} \big) + 10\xi_v^2 \Delta\xi_v \big(\eta_v + 2\eta_{v+1} \big) + 5\xi_v \Delta\xi_v^2 \big(\eta_v + 3\eta_{v+1} \big) + \Delta\xi_v^3 \big(\eta_v + 4\eta_{v+1} \big) \big),$$

$$s_{p9} = \int_{\Omega_p} \xi\eta^2 \, d\mathcal{A} = \frac{1}{60} \sum_{v=1}^{N_{v,p}} (\xi_v - \xi_{v+1}) \left(10\eta_v^3(\xi_v + \xi_{v+1}) + 10\eta_v^2\Delta\eta_v(\xi_v + 2\xi_{v+1}) \right.$$
$$\left. +5\eta_v\Delta\eta_v^2(\xi_v + 3\xi_{v+1}) + \Delta\eta_v^3(\xi_v + 4\xi_{v+1})\right),$$

$$s_{p10} = \int_{\Omega_p} \eta^3 \, d\mathcal{A} =$$
$$\frac{1}{80} \sum_{v=1}^{N_{v,p}} (\xi_v - \xi_{v+1})\Big(2\eta_{v+1}^2 + (1-\sqrt{5})\eta_v\eta_{v+1} + 2\eta_v^2\Big)\Big(2\eta_{v+1}^2 + (1+\sqrt{5})\eta_v\eta_{v+1} + 2\eta_v^2\Big),$$

$$s_{p11} = \int_{\Omega_p} \xi^4 \, d\mathcal{A} = \frac{1}{30} \sum_{v=1}^{N_{v,p}} (\eta_{v+1} - \eta_v)(\xi_{v+1} + \xi_v)(\xi_{v+1}^2 + \xi_v\xi_{v+1} + \xi_v^2)(\xi_{v+1}^2 - \xi_v\xi_{v+1} + \xi_v^2),$$

$$s_{p12} = \int_{\Omega_p} \xi^3\eta \, d\mathcal{A} = \frac{1}{120} \sum_{v=1}^{N_{v,p}} (\eta_{v+1} - \eta_v)\Big(15\xi_v^4(\eta_v + \eta_{v+1}) + 20\xi_v^3\Delta\xi_v(\eta_v + 2\eta_{v+1}) +$$
$$15\xi_v^2\Delta\xi_v^2(\eta_v + 3\eta_{v+1}) + 6\xi_v\Delta\xi_v^3(\eta_v + 4\eta_{v+1}) + \Delta\xi_v^4(\eta_v + 5\eta_{v+1})\Big),$$

$$s_{p13} = \int_{\Omega_p} \xi^2\eta^2 \, d\mathcal{A} =$$
$$\frac{1}{180} \sum_{v=1}^{N_{v,p}} (\eta_{v+1} - \eta_v)\Big(20\xi_v^3(\eta_v^2 + \eta_v\eta_{v+1} + \eta_{v+1}^2) + 15\xi_v^2\Delta\xi_v(\eta_v^2 + 2\eta_v\eta_{v+1} + 3\eta_{v+1}^2) +$$
$$6\xi_v\Delta\xi_v^2(\eta_v^2 + 3\eta_v\eta_{v+1} + 6\eta_{v+1}^2) + \Delta\xi_v^3(\eta_v^2 + 4\eta_v\eta_{v+1} + 10\eta_{v+1}^2)\Big),$$

$$s_{p14} = \int_{\Omega_p} \xi\eta^3 \, d\mathcal{A} = \frac{1}{120} \sum_{v=1}^{N_{v,p}} (\xi_v - \xi_{v+1})\Big(15\eta_v^4(\xi_v + \xi_{v+1}) + 20\eta_v^3\Delta\eta_v(\xi_v + 2\xi_{v+1}) +$$
$$15\eta_v^2\Delta\eta_v^2(\xi_v + 3\xi_{v+1}) + 6\eta_v\Delta\eta_v^3(\xi_v + 4\xi_{v+1}) + \Delta\eta_v^4(\xi_v + 5\xi_{v+1})\Big),$$

and

$$s_{p15} = \int_{\Omega_p} \eta^4 \, d\mathcal{A} = \frac{1}{30} \sum_{v=1}^{N_{v,p}} (\xi_v - \xi_{v+1})(\eta_{v+1} + \eta_v)(\eta_{v+1}^2 + \eta_v\eta_{v+1} + \eta_v^2)(\eta_{v+1}^2 - \eta_v\eta_{v+1} + \eta_v^2).$$

Of the above equations, those that are not listed in Jibben *et al.* [29] (*i.e.* all s_{pi} where $i > 6$) are derived in Appendix D. Given the simplification resulting from Eq. (7.8), the remaining equations in this section are equivalent to Eqs. (3.18) – (3.21). Once again, the conservation of fluid 1 is approximately captured, and the associated errors are quantified in Section 9.3.

The application of the above interface reconstruction scheme in this work is primarily for surface tension modelling purposes. The procedure for doing so in a vertex-centred finite volume context is detailed next.

Chapter 8

Surface Tension Modelling Mathematical Formulation

8.1 Governing Equations

The incompressible momentum and mass conservation equations with surface tension for two-phase Newtonian fluid flow can be expressed as

$$\frac{\partial(\rho \boldsymbol{u})}{\partial t} + \nabla\cdot(\rho \boldsymbol{u} \otimes \boldsymbol{u}) - \nabla\cdot\big(\mu\big(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\big)\big) + \nabla p = \boldsymbol{f}_\sigma, \tag{8.1}$$
$$\nabla\cdot\boldsymbol{u} = 0,$$

where $\boldsymbol{u} \equiv \boldsymbol{u}(\boldsymbol{x}, t)$ is the velocity field, $p \equiv p(\boldsymbol{x}, t)$ is the pressure field, $\rho \equiv \rho(\boldsymbol{x}, t)$ is the fluid density, and $\mu \equiv \mu(\boldsymbol{x}, t)$ is the dynamic viscosity. Further, $\boldsymbol{f}_\sigma$ is the surface tension source term which is defined as [62]

$$\boldsymbol{f}_\sigma = \sigma\kappa\delta_s\boldsymbol{n}, \tag{8.2}$$

where σ is the surface tension coefficient, κ is the interface curvature, $\boldsymbol{n}$ is the interface normal, and δ_s is the Dirac delta function which places the surface tension contribution on the interface. Note that the contribution of the acceleration source term is not accounted for in this work. The interface is tracked by the volume fraction equation

$$\frac{\partial\alpha}{\partial t} + \nabla\cdot(\alpha\boldsymbol{u}) = 0, \tag{8.3}$$

where, given two fluids — fluid 0 and fluid 1 — α is the volume fraction of fluid 1. Similar to Tryggvason *et al.* [62], the density and viscosity in Eq. (8.1) are computed as a function of the volume fraction field as

$$\begin{aligned}\rho(\alpha) &= \alpha\rho_1 + (1-\alpha)\,\rho_0,\\ \mu(\alpha) &= \alpha\mu_1 + (1-\alpha)\,\mu_0,\end{aligned}$$

where the 0 and 1 subscripts denote the attributes associated to the respective fluids. The surface tension source term in Eq. (8.2) is approximated using the continuum-surface-force (CSF) method of Brackbill *et al.* [4] as

$$\sigma\kappa\delta_s\boldsymbol{n} \approx \sigma\kappa\nabla\alpha,$$

where $\nabla\alpha$ is the gradient of the volume fraction field. The discretisation procedures for the terms in the governing equations are detailed in the sections to follow. The reader is reminded that although both the 2D and 3D methods are detailed here, the contents of this chapter are only validated in 2D (see Sections 9.4 and 9.5). The 3D validation is left for future work as a result of time constraints in completing this thesis.

8.2 Spatial Discretisation

For spatial discretisation purposes, a vertex-centred, edge-based, finite volume method is employed in this work. An example in $\mathbb{R}^2$ of a typical median dual-cell Ω_i associated to a grid vertex i is depicted in Fig. 8.1(a).

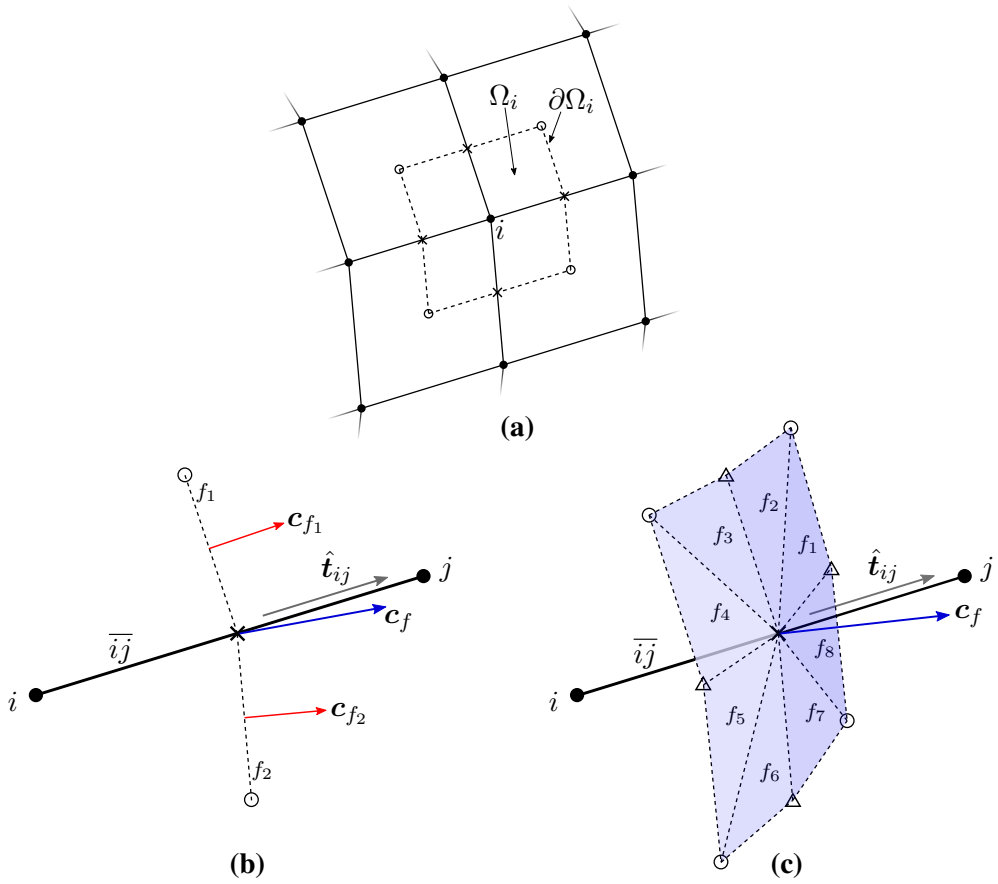


Figure 8.1: Vertex-centred, edge-based, finite volume spatial discretisation — (a) median dual-cell Ω_i with boundary $\partial\Omega_i$ in $\mathbb{R}^2$, while (b) and (c) respectively define the face coefficient vector $\boldsymbol{c}_f$ and the unit edge tangent vector $\hat{\boldsymbol{t}}_{ij}$ associated to an edge $\overline{ij}$ in $\mathbb{R}^2$ and $\mathbb{R}^3$.

The boundary, $\partial\Omega_i$, enclosing Ω_i is composed of faces f attached to each edge $\overline{ij}$ that connects vertex i to a neighbouring vertex j. Each face $f \in \partial\Omega_i$ is defined as

$$f = \bigcup_{r=1}^{M_f} f_r,$$

where f_r is a sub-face attached to edge $\overline{ij}$ (see Figs. 8.1(b) and (c)), and M_f is the total number of attached sub-faces constituting face f. In $\mathbb{R}^2$, each edge is attached to as many sub-faces as there are elements adjacent to the edge, and each sub-face is the straight line connecting the edge centre to the centroid of an adjacent element (respectively denoted by "$\times$" and "$\bigcirc$" in Fig. 8.1(b)). In $\mathbb{R}^3$, each edge has twice as many sub-faces (all triangular) as there are elements adjacent to the edge, and each sub-face is defined by the edge centre, the centroid of an adjacent element, and the centroid of one of two adjacent faces of the same element (respectively denoted by "$\times$", "$\bigcirc$", and "$\triangle$" in Fig. 8.1(c)).

For finite volume spatial discretisation purposes, the face coefficient vector, $\boldsymbol{c}_f$, of each face f is defined as

$$\boldsymbol{c}_f = \sum_{r=1}^{M_f} \boldsymbol{c}_{f_r} = \sum_{r=1}^{M_f} \boldsymbol{n}_{f_r} \mathcal{A}_{f_r}, \quad \forall f \in \partial\Omega_i, \tag{8.4}$$

where $\boldsymbol{n}_{f_r}$ and $\mathcal{A}_{f_r}$ are the outward-pointing (with respect to Ω_i) unit normal and area (length in $\mathbb{R}^2$) of each sub-face f_r, respectively.

8.3 Temporal Discretisation

At each time-step m, the volume fraction field is first updated using

$$\alpha^{m+1} = \alpha^m - \frac{\Delta t}{\mathcal{V}_i} \sum_{f \in \partial\Omega_i} \alpha_f \boldsymbol{u}_f \cdot \boldsymbol{c}_f,$$

where $\partial\Omega_i$ and $\mathcal{V}_i$ are as defined previously, Δt is the time-step size, and the face coefficient vector $\boldsymbol{c}_f$ is defined as per Eq. (8.4). Additionally, α_f is the volume fraction at face f which is evaluated using CICSAM [63], and $\boldsymbol{u}_f$ is the fluid velocity at face f which comprises the divergence-free velocity as computed on the right-hand side of Eq. (8.6) shown below.

Next, the standard semi-implicit fractional-step methodology [4, 36, 43, 47] is used to discretise the temporal term in Eq. (8.1) as

$$\frac{\partial(\rho\boldsymbol{u})}{\partial t} \approx \frac{\rho\boldsymbol{u}|^{m+1} - \rho\boldsymbol{u}|^m}{\Delta t} = \frac{\rho\boldsymbol{u}|^{m+1} - \rho\boldsymbol{u}|^*}{\Delta t} + \frac{\rho\boldsymbol{u}|^* - \rho\boldsymbol{u}|^m}{\Delta t}, \tag{8.5}$$

from which the projected velocity field, $\boldsymbol{u}^*$, is solved for explicitly by equating the terms

$$\frac{\rho\boldsymbol{u}|^* - \rho\boldsymbol{u}|^m}{\Delta t} = -\nabla\cdot(\rho\boldsymbol{u}\otimes\boldsymbol{u})\big|^m + \nabla\cdot\big(\mu\big(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T\big)\big)\big|^m$$

Next, the pressure field is solved for implicitly from the Pressure Poisson equation

$$0 = \nabla\cdot\left[\boldsymbol{u}^* - \frac{\Delta t}{\rho^{m+1}}\left(\nabla p - \sigma\kappa\nabla\alpha\right)\big|^{m+1}\right]. \tag{8.6}$$

Lastly, the velocity field is updated by equating the terms

$$\frac{\rho\boldsymbol{u}|^{m+1} - \rho\boldsymbol{u}|^*}{\Delta t} = \big(-\nabla p + \sigma\kappa\nabla\alpha\big)\big|^{m+1} \tag{8.7}$$

8.4 Balanced-force Surface Tension Discretisation

The purpose of adding the surface tension source term into Eqs. (8.6) and (8.7) is to ensure a well-balanced scheme. This is enforced by discretising the pressure and surface tension source terms in the aforementioned equations in a mutually consistent manner. First, the pressure term in Eq. (8.6) is discretised at a vertex i as

$$\begin{aligned}\nabla\cdot\left(\frac{1}{\rho}\nabla p\right)\bigg|_i &\approx \frac{1}{\mathcal{V}_i}\sum_{f\in\partial\Omega_i}\left(\frac{1}{\rho}\nabla p\right)_f\cdot\boldsymbol{c}_f \\ &= \frac{1}{\mathcal{V}_i}\sum_{f\in\partial\Omega_i}\left(\left(\frac{1}{\rho}\nabla p\right)_f^{\parallel}+\left(\frac{1}{\rho}\nabla p\right)_f^{\perp}\right)\cdot\boldsymbol{c}_f,\end{aligned} \tag{8.8}$$

where $(\nabla p/\rho)_f^{\parallel}$ and $(\nabla p/\rho)_f^{\perp}$ respectively, are the components of $(\nabla p/\rho)_f$ that are parallel and perpendicular to the edge $\overline{ij}$ associated to face f. By respectively defining the edge tangent and the unit edge tangent vectors (see Figs. 8.1(b) and (c)) of the edge $\overline{ij}$ as

$$\boldsymbol{t}_{ij} = \boldsymbol{x}_j - \boldsymbol{x}_i, \quad \hat{\boldsymbol{t}}_{ij} = \frac{\boldsymbol{t}_{ij}}{\|\boldsymbol{t}_{ij}\|}, \tag{8.9}$$

the perpendicular component of $(\nabla p/\rho)_f$ is computed as

$$\left(\frac{1}{\rho}\nabla p\right)_f^{\perp} = \boldsymbol{v}_f - \left(\boldsymbol{v}_f\cdot\hat{\boldsymbol{t}}_{ij}\right)\hat{\boldsymbol{t}}_{ij}, \quad \boldsymbol{v}_f = \frac{\nabla p_i + \nabla p_j}{\rho_i+\rho_j}. \tag{8.10}$$

The computation of the pressure gradient ∇p_i at each vertex i will be detailed shortly in Eqs. (8.13) and (8.14). Next, to compute the parallel component of $(\nabla p/\rho)_f$, a compact stencil approach [9] is employed. To this end, based on an approach similar to that of Panahi *et al.* [44], the following is assumed:

$$\left(\frac{1}{\rho}\nabla p\right)_f^{\parallel} = \frac{1}{\rho}\frac{\partial p}{\partial\hat{\boldsymbol{t}}_{ij}} = c_{p_{ij}},$$

where $\partial p/\partial\hat{\boldsymbol{t}}_{ij}$ is the directional derivative of p in the direction of $\hat{\boldsymbol{t}}_{ij}$, and $c_{p_{ij}}$ is assumed to be constant over edge $\overline{ij}$. By integrating the above along the edge from vertex i to face f, and assuming a piecewise-constant density distribution in the dual-cells of vertices i and j, the change in pressure between the vertex and the face can be expressed as

$$p_f - p_i = \frac{\|\boldsymbol{t}_{ij}\|}{2}\rho_i c_{p_{ij}}. \tag{8.11}$$

Likewise, integrating along the edge from face f to vertex j gives

$$p_j - p_f = \frac{\|\boldsymbol{t}_{ij}\|}{2}\rho_j c_{p_{ij}}. \tag{8.12}$$

The parallel component of $(\nabla p/\rho)_f$ can now be computed by solving for the constant $c_{p_{ij}}$, by adding Eqs. (8.11) and (8.12) to get

$$\left(\frac{1}{\rho}\nabla p\right)_f^{\parallel} = c_{p_{ij}} = \frac{2(p_j-p_i)}{\|\boldsymbol{t}_{ij}\|(\rho_i+\rho_j)}.$$

Lastly, the pressure at face f is found by subtracting Eq. (8.12) from Eq. (8.11), which gives

$$p_f = \frac{\rho_j p_i + \rho_i p_j}{\rho_i + \rho_j}. \tag{8.13}$$

This is used to compute the pressure gradients ∇p_i at each vertex i (as required by Eqs. (8.7) and (8.10)) as follows:

$$\nabla p_i \approx \frac{1}{\mathcal{V}_i} \sum_{f \in \partial\Omega_i} p_f \boldsymbol{c}_f. \tag{8.14}$$

As mentioned previously, the surface tension source term in Eq. (8.7) must be discretised in a manner that is consistent with the discretisation of the pressure term in Eq. (8.8). This yields

$$\nabla\cdot\left(\frac{\sigma\kappa}{\rho}\nabla\alpha\right)\bigg|_i \approx \frac{\sigma}{\mathcal{V}_i} \sum_{f \in \partial\Omega_i} \kappa_f \left(\frac{1}{\rho}\nabla\alpha\right)_f \cdot \boldsymbol{c}_f, \tag{8.15}$$

where $(\nabla\alpha/\rho)_f$ is discretised as per $(\nabla p/\rho)_f$. Further, $\kappa_f = (\kappa_i + \kappa_j)/2$ is the central-differenced curvature[22] at face f, where κ_i is computed at each vertex i in the manner detailed in the next section. This κ_i is also employed to discretize Eq. (8.7) at each vertex i. Lastly, the gradient in α is discretised similar to Eq. (8.14) with α_f computed as

$$\alpha_f = \frac{\rho_j \alpha_i + \rho_i \alpha_j}{\rho_i + \rho_j}.$$

The procedure for computing the interface curvature at vertices surrounding the interface is detailed next.

8.5 Computing Curvature from a Higher-order Interface Reconstruction

The equations related to curvature computation from polynomial-based interface reconstructions (Eqs. (3.12) and (3.21)) are restated here for convenience:

$$\kappa = \begin{cases} \dfrac{2c_3}{(1+c_2^2)^{3/2}} & \text{in } \mathbb{R}^2 \\ \dfrac{2\big(c_4(1+c_3^2) - c_2 c_3 c_5 + c_6(1+c_2^2)\big)}{(1+c_2^2+c_3^2)^{3/2}} & \text{in } \mathbb{R}^3 \end{cases}, \tag{8.16}$$

where the polynomial coefficients c_i are computed as per Chapter 7. To ensure that the contributions of the $\nabla\cdot(\sigma\kappa\nabla\alpha/\rho)$ and $\sigma\kappa\nabla\alpha$ terms (see Eqs. (8.6) and (8.7) respectively) are accounted for at all interface vertices[23] over a potentially smeared VOF interface, it is necessary to assign curvature values to a *band* of vertices surrounding the interface (see black region in Fig. 8.2).

[22] Note that the mean curvature estimate at the face retains the order of convergence of the curvature values associated to the adjacent vertices. This is because the arithmetic mean is always bounded between these values. Therefore, the mean curvature will always converge at the same rate as that of the curvature of the adjacent vertices.

[23] Refer to the beginning of Section 4.1 for the definition of an *interface vertex*.

The aforementioned band includes all vertices in the vicinity of the interface that are connected to at least one central column of an aligned stencil[24].

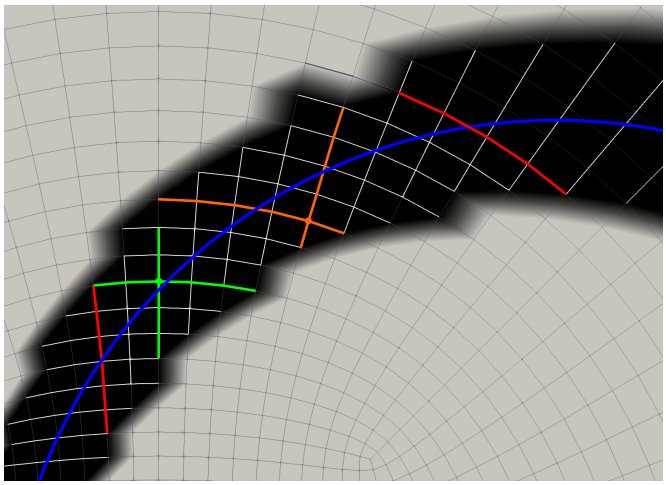


Figure 8.2: Portion of an interface (blue curve) in $\mathbb{R}^2$ showing the curvature band (black region) and a region of overlapping interface columns (the extent of overlap is exaggerated for illustrative purposes). The green columns are connected to an interface vertex, while the orange columns are connected to a vertex that is not an interface vertex. The red columns are associated to misaligned stencils and are excluded as curvature is only computed in aligned stencils.

The first step is to associate a curvature value to the central column of each aligned stencil *via* Eq. (8.16). The curvature at each vertex in the curvature band is then computed as a function of the curvature associated to its connected column(s). All vertices that are connected to one column inherit the curvature associated to the column itself. However, when a vertex is connected to more than one column (overlapping region), the assigned curvature is computed as a weighted sum of the curvature values associated to each connected column. Therefore, the interface curvature at all vertices i within the *band* is expressed as

$$\kappa_i = \begin{cases} \kappa_{\zeta_r},\ r \in I_i & \text{if } M_i = 1 \\ \dfrac{\sum_{r \in I_i} \omega_{\zeta_r} \kappa_{\zeta_r}}{\sum_{r \in I_i} \omega_{\zeta_r}} & \text{otherwise} \end{cases}, \tag{8.17}$$

where, in the d-dimensional case, $M_i \leq d$ is the number of interface columns connected to vertex i, I_i is the set of global indices of each column ζ_r connected to i, and κ_{ζ_r} and ω_{ζ_r} respectively are the curvature and the curvature weighting factor associated to ζ_r. It is important to note that, for all vertices in the overlapping region, the weighting factors ω_{ζ_r} must be pre-computed for each ζ_r before Eq. (8.17) is applied. This is because ω_{ζ_r} (whose involved computation procedure is detailed below) is required when applying Eq. (8.17) to each vertex connected to column ζ_r.

[24] Curvature is not computed for misaligned stencils as it cannot be accurately interpreted at the centroid of the central PLIC facet, which in general does not coincide with the origin of the local coordinate system (*i.e.* the point at which Eq. (8.16) is evaluated) of the polynomial fit.

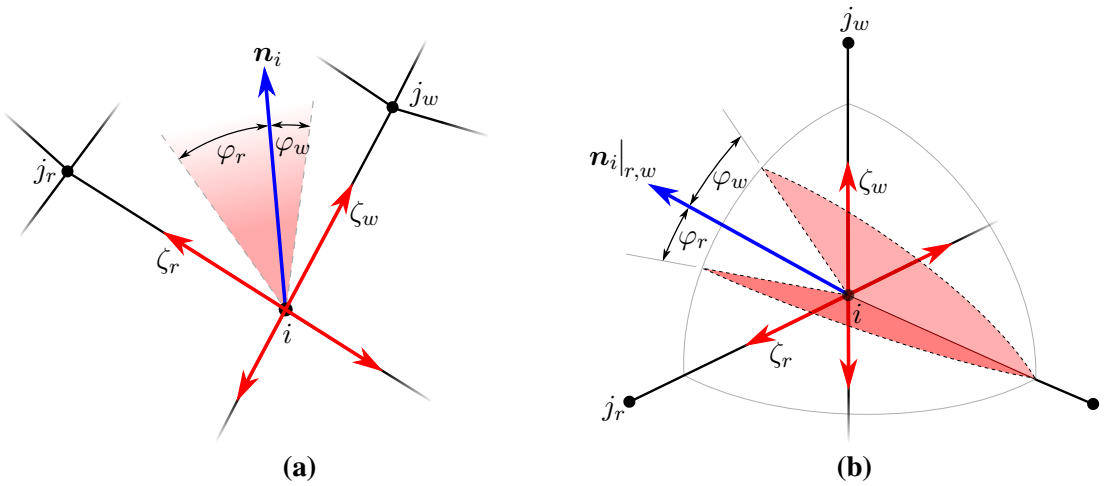


Figure 8.3: Calculation of the angles (with respect to an interface normal $\boldsymbol{n}_i$) associated to the $\varrho_{r,w}$ factors of a pair of interface columns, ζ_r and ζ_w, connected to an interface vertex i in—(a) $\mathbb{R}^2$ and (b) $\mathbb{R}^3$.

The weighting factors ω_{ζ_r} are computed as a function of the normals at the interface vertices in the overlapping region. This allows for a gradual transition of the interface curvature computed over the overlapping region, which is particularly important if there is a variation in grid spacing between two (or more) contiguous regions of columns. For an interface column ζ_r, which is one of M_i columns connected to an interface vertex i, ω_{ζ_r} is computed as

$$\omega_{\zeta_r} = \begin{cases} \varrho_{r,s}, \quad r,s \in I_i,\ r \neq s & \text{if } M_i = 2 \\ \dfrac{1}{3}(\varrho_{r,s} + \varrho_{r,t}), \quad r,s,t \in I_i,\ r \neq s,\ r \neq t,\ s \neq t & \text{if } M_i = 3 \end{cases}, \tag{8.18}$$

where

$$\varrho_{r,w} = \frac{\varphi_w}{\varphi_r + \varphi_w},\ w \in \{s,t\},$$

and φ_r and φ_w are the angles associated to the interface columns, ζ_r and ζ_w, computed with respect to the interface normal[25] $\boldsymbol{n}_i$ at vertex i, in the manner illustrated in Figs. 8.3(a) and (b) for $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively. In $\mathbb{R}^3$, the angles are computed in the plane defined by the edges $\overline{ij_r}$ and $\overline{ij_w}$, with respect to the vector projection, $\boldsymbol{n}_i|_{r,w}$, of $\boldsymbol{n}_i$ onto the same plane. The regions within which the angles are computed (shaded in red in Fig. 8.3) are determined as per Sections 4.2 and 4.3 for $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively.

[25]This is the interface normal computed at vertex i as per Eq. (4.2), after the completion of the interface normal refinement procedure.

Chapter 9

Results

A set of three grids of varying fineness in both $\mathbb{R}^2$ and $\mathbb{R}^3$ was used to evaluate the accuracy and performance characteristics of the developed VOF interface reconstruction and curvature computation schemes. Four interface types were considered – a circular/spherical interface of radius $r = 0.25m$ and an elliptical/ellipsoidal interface with major and minor axis half-lengths $r_x = 0.3m$ and $r_y = 0.25m$ in $\mathbb{R}^2$, and $r_x = 0.3m$, $r_y = 0.275m$, and $r_z = 0.25m$ in $\mathbb{R}^3$. The volume fraction field for each interface was initialised to machine-precision accuracy using the Arbitrary Grid Initialiser (AGI) tool of Jones *et al.* [30]. The interface was centred at four positions, $x \in \{0.0,\ 0.067,\ 0.133,\ 0.2\}$, in each grid (see Fig. 9.1). The positions were chosen to ensure that two of them would result in a region of overlapping interface columns for all interface types – this occurs when $x = 0.133$ and $x = 0.2$. However, note that position $x = 0.133$ was excluded from the 3D test cases as it resulted in complications when constructing interface columns across the *singularities* (see Fig. 1.1) in the innermost structured region of the grid.

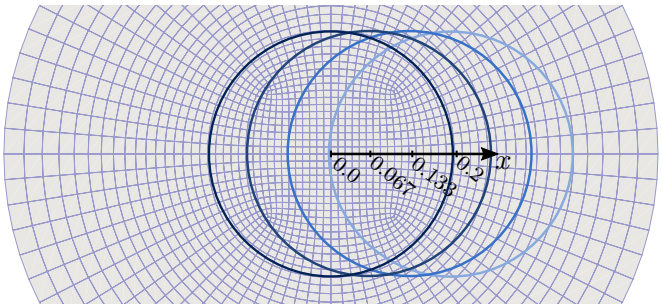


Figure 9.1: The four interface centre positions marked on the x-axis, illustrated for the circular interface on the second coarsest grid in $\mathbb{R}^2$. The cross-sections of the grids in $\mathbb{R}^3$ are of a similar form.

Tables 9.1–9.4 list the PLIC facet length information for each grid. For each interface centre position, the total number of interface vertices, N_i, and the characteristic PLIC facet length, $L_2(\Delta l)$, are given. The latter is the L_2-norm of the characteristic lengths[26], Δl_p, of each PLIC facet p. For some scalar attribute φ, the following definition of the L_2-norm (equivalent to the root-mean-square (RMS)) is employed:

$$L_2(\varphi) = \sqrt{\frac{1}{N_p}\sum_{p}^{N_p}(\varphi_p)^2}, \tag{9.1}$$

[26] In $\mathbb{R}^3$, $\Delta l_p = \sqrt{A_p}$, where A_p is the PLIC facet area.

where N_p is the number of PLIC facets and φ_p is the value of the attribute associated to PLIC facet p. Another metric that is employed to quantify accuracy in the sections to follow is the L_∞-norm, which is defined as

$$L_\infty(\varphi) = \max\{\varphi_p \mid \forall p\}. \tag{9.2}$$

The reader is here reminded that the curvature is only computed at aligned stencils. Therefore, Eqs. (9.1) and (9.2) only apply to all PLIC facets p associated to interface columns at which aligned stencils are constructed.

Table 9.1: 2D grid information for the circular interface. N_i denotes the number of interface vertices and $L_2(\Delta l)$ is the L_2-norm of the lengths of the PLIC facets.

Grid	No. of	$x = 0.0$		$x = 0.067$		$x = 0.133$		$x = 0.2$	
#	Vertices	N_i	$L_2(\Delta l)$	N_i	$L_2(\Delta l)$	N_i	$L_2(\Delta l)$	N_i	$L_2(\Delta l)$
1	561	40	3.93e-2	40	4.00e-2	38	4.31e-2	36	4.70e-2
2	1121	56	2.81e-2	56	2.86e-2	52	3.02e-2	50	3.32e-2
3	2361	80	1.96e-2	80	2.00e-2	76	2.16e-2	74	2.30e-2

Table 9.2: 2D grid information for the elliptical interface.

Grid	No. of	$x = 0.0$		$x = 0.067$		$x = 0.133$		$x = 0.2$	
#	Vertices	N_i	$L_2(\Delta l)$	N_i	$L_2(\Delta l)$	N_i	$L_2(\Delta l)$	N_i	$L_2(\Delta l)$
1	561	40	3.61e-2	40	3.69e-2	34	4.42e-2	32	4.58e-2
2	1121	56	2.58e-2	56	2.64e-2	48	3.16e-2	46	3.27e-2
3	2361	80	1.81e-2	80	1.85e-2	70	2.16e-2	68	2.22e-2

Table 9.3: 3D grid information for the spherical interface.

Grid	No. of	$x = 0.0$		$x = 0.067$		$x = 0.2$	
#	Vertices	N_i	$L_2(\Delta l)$	N_i	$L_2(\Delta l)$	N_i	$L_2(\Delta l)$
1	13973	602	3.62e-2	602	3.66e-2	582	4.03e-2
2	57205	1538	2.26e-2	1538	2.28e-2	1494	2.52e-2
3	292631	4706	1.29e-2	4706	1.32e-2	4482	1.44e-2

Table 9.4: 3D grid information for the ellipsoidal interface.

Grid	No. of	$x = 0.0$		$x = 0.067$		$x = 0.2$	
#	Vertices	N_i	$L_2(\Delta l)$	N_i	$L_2(\Delta l)$	N_i	$L_2(\Delta l)$
1	13973	602	3.97e-2	602	3.99e-2	582	4.37e-2
2	57205	1538	2.49e-2	1538	2.52e-2	1494	2.74e-2
3	292631	4706	1.42e-2	4706	1.44e-2	4482	1.56e-2

9.1 Curvature Grid Convergence Study

9.1.1 Circular/Spherical Interface

The interface curvature, κ_p, was computed at all PLIC facets p representing a circle and sphere of radius $r = 0.25m$, and compared to the analytical curvature $\kappa_{exact} = -4.0\,m^{-1}$ and $\kappa_{exact} = -8.0\,m^{-1}$, respectively (an outward-pointing interface normal is assumed, resulting in negative

curvature). The two centre positions $x = 0.0$ and $x = 0.2$ respectively (see Fig. 9.1), resulted in the best and worst cases in terms of curvature error. Figs. 9.2–9.4 depict the distribution of the curvature relative error over the interface. In Figs. 9.2 and 9.3, the horizontal axes correspond to the angle subtended by the x-axis and the line joining the centre of the circular interface and the point on the polynomial fit at which curvature is computed, for each PLIC facet. In the local reference frame of a polynomial $P_n(\xi)$, this is the point $\big(0, P_n(0)\big)$. Given a computed numerical curvature κ_p, the relative error is computed as $\epsilon_{\kappa,p} = (\kappa_p - \kappa_{exact})/\kappa_{exact}$.

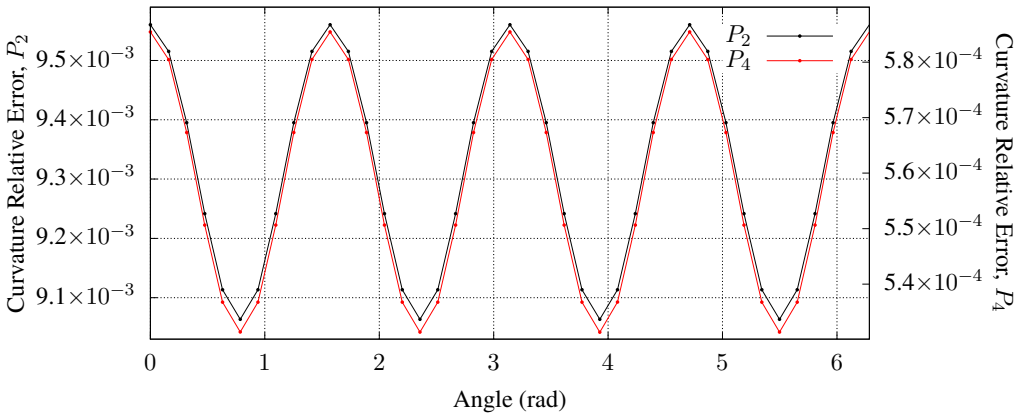


Figure 9.2: 2D curvature relative error distribution with second-order (P_2) and fourth-order (P_4) polynomial fits on grid 1 for the circular interface centred at $x = 0.0$. Note that the scales for the second- and fourth-order results are on the left and right sides of the plot, respectively.

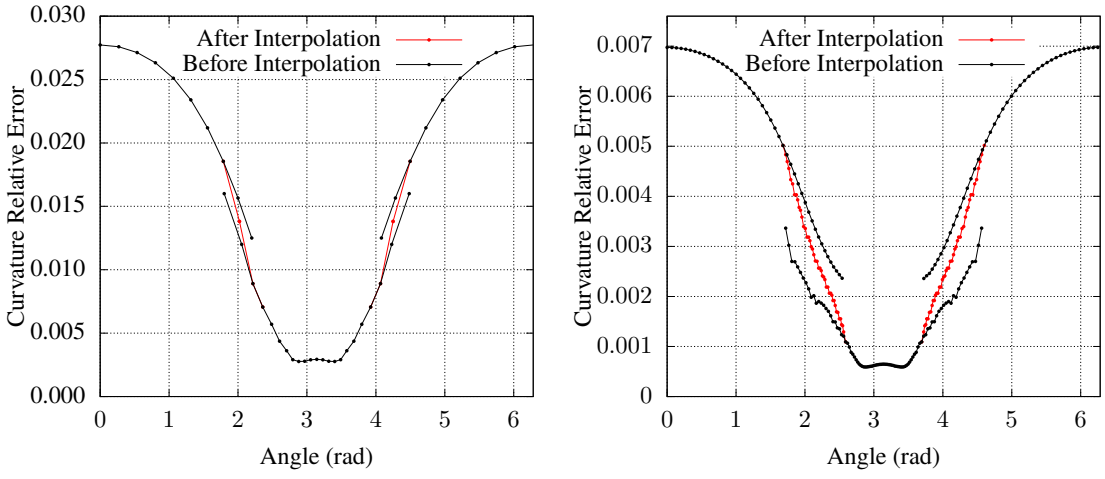


Figure 9.3: 2D curvature relative error distribution for the circular interface centred at $x = 0.2$ using second-order polynomial fits for (a) grid 1, and (b) grid 3. The red dotted line is the relative error of the proposed curvature interpolation procedure in the overlapping region (see Section 8.5).

Fig. 9.2 depicts a case with no overlapping interface columns over the interface. Figs. 9.3(a) and 9.3(b) show the relative error plots at $x = 0.2$, where there are two overlapping regions of columns, and the curvature is computed using second-order polynomial fits for the coarsest and

finest grids, respectively. The resulting discrepancy in the magnitude of the relative error in the overlapping regions is shown by the black dotted curves. This discrepancy will naturally be greater for larger degrees of anisotropy of the grid near these regions. The red dotted line represents the relative error of the interpolated curvature, computed as per Eq. (8.17). The figures demonstrate that a gradual and bounded transition of the interpolated curvature in the overlapping regions can be achieved by the developed scheme.

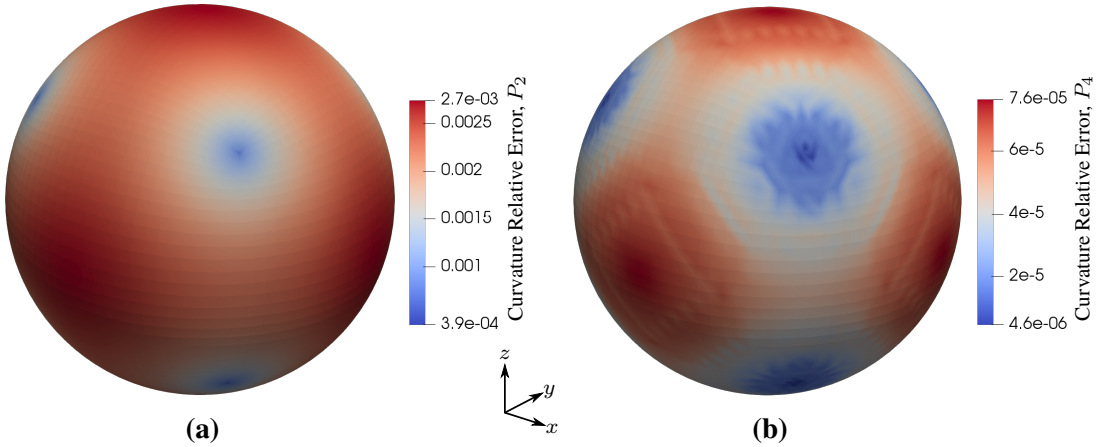


Figure 9.4: 3D curvature relative error distribution for the spherical interface centred at $x = 0.0$ on grid 3 using — (a) second-order and (b) fourth-order polynomial fits.

Fig. 9.4 illustrates curvature computed for the spherical interface at $x = 0.0$ using second- and fourth-order polynomial fits. Similar to the 2D case, 4^{th}-order results in significantly increased accuracy. The striations in the relative error distribution when using fourth-order polynomial fits is attributed to an increased sensitivity of the curvature computation to the smoothness of the underlying grid (a Laplacian smoothing algorithm was employed to smooth all grids). These striations become more pronounced with further grid refinement. Albeit beyond the scope of this work, it can be shown that non-orthogonal grids generated *via* smooth mappings of a uniform Cartesian grid would not result in such striations for higher-order polynomial fits.

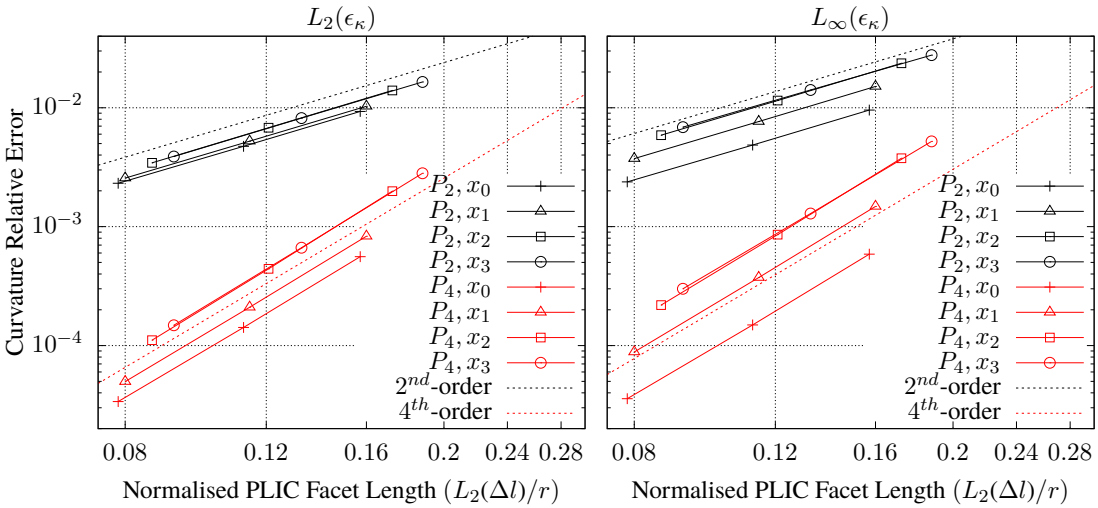


Figure 9.5: 2D grid convergence plots of $L_2(\epsilon_\kappa)$ and $L_\infty(\epsilon_\kappa)$ for the circular interface.

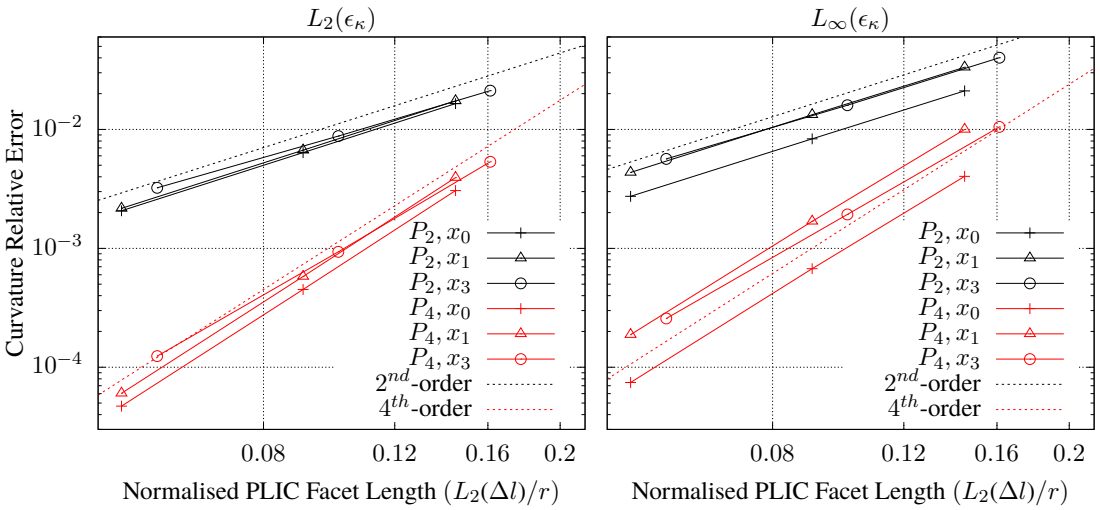


Figure 9.6: 3D grid convergence plots of $L_2(\epsilon_\kappa)$ and $L_\infty(\epsilon_\kappa)$ for the spherical interface.

Figs. 9.5 and 9.6 respectively depict the computed curvature grid convergence characteristics of $L_2(\epsilon_\kappa)$ and $L_\infty(\epsilon_\kappa)$ for the circular and spherical interfaces at the different interface positions. As shown, 4^{th}-order polynomials achieve a significant increase in accuracy as compared to 2^{nd}-order in all cases. The plots also clearly demonstrate formal second- and fourth-order accuracy of the scheme for both $L_2(\epsilon_\kappa)$ and $L_\infty(\epsilon_\kappa)$. However, with increased non-uniformity of the grids, it is reasonable to expect (see results in Section 3.4.2) that a drop of up to one order of accuracy can occur with further grid refinement. This is not apparent in the selected set of grids as the L_2-norms of the stretching ratios between neighbouring PLIC facets were 1.014 and 1.052 in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively. Therefore, a more detailed study on the effects of more pronounced grid non-uniformity on the order of accuracy (similar to Fig. 3.10) in the context of non-orthogonal structured grids is left for future work.

9.1.2 Elliptical/Ellipsoidal Interface

An elliptical/ellipsoidal interface was selected to validate the curvature computation scheme on interfaces with smoothly varying curvature. Assuming an outward-pointing interface normal, the analytical curvature at a point on an ellipse/ellipsoid can be computed as

$$\kappa_{exact} = \begin{cases} \dfrac{-r_x r_y}{\left(r_x^2 sin^2\theta + r_y^2 cos^2\theta\right)^{\frac{3}{2}}} & \text{in } \mathbb{R}^2 \\ \dfrac{-r_x r_y r_z \left[3(r_x^2+r_y^2)+2r_z^2+(r_x^2+r_y^2-2r_z^2)\cos(2\phi)-2(r_x^2-r_y^2)\cos(2\theta)\sin^2\phi\right]}{4\left[r_x^2 r_y^2\cos^2\phi + r_z^2(r_y^2\cos^2\theta + r_x^2\sin^2\theta)\sin^2\phi\right]^{\frac{3}{2}}} & \text{in } \mathbb{R}^3 \end{cases}.$$

Here, r_x, r_y, and r_z are the major and minor axis half-lengths of the ellipse/ellipsoid, and θ (and ϕ in $\mathbb{R}^3$) are the polar/spherical coordinates associated to some point $\boldsymbol{x}$ (see Fig. 9.7 for 2D analogy). Computing these angles requires the projection, $\boldsymbol{x}'$, of $\boldsymbol{x}$ onto an outer circle/sphere of radius r_x. As shown in the figure, $\boldsymbol{x}'$ is computed by scaling the y-component of $\boldsymbol{x}$ by a factor r_x/r_y (for the ellipsoid, both the y- and z-components need to be scaled by factors r_x/r_y and r_x/r_z, respectively). For the selected interfaces, $r_x = 0.3m$ and $r_y = 0.25m$ in $\mathbb{R}^2$; and $r_x = 0.3m$, $r_y = 0.275m$, and $r_z = 0.25m$ in $\mathbb{R}^3$.

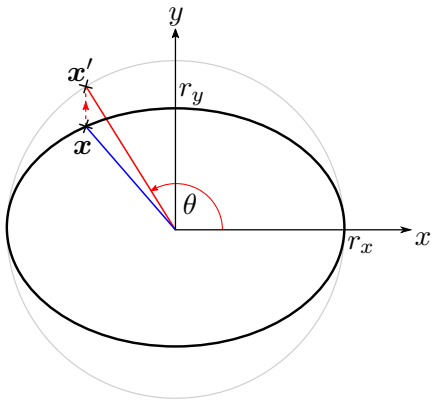


Figure 9.7: Calculation of the angle associated to a point $\boldsymbol{x}$ on an elliptical interface with major and minor axis half-lengths r_x and r_y, respectively.

For all PLIC facets that do not lie in an overlapping region, the point $\boldsymbol{x}$ is taken to be the point on the polynomial curve at which curvature was computed for the given PLIC facet. Given a computed numerical curvature κ_p, at the point $\boldsymbol{x}$ on the ellipse, the relative error in curvature, $\epsilon_{\kappa,p}$, is computed as previously.

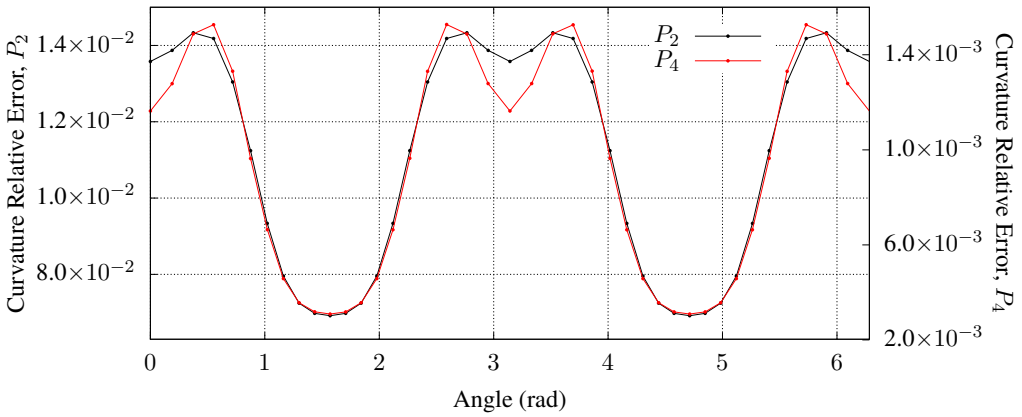
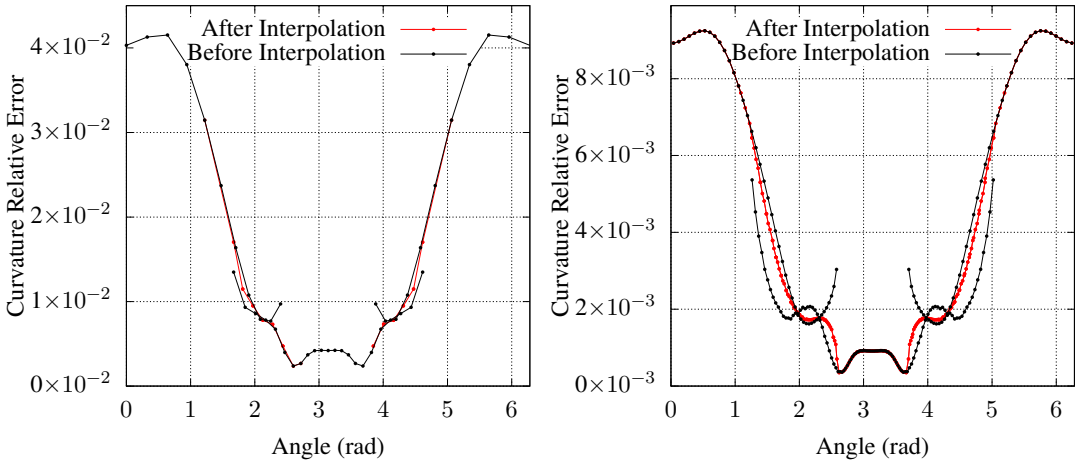


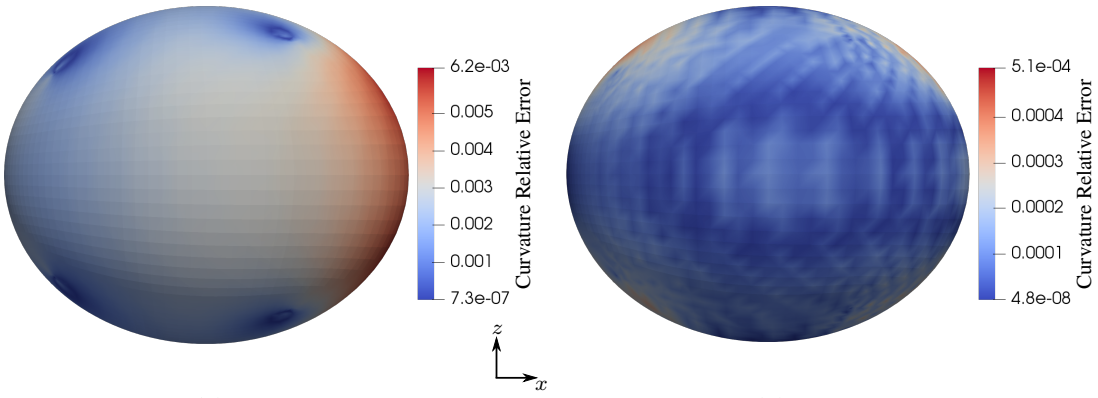
Figure 9.8: 2D curvature relative error distribution with second-order (P_2) and fourth-order (P_4) polynomial fits on grid 1 for the elliptical interface centred at $x = 0.0$.

Fig. 9.8 shows the relative error plots for the elliptical interface at $x = 0.0$ for grid 1. Fig. 9.9 depicts a case with overlapping interface columns, where the interface is positioned at $x = 0.2$ and the curvature is computed using second-order polynomial fits. The analytical curvature in the overlapping region is here computed by interpolating that associated to the two overlapping PLIC facets, using the same weights as those applied to the corresponding numerical curvatures in Eq. (8.17). The associated angle of each interpolation point is taken to be the angle θ such that $\kappa_{exact}(\theta)$ matches the aforementioned interpolated analytical curvature. Once again, Fig. 9.9 shows that Eq. (8.17) successfully bounds the error in the interpolated curvature between the errors of the overlapping layers.



(a) **(b)**

Figure 9.9: 2D curvature relative error distribution for (a) grid 1, and (b) grid 3, for the elliptical interface centred at $x = 0.2$. The curvature is computed using second-order polynomial fits.



(a) **(b)**

Figure 9.10: 3D curvature relative error distribution for the ellipsoidal interface centred at $x = 0.067$ on grid 3 using—(a) second-order and (b) fourth-order polynomial fits.

Fig. 9.10 illustrates curvature computed for the ellipsoidal interface at $x = 0.067$ using second- and fourth-order polynomial fits. Once again, the striations in the relative error distribution when using fourth-order polynomial fits are visible for the same reasons mentioned previously.

Figs. 9.11 and 9.12 respectively depict the curvature grid convergence characteristics of $L_2(\epsilon_\kappa)$ and $L_\infty(\epsilon_\kappa)$ for the elliptical and ellipsoidal interfaces. As before, 4^{th}-order achieves a significant increase in accuracy. Further, formal second- and fourth-order accuracy can be observed for all cases except for the $L_\infty(\epsilon_\kappa)$ plot of the ellipsoid. Here, a deterioration in the order of accuracy is apparent, and can be attributed to a larger L_2-norm of the stretching ratios between neighbouring PLIC facets (in comparison to the spherical case)—1.043 and 1.097 in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively.

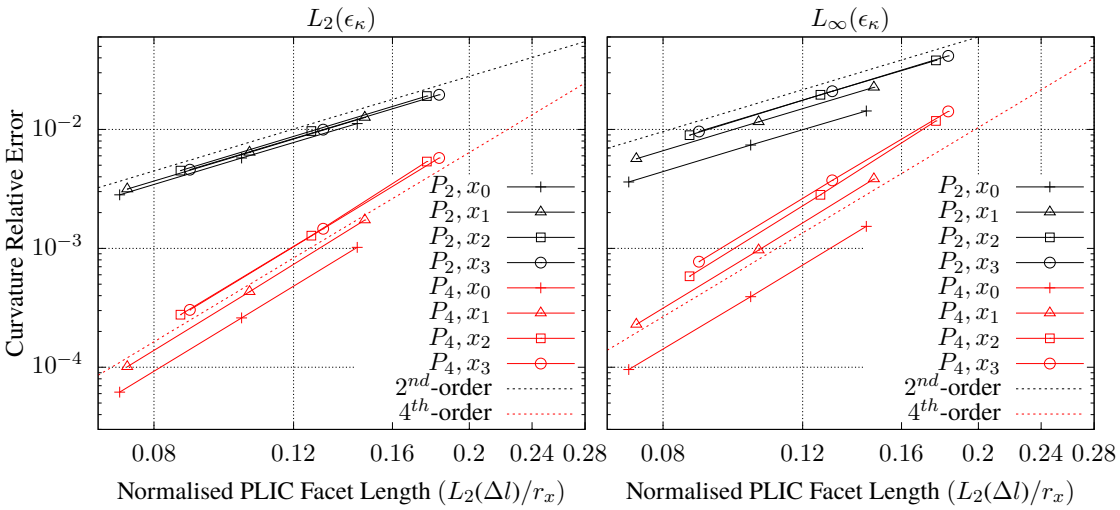


Figure 9.11: 2D grid convergence plots of $L_2(\epsilon_\kappa)$ and $L_\infty(\epsilon_\kappa)$ for the elliptical interface.

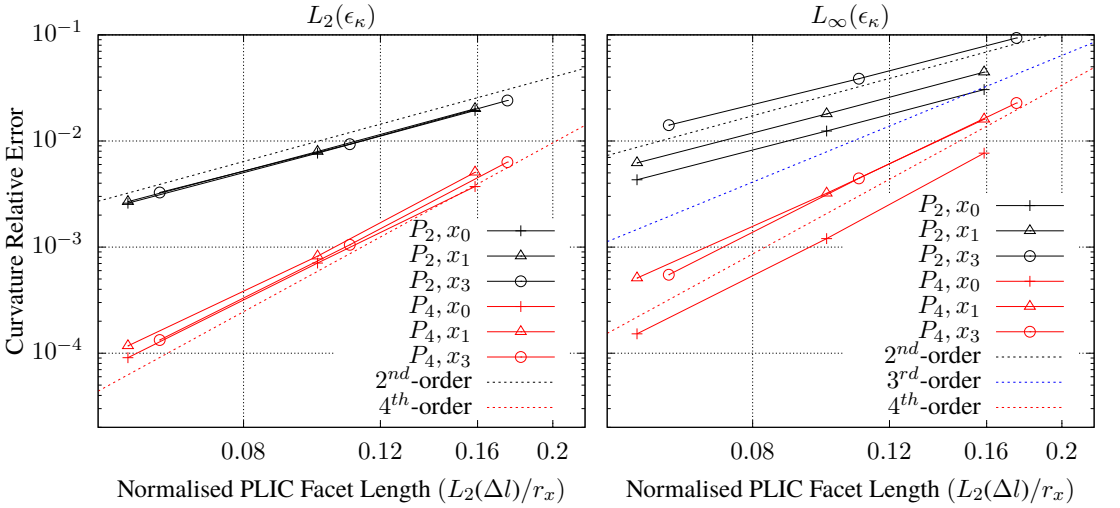


Figure 9.12: 3D grid convergence plots of $L_2(\epsilon_\kappa)$ and $L_\infty(\epsilon_\kappa)$ for the ellipsoidal interface.

9.2 Curvature–Interface Normal Sensitivity Study

Since the iterative interface normal refinement procedure was adopted as a means of increasing the accuracy of interface curvature, it is instructive to investigate the change in accuracy of the computed curvature after each iteration, k. Further, the impact of the normal refinement acceleration procedure is to be investigated as well. Figs. 9.13 and 9.14 show the convergence plots of the L_∞-norm of the curvature relative error for the elliptical and ellipsoidal interfaces at centre positions $x = 0.0$ and $x = 0.2$. For this purpose, the interface curvature is computed after each iteration using 4^{th}-order polynomial fits (similar results were achieved when using 2^{nd}-order fits). The graph labels *Accel on* and *Accel off* respectively denote the convergence histories with and without the acceleration procedure.

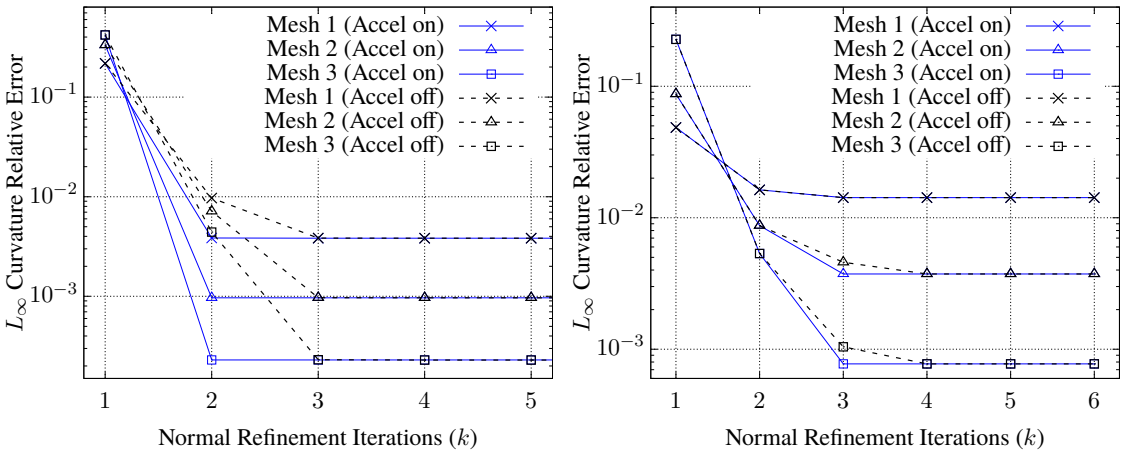


Figure 9.13: 2D convergence history of the L_∞-norm of the relative error in curvature during the iterative normal refinement procedure for the elliptical interface at (a) $x = 0.0$, and (b) $x = 0.2$.

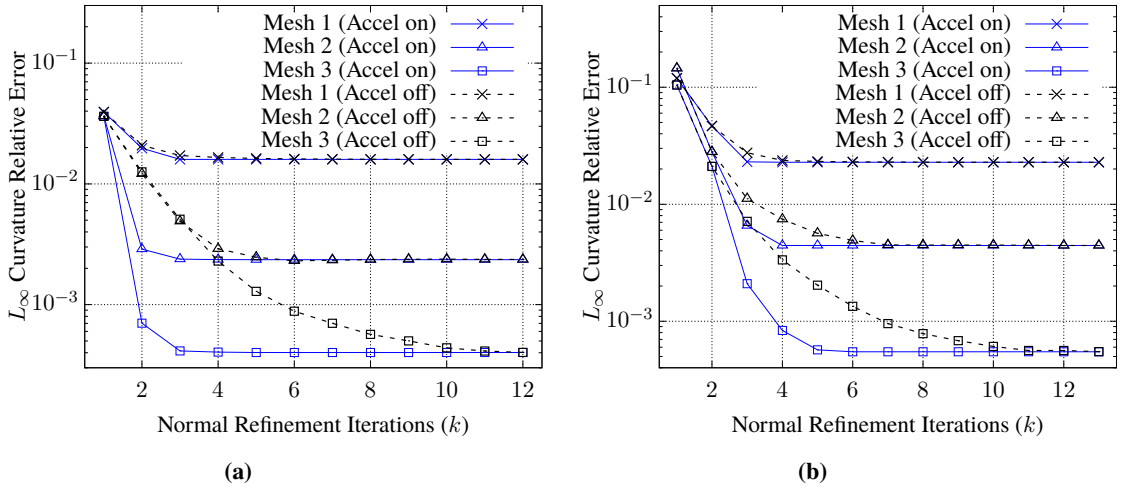


Figure 9.14: 3D convergence history of the L_∞-norm of the relative error in curvature during the iterative normal refinement procedure for the ellipsoidal interface at (a) $x = 0.0$, and (b) $x = 0.2$.

Fig. 9.13 demonstrates that in $\mathbb{R}^2$, the refinement acceleration procedure yields a marginal improvement in the number of iterations required to converge $L_\infty(\epsilon_\kappa)$. However, as shown in Fig. 9.14, a significant improvement is observed in $\mathbb{R}^3$, particularly for the two finer grids. This reduction in the required number of iterations is advantageous in that it significantly reduces the required cost of PLIC reconstruction which, as will be seen in Section 9.6, accounts for a large portion of the computational cost of the overall scheme.

For all cases investigated, at most 29 and 63 sub-iterations (λ) were required to converge the refinement acceleration procedure in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively. Further research is to be carried out in optimising the Newton minimisation procedure (see Section 6.4.3) such that a line-search [39] is performed along the Newton descent direction. This would be expected to reduce the number of sub-iterations further.

9.3 Volume Conservation Grid Convergence Study

It was mentioned in Chapter 7 that unlike on Cartesian grids, the least squares polynomial fitting procedure on non-orthogonal structured grids does not exactly conserve the volume of the reference fluid in each column of a stencil. Therefore, it is instructive to quantify the volume conservation error of the polynomial fit relative to the analytical volume of the reference fluid in each column upon initialisation of a given interface. For this purpose, the elliptical and ellipsoidal interfaces were positioned at $x = 0.067$. The relative volume conservation error is then computed as

$$\epsilon_{\mathcal{V},n} = \frac{1}{\mathcal{V}_{exact}} \sum_{\zeta} \left| \mathcal{V}_{ref,\zeta} - \mathcal{V}_{\overline{P}_n,\zeta} \right|$$

where $\mathcal{V}_{ref,\zeta}$ is the volume of the reference fluid (that which lies inside the ellipse/ellipsoid) contained in column ζ, and $\mathcal{V}_{\overline{P}_n,\zeta}$ is the estimated volume of the reference fluid enclosed by the polynomial $\overline{P}_n$ (where $n \in \{2, 4\}$) fitted to a stencil with ζ as the central column[27]. The enclosed volume in each column was computed *via* AGI [30]. Lastly, the error is normalised with respect to the analytical volume enclosed by the respective interface, which is $\mathcal{V}_{exact} = \pi r_x r_y$ for the ellipse and $\mathcal{V}_{exact} = \frac{4}{3}\pi r_x r_y r_z$ for the ellipsoid (where r_x, r_y, and r_z are as defined previously). The convergence characteristics of $\epsilon_{\mathcal{V},n}$ are depicted in the figure below.

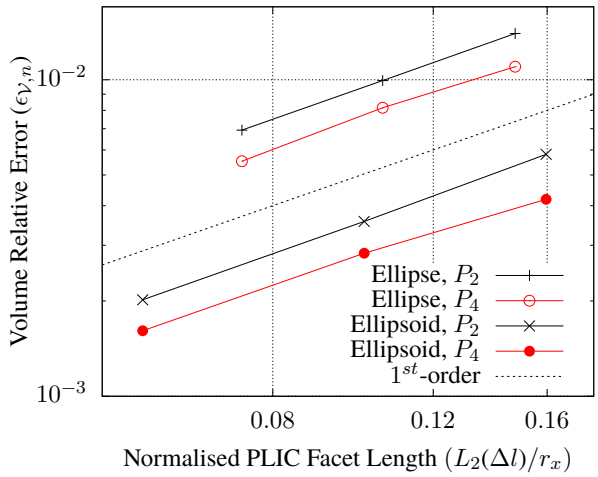


Figure 9.15: Grid convergence plots of $\epsilon_{\mathcal{V},n}$ for the elliptical and ellipsoidal interfaces positioned at $x = 0.067$.

Fig. 9.15 demonstrates that only first-order accuracy can be achieved for both the second- and fourth-order polynomial fitting schemes. This can be attributed to the variation in the cross-section of the columns being unaccounted for in the fitting procedure.

[27] Note that $\overline{P}_n$ is the modification of P_n (as per Eqs. (3.10) and (3.19)) that ensures volume conservation in the local coordinate system, in the central column ζ.

9.4 Stationary Bubble

This section investigates the scheme's ability to minimise the generation of spurious currents [46, 48] over time. This is done by simulating a stationary bubble [4] in $\mathbb{R}^2$. (The reader is again reminded that, as a result of time limitations of the project, the test case in $\mathbb{R}^3$ was left as future work). At the equilibrium state, the following condition should hold in Eq. (8.1):

$$-\nabla p + \sigma\kappa\nabla\alpha = 0.$$

However, discretisation errors typically introduce spurious velocities. The Laplace number, $La = \sigma\rho D/\mu^2$, is the ratio of surface tension to fluid momentum, where $D = 0.5m$ is the diameter of the bubble. Fig. 9.16 depicts the evolution of the non-dimensionalised RMS velocity over time on grid 1 (see Table 9.1) for La=$1.2e4$ and a liquid-gas density ratio of 1000. In addition, to investigate the effect of the interface crossing multiple co-centric rows of vertices, the grid is *compressed* by reducing all y-coordinates by 20%. The *compressed* and *uncompressed* cases are represented by the solid and dashed lines, respectively.

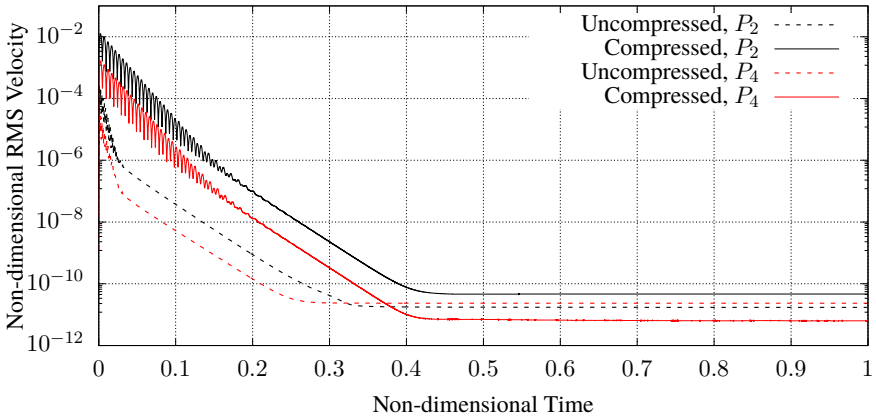


Figure 9.16: Evolution of the RMS velocity for a circular droplet in equilibrium for the compressed and uncompressed grid 1 with curvature computed using second-order and fourth-order polynomial fits.

Time is normalised with respect to the viscous time scale as per Popinet [46] as

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

and the RMS velocity is normalised with respect to the time scale

$$U_\sigma = \sqrt{\frac{\sigma}{\rho D}}.$$

The results demonstrate that, for both second- and fourth-order curvature computation, the scheme adequately damps spurious velocities down to similar levels as previous work using the HF method [1, 46]. In addition to investigating the velocity evolution, it is also instructive to demonstrate that the order convergence of curvature reflects in the Laplace pressure jump, $\Delta p = \sigma\kappa$, across the interface. To this end, the L_2-norm, $L_2(\Delta p)$, of the pressure jump across

each interface column was computed. Figs. 9.17 and 9.18 show the evolution of $L_2(\Delta p)$, which converges relatively quickly in comparison to the RMS velocity.

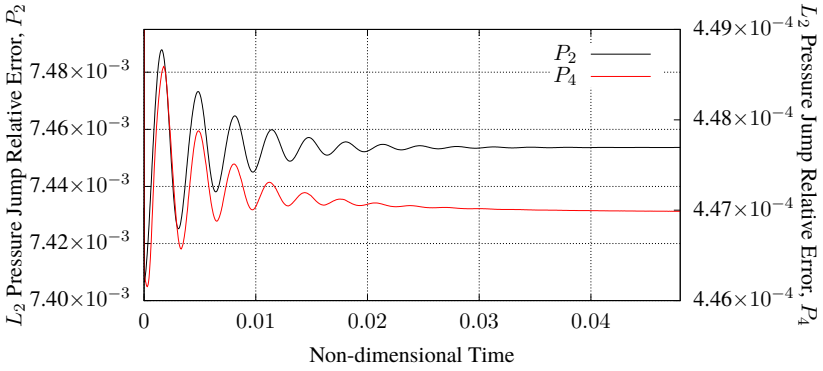


Figure 9.17: Evolution of the L_2-norm of the relative error in the Laplace pressure jump for a circular bubble in equilibrium, with curvature computed using second- and fourth-order polynomial fits, on the uncompressed version of grid 1.

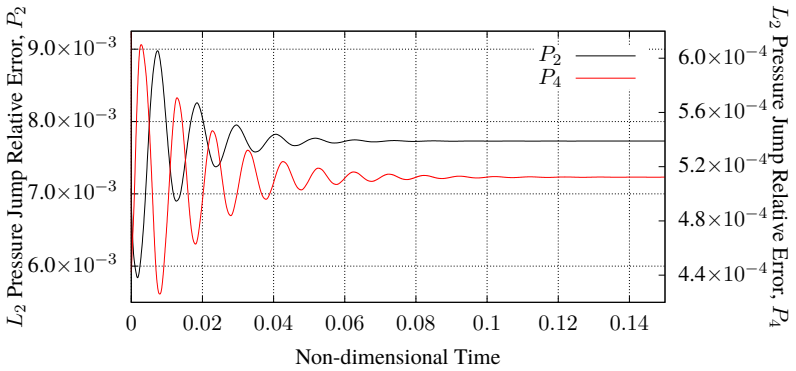


Figure 9.18: Evolution of the L_2-norm of the relative error in the Laplace pressure on the compressed version of grid 1.

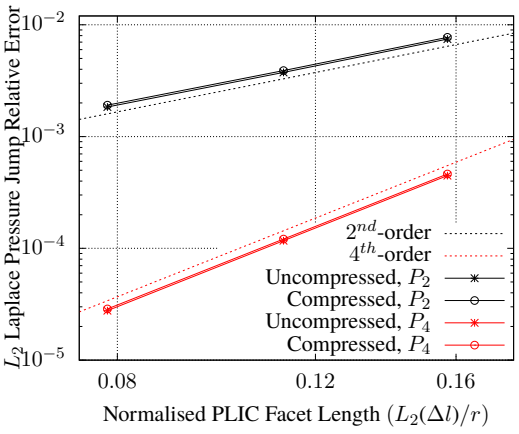


Figure 9.19: Laplace pressure jump convergence study with curvature computed using second-order (P_2) and fourth-order (P_4) polynomial fits.

The converged values of the pressure jump were used to perform a convergence study using all grids. Fig. 9.19 confirms that the scheme achieves formal second- and fourth-order accuracy for the pressure calculation as well.

9.5 Oscillating Droplet in Zero Gravity

Another standard test case for evaluating the accuracy of the surface tension scheme is that of an oscillating droplet in an inviscid fluid in zero gravity [17, 19, 22, 48, 61]. (Once again, this section is confined to the case in $\mathbb{R}^2$, and that in $\mathbb{R}^3$ is left as future work). Consider a droplet in $\mathbb{R}^2$ whose initial perturbed interface is given by

$$r = r_0 + \Delta r \cos(k\theta),$$

where k is the mode of oscillation and Δr is a small-amplitude perturbation about a mean radius r_0. The analytical oscillation frequency [20, 31] for the above interface can be expressed as

$$\omega = \sqrt{\frac{k(k^2-1)\sigma}{(\rho_0+\rho_1)r_0^3}}, \tag{9.3}$$

where σ is the surface tension coefficient, and ρ_1 and ρ_0 are the fluid densities inside and outside the droplet, respectively. The test case was set up similar to that in Torres and Brackbill [61] and Hermann [22], except with a density ratio of 1000, as used by Fuster *et al.* [19]. An interface with $r_0 = 2m$, $\Delta r = 0.001m$, $\rho_0 = 0.001kg/m^3$, $\rho_1 = 1kg/m^3$, and $\sigma = 1N/m$, was centred at $(0,0)$ in a circular domain of radius $2.0e4m$ with slip boundary conditions. A time-step size of $\Delta t = 1.0e-4s$ was used for all grids. The relatively large radius of the outer boundary was chosen because Eq. (9.3) strictly holds for a droplet surrounded by a fluid of infinite mass [31]. Torres and Brackbill [61] simulate this effect by employing doubly-periodic boundaries on a square domain.

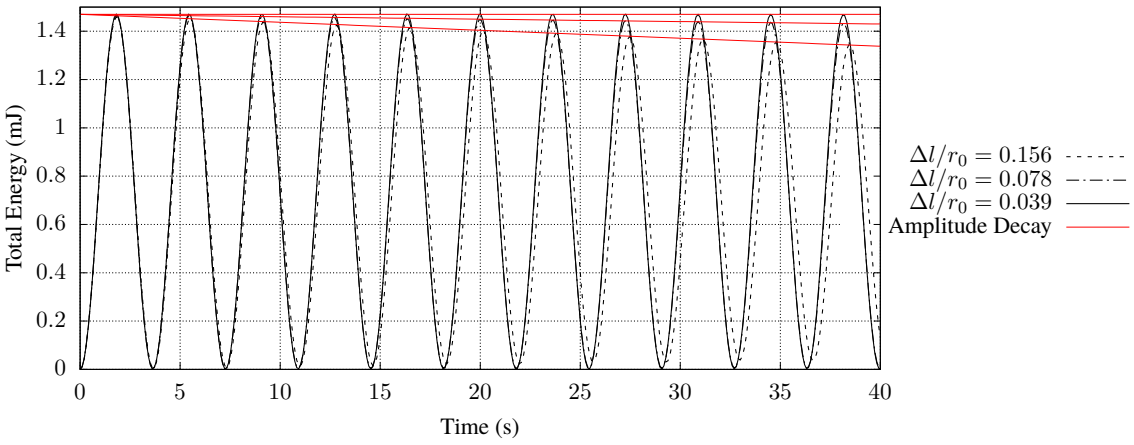


Figure 9.20: Comparison of the accuracy in computed frequency for a droplet oscillating with mode $k = 2$.

A new set of four block-structured grids was generated to match the circumferential and radial grid spacing with that employed by the compared schemes [19, 22, 61]. Fig. 9.20 depicts

the total kinetic energy of the oscillating droplet in the three coarsest grids. The red lines depict the amplitude decay due to numerical damping [19]. Fig. 9.21 shows the comparison of the accuracy in computed oscillation frequency for $k = 2$. This demonstrates that second-order accuracy is achieved for both the second- and fourth-order curvature computation schemes. The inability to achieve fourth-order accuracy can be attributed to the second-order accurate discretisation of the remaining terms in the governing equations.

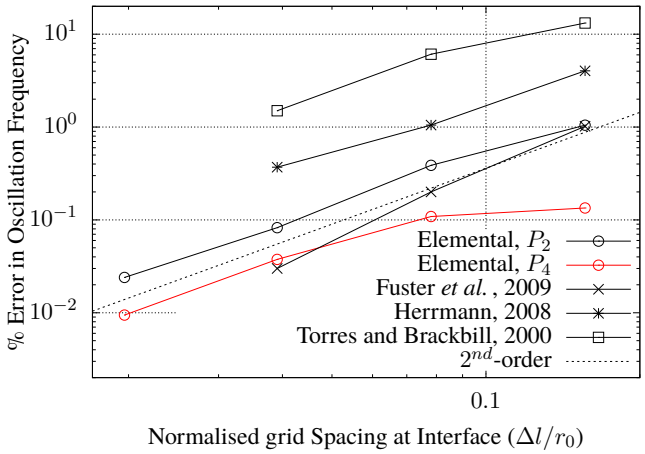


Figure 9.21: Comparison of the accuracy in computed frequency for a droplet oscillating with mode $k=2$.

9.6 Computational Cost

To assess the computational cost, the proposed interface reconstruction scheme was time-profiled over 1000 time-steps of the stationary bubble simulation, which was run on a workstation with a 3.5GHz AMD Ryzen Threadripper 2920X 12-core processor. Table 9.5 presents the time comparison of the components of the interface reconstruction procedure for a single column at a single iteration (k) of the interface normal refinement procedure.

Table 9.5: Time comparison of the various components of the interface reconstruction procedure in a single column at a single iteration k of the interface normal refinement procedure.

Component	Total Time (μs)		% of Total Time	
	2D	3D	2D	3D
Convex decomposition	6.92	74.22	46.11	39.26
Vertex sorting	4.20	41.99	27.95	22.21
PLIC facet positioning	2.19	38.41	14.59	20.32
Bracketing	1.69	34.39	11.29	18.19

It is clear from the above, that convex decomposition (CD) of each column is the most costly component of the algorithm. However, it is important to note that since a vertex-centred approach is employed in this work, storage costs are significantly reduced by not explicitly storing the geometry of median dual-cells. Instead, each dual-cell is computed *ad hoc* during PLIC reconstruction in a column. Therefore, the cost of CD is dominated by the caching costs

of traversing the local connectivity of a grid vertex, and cumulatively constructing the triangles/tetrahedra that compose its dual-cell. Since this would be required regardless of whether a CD approach is employed (in the case where the median-dual cells are not stored), a non-CD approach like that of López *et al.* [35] would yield a similar cost in the context of this work. An alternative approach, to be investigated in future work, is the decomposition of only a portion of the column that lies in the vicinity of where the PLIC facet is expected to be.

Table 9.6: Total time comparison of curvature computation scheme and the incompressible solution procedure in a single time-step of a flow simulation in $\mathbb{R}^2$.

Component	Total Time (ms)					
	Accel Off			Accel On		
	Grid 1	Grid 2	Grid 3	Grid 1	Grid 2	Grid 3
Incompressible solver	0.75	3.17	14.30	0.77	3.11	14.53
Curvature computation	1.71	3.18	6.31	0.95	2.15	4.01

Table 9.7: Percentage time comparison of curvature computation scheme and the incompressible solution procedure in a single time-step of a flow simulation in $\mathbb{R}^2$.

Component	% of Total Time					
	Accel Off			Accel On		
	Grid 1	Grid 2	Grid 3	Grid 1	Grid 2	Grid 3
Incompressible solver	30.55	49.92	69.40	43.68	60.21	77.15
Curvature computation	69.45	50.08	30.60	56.32	39.79	22.85

Tables 9.6 and 9.7 show the time comparisons between the curvature computation and the incompressible flow solver for an average time-step in $\mathbb{R}^2$. The results shown are specifically for the circular interface positioned at $x = 0.0$ on grids $1 - 3$, with curvature computed using second-order polynomial fits. The associated solver time comparisons in $\mathbb{R}^3$ shall be reported in future work.

It is worth noting that the additional cost introduced to the scheme when using fourth-order polynomial fits for curvature computation is insignificant. This is because the interface normal refinement and PLIC reconstruction procedures currently dominate the cost of the curvature computation scheme.

Chapter 10

Conclusions and Future Work

10.1 Conclusions

The VOF method [24] has been widely used for simulating two-phase flows numerically, but poses a significant challenge to the accurate computation of interface-related geometric properties such as position, normal, and curvature. While the height-function (HF) method [10, 16, 18, 42, 46, 53] offers accurate interface representation on orthogonal structured (Cartesian) grids, no work has hitherto been done to extend the HF philosophy to non-orthogonal structured grids. To this end, this thesis proposed a novel method for higher-order accurate (up to 4^{th}-order in this work) polynomial-based VOF interface reconstruction on non-orthogonal structured grids. The developed scheme is finally applied to surface tension modelling.

This thesis commenced with a brief introduction to the HF method as well as its recent extension for arbitrary-order accurate curvature computations by Evrard *et al.* [16] on non-uniform Cartesian grids. This was for the purpose of outlining the philosophy of the interface reconstruction component of HF, which is later extended to non-orthogonal structured grids. To aid in this extension, the least squares polynomial fitting method of Jibben *et al.* [29] was generalised to arbitrary-order accuracy and compared to the exact fitting method of Evrard *et al.* [16] on Cartesian grids. Although a marginal reduction in curvature accuracy was observed in $\mathbb{R}^3$, the former was chosen for extension to non-orthogonal structured grids owing to—(i) its versatility in being applicable to various regions of the grid, and (ii) the invariance of the computed mean curvature with respect to arbitrary orientations of the local coordinate system about its vertical axis.

Similar to HF, the proposed method first constructs interface columns that straddle the VOF interface. This is done in a manner that ensures overlapping regions of columns where the directions of columns switch. This is to enable the interpolation of curvature between two or more regions of contiguous columns where an abrupt change in grid spacing from one region to another may occur. Next, the scheme conservatively constructs *PLIC facets* in each interface column using a novel analytical PLIC reconstruction procedure based on the convex decomposition of dual-cells in the column. The PLIC facet normals are iteratively refined *via* the use of local stencils of neighbouring PLIC facets to enable higher-order accurate curvature computation. Owing to the slow convergence of the local refinement procedure, a novel *spring-based* normal refinement acceleration procedure was developed that targets both local and global errors in the computed PLIC facet normals. This was shown to significantly decrease the number of normal refinement iterations required to maximise the curvature accuracy available to the scheme. The proposed scheme was implemented into a balanced-force continuum-surface-force [4] surface tension model for two-phase Newtonian fluid flow.

The curvature scheme was validated against a variety of analytical interface types in $\mathbb{R}^2$ and $\mathbb{R}^3$, whose positions were varied on a set of block-structured grids. Formal second- and fourth-order accuracy of the L_2- and L_∞-norms of the computed curvature was demonstrated in $\mathbb{R}^2$ and $\mathbb{R}^3$. A slight reduction in the order of accuracy of the L_∞-norm was observed for the ellipsoidal interface in $\mathbb{R}^3$, owing to the increased non-uniformity between neighbouring PLIC facets. Formal first-order accurate volume conservation was demonstrated in the central column of each column stencil. Formal second- and fourth-order accuracy was also demonstrated for the Laplace pressure jump across the interface of a 2D stationary bubble with a density ratio of 1000. A significant reduction (comparable to previous work [1, 46]) in the magnitudes of spurious velocities was observed in this case, and a stable result was obtained for both the second- and fourth-order curvature schemes. Similarly, the frequency of a 2D inviscid oscillating droplet in zero gravity was computed to second-order accuracy for both curvature schemes, owing to the second-order discretisation of the remaining terms in the governing equations.

The dominant cost of the conservative PLIC reconstruction procedure, and thereby the curvature computation scheme, was found to be that of the convex decomposition of interface columns. This was due to the caching costs of traversing the local connectivity tables of the grid vertices in each column to construct their median dual-cells in an *ad hoc* manner, to avoid significant increases in memory costs.

In conclusion, this thesis advances existing VOF-based free-surface modelling capabilities through the development of a novel higher-order polynomial-based VOF interface reconstruction scheme for non-orthogonal structured grids. While its present application was for surface tension modelling, the proposed scheme lays the foundation for highly accurate interface representation in numerous applications where non-orthogonal structured grids are indispensable.

10.2 Future Work

In the interest of building on the foundation laid by the proposed scheme, the following recommendations are made for potential future work:

- The current interface column construction and polynomial reconstruction procedures cannot handle *singularity* vertices in the grid. A possible solution for the handling of such cases is to avoid constructing columns through such vertices entirely, and to solely employ stencils comprising the surrounding columns for polynomial fitting purposes. An important consideration here is the accurate computation of the normals of the PLIC facets surrounding the singularity vertex. These must be computed in a manner that does not result in an abrupt spike in the local error of the computed curvature.

- Since convex decomposition of the interface columns was found to be the dominant cost of the PLIC reconstruction procedure (see Section 9.6), a significant reduction in this cost can be achieved by decomposing only a portion of the dual-cells in each column where the final PLIC facet is expected to be positioned. A robust means of doing so is required (*i.e.* one that does not compromise conservation).

- In the least squares polynomial fitting procedure (see Chapters 3 and 7), the enforcement of exact volume conservation in the central column of each stencil may be of future interest (*e.g.* for phase-change modelling). This can be achieved by accounting for the variation of the central column width/cross-section in the direction of the vertical axis of the local coordinate

system of the fitted polynomial. This would be straightforward to implement in $\mathbb{R}^2$ since the polynomial curve intersects exactly two edges in a given column. However, in $\mathbb{R}^3$, the polynomial surface intersects multiple edges and faces of the column, which would require a more involved solution.

- The early onset of ill-conditioning of the $\boldsymbol{A}$ matrices associated to the least squares polynomial fitting precedure is currently a bottle-neck to further grid refinement (relative to the chosen range of grid spacing values in this work). Further investigation is required to formulate a solution similar to that proposed for coordinate-based least squares fitting problems by Brubeck *et al.* [5], who employed an iterative Arnoldi orthogonalisation procedure.

- The impact of Laplacian smoothing of the underlying non-orthogonal structured grid is to be analysed in the context of the striations observed for higher-order polynomial fits. This is to be compared to non-orthogonal grids generated *via* smooth mappings of Cartesian grids.

- The investigation of contact-line phenomena is of increasing relevance and importance in the modelling of two-phase flow. Therefore, the extension of the developed polynomial-based interface reconstruction capabilities to account for the interaction of the liquid-gas interface with the domain boundary is necessary. Of particular importance is the accurate computation of the interface normal and curvature at PLIC facets adjacent to the boundary.

- The extension of the present work to highly anisotropic non-orthogonal structured grids ought to be of future interest, particularly for phase-change and contact-line modelling purposes. The main component in this work that does not extend straightforwardly to such grids is the PLIC facet normal refinement acceleration procedure in $\mathbb{R}^3$ (see Section 6.4.2). This is due to the PLIC facets being represented by circular disks. Representing large aspect-ratio PLIC facets by ellipses instead is a tractable solution. However, this would require an additional unknown parameter for each PLIC facet, to account for the orientation of the ellipse about its normal. An alternate solution is to employ multiple co-planar and co-centric circular disks of varying radii that are adjusted according to the proximity of each neighbouring PLIC facet.

- In the iterative Newton minimisation scheme of the normal refinement acceleration procedure (see Section 6.4.3), the incorporation of a line-search [39] along each Newton descent direction could significantly decrease the number of iterations required to minimise the total spring energy in the system.

- The normal refinement acceleration procedure is the only component of the present work that is yet to be parallelised. However, a major bottleneck to the efficient parallel scaling of the overall scheme is load-balancing over the VOF interface. This is exacerbated by the fact that the curvature computation accounts for a significant portion of the overall cost of each simulation (see Table 9.6). Therefore, an efficient load-balancing strategy over the interface is crucial.

- Whether the spring-based acceleration procedure introduces unwanted numerical damping of high-frequency capillary oscillations is to be investigated.

- Investigate the convergence properties of the curvature computation scheme for a circular, elliptical, spherical, and ellipsoidal bubble transported at a constant velocity across the flow domain. Further, a rigorous analysis of the sensitivity of the scheme to the accumulated errors of the VOF advection scheme is needed.

- The stationary bubble and oscillating droplet test cases are to be extended to 3D. The ability of the scheme to obtain a stable solution as well as formal second- and fourth-accuracy in 3D is to be demonstrated.
- It is instructive to investigate the performance of the developed scheme on more industrially relevant test cases with complex interface topologies, and compare to other curvature computation schemes in the literature.
- To aid researchers in comparing their schemes to the results generated in this work, it would be useful to release quantitative data for the various test cases. Examples of these are interface normals, curvature, PLIC facet definitions, polynomial coefficients, mesh coordinates and connectivity, *etc.*
- The robustness of the scheme when applied to industrially relevant cases is of utmost importance. A rigorous analysis of the limits of the developed scheme's ability to compute curvature accurately is required (*e.g.* when the radius of curvature of the interface is similar to the cell size). The scheme needs to be equipped with a fail-safe procedure to — (i) recognise when these limits are surpassed and subsequently (ii) switch to an alternate method to compute curvature to a reasonable accuracy. An appropriate case to test the robustness of the scheme is the modelling of interface coalescence and breakup.
- Similar to the methods of Ivey and Moin [28] and Evrard *et al.* [15], it would be worthwhile investigating the adaptability of the present work to fully unstructured grids in $\mathbb{R}^2$ and $\mathbb{R}^3$.
- Similar to the work of Qi *et al.* [50], it is worth exploring the applicability of artificial neural networks in the context of curvature computations on the grids employed in this work. A key consideration when training such a neural network would be the large variety of possible topologies for a given curvature stencil.

Bibliography

[1] T. Abadie, J. Aubin, and 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), pp. 611–636.

[2] M. O. Abu-Al-Saud, S. Popinet, and H. A. Tchelepi, *A conservative and well-balanced surface tension model*, Journal of Computational Physics, 371 (2018), pp. 896–913.

[3] H. T. Ahn and M. Shashkov, *Multi-material interface reconstruction on generalized polyhedral meshes*, Journal of Computational Physics, 226 (2007), pp. 2096–2132.

[4] J. Brackbill, D. Kothe, and C. Zemach, *A continuum method for modeling surface tension*, Journal of Computational Physics, 100 (1992), pp. 335–354.

[5] P. D. Brubeck, Y. Nakatsukasa, and L. N. Trefethen, *Vandermonde with Arnoldi*, SIAM Review, 63 (2021), pp. 405–415.

[6] Z. Cao, D. Sun, J. Wei, B. Yu, and J. Li, *A coupled volume-of-fluid and level set method based on general curvilinear grids with accurate surface tension calculation*, Journal of Computational Physics, 396 (2019), pp. 799–818.

[7] Z. Cao, J. Zhou, A. Liu, D. Sun, B. Yu, and J. Wei, *A three dimensional coupled VOF and Level set (VOSET) method with and without phase change on general curvilinear grids*, Chemical Engineering Science, 223 (2020).

[8] D. M. Changfoot, A. G. Malan, and J. Nordström, *Hybrid Computational-Fluid-Dynamics Platform to Investigate Aircraft Trailing Vortices*, Journal of Aircraft, 56 (2019), pp. 344–355.

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

[10] S. J. Cummins, M. M. Francois, and D. B. Kothe, *Estimating curvature from volume fractions*, Computers and Structures, 83 (2005), pp. 425–434.

[11] D. Dai and A. Y. Tong, *An analytical interface reconstruction algorithm in the PLIC-VOF method for 2D polygonal unstructured meshes*, International Journal for Numerical Methods in Fluids, 88 (2018), pp. 265–276.

[12] O. Desjardins, V. Moureau, and H. Pitsch, *An accurate conservative level set/ghost fluid method for simulating turbulent atomization*, Journal of Computational Physics, 227 (2008), pp. 8395–8416.

[13] S. Diot and M. M. François, *An interface reconstruction method based on an analytical formula for 3D arbitrary convex cells*, Journal of Computational Physics, 305 (2016), pp. 63–74.

[14] S. Diot, M. M. François, and E. D. Dendy, *An interface reconstruction method based on analytical formulae for 2D planar and axisymmetric arbitrary convex cells*, Journal of Computational Physics, 275 (2014), pp. 53–64.

[15] F. Evrard, F. Denner, and B. van Wachem, *Estimation of curvature from volume fractions using parabolic reconstruction on two-dimensional unstructured meshes*, Journal of Computational Physics, 351 (2017), pp. 271–294.

[16] F. Evrard, F. Denner, and B. van Wachem, *Height-function curvature estimation with arbitrary order on non-uniform Cartesian grids*, Journal of Computational Physics: X, 7 (2020), p. 100060.

[17] M. M. Francois, S. J. Cummins, E. D. Dendy, D. B. Kothe, J. M. Sicilian, and 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 (2006), pp. 141–173.

[18] M. M. Francois and B. K. Swartz, *Interface curvature via volume fractions, heights, and mean values on nonuniform rectangular grids*, Journal of Computational Physics, 229 (2010), pp. 527–540.

[19] D. Fuster, G. Agbaglah, C. Josserand, S. Popinet, and S. Zaleski, *Numerical simulation of droplets, bubbles and waves: state of the art*, Fluid Dynamics Research, 41 (2009), p. 065001.

[20] D. Fyfe, E. Oran, and M. Fritts, *Surface tension and viscosity with lagrangian hydrodynamics on a triangular mesh*, Journal of Computational Physics, 76 (1988), pp. 349–384.

[21] M. Haghshenas, J. A. Wilson, and R. Kumar, *Algebraic coupled level set-volume of fluid method for surface tension dominant two-phase flows*, International Journal of Multiphase Flow, 90 (2017), pp. 13–28.

[22] M. Herrmann, *A balanced force refined level set grid method for two-phase flows on unstructured flow solver grids*, Journal of Computational Physics, 227 (2008), pp. 2674–2706.

[23] J. A. Heyns, A. G. Malan, T. M. Harms, and 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 (2013), pp. 788–804.

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

[25] J. Huang, P. M. Carrica, and F. Stern, *A geometry-based level set method for curvilinear overset grids with application to ship hydrodynamics*, International Journal for Numerical Methods in Fluids, 68 (2012), pp. 494–521.

[26] N. A. Ilangakoon, A. G. Malan, and B. W. Jones, *A higher-order accurate surface tension modelling volume-of-fluid scheme for 2D curvilinear meshes*, Journal of Computational Physics, 420 (2020), p. 109717.

[27] K. Ito, T. Kunugi, S. Ohno, H. Kamide, and H. Ohshima, *A high-precision calculation method for interface normal and curvature on an unstructured grid*, Journal of Computational Physics, 273 (2014), pp. 38–53.

[28] C. B. Ivey and P. Moin, *Accurate interface normal and curvature estimates on three-dimensional unstructured non-convex polyhedral meshes*, Journal of Computational Physics, 300 (2015), pp. 365–386.

[29] Z. Jibben, N. Carlson, and M. Francois, *A paraboloid fitting technique for calculating curvature from piecewise-linear interface reconstructions on 3D unstructured meshes*, Computers and Mathematics with Applications, 78 (2019), pp. 643–653.

[30] B. W. Jones, A. G. Malan, and N. A. Ilangakoon, *The initialisation of volume fractions for unstructured grids using implicit surface definitions.*, Computers and Fluids, 179 (2019), pp. 194–205.

[31] H. Lamb, *Hydrodynamics*, Dover Publications, New York, 1945.

[32] R. W. Lewis and A. G. Malan, *Continuum thermodynamic modeling of drying capillary particulate materials via an edge-based algorithm*, Computer Methods in Applied Mechanics and Engineering, 194 (2005), pp. 2043–2057.

[33] J. López and J. Hernández, *Analytical and geometrical tools for 3D volume of fluid methods in general grids*, Journal of Computational Physics, 227 (2008), pp. 5939–5948.

[34] J. López, J. Hernández, P. Gómez, and F. Faura, *A new volume conservation enforcement method for PLIC reconstruction in general convex grids*, Journal of Computational Physics, 316 (2016), pp. 338–359.

[35] J. López, J. Hernández, P. Gómez, and F. Faura, *Non-convex analytical and geometrical tools for volume truncation, initialization and conservation enforcement in VOF methods*, Journal of Computational Physics, 392 (2019), pp. 666–693.

[36] A. Malan and O. Oxtoby, *An accelerated, fully-coupled, parallel 3D hybrid finite-volume fluid–structure interaction scheme*, Computer Methods in Applied Mechanics and Engineering, 253 (2013), pp. 426–438.

[37] L. C. Malan, A. G. Malan, S. Zaleski, and P. G. Rousseau, *A geometric vof method for interface resolved phase change and conservative thermal energy advection*, (2020), pp. 1–24.

[38] T. Marić, D. B. Kothe, and D. Bothe, *Unstructured un-split geometrical Volume-of-Fluid methods – A review*, Journal of Computational Physics, 420 (2020), p. 109695.

[39] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, Springer New York, jul 2006.

[40] S. Osher and J. A. Sethian, *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations*, Journal of Computational Physics, 79 (1988), pp. 12–49.

[41] M. Owkes and O. Desjardins, *A discontinuous Galerkin conservative level set scheme for interface capturing in multiphase flows*, Journal of Computational Physics, 249 (2013), pp. 275–302.

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

[43] O. F. Oxtoby, A. G. Malan, and J. A. Heyns, *A computationally efficient 3D finite-volume scheme for violent liquid-gas sloshing*, International Journal for Numerical Methods in Fluids, 79 (2015), pp. 306–321.

[44] R. Panahi, E. Jahanbakhsh, and M. S. Seif, *Development of a VoF-fractional step solver for floating body motion simulation*, Applied Ocean Research, 28 (2006), pp. 171–181.

[45] J. E. Pilliod and E. G. Puckett, *Second-order accurate volume-of-fluid algorithms for tracking material interfaces*, Journal of Computational Physics, 199 (2004), pp. 465–502.

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

[47] S. Popinet and S. Zaleski, *A front-tracking algorithm for accurate representation of surface tension*, International Journal for Numerical Methods in Fluids, 30 (1999), pp. 775–793.

[48] S. Popinet, *Numerical Models of Surface Tension*, Annual Review of Fluid Mechanics, 50 (2018), pp. 49–75.

[49] A. Pressley, *Elementary Differential Geometry*, Springer Undergraduate Mathematics Series, Springer London, London, 2010.

[50] Y. Qi, J. Lu, R. Scardovelli, S. Zaleski, and G. Tryggvason, *Computing curvature for volume of fluid methods using machine learning*, Journal of Computational Physics, 377 (2019), pp. 155–161.

[51] Y. Renardy and M. Renardy, *PROST: A parabolic reconstruction of surface tension for the volume-of-fluid method*, Journal of Computational Physics, 183 (2002), pp. 400–421.

[52] W. J. Rider and D. B. Kothe, *Reconstructing Volume Tracking*, Journal of Computational Physics, 141 (1998), pp. 112–152.

[53] M. Rudman, *Volume-tracking methods for interfacial flow calculations*, International Journal for Numerical Methods in Fluids, 24 (1997), pp. 671–691.

[54] Y. Sato and B. Ničeno, *A sharp-interface phase change model for a mass-conservative interface tracking method*, Journal of Computational Physics, 249 (2013), pp. 127–161.

[55] R. Scardovelli and S. Zaleski, *Direct Numerical Simulation of Free-surface and Interfacial Flow*, Annual Review of Fluid Mechanics, 31 (1999), pp. 567–603.

[56] H. Scheufler and J. Roenby, *Accurate and efficient surface reconstruction from volume fraction data on general meshes*, Journal of Computational Physics, 383 (2019), pp. 1–23.

[57] J. Schlottke and B. Weigand, *Direct numerical simulation of evaporating droplets*, Journal of Computational Physics, 227 (2008), pp. 5215–5237.

[58] M. Skarysz, A. Garmory, and M. Dianat, *An iterative interface reconstruction method for PLIC in general convex grids as part of a Coupled Level Set Volume of Fluid solver*, Journal of Computational Physics, 368 (2018), pp. 254–276.

[59] M. Sussman and M. Ohta, *High-order Techniques for Calculating Surface Tension Forces*, in Free Boundary Problems, vol. 154, Birkhäuser Basel, Basel, 2007, pp. 425–434.

[60] M. Sussman and E. G. Puckett, *A Coupled Level Set and Volume-of-Fluid Method for Computing 3D and Axisymmetric Incompressible Two-Phase Flows*, Journal of Computational Physics, 162 (2000), pp. 301–337.

[61] D. Torres and J. Brackbill, *The Point-Set Method: Front-Tracking without Connectivity*, Journal of Computational Physics, 165 (2000), pp. 620–644.

[62] G. Tryggvason, R. Scardovelli, and S. Zaleski, *Direct Numerical Simulations of Gas–Liquid Multiphase Flows*, Cambridge University Press, Cambridge, jan 2011.

[63] O. Ubbink and R. Issa, *A Method for Capturing Sharp Fluid Interfaces on Arbitrary Meshes*, Journal of Computational Physics, 153 (1999), pp. 26–50.

[64] D. T. Waidyaratne, *Analysis of the Accuracy of Free-surface Curvature for Surface-tension Modelling*, tech. rep., University of Cape Town, Cape Town, 2020.

[65] Z. Wang, J. Yang, and F. Stern, *A new volume-of-fluid method with a constructed distance function on general structured grids*, Journal of Computational Physics, 231 (2012), pp. 3703–3722.

[66] S. W. Welch and J. Wilson, *A Volume of Fluid Based Method for Fluid Flows with Phase Change*, Journal of Computational Physics, 160 (2000), pp. 662–682.

[67] Q. Zhang, *HFES: A Height Function Method with Explicit Input and Signed Output for High-Order Estimations of Curvature and unit Vectors of Planar Curves*, SIAM Journal on Numerical Analysis, 55 (2017), pp. 1024–1056.

Appendix A

Curvature of an Intersection Curve Between a Surface and a Slicing Plane

This chapter details the procedure for computing the curvature of the curve of intersection between a regular surface S and a vertical slicing plane P that passes through the origin. Let S be described by the graph of the function $z = f(x, y)$, and the slicing plane P by $\boldsymbol{n}_P \cdot \boldsymbol{x} = 0$, where $\boldsymbol{n}_P = [n_x \;\; n_y \;\; 0]^T$. The components, n_x and n_y, of the plane normal can be defined as a function of the angle $\theta \in [0, 2\pi]$ between P and the xz-plane as $n_x = -\sin\theta$ and $n_x = \cos\theta$. The curvature in the plane P at a point on the intersection curve between S and P can be computed as a function of the normal curvature, κ_n, of S in the direction of the tangent to the curve. This procedure is detailed next.

Let an arbitrary point on S be defined by the vector-valued function $\boldsymbol{r} = [x \;\; y \;\; f(x,y)]^T$. The first- and second-order partial derivatives of $\boldsymbol{r}$ with respect to x and y are

$$\boldsymbol{r}_x = \begin{bmatrix} 1 \\ 0 \\ f_x \end{bmatrix}, \;\; \boldsymbol{r}_y = \begin{bmatrix} 0 \\ 1 \\ f_y \end{bmatrix}, \;\; \boldsymbol{r}_{xx} = \begin{bmatrix} 0 \\ 0 \\ f_{xx} \end{bmatrix}, \;\; \boldsymbol{r}_{xy} = \boldsymbol{r}_{yx} = \begin{bmatrix} 0 \\ 0 \\ f_{xy} \end{bmatrix}, \text{ and } \boldsymbol{r}_{yy} = \begin{bmatrix} 0 \\ 0 \\ f_{yy} \end{bmatrix}.$$

From the above, the unit normal of S at $\boldsymbol{r}$ can be defined as $\boldsymbol{n}_S = (\boldsymbol{r}_x \times \boldsymbol{r}_y)/|\boldsymbol{r}_x \times \boldsymbol{r}_y|$. The coefficients of the first and second fundamental forms [49], I and II, of S at $\boldsymbol{r}$ are defined as

$$E = \boldsymbol{r}_x \cdot \boldsymbol{r}_x, \;\; F = \boldsymbol{r}_x \cdot \boldsymbol{r}_x, \;\; G = \boldsymbol{r}_x \cdot \boldsymbol{r}_x,$$

and

$$e = \boldsymbol{r}_{xx} \cdot \boldsymbol{n}_S, \;\; f = \boldsymbol{r}_{xy} \cdot \boldsymbol{n}_S, \;\; g = \boldsymbol{r}_{yy} \cdot \boldsymbol{n}_S,$$

respectively. Next, the tangent to the intersection curve, $\boldsymbol{t} = \boldsymbol{n}_P \times \boldsymbol{n}_S$, can be expressed as a linear combination of the basis vectors, $\boldsymbol{r}_x$ and $\boldsymbol{r}_y$, of the tangent space of S at $\boldsymbol{r}$ as

$$\mathbf{t} = \left(\mathbf{t} \cdot \frac{\mathbf{r}_x}{|\mathbf{r}_x|}\right)\frac{\mathbf{r}_x}{|\mathbf{r}_x|} + \left(\mathbf{t} \cdot \frac{\mathbf{r}_y}{|\mathbf{r}_y|}\right)\frac{\mathbf{r}_y}{|\mathbf{r}_y|} = \left(\frac{\mathbf{t} \cdot \mathbf{r}_x}{\mathbf{r}_x \cdot \mathbf{r}_x}\right)\mathbf{r}_x + \left(\frac{\mathbf{t} \cdot \mathbf{r}_y}{\mathbf{r}_y \cdot \mathbf{r}_y}\right)\mathbf{r}_y = a\mathbf{r}_x + b\mathbf{r}_y.$$

The normal curvature of the intersection curve at $\boldsymbol{r}$ in the direction $\boldsymbol{t}$ is then defined as

$$\kappa_n = \frac{II(\boldsymbol{t}, \boldsymbol{t})}{I(\boldsymbol{t}, \boldsymbol{t})} = \frac{ea^2 + 2fab + gb^2}{Ea^2 + 2Fab + Gb^2}$$

Finally, the curvature, κ, of the intersection curve at $\boldsymbol{r}$ in the plane P can be computed *via* Meusnier's Theorem [49], which states that

$$\kappa_n = \kappa \sin\theta,$$

where θ is the angle between P and the tangent plane of S at $\boldsymbol{r}$.

Appendix B

Conservative PLIC Reconstruction Derivations

B.1 Derivations for Sweep-sections s_0 and s_2

Using Eqs. (5.7), (5.8), (5.12), (5.17), and (5.18), the incremental volume in sweep-sections s_0 and s_2 can be derived as

$$\begin{aligned}
\Delta\mathcal{V}_\tau(\Delta\eta)\big|_{s_0/s_2} &= \frac{1}{6}\left\|\left(h\hat{\boldsymbol{t}}_{03}\right)\cdot\left(\left(h\hat{\boldsymbol{t}}_{ab}\right)\times\left(h\hat{\boldsymbol{t}}_{cd}\right)\right)\right\| - \frac{1}{6}\left\|\left(h_{lo}\hat{\boldsymbol{t}}_{03}\right)\cdot\left(\left(h_{lo}\hat{\boldsymbol{t}}_{ab}\right)\times\left(h_{lo}\hat{\boldsymbol{t}}_{cd}\right)\right)\right\| \\
&= \frac{1}{6}\left\|\hat{\boldsymbol{t}}_{03}\cdot\left(\hat{\boldsymbol{t}}_{ab}\times\hat{\boldsymbol{t}}_{cd}\right)\right\|\left(h^3 - h_{lo}^3\right) \\
&= \frac{1}{6}\left\|\hat{\boldsymbol{t}}_{03}\cdot\left(\hat{\boldsymbol{t}}_{ab}\times\hat{\boldsymbol{t}}_{cd}\right)\right\|\left(\left(h_{lo}+\Delta\eta\right)^3 - h_{lo}^3\right) \\
&= \frac{1}{6}\left\|\hat{\boldsymbol{t}}_{03}\cdot\left(\hat{\boldsymbol{t}}_{ab}\times\hat{\boldsymbol{t}}_{cd}\right)\right\|\left(\Delta\eta^3 + 3h_{lo}\Delta\eta^2 + 3h_{lo}^2\Delta\eta\right).
\end{aligned}$$

B.2 Derivations for Sweep-section s_1

Using Eqs. (5.7) – (5.9) and (5.20) – (5.22), the volume $\mathcal{V}_\tau(h)|_{s_1}^{mid}$ in Fig. 5.7 can be derived as

$$\begin{aligned}
\mathcal{V}_\tau(h)|_{s_1}^{mid} &= \frac{1}{6}\boldsymbol{t}_{df}\cdot\left(\boldsymbol{t}_{da}\times\boldsymbol{t}_{dc}\right) \\
&= \frac{1}{6}\left(\boldsymbol{x}_f - \boldsymbol{x}_d\right)\cdot\Big(\left(\boldsymbol{x}_a - \boldsymbol{x}_d\right)\times\left(\boldsymbol{x}_d - \boldsymbol{x}_c\right)\Big) \\
&= \frac{1}{6}\Big(\boldsymbol{t}_{ab} + h\left(\hat{\boldsymbol{t}}_{02} - \hat{\boldsymbol{t}}_{13}\right)\Big)\cdot\Big(\left(-h\hat{\boldsymbol{t}}_{13}\right)\times\left(\boldsymbol{t}_{ac} - h\hat{\boldsymbol{t}}_{13}\right)\Big) \\
&= -\frac{h}{6}\left[\boldsymbol{t}_{ab}\cdot\Big(\hat{\boldsymbol{t}}_{13}\times\left(\boldsymbol{t}_{ac} - h\hat{\boldsymbol{t}}_{13}\right)\Big) + h\left(\hat{\boldsymbol{t}}_{02} - \hat{\boldsymbol{t}}_{13}\right)\cdot\Big(\hat{\boldsymbol{t}}_{13}\times\left(\boldsymbol{t}_{ac} - h\hat{\boldsymbol{t}}_{13}\right)\Big)\right] \\
&= -\frac{h}{6}\left[\boldsymbol{t}_{ab}\cdot\left(\hat{\boldsymbol{t}}_{13}\times\boldsymbol{t}_{ac}\right) + h\left(\hat{\boldsymbol{t}}_{02} - \hat{\boldsymbol{t}}_{13}\right)\cdot\left(\hat{\boldsymbol{t}}_{13}\times\boldsymbol{t}_{ac}\right)\right] \\
&= -\frac{h}{6}\left[\boldsymbol{t}_{ab}\cdot\left(\hat{\boldsymbol{t}}_{13}\times\boldsymbol{t}_{ac}\right) + h\hat{\boldsymbol{t}}_{02}\cdot\left(\hat{\boldsymbol{t}}_{13}\times\boldsymbol{t}_{ac}\right)\right] \\
&= -\frac{h}{6}\left(\hat{\boldsymbol{t}}_{13}\times\boldsymbol{t}_{ac}\right)\cdot\left(\boldsymbol{t}_{ab} + h\hat{\boldsymbol{t}}_{02}\right) \\
&= \frac{h}{6}\left(\boldsymbol{t}_{ac}\times\hat{\boldsymbol{t}}_{13}\right)\cdot\left(\boldsymbol{t}_{ab} + h\hat{\boldsymbol{t}}_{02}\right).
\end{aligned}$$

Likewise, by defining $\hat{\boldsymbol{t}}_A = (\hat{\boldsymbol{t}}_{02} - \hat{\boldsymbol{t}}_{13})$ and $\hat{\boldsymbol{t}}_B = (\hat{\boldsymbol{t}}_{03} - \hat{\boldsymbol{t}}_{12})$, the volume $\mathcal{V}_\tau(h)|_{s_1}^{top}$ is derived as

$$\begin{aligned}
\mathcal{V}_\tau(h)|_{s_1}^{top} &= \frac{h}{6}\boldsymbol{n}_p^k \cdot \big(\boldsymbol{t}_{df} \times (\boldsymbol{t}_{dg} - \boldsymbol{t}_{de})\big) \\
&= \big(\boldsymbol{t}_{ab} + h\hat{\boldsymbol{t}}_A\big) \times \big(\boldsymbol{t}_{ac} + h\hat{\boldsymbol{t}}_B\big) \\
&= \frac{h}{6}\boldsymbol{n}_p \cdot \big(h^2\hat{\boldsymbol{t}}_A \times \hat{\boldsymbol{t}}_B + h(\boldsymbol{t}_{ab} \times \hat{\boldsymbol{t}}_B - \boldsymbol{t}_{ac} \times \hat{\boldsymbol{t}}_A) + \boldsymbol{t}_{ab} \times \boldsymbol{t}_{ac}\big).
\end{aligned}$$

The coefficient m_1 in Eq. (5.27) is derived as

$$\begin{aligned}
m_1 &= \frac{1}{6}\big(2\boldsymbol{n}_p^k \cdot (\boldsymbol{t}_{ab} \times \boldsymbol{t}_{ac}) + \boldsymbol{t}_{ab} \cdot (\boldsymbol{t}_{ac} \times \hat{\boldsymbol{t}}_{13})\big) \\
&= \frac{1}{6}\big(2\boldsymbol{t}_{ab} \cdot (\boldsymbol{t}_{ac} \times \boldsymbol{n}_p^k) + \boldsymbol{t}_{ab} \cdot (\boldsymbol{t}_{ac} \times \hat{\boldsymbol{t}}_{13})\big) \\
&= \frac{1}{6}\boldsymbol{t}_{ab} \cdot \big(2\boldsymbol{t}_{ac} \times \boldsymbol{n}_p^k + \boldsymbol{t}_{ac} \times \hat{\boldsymbol{t}}_{13}\big) \\
&= \frac{1}{6}\boldsymbol{t}_{ab} \cdot \big(\boldsymbol{t}_{ac} \times (2\boldsymbol{n}_p^k + \hat{\boldsymbol{t}}_{13})\big).
\end{aligned}$$

Lastly, the incremental volume $\Delta\mathcal{V}_\tau(\Delta\eta)|_{s_1}$ is derived as

$$\begin{aligned}
\Delta\mathcal{V}_\tau(\Delta\eta)|_{s_1} &= \mathcal{V}_\tau(h)|_{s_1} - \mathcal{V}_\tau(h_{lo})|_{s_1} \\
&= m_1\big(h - h_{lo}\big) + m_2\big(h^2 - h_{lo}^2\big) + m_3\big(h^3 - h_{lo}^3\big) \\
&= m_1\big((h_{lo} + \Delta\eta) - h_{lo}\big) + m_2\big((h_{lo} + \Delta\eta)^2 - h_{lo}^2\big) + m_3\big((h_{lo} + \Delta\eta)^3 - h_{lo}^3\big) \\
&= m_1\Delta\eta + m_2\big(\Delta\eta^2 + 2h_{lo}\Delta\eta\big) + m_3\big(\Delta\eta^3 + 3h_{lo}\Delta\eta^2 + 3h_{lo}^2\Delta\eta\big) \\
&= \big(3m_3h_{lo}^2 + 2m_2h_{lo} + m_1\big)\Delta\eta + \big(3m_3h_{lo} + m_2\big)\Delta\eta^2 + m_3\Delta\eta^3
\end{aligned}$$

Appendix C

PLIC Facet Normal Refinement Derivations

C.1 Normal Refinement on Misaligned Stencils

C.1.1 2D Derivations

Consider the parabola

$$P_2(\xi) = \sum_{i=1}^{3} c_i \xi^{i-1}.$$

An arbitrary point on this parabola is $\boldsymbol{x}_{P_2} = (\xi, P_2(\xi))$. Let $\boldsymbol{x}_p^k = (\xi_p^k, \eta_p^k)$ be the definition of the centroid of PLIC facet p in the local coordinate system. Then, the quadrance (distance squared) between $\boldsymbol{x}_p^k$ and $P_2(\xi)$ is

$$Q = \|\boldsymbol{x}_{P_2} - \boldsymbol{x}_p^k\|^2 = (\xi - \xi_p^k)^2 + (c_1 + c_2\xi + c_3\xi^2 - \eta_p^k)^2.$$

Minimising Q with respect to η gives

$$\begin{aligned}
\frac{dQ}{d\xi} &= 2(\xi - \xi_p^k) + 2(c_1 + c_2\xi + c_3\xi^2 - \eta_p^k)(c_2 + 2c_3\xi) \\
&= 2(\xi - \xi_p^k) + 2\left(c_2(c_1 - \eta_p^k) + \big(c_2^2 + 2c_3(c_1 - \eta_p^k)\big)\xi + 3c_2c_3\xi^2 + 2c_3^2\xi^3\right) \\
&= 2\big(c_2(c_1 - \eta_p^k) - \xi_p^k\big) + 2\big(c_2^2 + 2c_3(c_1 - \eta_p^k) + 1\big)\xi + 6c_3c_2\xi^2 + 4c_3^2\xi^3.
\end{aligned}$$

C.1.2 3D Derivations

Consider the paraboloid

$$P_2(\xi, \eta) = \sum_{s=0}^{2} \sum_{r=0}^{2-s} c_i \xi^r \eta^s,$$
$$i = s + 1 + \frac{1}{2}(r + s)(r + s + 1) \in [1, 6].$$

An arbitrary point on this paraboloid is $\boldsymbol{x}_{P_2} = (\xi, \eta, P_2(\xi, \eta))$. Let $\boldsymbol{x}_p^k = (\xi_p^k, \eta_p^k, \psi_p^k)$ be the definition of the centroid of PLIC facet p in the local coordinate system. Then, the quadrance

between $\boldsymbol{x}_p^k$ and $P_2(\xi,\eta)$ is

$$\begin{aligned} Q(\xi,\eta) &= \|\boldsymbol{x}_{P_2} - \boldsymbol{x}_p^k\|^2 \\ &= (\xi-\xi_p^k)^2 + (\eta-\eta_p^k)^2 + (c_1 + c_2\xi + c_3\eta + c_4\xi^2 + c_5\xi\eta + c_6\eta^2 - \psi_p^k)^2. \end{aligned}$$

The first-order partial derivatives of Q with respect to ξ and η are derived as follows:

$$\begin{aligned} Q_\xi &= 2(\xi-\xi_p^k) + 2(c_1 + c_2\xi + c_3\eta + c_4\xi^2 + c_5\xi\eta + c_6\eta^2 - \psi_p^k)(c_2 + 2c_4\xi + c_5\eta) \\ &= 2\Big(c_2(c_1-\psi_p^k) - \xi_p^k + \big(c_2^2 + 2c_4(c_1-\psi_p^k) + 1\big)\xi + \big(c_5(c_1-\psi_p^k) + c_2c_3\big)\eta + 3c_2c_5\xi^2 + \\ &\qquad 2(c_2c_5 + c_3c_4)\xi\eta + (c_2c_6 + c_3c_5)\eta^2 + 2c_5^2\xi^3 + 3c_4c_5\xi^2\eta + (c_5^2 + 2c_4c_6)\xi\eta^2 + c_4c_6\eta^3\Big) \\ &= 2\big(c_2d_1 - \xi_p^k + d_2\xi + d_3\eta + 3c_2c_5\xi^2 + 2d_4\xi\eta + d_5\eta^2 + 2c_5^2\xi^3 + 3c_4c_5\xi^2\eta + \\ &\qquad\qquad d_6\xi\eta^2 + c_4c_6\eta^3\big), \end{aligned}$$

where

$$\begin{gathered} d_1 = c_1 - \psi_p^k, \quad d_2 = c_2^2 + 2c_4d_1 + 1, \quad d_3 = c_5d_1 + c_2c_3 \\ d_4 = c_2c_5 + c_3c_4, \quad d_5 = c_2c_6 + c_3c_5, \text{ and } d_6 = c_5^2 + 2c_4c_6. \end{gathered}$$

Likewise,

$$\begin{aligned} Q_\eta &= 2(\eta-\psi_p^k) + 2(c_1 + c_2\xi + c_3\eta + c_4\xi^2 + c_5\xi\eta + c_6\eta^2 - \psi_p^k)(c_3 + c_5\xi + 2c_6\eta) \\ &= 2\Big(c_3(c_1-\psi_p^k) - \xi_p^k + \big(c_5(c_1-\psi_p^k) + c_2c_3\big)\xi + (c_3^2 + 2c_6d_1 + 1)\eta + (c_2c_5 + c_3c_4)\xi^2 + \\ &\qquad 2(c_2c_6 + c_3c_5)\xi\eta + 3c_3c_6\eta^2 + c_4c_5\xi^3 + (c_5^2 + 2c_4c_6)\xi^2\eta + 3c_4c_6\xi\eta^2 + 2c_6^2\eta^3\Big) \\ &= 2\big(c_3d_1 - \xi_p^k + d_3\xi + d_7\eta + d_4\xi^2 + 2d_5\xi\eta + 3c_3c_6\eta^2 + c_4c_5\xi^3 + d_6\xi^2\eta + \\ &\qquad\qquad 3c_4c_6\xi\eta^2 + 2c_6^2\eta^3\big), \end{aligned}$$

where

$$d_7 = c_3^2 + 2c_6d_1 + 1.$$

Lastly, the second-order partial derivatives $Q_{\xi\xi}$, $Q_{\eta\eta}$, and $Q_{\xi\eta} = Q_{\eta\xi}$ can be straightforwardly derived from the above expressions for Q_ξ and Q_η.

C.2 Normal Refinement Acceleration Procedure

C.2.1 2D Derivations

Using the definitions in Eqs. (6.9) – (6.15), the total spring energy in the system is derived as follows:

$$\begin{aligned}
Q(\boldsymbol{\theta}) &= \sum_{(p,q)\in\mathfrak{P}} \|\boldsymbol{x}_{pq} - \boldsymbol{x}_{qp}\|^2 \\
&= \sum_{(p,q)\in\mathfrak{P}} (\boldsymbol{x}_{pq} - \boldsymbol{x}_{qp})\cdot(\boldsymbol{x}_{pq} - \boldsymbol{x}_{qp}) \\
&= \sum_{(p,q)\in\mathfrak{P}} (\boldsymbol{x}_p^k + r_p\hat{\boldsymbol{t}}_{pq} - \boldsymbol{x}_q^k - r_q\hat{\boldsymbol{t}}_{qp})\cdot(\boldsymbol{x}_p^k + r_p\hat{\boldsymbol{t}}_{pq} - \boldsymbol{x}_q^k - r_q\hat{\boldsymbol{t}}_{qp}) \\
&= \sum_{(p,q)\in\mathfrak{P}} (r_p\hat{\boldsymbol{t}}_{pq} - r_q\hat{\boldsymbol{t}}_{qp} - \boldsymbol{d}_{pq})\cdot(r_p\hat{\boldsymbol{t}}_{pq} - r_q\hat{\boldsymbol{t}}_{qp} - \boldsymbol{d}_{pq}) \\
&= \sum_{(p,q)\in\mathfrak{P}} \big(r_p^2\hat{\boldsymbol{t}}_{pq}\cdot\hat{\boldsymbol{t}}_{pq} + r_q^2\hat{\boldsymbol{t}}_{qp}\cdot\hat{\boldsymbol{t}}_{qp} - 2r_pr_q\hat{\boldsymbol{t}}_{pq}\cdot\hat{\boldsymbol{t}}_{qp} + (\boldsymbol{d}_{pq} + 2r_q\hat{\boldsymbol{t}}_{qp} - 2r_p\hat{\boldsymbol{t}}_{pq})\cdot\boldsymbol{d}_{pq}\big),
\end{aligned}$$

Given the definition of $\boldsymbol{s}_{pq}$ in Eq. (6.18), and noting that the contributions from all $q\in\mathfrak{N}_p$ are to be accounted for, the first-order partial derivatives of $Q(\boldsymbol{\theta})$ with respect to each θ_p can be derived as

$$\begin{aligned}
\frac{\partial Q}{\partial\theta_p} &= \sum_{q\in\mathfrak{N}_p}\left(2r_p^2\hat{\boldsymbol{t}}_{pq}\cdot\frac{\partial\hat{\boldsymbol{t}}_{pq}}{\partial\theta_p} - 2r_pr_q\hat{\boldsymbol{t}}_{qp}\cdot\frac{\partial\hat{\boldsymbol{t}}_{pq}}{\partial\theta_p} - 2r_p\boldsymbol{d}_{pq}\cdot\frac{\partial\hat{\boldsymbol{t}}_{pq}}{\partial\theta_p}\right) \\
&= 2r_p\sum_{q\in\mathfrak{N}_p}\frac{\partial\hat{\boldsymbol{t}}_{pq}}{\partial\theta_p}\cdot\big(r_p\hat{\boldsymbol{t}}_{pq} - r_q\hat{\boldsymbol{t}}_{qp} - \boldsymbol{d}_{pq}\big) \\
&= 2r_p\sum_{q\in\mathfrak{N}_p}\frac{\partial\hat{\boldsymbol{t}}_{pq}}{\partial\theta_p}\cdot\boldsymbol{s}_{pq}, \quad \forall p\in\{1,2,...,\mathcal{M}\}.
\end{aligned}$$

Likewise, the second-order partial derivatives of $Q(\boldsymbol{\theta})$ with respect to θ_p follow straightforwardly from the final expression above.

C.2.2 3D Derivations

The total spring energy, $Q(\boldsymbol{\gamma})$, in $\mathbb{R}^3$ (see Eq. (6.25)) is derived in the same manner as $Q(\boldsymbol{\theta})$ above. Likewise, by letting $\gamma\in\{\theta,\phi\}$, the first-order partial derivatives of $Q(\boldsymbol{\gamma})$ are derived in the same manner as those of $Q(\boldsymbol{\theta})$ above. The first-order partial derivatives of $\hat{\boldsymbol{t}}_{pq}$ that feature in Eq. (6.26) are derived from Eq. (6.22) as

$$\begin{aligned}
\frac{\partial\hat{\boldsymbol{t}}_{pq}}{\partial\gamma_p} &= \frac{\|\boldsymbol{r}_{pq}\|\left(\frac{\partial\boldsymbol{r}_{pq}}{\partial\gamma_p}\times\boldsymbol{n}_p + \boldsymbol{r}_{pq}\times\frac{\partial\boldsymbol{n}_p}{\partial\gamma_p}\right) - \|\boldsymbol{r}_{pq}\|^{-1}\left(\boldsymbol{r}_{pq}\cdot\frac{\partial\boldsymbol{r}_{pq}}{\partial\gamma_p}\right)(\boldsymbol{r}_{pq}\times\boldsymbol{n}_p)}{\|\boldsymbol{r}_{pq}\|^2} \\
&= \frac{\|\boldsymbol{r}_{pq}\|\left(\boldsymbol{v}_{pq}^{\gamma}\times\boldsymbol{n}_p + \boldsymbol{r}_{pq}\times\frac{\partial\boldsymbol{n}_p}{\partial\gamma_p}\right) - (\boldsymbol{r}_{pq}\cdot\boldsymbol{v}_{pq}^{\gamma})\hat{\boldsymbol{t}}_{pq}}{\|\boldsymbol{r}_{pq}\|^2},
\end{aligned}$$

where $\boldsymbol{r}_{pq}$ and $\boldsymbol{v}_{pq}^{\gamma}$ are computed as per Eqs. (6.23) and (6.28), respectively.

The second-order partial derivatives of $Q(\boldsymbol{\gamma})$ with respect to θ_p and ϕ_p follow straightforwardly from the expression for $\partial Q/\partial\gamma_p$ (see Eq. (6.26)), and those of $\hat{\boldsymbol{t}}_{pq}$ (that feature in the second-order partial derivatives of $Q(\boldsymbol{\gamma})$) are derived from Eq. (6.27) as

$$\begin{aligned}
\frac{\partial^2\hat{\boldsymbol{t}}_{pq}}{\partial\gamma_p^2} &= \frac{1}{\|\boldsymbol{r}\|^4}\Big\{\|\boldsymbol{r}\|^2\Big[\,\|\boldsymbol{r}\|^{-1}\big(\boldsymbol{r}\cdot\boldsymbol{v}^{\gamma}\big)\big(\boldsymbol{r}\times\boldsymbol{n}_{\gamma}+\boldsymbol{v}^{\gamma}\times\boldsymbol{n}\big)+\\
&\quad\|\boldsymbol{r}\|\big(\boldsymbol{r}\times\boldsymbol{n}_{\gamma\gamma}+\tfrac{\partial\boldsymbol{r}}{\partial\gamma}\times\boldsymbol{n}_{\gamma}+\boldsymbol{v}^{\gamma}\times\boldsymbol{n}_{\gamma}+\tfrac{\partial\boldsymbol{r}^{\gamma}}{\partial\gamma}\times\boldsymbol{n}\big)\;-\Big(\big(\boldsymbol{r}\cdot\tfrac{\partial\boldsymbol{v}^{\gamma}}{\partial\gamma}+\tfrac{\partial\boldsymbol{r}}{\partial\gamma}\cdot\boldsymbol{v}^{\gamma}\big)\hat{\boldsymbol{t}}+\big(\boldsymbol{r}\cdot\boldsymbol{v}^{\gamma}\big)\hat{\boldsymbol{t}}_{\gamma}\Big)\Big]-\\
&\qquad 2\big(\boldsymbol{r}\cdot\boldsymbol{v}^{\gamma}\big)\Big(\,\|\boldsymbol{r}\|\,\big(\boldsymbol{r}\times\boldsymbol{n}_{\gamma}+\boldsymbol{v}^{\gamma}\times\boldsymbol{n}\big)-\big(\boldsymbol{r}\cdot\boldsymbol{v}^{\gamma}\big)\hat{\boldsymbol{t}}\Big)\Big\}\\
&= \frac{1}{\|\boldsymbol{r}\|^4}\Big\{\|\boldsymbol{r}\|^3\big(\boldsymbol{r}\times\boldsymbol{n}_{\gamma\gamma}+2\boldsymbol{v}^{\gamma}\times\boldsymbol{n}_{\gamma}+\boldsymbol{w}^{\gamma}\times\boldsymbol{n}\big)-\\
&\quad\|\boldsymbol{r}\|^2\Big(\big(\boldsymbol{r}\cdot\boldsymbol{w}^{\gamma}+\|\boldsymbol{v}^{\gamma}\|^2\big)\hat{\boldsymbol{t}}+\big(\boldsymbol{r}\cdot\boldsymbol{v}^{\gamma}\big)\hat{\boldsymbol{t}}_{\gamma}\Big)\;+\big(\boldsymbol{r}\cdot\boldsymbol{v}^{\gamma}\big)\Big(2\big(\boldsymbol{r}\cdot\boldsymbol{v}^{\gamma}\big)\hat{\boldsymbol{t}}-\|\boldsymbol{r}\|\big(\boldsymbol{r}\times\boldsymbol{n}_{\gamma}+\boldsymbol{v}^{\gamma}\times\boldsymbol{n}\big)\Big)\Big\}\\
&= \frac{1}{\|\boldsymbol{r}\|^2}\Big[\|\boldsymbol{r}\|\big(\boldsymbol{r}\times\boldsymbol{n}_{\gamma\gamma}+2\boldsymbol{v}^{\gamma}\times\boldsymbol{n}_{\gamma}+\boldsymbol{w}^{\gamma}\times\boldsymbol{n}\big)-\big(\boldsymbol{r}\cdot\boldsymbol{w}^{\gamma}+\|\boldsymbol{v}^{\gamma}\|^2\big)\hat{\boldsymbol{t}}-\big(\boldsymbol{r}\cdot\boldsymbol{v}^{\gamma}\big)\hat{\boldsymbol{t}}_{\gamma}\Big]+\\
&\qquad\frac{\boldsymbol{r}\cdot\boldsymbol{v}^{\gamma}}{\|\boldsymbol{r}\|^4}\Big[2\big(\boldsymbol{r}\cdot\boldsymbol{v}^{\gamma}\big)\hat{\boldsymbol{t}}-\|\boldsymbol{r}\|\big(\boldsymbol{r}\times\boldsymbol{n}_{\gamma}+\boldsymbol{v}^{\gamma}\times\boldsymbol{n}\big)\Big],
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2\hat{\boldsymbol{t}}_{pq}}{\partial\theta_p\partial\phi_p} &= \frac{\partial^2\hat{\boldsymbol{t}}_{pq}}{\partial\phi_p\partial\theta_p} =\\
&\frac{1}{\|\boldsymbol{r}\|^4}\Big\{\|\boldsymbol{r}\|^2\Big[\,\|\boldsymbol{r}\|^{-1}\big(\boldsymbol{r}\cdot\boldsymbol{v}^{\phi}\big)\big(\boldsymbol{r}\times\boldsymbol{n}_{\theta}+\boldsymbol{v}^{\theta}\times\boldsymbol{n}\big)+\\
&\|\boldsymbol{r}\|\big(\boldsymbol{r}\times\boldsymbol{n}_{\theta\phi}+\boldsymbol{v}^{\theta}\times\boldsymbol{n}_{\phi}+\boldsymbol{v}^{\phi}\times\boldsymbol{n}_{\theta}+(\boldsymbol{n}_{\theta\phi}\times\boldsymbol{d})\times\boldsymbol{n}\big)-\Big(\big((\boldsymbol{n}_{\theta\phi}\times\boldsymbol{d})\cdot\boldsymbol{r}+\boldsymbol{v}^{\theta}\cdot\boldsymbol{v}^{\phi}\big)\hat{\boldsymbol{t}}+\big(\boldsymbol{r}\cdot\boldsymbol{v}^{\theta}\big)\hat{\boldsymbol{t}}_{\phi}\Big)\Big]-\\
&\qquad 2\big(\boldsymbol{r}\cdot\boldsymbol{v}^{\phi}\big)\big(\|\boldsymbol{r}\|(\boldsymbol{r}\times\boldsymbol{n}_{\theta}+\boldsymbol{v}^{\theta}\times\boldsymbol{n})-(\boldsymbol{r}\cdot\boldsymbol{v}^{\theta})\hat{\boldsymbol{t}}\big)\Big\}\\
&= \frac{1}{\|\boldsymbol{r}\|^4}\Big\{\|\boldsymbol{r}\|^3\Big[\boldsymbol{r}\times\boldsymbol{n}_{\theta\phi}+\boldsymbol{v}^{\theta}\times\boldsymbol{n}_{\phi}+\boldsymbol{v}^{\phi}\times\boldsymbol{n}_{\theta}+\big(\boldsymbol{n}_{\theta\phi}\times\boldsymbol{d}\big)\times\boldsymbol{n}\Big]-\\
&\|\boldsymbol{r}\|^2\Big[\Big(\big(\boldsymbol{n}_{\theta\phi}\times\boldsymbol{d}\big)\cdot\boldsymbol{r}+\boldsymbol{v}^{\theta}\cdot\boldsymbol{v}^{\phi}\Big)\hat{\boldsymbol{t}}+\big(\boldsymbol{r}\cdot\boldsymbol{v}^{\theta}\big)\hat{\boldsymbol{t}}_{\phi}\Big]+\big(\boldsymbol{r}\cdot\boldsymbol{v}^{\phi}\big)\Big[2\big(\boldsymbol{r}\cdot\boldsymbol{v}^{\theta}\big)\hat{\boldsymbol{t}}-\|\boldsymbol{r}\|\big(\boldsymbol{r}\times\boldsymbol{n}_{\theta}+\boldsymbol{v}^{\theta}\times\boldsymbol{n}\big)\Big]\Big\}\\
&= \frac{1}{\|\boldsymbol{r}\|^2}\Big[\|\boldsymbol{r}\|\big(\boldsymbol{r}\times\boldsymbol{n}_{\theta\phi}+\boldsymbol{v}^{\theta}\times\boldsymbol{n}_{\phi}+\boldsymbol{v}^{\phi}\times\boldsymbol{n}_{\theta}+(\boldsymbol{n}_{\theta\phi}\times\boldsymbol{d})\times\boldsymbol{n}\big)-\\
&\qquad\big((\boldsymbol{n}_{\theta\phi}\times\boldsymbol{d})\cdot\boldsymbol{r}+\boldsymbol{v}^{\theta}\cdot\boldsymbol{v}^{\phi}\big)\hat{\boldsymbol{t}}-\big(\boldsymbol{r}\cdot\boldsymbol{v}^{\theta}\big)\hat{\boldsymbol{t}}_{\phi}\Big]+\frac{\boldsymbol{r}\cdot\boldsymbol{v}^{\phi}}{\|\boldsymbol{r}\|^4}\Big[2\big(\boldsymbol{r}\cdot\boldsymbol{v}^{\theta}\big)\hat{\boldsymbol{t}}-\|\boldsymbol{r}\|\big(\boldsymbol{r}\times\boldsymbol{n}_{\theta}+\boldsymbol{v}^{\theta}\times\boldsymbol{n}\big)\Big].
\end{aligned}$$

Appendix D

Fourth-order Polynomial Surface Fitting Derivations

The double integrals, $\int_{\Omega_p} \phi \, d\mathcal{A}$ for each $\phi \in \{\xi^r \eta^s |\ r, s \in [0, n],\ r+s \leq n\}$, in Section 7.2, are evaluated using Green's theorem, which states that

$$\iint_{\Omega_p} \left(\frac{\partial Q}{\partial \xi} - \frac{\partial P}{\partial \eta} \right) d\mathcal{A} = \oint_{\partial \Omega_p} P \, d\xi + Q \, d\eta, \tag{D.1}$$

where $\partial\Omega_p$ is the boundary of the arbitrary polygonal integration domain Ω_p, and $d\mathcal{A} \equiv d\xi d\eta$. All the integrals are first cast into the form of the right-hand side of Eq. (D.1), and are then evaluated by parametrising each line segment that constitutes the boundary $\partial\Omega_p$. This chapter derives the s_{pi} terms in Section 7.2 for $i > 6$, that are required for polynomial fits of order $n \geq 4$. The remaining terms, for $i \leq 6$, are stated in Jibben *et al.* [29]. By defining $\Delta\xi_v = (\xi_{v+1} - \xi_v)$, $\Delta\eta_v = (\eta_{v+1} - \eta_v)$, and $N_{v,p}$ as the number of vertices of PLIC facet p, whose vertices are ordered counter-clockwise about its normal, the required integrals are evaluated as shown below. The subscript p applies to all terms within the summation operators and is therefore suppressed for brevity.

$$\begin{aligned}
\int_{\Omega_p} \xi^3 \, d\mathcal{A} &= \oint_{\partial\Omega_p} 0 \, d\xi + \frac{\xi^4}{4} \, d\eta \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{4} \int_0^1 (\xi_v + \Delta\xi_v t)^4 \, dt \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{4} \int_0^1 \left(\xi_v^4 + 4\xi_v^3 \Delta\xi_r t + 6\xi_v^2 \Delta\xi_v^2 t^2 + 4\xi_v \Delta\xi_v^3 t^3 + \Delta\xi_v^4 t^4 \right) dt \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{20} \left(5\xi_v^4 + 10\xi_v^3 \Delta\xi_v + 10\xi_v^2 \Delta\xi_v^2 + 5\xi_v \Delta\xi_v^3 + \Delta\xi_v^4 \right) \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{20} \frac{\left((\xi_v + \Delta\xi_v)^5 - \xi_v^5 \right)}{\Delta\xi_v} \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{20} \frac{\left(\xi_{v+1}^5 - \xi_v^5 \right)}{\Delta\xi_v} \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{80} \left(2\xi_{v+1}^2 + (1 - \sqrt{5}) \xi_v \xi_{v+1} + 2\xi_v^2 \right) \left(2\xi_{v+1}^2 + (1 + \sqrt{5}) \xi_v \xi_{v+1} + 2\xi_v^2 \right).
\end{aligned}$$

Similarly,

$$\begin{aligned}\int_{\Omega_p} \eta^3 \, d\mathcal{A} &= \oint_{\partial\Omega_p} \frac{-\eta^4}{4} \, d\xi + 0 \, d\eta \\ &= \sum_{v=1}^{N_{v,p}} \frac{-\Delta\xi_v}{80} \Big(2\eta_{v+1}^2 + (1-\sqrt{5})\eta_v\eta_{v+1} + 2\eta_v^2\Big)\Big(2\eta_{v+1}^2 + (1+\sqrt{5})\eta_v\eta_{v+1} + 2\eta_v^2\Big) .\end{aligned}$$

$$\begin{aligned}\int_{\Omega_p} \xi^2\eta \, d\mathcal{A} &= \oint_{\partial\Omega_p} 0 \, d\xi + \frac{\xi^3\eta}{3} \, d\eta \\ &= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{3} \int_0^1 (\xi_v + \Delta\xi_v t)^3 \, (\eta_v + \Delta\eta_v t) \, dt \\ &= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{3} \int_0^1 \left(\xi_v^3 + 3\xi_v^2\Delta\xi_r t + 3\xi_v\Delta\xi_v^2 t^2 + \Delta\xi_v^3 t^3\right) (\eta_v + \Delta\eta_v t) \; dt \\ &= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{3} \int_0^1 \Big(\xi_v^3\eta_v + \xi_v^2\big(\xi_v\Delta\eta_v + 3\eta_v\Delta\xi_v\big)t + 3\xi_v\Delta\xi_v\big(\xi_v\Delta\eta_v + \eta_v\Delta\xi_v\big)t^2 + \\ &\qquad \Delta\xi_v^2\big(3\xi_v\Delta\eta_v + \eta_v\Delta\xi_v\big)t^3 + \Delta\xi_v^3\Delta\eta_r t^4\Big) \, dt \\ &= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{60} \Big(10\xi_v^3\big(\eta_v + \eta_{v+1}\big) + 10\xi_v^2\Delta\xi_v\big(\eta_v + 2\eta_{v+1}\big) + 5\xi_v\Delta\xi_v^2\big(\eta_v + 3\eta_{v+1}\big) + \\ &\qquad \Delta\xi_v^3\big(\eta_v + 4\eta_{v+1}\big)\Big).\end{aligned}$$

Similarly,

$$\begin{aligned}\int_{\Omega_p} \xi\eta^2 \, d\mathcal{A} &= \oint_{\partial\Omega_p} \frac{-\xi\eta^3}{3} \, d\xi + 0 \, d\eta \\ &= \sum_{v=1}^{N_{v,p}} \frac{-\Delta\xi_v}{60} \Big(10\eta_v^3\big(\xi_v + \xi_{v+1}\big) + 10\eta_v^2\Delta\eta_v\big(\xi_v + 2\xi_{v+1}\big) + 5\eta_v\Delta\eta_v^2\big(\xi_v + 3\xi_{v+1}\big) + \\ &\qquad \Delta\eta_v^3\big(\xi_v + 4\xi_{v+1}\big)\Big).\end{aligned}$$

$$
\begin{aligned}
\int_{\Omega_p} \xi^4 \, d\mathcal{A} &= \oint_{\partial\Omega_p} 0 \, d\xi + \frac{\xi^5}{5} \, d\eta \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{5} \int_0^1 (\xi_v + \Delta\xi_v t)^5 \, dt \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{5} \int_0^1 \left(\xi_v^5 + 5\xi_v^4\Delta\xi_r t + 10\xi_v^3\Delta\xi_v^2 t^2 + 10\xi_v^2\Delta\xi_v^3 t^3 + 5\xi_v\Delta\xi_v^4 t^4 + \Delta\xi_v^5 t^5\right) \, dt \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{30} \left(6\xi_v^5 + 15\xi_v^4\Delta\xi_v + 20\xi_v^3\Delta\xi_v^2 + 15\xi_v^2\Delta\xi_v^3 + 6\xi_v\Delta\xi_v^4 + \Delta\xi_v^5\right) \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{30} \frac{\left((\xi_v + \Delta\xi_v)^6 - \xi_v^6\right)}{\Delta\xi_v} \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{30} \frac{(\xi_{v+1}^6 - \xi_v^6)}{\Delta\xi_v} \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{30} (\xi_{v+1} + \xi_v) \left(\xi_{v+1}^2 + \xi_v\xi_{v+1} + \xi_v^2\right) \left(\xi_{v+1}^2 - \xi_v\xi_{v+1} + \xi_v^2\right).
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
\int_{\Omega_p} \eta^4 \, d\mathcal{A} &= \oint_{\partial\Omega_p} \frac{-\eta^5}{5} \, d\xi + 0 \, d\eta \\
&= \sum_{v=1}^{N_{v,p}} \frac{-\Delta\xi_v}{30} (\eta_{v+1} + \eta_v) \left(\eta_{v+1}^2 + \eta_v\eta_{v+1} + \eta_v^2\right) \left(\eta_{v+1}^2 - \eta_v\eta_{v+1} + \eta_v^2\right).
\end{aligned}
$$

$$
\begin{aligned}
&\int_{\Omega_p} \xi^2\eta^2 \, d\mathcal{A} = \oint_{\partial\Omega_p} 0 \, d\xi + \frac{\xi^3\eta^2}{3} \, d\eta \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{3} \int_0^1 (\xi_v + \Delta\xi_v t)^3 (\eta_v + \Delta\eta_v t)^2 \, dt \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{3} \int_0^1 \left(\xi_v^3 + 3\xi_v^2\Delta\xi_v t + 3\xi_v\Delta\xi_v^2 t^2 + \Delta\xi_v^3 t^3\right) \left(\eta_v^2 + 2\eta_v\Delta\eta_v + \Delta\eta_v^2\right) dt \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{3} \int_0^1 \Big(\xi_v^3\eta_v^2 + \left(3\xi_v^2\eta_v^2\Delta\xi_v + 2\xi_v^3\eta_v\Delta\eta_v\right)t + \left(\xi_v^3\Delta\eta_v^2 + 6\xi_v^2\eta_v\Delta\xi_v\Delta\eta_v + 3\xi_v\eta_v^2\Delta\xi_v^2\right)t^2 + \\
&\quad \left(3\xi_v^2\Delta\xi_v\Delta\eta_v^2 + 6\xi_v\eta_v\Delta\xi_v^2\Delta\eta_v + \eta_v^2\Delta\xi_v^3\right)t^3 + \left(3\xi_v\Delta\xi_v^2\Delta\eta_v^2 + 2\eta_v\Delta\xi_v^3\Delta\eta_v\right)t^4 + \Delta\xi_v^3\Delta\eta_v^2 t^5\Big) dt \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{180} \Big(20\xi_v^3\left(\eta_v^2 + \eta_v\eta_{v+1} + \eta_{v+1}^2\right) + 15\xi_v^2\Delta\xi_v\left(\eta_v^2 + 2\eta_v\eta_{v+1} + 3\eta_{v+1}^2\right) + \\
&\qquad 6\xi_v\Delta\xi_v^2\left(\eta_v^2 + 3\eta_v\eta_{v+1} + 6\eta_{v+1}^2\right) + \Delta\xi_v^3\left(\eta_v^2 + 4\eta_v\eta_{v+1} + 10\eta_{v+1}^2\right)\Big).
\end{aligned}
$$

$$
\begin{aligned}
\int_{\Omega_p} \xi^3 \eta \, d\mathcal{A} &= \oint_{\partial\Omega_p} 0 \, d\xi + \frac{\xi^4 \eta}{4} \, d\eta \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{4} \int_0^1 (\xi_v + \Delta\xi_v t)^4 \, (\eta_v + \Delta\eta_v t) \, dt \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{4} \int_0^1 \left(\xi_v^4 + 4\xi_v^3\Delta\xi_r t + 6\xi_v^2\Delta\xi_v^2 t^2 + 4\xi_v\Delta\xi_v^3 t^3 + \Delta\xi_v^4 t^4\right) (\eta_v + \Delta\eta_v t) \, dt \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{4} \int_0^1 \Big(\xi_v^4\eta_v + \xi_v^3\big(\xi_v\Delta\eta_v + 4\eta_v\Delta\xi_v\big)t + 2\xi_v^2\Delta\xi_v\big(2\xi_v\Delta\eta_v + 3\eta_v\Delta\xi_v\big)t^2 + \\
&\qquad 2\xi_v\Delta\xi_v^2\big(3\xi_v\Delta\eta_v + 2\eta_v\Delta\xi_v\big)t^3 + \Delta\xi_v^3\big(4\xi_v\Delta\eta_v + \eta_v\Delta\xi_v\big)t^4 + \Delta\xi_v^4\Delta\eta_r t^5\Big) dt \\
&= \sum_{v=1}^{N_{v,p}} \frac{\Delta\eta_v}{120} \Big(15\xi_v^4\big(\eta_v + \eta_{v+1}\big) + 20\xi_v^3\Delta\xi_v\big(\eta_v + 2\eta_{v+1}\big) + 15\xi_v^2\Delta\xi_v^2\big(\eta_v + 3\eta_{v+1}\big) + \\
&\qquad 6\xi_v\Delta\xi_v^3\big(\eta_v + 4\eta_{v+1}\big) + \Delta\xi_v^4\big(\eta_v + 5\eta_{v+1}\big)\Big).
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
\int_{\Omega_p} \xi\eta^3 \, d\mathcal{A} &= \oint_{\partial\Omega_p} \frac{-\xi\eta^4}{4} \, d\xi + 0 \, d\eta \\
&= \sum_{v=1}^{N_{v,p}} \frac{-\Delta\xi_v}{120} \Big(15\eta_v^4\big(\xi_v + \xi_{v+1}\big) + 20\eta_v^3\Delta\eta_v\big(\xi_v + 2\xi_{v+1}\big) + 15\eta_v^2\Delta\eta_v^2\big(\xi_v + 3\xi_{v+1}\big) + \\
&\qquad 6\eta_v\Delta\eta_v^3\big(\xi_v + 4\xi_{v+1}\big) + \Delta\eta_v^4\big(\xi_v + 5\xi_{v+1}\big)\Big).
\end{aligned}
$$