On estimating the interface normal and curvature in PLIC-VOF approach for 3D arbitrary meshes

Ahmad Amani[a,*], Jordi Muela[a], Eugenio Schillaci[b], Jesús Castro[a]

[a] *Heat and Mass Transfer Technological Center (CTTC), Universitat Politécnica de Catalunya-Barcelona Tech((UPC), ESEIAAT, Colom 11, 08222 Terrassa (Barcelona), Spain*
[b] *Termo Fluids S.L. Carrer Magi Colet 8, Sabadell, Barcelona, Spain http://www.termofluids.com*

Abstract

Volume-of-Fluid (VOF) method with its Piecewise-Linear-Interface-Calculation (PLIC) reconstruction algorithm is one of the most popular approaches in numerical simulation of interfacial flows with a wide range of applications in different areas. In an effort to evaluate the similarity of the PLIC-generated planes in comparison with the exact interface, a point-cloud, based on the polygon centers of PLIC planes is extracted, which later is used to form a triangular grid that represents the estimated interface. The main objective of this paper is to evaluate the interface geometrical properties based on the extracted triangular grid of the interface. The methods presented in this paper, characterized by a higher spatially convergence ratio, are compared with the commonly used methods. The proposed methods are tested for two 3D general test cases, where an evident improvement is seen in calculation accuracy and spatial convergence of the errors of interface normal vector and curvature.

Keywords: Multiphase flows, Volume-of-Fluid (VOF), PLIC reconstruction approach, interface normal, interface curvature

*Corresponding author
Email addresses: ahmad@cttc.upc.edu.com (Ahmad Amani), jordim@cttc.upc.edu (Jordi Muela), eugenio@cttc.upc.edu (Eugenio Schillaci), jesus@cttc.upc.edu (Jesús Castro)

Introduction

The numerical solution of interfacial flows is of huge interest for a large variety of applications, i.e. engineering, environmental, and geophysical fields,[1] just to name a few. Different approaches have been developed by the researchers to numerically resolve the moving interface boundaries in multiphase flows. These approaches could be characterized in two groups: (i) Interface tracking methods, where the interface is treated as the boundary between two sub-domains in a moving grid arrangement[2] or as the trajectories of massless particles[3] as it is in Front Tracking (FT) approach. (ii) Interface capturing methods, where scalar fields are used to define the location of different fluids embedded in a fixed grid, e.g. Volume-of-Fluid (VOF),[4,5] Level-Set (LS),[6,7,8] and Coupled Level-Set/Volume-of- Fluid approaches.[9]

Although there might be similarities in these methods, their numerical implementation may differ greatly, resulting in different advantages and disadvantages, thus making them appropriate for the solution of different multiphase flow problems. For example in the Front Tracking method, the interface is being tracked explicitly by a separate Lagrangian grid, which leads to extremely accurate results in some problems, e.g. dense bubbly flows[10] where coalescence of bubbles can be explicitly controlled, but since dynamic re-meshing of the Lagrangian interface mesh is required, it would be rather complex to implement in general multiphase flow problems where one can expect large topological changes.

On the other hand, methods of the category (ii) perform better in resolving general topological changes, appearing in many applications, i.e. breakup and coalescence of bubbles/droplets.[11,12,13] LS method describes the interface as the zero contours of an auxiliary signed distance function, while in VOF method, the interface is captured implicitly using a color function scalar field, representing the volume fraction of a phase inside each cell of the discretized domain. It can be concluded that the main difference between these

two methods is that VOF methods resolve the interface motion by using discontinuous color function while LS methods describe the interface using a continuous auxiliary function. Consequently, compared to VOF methods, LS methods are more accurate in calculating geometric quantities such as curvature and normal vectors. However in standard LS methods, the discrete solution of the advection equation of LS function alters the distance-function nature of the auxiliary function, thus there is a need for a re-initialization step which contrary to VOF methods, leads to loss or gain of the mass in standard LS methods.

The main focus of this work is to analyse and improve the calculation of geometric quantities, curvature and normal vectors, using available information of discontinuous color function in VOF approach.

Navier-Stokes equations are used to describe the conservation of mass and momentum of two incompressible immiscible Newtonian fluids on a spacial domain Ω with boundary $\partial\Omega$ as following:[12]

$$\frac{\partial}{\partial t}(\rho\mathbf{u}) + \nabla\cdot(\rho\mathbf{u}\mathbf{u}) = \nabla\cdot\mathbf{S} + \rho\mathbf{g} + \sigma\kappa\mathbf{n}\delta_\Gamma \text{ in } \Omega \quad (1)$$

$$\mathbf{S} = -p\mathbf{I} + \mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^T) \quad (2)$$

$$\nabla\cdot\mathbf{u} = 0 \text{ in } \Omega \quad (3)$$

where ρ and μ are density and dynamic viscosity of the fluids, $\mathbf{u}$ is the velocity field, $\mathbf{S}$ is the stress tensor, p pressure field, $\mathbf{g}$ gravitational acceleration and δ_Γ is the Dirac delta function concentrated at the interface (Γ). In this formulation, $\mathbf{n}$ is the normal unit vector outward to interface, σ is the interface tension coefficient, and $\kappa = \kappa_1 + \kappa_2$ is the interface curvature, with κ_1 and κ_2 as principal curvatures of the interface.

The correct calculation of surface tension term in equation 1 is of vital importance for the accuracy of the momentum solver. The accuracy of this term depends on extracting the geometrical properties of interface,

i.e. interface normal vector $\mathbf{n}$ and curvature κ. In the VOF method, in addition to momentum equation, there is need for an advection equation to resolve the temporal evolution of the color function by the known velocity field using:

$$\frac{\partial \alpha_k}{\partial t} + \nabla \cdot (\alpha_k \mathbf{u}) = 0 \tag{4}$$

where α_k is the volume fraction of fluid k, and $\mathbf{u}$ is the known velocity vector in the associated grid cell. In the context of finite-volume methods, the application of the divergence theorem and the spatial discretization of the latter equation over a cell with volume ∂v and face set F with area vectors of $d\mathbf{A}_f$ leads to:

$$\int_{\partial v} \frac{\partial \alpha_k}{\partial t} + \sum_{f_i \in F} (\alpha_k \mathbf{u})_{f_i} \cdot d\mathbf{A}_{f_i} = 0 \tag{5}$$

where the first term concerns the temporal evolution of the interface and the second term is the total volumetric flux of fluid k across the faces of the associated grid cell in a timestep based on the selected temporal scheme. While different approaches could be used in temporal discretization of the first term, the main difficulties are associated with correctly calculating the second term.

There are two methods to approach the solution of this equation, (i) using standard numerical convection schemes, (ii) using geometrically reconstructed interface calculated based on volume fraction values at interface cells. The first and second approaches are usually known as Algebraic and Geometric VOF methods, respectively. In the first approach, the sharpness of the interface is not preserved in the advection process, resulting in the smearing of the interface. There are anti-diffusion or artificial steepening approaches, which can be applied by high-resolution and compressive schemes[14,15,16] which reduce, but not completely elimi-

nate this issue. As a result the solution quality is usually inferior compared to geometric VOF methods.[16] In the geometrical approach, the second term in left hand side of equation 5 considered as the volumetric flow of fluid k across face f is evaluated using geometrical tools starting by first reconstructing the interface at the associated grid cell and then using this interface to calculate the total volumetric flux of fluid k across the faces of each interface cell. In the simplest approach to reconstruct the interface known as (SLIC) scheme,[17,5] the interface is considered to be a line (2D) or plane (3D) parallel to coordinate axis which divides the cell into two sub-cells associated with volumes α_k and $(1 - \alpha_k)$. However, the accuracy of this method is at question, as it provides only a first-order accurate representation of the interface. Eliminating the constraints that the interface line (2D) or plane (3D) has to be parallel to the coordinate axis, results in development of different approaches categorized as piecewise linear interface calculation (PLIC) methods where the interface can have an arbitrary orientation in each cell. Although there have been studies to reconstruct the interface not as a line (2D) or plane (3D) but as parabolic,[18] or spline[19] functions, the application of these methods on arbitrary unstructured meshes raises difficulties in implementation as well as increased computational cost, leaving PLIC-VOF methods the most common geometric VOF approaches. In this method, regardless of grid cell type, i.e. structured or unstructured, the interface in each cell is defined as:

$$\mathbf{x} \cdot \mathbf{n} - d = 0 \tag{6}$$

where $\mathbf{n}$ is the unit-normal vector pointing outward with respect to the phase α_k, $\mathbf{x}$, position vector of a point in the interface and d is the signed distance from the origin to the plane. As the main constraint is

the local volume conservation of α_k in the associated grid cell, thus, the value of d could be found in which, for a given normal vector $\mathbf{n}$, enforces the α_k volume fraction in the associated grid cell. In other words, the interface truncates the grid cell in a way that the volume ratio of the two resulting polyhedrons is equal to α_k. Thus, for a given normal vector $\mathbf{n}$, in each grid cell, the value of d could be accurately calculated using root-finding algorithms, e.g. Brent's method.[20] As a result, two functions are required in a PLIC-VOF method. One named volume truncation, to calculate the volume fractions of a cell truncated by a fixed plane (given $\mathbf{n}$ and d). This function is usually denoted by β. And another named volume enforcement, to enforce a volume fraction α_k by a movable plane, i.e. to determine d for a given $\mathbf{n}$ and α_k. There have been developments to perform these tasks using analytical and geometrical tools for rectangular and hexahedral,[21] triangular and tetrahedral,[22] convex,[23] and non-convex[24] polyhedral elements.

Thus the correct estimation of the interface normal in arbitrary grid cells, i.e. structured and unstructured meshes, is the key to any PLIC algorithm and has been an active research field. In the next sub-section, different advancements on this topic in literature are evaluated briefly.

Interface Normal Estimation

Various methods have been proposed for structured and unstructured grid cells, including Parker–Youngs (PY) method,[25] the least-square fit,[26] the least-squares volume-of-fluid interface reconstruction algorithm (LVIRA) and its efficient variation for structured grids,[27] the geometric least-squares (GLS) method for unstructured grids,[28] Height Function (HF) interface reconstruction method,[4] and Mosso–Swartz algorithm[29] along with its modified variations.[30,31] Since the objective of this study is to improve the accuracy of PLIC-VOF method in calculating interface geometrical properties, each of these approaches will be briefly discussed. There are other variations of PLIC-VOF method that have been presented recently which are in

one way or another relying on the fundamental ideas of PLIC-VOF method. For example, in Moment-of-Fluid (MoF) algorithm[30] a variant of Swartz method[32] is used to estimate the interface normal where, in addition to the color function, centroids of the interface are advected as well. This method increases the spatial-convergence of error in calculating interface normals on 3D arbitrary/unstructured meshes, however the additional advection and minimization steps of this approach, increase implementation complexity and associated computational cost.[33] The focus of the current work is to compare/improve the interface reconstruction based on classical PLIC-VOF method, without the need for additional advection, minimization, or formulations. Thus more complicated ideas like MoF, EHF, etc, are not discussed in details. An overview of earlier interface reconstruction problems,[5] along with a more recent review of reconstruction algorithms on structured Cartesian grid cells[27,26] could be found in literature.

The PLIC algorithm, known as Parker–Youngs (PY) method, was introduced[25] based on a simple normalized gradient of the volume fraction in the grid cell as $\mathbf{n} = \nabla\alpha_k/|\nabla\alpha_k|$. It has been reported that using a vertex-connectivity least-squares method to calculate the term $\nabla\alpha_k$, will increase the accuracy in case of 3-D unstructured meshes.[34] Despite the simplicity and its computationally inexpensive nature, this method is at most first-order accurate. A least-square fit procedure was presented[26] which showed better results compared to PY method, and later was extended to unstructured grid cells.[35] However, similar to PY method, this approach lacks the second-order accuracy, as well. In the LVIRA method, as depicted in figure 1, the normal vector in a cell is calculated such that if the linear interface is extended to the neighbourhood cells of associated cell, the discrepancy in volume truncation ratio resulted from this linear interface (β_k)

and cell's phase volume fraction ratio (α_k) is minimized in a least-square approach:

$$e(\mathbf{n}) = \sqrt{\sum_{F \in nbCells} (\alpha_{kF} - \beta_{kF})^2} \tag{7}$$

where nbCells refers to the set of all the neighbouring cells of associated grid cell. Thus for each grid cell, in an iterative procedure, costly geometrical steps of volume enforcement and volume truncation must be done.[27]

Fig. 1 should be placed here

For structured meshes, ELVIRA eliminates the need of iterations by selecting the normal vector $\mathbf{n}$ from a set of candidates obtained from backward, central and forward estimations in each direction. Similar to LVIRA, geometric least-square (GLS) method requires costly geometric iterations within an unstructured framework.[28]

The HF method,[36] is another prevalent second-order approach originally developed for curvature calculations. In structured meshes, first an initial approximation of normal vector $\mathbf{n}$ is calculated using a simpler method. Afterwards, the volume fraction (α_k) is integrated in vertical or horizontal direction, based on which direction is closest to the approximated normal, to estimate a height function denoted by H. The slopes of this function in the other two cartesian directions are used to correct the approximated normal vector $\mathbf{n}$.[36,37] In unstructured meshes with irregular cell arrangements, this integration could be cumbersome, since the area for integration could not be determined appropriately, imposing extra implementation complexities to the algorithm.[38] Figure 2 illustrates the definition of HF using the predicted normal vector on arbitrary unstructured meshes.

Fig. 2 should be placed here

This method was extended[39] to 2D nonuniform rectangular grids; to 2D unstructured rectangular/triangular;[38] and to 3D unstructured meshes.[24] There are different variations of HF methods reported in the literature, including unstructured-HF,[38] **n** aligned HF,[40] and embedded HF (EHF),[24] aiming to advance further upon the existing method. However, a loss in second-order convergence of HF method[24] on finer unstructured meshes was reported.[41]

The formulation of Swartz method[32] was extracted for unstructured meshes[29] . In their approach, PY method is used to calculate an initial estimate of the normal vector **n** in an interface cell p, denoted by $\mathbf{n}_p$, and extract interface plane in a volume enforcement step. Interface center is calculated as the center of a polygon extracted from interface plane-cell intersection, denoted by $\mathbf{x}_p$. For cell p a set of estimates of normal vectors based on each neighbouring cell F and their interface polygon center $\mathbf{x}_F$ is calculated such that $\mathbf{n}_p^F$ is perpendicular to the vector $(\mathbf{x}_p - \mathbf{x}_F)$ as:

$$\mathbf{n}_p^F \cdot (\mathbf{x}_p - \mathbf{x}_F) = 0 \tag{8}$$

The modified interface normal $\mathbf{n}_p^m$ is selected such that its least-square error with regard to other $\mathbf{n}_p^F$ values is minimized:

$$e(n) = \sum_{F \in nbCells} \left(\mathbf{n}_p^m - \mathbf{n}_p^F\right)^2 \tag{9}$$

Using the new estimate of interface normal vector **n**, the interface can be reconstructed again to obtain a new set of interface polygon centers **x**, to be used in further improving the interface normal estimates. To

achieve second-order convergence, this process has to repeat four times.[29]

In an effort to eliminate the need for these four iterations, the modified interface normal was calculated as an arithmetic average of all the normal estimates,[30] including the one extracted by PY method:

$$\mathbf{n}_p^m = 0.5 \left(\mathbf{n}_{PY} + \frac{\sum_{F \in nbCells} \mathbf{n}_p^F}{N_{nbCells}} \right) \tag{10}$$

The $\mathbf{n}_{PY}$ normal vector in equation 10 was eliminated[31] and the arithmetic average of estimated normal was changed to sum of these values as long as they angle under 30 degrees with $\mathbf{n}_{PY}$. This 30 degree parameter was extracted based on numerical tests:[31]

$$\mathbf{n}_p^m = \sum_{F \in nbCells} \mathbf{n}_p^F \quad if \quad \angle(\mathbf{n}_p^F, \mathbf{n}_{PY}) < 30^\circ \tag{11}$$

The extracted interface normal then needs to be normalized.

Interface Curvature Estimation

As mentioned in Introduction section, correct calculation of surface tension term in equation 1 is vital for the accuracy of momentum solver. Assuming an exact estimation of normal vector $\mathbf{n}$, the main remaining challenge of this term is calculating the local curvature of the interface κ which is based on second derivative of the volume fraction distribution. However, since volume fraction distribution is a discontinuous and non-differentiable function, an accurate calculation of the second derivative is not easy. There have been efforts reported in literature to tackle this issue:

Continuous surface force model (CSF) was proposed for surface tension computation[42] which converts

the term $\sigma\kappa\mathbf{n}\delta_\Gamma$ in Eq. 1 to a volume force term of $\sigma\kappa(\alpha)\nabla\alpha$. This approach could be interpreted as using a convolution of the volume fraction with a kernel function to make a smooth function of the α distribution. By applying this approach, the explicit tracking of the interface is not necessary, and κ is evaluated by net tensile force acting on the interface, which is expressed as $\kappa = -\nabla \cdot \mathbf{n}$ where $\mathbf{n}$ in this formulation is traditionally replaced by $\nabla\alpha/|\nabla\alpha|$, resulting to $\kappa = -\nabla \cdot (\nabla\alpha/|\nabla\alpha|)$. The CSF method is extensively used by researchers in methods developed on both structured and unstructured meshes.

Another family of methods developed to improve calculation of curvature from discontinuous volume fraction information are reconstructed-distance function (RDF) methods, also referred to as "coupled LS-VOF" methods. In these methods, based on the location of interface extracted from PLIC, a signed-distance function is created for a stencil of cells surrounding the interface, to provide a smooth differentiable field, ϕ, which is used to calculate the curvature usually as $\kappa = -\nabla \cdot \phi$. There are different ways reported to construct the distance function, e.g. distance constructed in a cell based on weighted average of all nearby PLIC-interface cells,[37] or based on single value distance from PLIC-interface cells.[43] There have been successful extension of RDF method on arbitrary unstructured meshes.[44,43]

HF method described in Interface Normal Estimation section has been used to calculate the curvature[37] in addition to normal vectors for 2D uniform structured meshes,[45,46] 2D nonuniform rectangular meshes,[39] unstructured rectangular/triangular meshes[38] and 3D unstructured meshes.[24] Despite the advances in this approach, there are difficulties associated with HF method to calculate curvature. For instance, to improve the convergence ratio of HF method, the **n**-aligned column approach was proposed which improved the accuracy of the method on coarse meshes. However, an extra neighbourhood search step was required to compute the intersections with the mesh and the columns which added to the already high complications of

the method and its computational costs.[40] Besides, relatively larger computational stencil required in HF methods increases the communication complexity and overhead in parallel simulations.

Other approaches to calculate the curvature based on fitting a function to available information from the interface are also reported in the literature. For example, there are reports where the curvature is calculated from a quadratic function that is best-fitted to the volume fraction data,[47] or calculated using a paraboloid function that was best-fitted to the HF data.[48] There are alternative approaches that deviate from classical PLIC method. For instance, in the Parabolic Reconstruction Of Surface Tension method, PROST,[18] an implicit quadratic interface is being fitted to the volume fraction information in structured[18] and 2D unstructured[41] grid cells. As mentioned before, the intersection of a cell with relatively higher-order interfaces imposes complexities in implementations and computational costs. In the next section, the available information in PLIC-VOF method will be analysed. A novel approach for calculating normal vector and curvature of interface based on these available information will be studied. Two test cases are used to examine these approaches in Numerical experiments and discussions section. Finally, concluding remarks are presented in Conclusion section.

Numerical method

Available information in PLIC interfaces

As described in Interface Normal Estimation section, in PLIC-VOF method the interface is estimated by a line (2D) or plane (3D) with the forced constraint that the volume truncation ratio of the resulting polyhedrons β_k is equal to the volume fraction of the cell α_k. Figure 3 illustrates the interface polygons extracted by PLIC-VOF method in a domain with arbitrary unstructured cell arrangements compared

to exact interface. In this figure, LVIRA method is used to extract normal vectors in PLIC approach. Although the PLIC interface conserves the mass of the phases, as can be seen in this figure, it estimates the interface with lines/planes which do not necessarily represent the exact location of the interface, but a rough approximation of it.

Fig. 3 should be placed here

In this paper, the centers of polygons resulting from interface plane-cell intersection, here called interface polygons, are used to form a set of points, herein after called point-cloud, very close to the not-available exact interface. Figure 4 illustrates the point-cloud of the interface polygon centers (Pi) in PLIC-VOF method, compared with the exact surface of the associated analytical function (top: an oblate spheroid, bottom a wave function). The color palette in this figure represents the error $E = (\mathbf{P}_i - \mathbf{P}) \cdot \mathbf{n}_P / h \times 100$, where $\mathbf{P}$ is the point on the spheroid surface closest to $\mathbf{P}_i$, $\mathbf{n}_P$ is the exact normal vector, normal to the spheroid surface at point $\mathbf{P}$, and h=0.2 is characteristic size of grid cell. For each case (spheroid, and wave implicit functions), the method of Lagrange multipliers is used to extract a system of non-linear equations which its solution using an iterative solver have led to obtaining the point $\mathbf{P}$ closest to point $\mathbf{P}_i$ on the associated surface. According to this figure, in the simulation of an oblate spheroid (figure 4 (top)) and the wave function interfaces (figure 4 (bottom)), each with characteristic grid size of h=0.2, the center points of PLIC-VOF interface polygons have at most $0.02h$ error compared with the exact interface. This means the distance of the worst point $\mathbf{P}_i$ to the exact interface is around 2% of the characteristic grid size. The majority of these points have a negative error, meaning the extracted points from PLIC-VOF method of phase k fall inside the phase k.

Fig. 4 should be placed here

Figure 5 illustrates the spatial convergence of first and infinite norms of error, calculated as: $L_1 = |\sum(e_i V_i)|/\sum(V_i)$ and $L_{inf} = max(|e_i|)$ where $e_i = (\mathbf{P}_i - \mathbf{P}) \cdot \mathbf{n}_P$ and V_i is the volume of associated cell i. As can be seen, the interface polygon center points are converging to the exact surface of spheroid, as the mesh is refined. According to this figure, the L_1 and L_{inf} are spatially converging with an order around $\mathcal{O}(2)$. It is also evident that by refining the mesh from h=0.1 to h=0.06, the spatial convergence order of L_{inf} is reduced to almost $\mathcal{O}(0)$, meaning that further mesh refinements are not reducing the L_{inf} anymore, or in other words, mesh refinements do not improve the quality of estimated interface in the region with the worse quality.

Fig. 5 should be placed here

Considering implicit surfaces of $F(x, y, z) = 0$ and $F^*(x, y, z) = 0$ representing the exact, and estimated interfaces, respectively, where the error of estimated interface with regards to the exact interface is spatially converging with an order of m, one can write $F(x, y, z) = F^*(x, y, z) + \mathcal{O}(h^m)$ where here $m = 2$. For the given implicit surface F, the normal vector and curvature of the surface could be extracted as:

$$\mathbf{N} = \nabla F \tag{12}$$

$$\kappa = \frac{\nabla F \ \mathbf{H}(F) \ \nabla F^T - |\nabla F|^2 \ Trace(\mathbf{H}(F))}{|\nabla F|^3} \tag{13}$$

with $\mathbf{H}$(F) as the 3×3 Hessian matrix of F. In this formulation, the unit normal vector is $\mathbf{n} = \mathbf{N}/|\mathbf{N}|$. Hence, for the implicit surface F and its estimated counterpart, the normal vector is given by $\mathbf{N} = \nabla F = (\partial F/\partial x, \partial F/\partial y, \partial F/\partial z) = \nabla F^* + \mathcal{O}(h^{m-1})$. Thus, if no special care is taken into account and if only the local information of estimated surface is used to calculate normal vector $\mathbf{N}$, spatial convergence of $\mathbf{N}$

would be at best one order smaller than F^*. However, it is important to note that based on formulation of equation 12, in order to calculate normal vector $\mathbf{N}$, the estimated interface F^* must be at least one degree differentiable. If not, the error of applying gradient operator (∇) to the non-differentiable function F^* will be added to the already one-order lower estimated normal vector $\mathbf{N}$. This is what one witnesses in PY method, where estimated surface F^* is the discontinuous, non-differentiable color function α_k and normal vector at each point is evaluated using $\mathbf{N} = \nabla F^* = \nabla \alpha_k$. As depicted in figure 6, for a spheroid interface, although the estimated surface F^* is spatially second-order accurate, the added error due to differentiating the discontinuous function F^* causes a spatial convergence error of not $\mathcal{O}(1)$ but almost $\mathcal{O}(0)$ in calculating normal vector $\mathbf{N}$. One the other hand, the LVIRA method does not suffer from this extra reduction of order of spatial convergence as this approach does not require differentiating the discontinuous surface function. Details of the numerical simulations related to the results presented in figure 6 will be discussed in Numerical experiments and discussions section.

Fig. 6 should be placed here

Regarding the curvature calculations, due to the presence of second-order derivatives in its formulation, it is expected that the spatial convergence of κ to be two orders smaller than F^* as $\kappa = \kappa^* + \mathcal{O}(h^{m-2})$. As a result, if no special care is taken into account and if only the local information of the estimated surface is used to calculate curvature, spatial convergence of κ would be two orders smaller than F^*. It is important to note that the mentioned two orders reduction occurs if in the discretized form, the surface function F^* is two degrees differentiable, and there are no difficulties in calculating the first and second order spatial-derivatives. But if, as in its usual form, the curvature is calculated based on the discontinuous color function α , $\kappa = \nabla \cdot \left(\frac{\nabla \alpha}{|\nabla \alpha|} \right)$, then the error of two times spatially differentiating a discontinuous function will be

added to the already two-order lower estimated curvature of $\kappa^* + \mathcal{O}(h^{m-2})$, and one can even witness a negative order of spatial convergence in curvature calculation. Figure 7 illustrates the L_1 and L_{inf} of spatial convergence for surface curvature κ using original formulation method as $\kappa = \nabla \cdot \left(\frac{\nabla \alpha}{|\nabla \alpha|} \right)$ compared with RDF method as $\kappa = \nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right)$ with ϕ as distance function for a spheroid interface. As can be seen in this figure, by generating a continues representation of interface through ϕ and using this distance function to calculate the curvature of the interface, the first-norm of error is showing a convergence ratio between $\mathcal{O}(0)$ and $\mathcal{O}(1)$, almost two times smaller than the spatial convergence error of the interface itself, as represented in figure 5 left. However for this figure, L_1 error of calculating κ by directly using α_k is one order of convergence smaller than what was expected and is negative which could be associated with the two-times differentiating a discontinuous function. Similar pattern can be seen in L_{inf} norm of error as depicted in figure 7 (right).

Fig. 7 should be placed here

In conclusion, if no special care is taken into account, the normal and curvature calculations from a surface that is spatially second-order accurate would be at best first, and zero order accurate, respectively. Different methods in calculating normal vector and curvature e.g. LVIRA, HF etc. are trying to overcome this limit by involving information of neighbouring cells in calculation of these geometrical properties in each cell.

Proposed Normal Estimation method

In this section a normal estimation method is presented where for each cell, the interface normal vector is calculated using the interface point-cloud extracted from PLIC polygon centers. For each point $\mathbf{P}_i$, a set of immediate-adjacent neighbouring points are extracted and sorted in a clockwise (or counter-clockwise) orientation. Using these neighbouring points, a triangular grid could be formed by connecting point $\mathbf{P}_i$ to

two consecutive neighbour point, j and k, as depicted in figure 8.

Fig. 8 should be placed here

For each triangular grid of $\triangleleft jik$, an estimation of unit normal vector could be calculated by vector-producting the edges associated to vertex i as:

$$n_{\triangleleft jik} = \frac{\vec{P_iP_j} \times \vec{P_iP_k}}{|\vec{P_iP_j} \times \vec{P_iP_k}|} \tag{14}$$

This normal vector is a rough estimation of the normal vector at point $\mathbf{P}_i$. Performing this step to all the triangular grids with point $\mathbf{P}_i$ as center piece leads to a set of estimated normal vectors, N_i, for this point. The size of N_i is equal to the number of neighbour points surrounding $\mathbf{P}_i$. This ensures the involvement of information of neighbour cells in calculating the normal vector of each cell. The final value of unit-normal vector n_i at point $\mathbf{P}_i$ is calculated as a weighted average of all the members of N_i, as:

$$n_i = \frac{\sum_{n_j \in N_i} w_j n_j}{\sum w_j} \tag{15}$$

Different weighting parameters were tested for averaging, and the best results were obtained where w_j is the area of associated triangle used in calculating n_j.

Proposed Curvature Estimation

In this section, three different approaches are being used to calculate the interface curvature from the spacially second-order points extracted from PLIC polygon centers as discussed in Available information in PLIC interfaces section. Different methods have been introduced in the literature for calculating the surface

curvature from a given point-cloud which are developed mainly for applications in computer graphics, reverse engineering, medial visualization, etc. In general, methods of calculating curvature from point-cloud data could be categorized in two types:

1. Discrete method,[49] which tries to calculate curvature directly from the point-cloud information by forming grids from data points and estimating fundamental forms per grid.

2. Surface fitting methods[50,51,52] which are based on finding a local approximation of the surface and calculating the curvature based on obtained formulation of this approximation.

In this section we analyse the accuracy of these approaches in calculating the interface curvature from data points extracted by PLIC-VOF method.

Extracting Fundamental Forms

For the interface surface formed by a point set, one can define the two Fundamental Forms, I and II, at each given edge formed by connecting two points, $\mathbf{P}_i$ and $\mathbf{P}_j$, describing the first and second order derivatives of parameterization of surface for that edge, respectively.[49] These derivatives define the extrinsic invariants of the surface, its principal curvatures as needed to calculate the curvature in surface tension term of equation 1. The Basic theory, details and proofs of theorems of fundamental forms of surfaces could be found in literature.[53]

For each given point $\mathbf{P}_i$, one can define m edges, by connecting $\mathbf{P}_i$ to its immediate-adjacent neighbours. For each of these edges, formed by $\mathbf{P}_i$ and $\mathbf{P}_j$, a local coordinate system ($\mathbf{T}_u$,$\mathbf{T}_v$, $\mathbf{n}$) could be defined with two arbitrary (and not necessarily orthogonal to each other) tangent vectors of $\mathbf{T}_u$ and $\mathbf{T}_v$ which are orthogonal to the normal vector $\mathbf{n}$. Coefficients of first fundamental form (I) are the inner product of the

tangent space of a surface in three-dimensions as:

$$I = \begin{pmatrix} A & B \\ B & C \end{pmatrix} \tag{16}$$

where $A = \mathbf{T}_u \cdot \mathbf{T}_u$, $B = \mathbf{T}_u \cdot \mathbf{T}_v$, and $C = \mathbf{T}_v \cdot \mathbf{T}_v$. The second fundamental form, or shape tensor, is calculated from the normal vector and second derivatives of tangent vectors as:

$$II = \begin{pmatrix} D & E \\ E & F \end{pmatrix} \tag{17}$$

where $D = \mathbf{T}_{uu} \cdot \mathbf{n}$, $E = \mathbf{T}_{uv} \cdot \mathbf{n}$, and $F = \mathbf{T}_{vv} \cdot \mathbf{n}$. Using these fundamental forms, one can form the shape operator, usually denoted by W standing for Weingarten, as $W = I^{-1}II$. For the given edge, this 2×2 symmetric matrix contains information on variation of the normalized normal vector in the direction of a tangent vector. The eigenvalues (λ_1 and λ_2) of this operator correspond to the principal curvatures of the surface at the given point. Thus the curvature of the surface used in equation 1 could be calculated as $\kappa = \lambda_1 + \lambda_2$. If selected tangent vectors in local coordinate system are orthogonal, i.e. $T_u \cdot T_v = 0$, the first fundamental form is equal to the Identity matrix, resulting in $W = II$.

By forming W matrix for each edge, one would obtain the change of the normal vector along that edge. However, the resulting system of equations is under-determined, as three unknowns of D, E, and F are more than the available information, i.e. variation of two normal vectors along the edge. That is why there is a need to involve more neighbour vertices in calculating the W matrix. As discussed in Proposed Normal Estimation method section, in the data point extracted, one can connect the neighbour points and form a

set of triangular meshes that represent the surface of the interface. The objective is to find the W matrix at each given point $\mathbf{P}_i$ using associated triangular grid of $\triangleleft jik$ and corresponding normal vectors at each vertex.

For each triangle $\triangleleft jik$, a local coordinate system, i.e. tangent space, can be defined with origin at $\mathbf{P}_i$, as depicted in figure 9, where $\mathbf{T}_u = \frac{\mathbf{P}_j - \mathbf{P}_i}{|\mathbf{P}_j - \mathbf{P}_i|}$, $\mathbf{n} = \frac{\mathbf{T}_u \times (\mathbf{P}_k - \mathbf{P}_i)}{|\mathbf{T}_u \times (\mathbf{P}_k - \mathbf{P}_i)|}$, and $\mathbf{T}_v = \frac{\mathbf{T}_u \times \mathbf{n}}{|\mathbf{T}_u \times \mathbf{n}|}$. By this transformation, one of the dimensions is constant and thus reduced. So the resulting problem could be studied in 2D space instead of 3D with saves in computational cost.

Fig. 9 should be placed here

The edges of this triangle are being transformed to the local coordinate system using transformation matrix of the tangent space. This transformation for point $\mathbf{P}_j$ could be written as:

$$\mathbf{P}_{Lj} = \begin{pmatrix} \mathbf{T}_{u1} & \mathbf{T}_{u2} & \mathbf{T}_{u3} \\ \mathbf{T}_{v1} & \mathbf{T}_{v2} & \mathbf{T}_{v3} \\ \mathbf{n}_1 & \mathbf{n}_2 & \mathbf{n}_3 \end{pmatrix} (\mathbf{P}_j - \mathbf{P}_i) \tag{18}$$

In this formulation, L stands for Local coordinate system. In addition to the locations, already known normal vectors at vertices need to be transformed to the local coordinate as well. However, since the tangent space is orthogonal, the transformation matrix would be the same as the one formulated in equation 18. For each edge, formed by $\mathbf{P}_i$ and $\mathbf{P}_j$, the following under-determined system of equations could be written as:

$$\begin{pmatrix} D & E \\ E & F \end{pmatrix} \begin{pmatrix} (\mathbf{P}_{Lj} - \mathbf{P}_{Li})_u \\ (\mathbf{P}_{Lj} - \mathbf{P}_{Li})_v \end{pmatrix} = \begin{pmatrix} (\mathbf{n}_{Lj} - \mathbf{n}_{Li})_u \\ (\mathbf{n}_{Lj} - \mathbf{n}_{Li})_v \end{pmatrix} \tag{19}$$

By constructing the same system of equations for the all three edges of triangle $\triangleleft jik$, an over-determined system of equation would be formed that can be solved using least-square approach as $\mathbf{X} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{Y}$.

$$\underbrace{\begin{pmatrix} (\mathbf{P}_{Lj}-\mathbf{P}_{Li})_u & (\mathbf{P}_{Lj}-\mathbf{P}_{Li})_v & 0 \\ 0 & (\mathbf{P}_{Lj}-\mathbf{P}_{Li})_u & (\mathbf{P}_{Lj}-\mathbf{P}_{Li})_v \\ (\mathbf{P}_{Lk}-\mathbf{P}_{Li})_u & (\mathbf{P}_{Lk}-\mathbf{P}_{Li})_v & 0 \\ 0 & (\mathbf{P}_{Lk}-\mathbf{P}_{Li})_u & (\mathbf{P}_{Lk}-\mathbf{P}_{Li})_v \\ (\mathbf{P}_{Lj}-\mathbf{P}_{Lk})_u & (\mathbf{P}_{Lj}-\mathbf{P}_{Lk})_v & 0 \\ 0 & (\mathbf{P}_{Lj}-\mathbf{P}_{Lk})_u & (\mathbf{P}_{Lj}-\mathbf{P}_{Lk})_v \end{pmatrix}}_{\mathbf{M}} \underbrace{\begin{pmatrix} D \\ E \\ F \end{pmatrix}}_{\mathbf{X}} = \underbrace{\begin{pmatrix} (\mathbf{n}_{Lj}-\mathbf{n}_{Li})_u \\ (\mathbf{n}_{Lj}-\mathbf{n}_{Li})_v \\ (\mathbf{n}_{Lk}-\mathbf{n}_{Li})_u \\ (\mathbf{n}_{Lk}-\mathbf{n}_{Li})_v \\ (\mathbf{n}_{Lj}-\mathbf{n}_{Lk})_u \\ (\mathbf{n}_{Lj}-\mathbf{n}_{Lk})_v \end{pmatrix}}_{\mathbf{Y}} \tag{20}$$

For each triangle, the obtained values of D, E, and F are being used to calculate the value of $\kappa = D + F$. This process is repeated for all the triangles created by connecting point $\mathbf{P}_i$ to its immediate-adjacent neighbours. The value of κ for points $\mathbf{P}_i$ could be calculated by weighted averaging over all the corresponding triangles. Different weighting parameters were tested for averaging, and the best result was obtained when this weighting parameter is the area of associated triangle.

Surface fitting

Different formulations could be used to best-fit a local surface to each point of interest $\mathbf{P}_i$ and its neighbours. In a pre-processing step, point $\mathbf{P}_i$ and its neighbours are transformed to a local coordinate system, similar to the one discussed in Extracting Fundamental Forms section. In this section, three different surfaces/approaches are described that can be used to calculate the curvature from the fitted surface:

1. Best-fit a quadratic surface:

$$F(a_i, u, v, w) = a_0u^2 + a_1v^2 + a_2uv + a_3u + a_4v - w = 0 \tag{21}$$

Curvature could be calculated either by using W matrix or using the implicit formulation as presented in equation 12.

2. In an effort to use all the available information to best-fit a local surface, one can use a cubic surface, by including the information of normal vectors at each points, as described in:[52]

$$F(a_i, u, v, w) = a_0u^2 + a_1v^2 + a_2uv + a_3u^3 + a_4u^2v + a_5uv^2 + a_6v^3 - w = 0 \tag{22}$$

The curvature could be calculated either by using W matrix of this formulation or using the implicit formulation as presented in equation 12. For the implicit function presented in equation 22, the normal vector at each point would be:

$$\mathbf{N}(a_i, u, v, w) = (F_u, F_v, -1) = (2a_0u+a_2v+3a_3u^2+2a_4uv+a_5v^2, \quad 2a_1v+a_2u+a_4u^2+2a_5uv+3a_6v2, \quad -1)$$

Since the normal vector for each point $\mathbf{P}_i$ is known from Proposed Normal Estimation method section, one can convert the $\mathbf{N}_i = (N_u, N_v, N_w)$ to $(-N_u/N_w, -N_v/N_w, -1)$. Thus the under-determined

system of equation for each arbitrary point can be written as:

$$\begin{pmatrix} u^2 & v^2 & uv & u^3 & u^2v & uv^2 & v^3 \\ 2u & 0 & v & 3u^2 & 2uv & v^2 & 0 \\ 0 & 2v & u & 0 & u^2 & 2uv & 3v^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} = \begin{pmatrix} w \\ -N_u/N_w \\ -N_v/N_w \end{pmatrix} \tag{23}$$

By including all the immediate-adjacent neighbours of $\mathbf{P}_i$ to this formulation, one can obtain a determined/over-determined system of equation which could be solved using direct/least-square approaches.

3. In the proposed approach, a quadratic surface with formulation of equation 21 is best-fitted to point $\mathbf{P}_i$ and all its immediate-adjacent neighbouring points as:

$$F(a_i, u, v, w) = a_0u^2 + a_1v^2 + a_2uv + a_3u + a_4v - w = 0 \tag{24}$$

The difference of this approach is that the fitted surface is not used to calculate the curvature only at $\mathbf{P}_i$, but also to calculate the curvature at all the neighbouring points used in fitting process. Thus, as each point $\mathbf{P}_j$ appears in m different neighbouring sets, and as a result, in m different fitted surfaces, different values of κ can be set for $\mathbf{P}_j$. The final curvature value for this point, κ_j^f, is the average of

all curvature values calculated from all the obtained m surfaces:

$$\kappa_j^f = \frac{\sum^m \kappa_j}{m} \tag{25}$$

In next section the results and comparisons of different discussed approaches would be presented. The numerical methods are implemented in an in-house parallel c++/MPI code called TermoFluids.[54] Validations and verifications of the general framework of numerical tools used in this work have been reported in.[12,13,55,56,57,58,59]

Numerical experiments and discussions

Two different test cases are introduced to perform comparisons and verification on accuracy of aforementioned methods. For each case, at each given interface cell i, the errors of estimated normal vector (n_i) and curvature (κ_i) are calculated as:

$$e_i^n = 1 - n_i \cdot n_{exact} \tag{26}$$

$$e_i^k = 1 - k_i/k_{exact} \tag{27}$$

Based on these values, the first and infinite norms of error are calculated as $L_1 = |\sum(e_i V_i)|/\sum(V_i)$ and $L_{inf} = max(|e_i|)$ with V_i as the volume of the associated cell.

Spheroid test case

Numerical tests are performed for a spheroid interface with implicit formulation of:

$$F(x,y,z) = \frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2} + \frac{(z-z_0)^2}{c^2} - 1 = 0 \tag{28}$$

The simulations are done in a cube domain with side length of L. For simplicity in extracting normal and curvature formulations, an oblate spheroid with $a = b = L/3.33$, and $c = L/4.44$ is selected. Two different grid types of structured and unstructured (triangular) meshes have been used to discretize the domain as depicted in figure 10 with characteristic grid size of $h = L/n$, for $n \in \{10, 20, 40, 80, 160\}$. Although the simulations have been done in both of these meshes, since the proposed methods are based on the extracted point-cloud representation of the interface, the results have shown to be independent of the grid type. Hence, to avoid redundancy, only the results obtained with the unstructured meshes are presented in the rest of the paper. In each case the simulations are repeated 100 times where (x_0, y_0, z_0) is randomly changed to eliminate the effect of initial location on results.

Fig. 10 should be placed here

For each point in point-cloud extracted from PLIC-VOF interface polygon centers, in a minimization problem, using Lagrange Multiplier approach and solving the resulting equations by Newton–Raphson method, the closest point on the surface of associated spheroid is extracted to compare the results. For the extracted point, the exact values of normal and curvature are calculated by applying equations 12 to the implicit formulation of equation 28. Figure 4 (top) illustrates the surface of selected spheroid.

3D wave test case

Numerical tests are performed for a three-dimensional wave interface with implicit formulation of:

$$F(x, y, z) = (z - z_0) - sin\,(x - x_0)\,cos\,(y - y_0) - \frac{L}{2} = 0 \tag{29}$$

The simulations are done in a cube domain with side length of $L = 8$. The domain is discretized with the same characteristic grid sizes as Spheroid test case. As before, the simulations are repeated 100 times, randomly selecting (x_0, y_0, z_0) coordinates to eliminate the effect of initial location on results. Similar to the spheroid test case, for each point in point-cloud extracted from PLIC-VOF interface polygon centers, in a minimization problem, using Lagrange Multiplier approach and solving the resulting equation by Newton–Raphson method, the closest point on the surface of associated wave is extracted to compare the results. For the extracted point, the exact values of normal and curvature are calculated by applying equations 12 to the implicit formulation of 29. Figure 4 (bottom) illustrates the surface of selected wave.

Results

Figures 11 and 12 illustrate the L_1 (left) and L_{inf} (right) norms of errors in interface normal vector $\mathbf{n}$ for the spheroid and wave interface test cases, respectively. Different approaches of PY,[25] LVIRA,[27] Mosso-Swartz,[29] and Modified Swatrz,[30] are used and the results of these approaches are compared with the proposed method presented in Proposed Normal Estimation method section.

Fig. 11 should be placed here

Fig. 12 should be placed here

As can be seen in these figures, for both cases of spheroid and 3D wave interfaces, the PY and Swartz

methods show similar results, with around 1st order of accuracy for coarser meshes, and almost zero order of accuracy for finer meshes in L_1. For L_{inf}, these two methods represent an order of convergence of less than one and even close to zero. The Modified Swartz algorithm is an improvement on Swartz approach but still does not present an order of accuracy better than one. LVIRA method, illustrates an order of accuracy around 2 for L_1 in both cases, but fails to achieve anything better than 0.8, and 1.5 for spheroid and wave test cases, respectively. The method proposed in this paper, discussed in Proposed Normal Estimation method section, shows 2nd and 3rd order of spacial convergence for L_1 in spheroid and wave test cases, respectively. For L_{inf}, however, these values are around 2 for both test cases. It is noticeable that the proposed method for calculating the normal vector $\mathbf{n}$ outperforms other methods studied in this section, leading to higher spatial convergence, while maintaining simplicity in implementation.

Figures 13 and 14, illustrate the L_1 (left) and L_{inf} (right) norms of errors in interface curvature κ for the spheroid and 3D wave test cases. Different approaches of fundamental forms (Extracting Fundamental Forms section), quadratic surface fitting (Surface fitting section item 1), and cubic surface fitting (Surface fitting section item 2), are used and their results are compared with the method proposed in Surface fitting section (item 3).

Fig. 13 should be placed here

Fig. 14 should be placed here

As can be seen in these figures, in case of very coarse meshes, both L_1 and L_{inf} norms of error for methods of Fundamental Forms, Quadratic and Cubic surfaces fittings show an error of convergence around 2 in both test cases of spheroid and 3D wave. However, this order of convergence converts to zero for L_1 and even negative for L_{inf} for moderate to fine mesh sizes in both test cases. The method proposed

in this paper, however maintains an order of convergence close to two for both L_1 and L_{inf} in both test cases. This convergence order is indeed an improvement on curvature calculation using classical formulation of $\kappa = \nabla \cdot \left(\frac{\nabla \alpha}{|\nabla \alpha|}\right)$ or RDF methods illustrated in figure 7. For very fine meshes, however, the order of convergence in the proposed method reduces to zero. This pattern is also visible in curvature calculations using the classical formulation and RDF methods as depicted in figure 7.

According to figure 5, one can notice that for very fine meshes, the convergence error in L_{inf} is reducing, which means that further mesh refinements does not lead to improvements on the information available in the area on the interface surface responsible for maximum error. In other words, if a region on the estimated interface has the maximum error compared to the exact interface, by refining the mesh size, this region is not resolved any better, and still introducing the same error in the solution. This can be associated to the nature of PLIC method, as its objective is not to estimate the exact location of the interface, but to satisfy the volume fraction of the phases in each cell by a line (in 2D) or plane (in 3D). Thus the order of spatial convergence of this region is zero, meaning no improvement due to the refinement of the mesh. Since the curvature calculation is based on the available information of the interface, and since for very fine meshes, the further refinement does not provide extra information of the estimated interface, and the curvature calculation shows a reduction in convergence order.

Figure 15 illustrates the execution time of curvature and normal vector evaluation functions in 100-times repeated spheroid test case. The curvature evaluation execution times are normalized by the average execution time of the classical formulation function, and the normal vector execution times are normalized by the average execution time of PY method. According to this figure, the proposed method for normal vector evaluation, the Swartz and modified Swartz approaches have similar execution times, which is around

twice the execution time of PY method. The LVIRA function execution time however ranges between 36 to around 50 times the execution of the PY method.

For curvature, however, as expected, the execution time of Classical formulation is lowest, followed by execution time of RDF Quadratic Surface, Proposed method, Fundamental Forms, and Cubic Surface. These approaches range from 2 to around 4.5 times the average execution time of Classical formulation.

Fig. 15 should be placed here

Conclusion

In this work, difficulties in calculating the geometrical properties of the interface, i.e. normal vector and curvature, using the PLIC-VOF approach were studied. Different methods available in the literature for calculating these parameters have been analyzed. For two 3D test cases of spheroid and a wave function interface, evaluations were performed to understand how well the PLIC interface estimates the actual location of the interface. In other terms, the scope of the test consists in quantifying, how much information of the actual interface is available in estimated PLIC interface. A point-cloud, based on the interface polygon centers, was formed to estimate the interface. Different approaches for calculating the interface normal and curvature from this point-cloud were tested. The method proposed in this study to evaluate the interface normal vector has shown to be spatially second-order accurate for both L_1 and challenging L_{inf} norms of error, outperforming well-known widely-used methods, e.g. LVIRA, while maintaining simplicity in implementation and lower computational costs as there is no need for expensive optimization steps. Different methods for evaluating the interface curvature from the available information of PLIC interface were tested as well. Diverse ideas for the calculation of the interface curvature were tested and it was shown

that the method proposed in this study improves the spatial convergence of curvature error while being independent of the grid type. Indeed, the method has shown to be effectively applicable to both structured and unstructured meshes, presenting improvements in spatial convergence order.

Acknowledgments

This work has been financially supported by MCIN/AEI/10.13039/501100011033 Spain, project PID2020-115837RBI00. E. Schillaci acknowledges the financial support of the Programa Torres Quevedo (PTQ2018-010060)

References

[1] Ruben Scardovelli and Stéphane Zaleski. Direct numerical simulation of free-surface and interfacial flow. *Annual Review of Fluid Mechanics*, 31(1):567–603, 1999.

[2] B. Maury. Characteristics ALE method for the unsteady 3D Navier-Stokes equations with a free surface. *International Journal of Computational Fluid Dynamics*, 1996.

[3] Salih Ozen Unverdi and Grétar Tryggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. *Journal of Computational Physics*, 100(1):25–37, 1992.

[4] C. W. Hirt and B. D. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 1981.

[5] William J. Rider and Douglas B. Kothe. Reconstructing Volume Tracking. *Journal of Computational Physics*, 141(2):112–152, 1998.

[6] Stanley Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 1988.

[7] Mark Sussman. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics*, 1994.

[8] Elin Olsson and Gunilla Kreiss. A conservative level set method for two phase flow. *Journal of Computational Physics*, 210(1):225–246, 2005.

[9] Mark Sussman and Elbridge Gerry Puckett. A Coupled Level Set and Volume-of-Fluid Method for Computing 3D and Axisymmetric Incompressible Two-Phase Flows. *Journal of Computational Physics*, 2000.

[10] Asghar Esmaeeli and Grétar Tryggvason. Direct numerical simulations of bubbly flows. Part 1. Low Reynolds number arrays. *Journal of Fluid Mechanics*, 1998.

[11] S. Afkhami, A. M. Leshansky, and Y. Renardy. Numerical investigation of elongated drops in a microfluidic T-junction. *Physics of Fluids*, 2011.

[12] Ahmad Amani, Néstor Balcázar, Jesús Castro, and Assensi Oliva. Numerical study of droplet deformation in shear flow using a conservative level-set method. *Chemical Engineering Science*, 207:153–171, 2019.

[13] Ahmad Amani, Néstor Balcázar, Enrique Gutiérrez, and Assensi Oliva. Numerical study of binary droplets collision in the main collision regimes. *Chemical Engineering Journal*, pages 477–498, 2019.

[14] O. Ubbink and R. I. Issa. A Method for Capturing Sharp Fluid Interfaces on Arbitrary Meshes. *Journal of Computational Physics*, 1999.

[15] M. Darwish and F. Moukalled. Convective schemes for capturing interfaces of free-surface flows on unstructured grids. *Numerical Heat Transfer, Part B: Fundamentals*, 2006.

[16] B. Xie, S. Ii, and F. Xiao. An efficient and accurate algebraic interface capturing method for unstructured grids in 2 and 3 dimensions: The THINC method with quadratic surface representation. *International Journal for Numerical Methods in Fluids*, 2014.

[17] W. F. Noh and Paul Woodward. SLIC (Simple Line Interface Calculation). 1976.

[18] Yuriko Renardy and Michael Renardy. PROST: A parabolic reconstruction of surface tension for the volume-of-fluid method. *Journal of Computational Physics*, 183(2):400–421, 2002.

[19] Joaquin López, J. Hernández, P. Gómez, and F. Faura. A volume of fluid method based on multidimensional advection and spline interface reconstruction. *Journal of Computational Physics*, 2004.

[20] G.B. Ermentrout. Numerical recipes in C. *Mathematical Biosciences*, 1989.

[21] Ruben Scardovelli and Stephane Zaleski. Analytical Relations Connecting Linear Interfaces and Volume Fractions in

Rectangular Grids. *Journal of Computational Physics*, 2000.

[22] Xiaofeng Yang and Ashley J. James. Analytic relations for reconstructing piecewise linear interfaces in triangular and tetrahedral grids. *Journal of Computational Physics*, 2006.

[23] J. López and J. Hernández. Analytical and geometrical tools for 3D volume of fluid methods in general grids. *Journal of Computational Physics*, 227(12):5939–5948, 2008.

[24] Christopher B. Ivey and Parviz Moin. Accurate interface normal and curvature estimates on three-dimensional unstructured non-convex polyhedral meshes. *Journal of Computational Physics*, 300:365–386, 2015.

[25] D. L. Youngs. Time-dependent multi-material flow with large fluid distortion. 1982.

[26] E. Aulisa, S. Manservisi, R. Scardovelli, and S. Zaleski. Interface reconstruction with least-squares fit and split advection in three-dimensional Cartesian geometry. *Journal of Computational Physics*, 2007.

[27] James Edward Pilliod and Elbridge Gerry Puckett. Second-order accurate volume-of-fluid algorithms for tracking material interfaces. *Journal of Computational Physics*, 199(2):465–502, 2004.

[28] S J Mosso, B K Swartz, D B Kothe, and S P Clancy. Recent enhancements of volume tracking algorithms for irregular grids. *Los Alamos National Laboratory, Los Alamos, NM, LA_UR_96_277*, (June):20–23, 1996.

[29] S J Mosso, B K Swartz, D B Kothe, and R C Ferrell. - A Parallel, Volume-Tracking Algorithm for Unstructured Meshes. In P Schiano, A Ecer, J Periaux, and N Satofuka, editors, *Parallel Computational Fluid Dynamics 1996*, pages 368–375. North-Holland, Amsterdam, 1997.

[30] Vadim Dyadechko and Mikhail Shashkov. Moment-of-fluid interface reconstruction. *Mathematical Modeling And Analysis*, 2005.

[31] Tomislav Marić, Holger Marschall, and Dieter Bothe. An enhanced un-split face-vertex flux-based VoF method. *Journal of Computational Physics*, 371:967–993, 2018.

[32] Blair Swartz. The Second-Order Sharpening of Blurred Smooth Borders. *Mathematics of Computation*, 1989.

[33] Hyung Taek Ahn and Mikhail Shashkov. Geometric algorithms for 3D interface reconstruction. In *Proceedings of the 16th International Meshing Roundtable, IMR 2007*, 2008.

[34] Andreas Haselbacher and Oleg V. Vasilyev. Commutative discrete filtering on unstructured grids based on least-squares techniques. *Journal of Computational Physics*, 2003.

[35] Nasser Ashgriz, Tiberiu Barbat, and Gang Wang. A computational Lagrangian-Eulerian advection remap for free surface flows. *International Journal for Numerical Methods in Fluids*, 44(1):1–32, 2004.

[36] Poorya A. Ferdowsi and Markus Bussmann. Second-order accurate normals from height functions. *Journal of Computational Physics*, 2008.

[37] Sharen J. Cummins, Marianne M. Francois, and Douglas B. Kothe. Estimating curvature from volume fractions. *Computers and Structures*, 83(6-7):425–434, 2005.

[38] Kei Ito, Tomoaki Kunugi, Shuji Ohno, Hideki Kamide, and Hiroyuki Ohshima. A high-precision calculation method for interface normal and curvature on an unstructured grid. *Journal of Computational Physics*, 273:38–53, 2014.

[39] Marianne M. Francois and Blair K. Swartz. Interface curvature via volume fractions, heights, and mean values on nonuniform rectangular grids. *Journal of Computational Physics*, 2010.

[40] Mark Owkes and Olivier Desjardins. A mesh-decoupled height function method for computing interface curvature. *Journal of Computational Physics*, 2015.

[41] Fabien Evrard, Fabian Denner, and Berend van Wachem. Estimation of curvature from volume fractions using parabolic reconstruction on two-dimensional unstructured meshes. *Journal of Computational Physics*, 2017.

[42] J U Brackbill, D B Kothe, and C Zemach. A continuum method for modeling surface tension. *Journal of Computational Physics*, 100(2):335–354, 1992.

[43] Néstor Balcázar, Oriol Lehmkuhl, Lluís Jofre, Joaquim Rigola, and Assensi Oliva. A coupled volume-of-fluid/level-set method for simulation of two-phase flows on unstructured meshes. *Computers & Fluids*, 124:12–29, 2016.

[44] Kei Ito, Tomoaki Kunugi, Hiroyuki Ohshima, and Takumi Kawamura. A volume-conservative PLIC algorithm on three-dimensional fully unstructured meshes. *Computers and Fluids*, 88:250–261, 2013.

[45] J. López, C. Zanzi, P. Gómez, R. Zamora, F. Faura, and J. Hernández. An improved height function technique for computing interface curvature from volume fractions. *Computer Methods in Applied Mechanics and Engineering*, 2009.

[46] G. Bornia, A. Cervone, S. Manservisi, R. Scardovelli, and S. Zaleski. On the properties and limitations of the height function method in two-dimensional Cartesian geometry. *Journal of Computational Physics*, 2011.

[47] Fabian Denner and Berend G M van Wachem. Fully-Coupled Balanced-Force VOF Framework for Arbitrary Meshes with Least-Squares Curvature Evaluation from Volume Fractions. *Numerical Heat Transfer, Part B: Fundamentals*,

65(3):218–255, 2014.

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

[49] Szymon Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *Proceedings - 2nd International Symposium on 3D Data Processing, Visualization, and Transmission. 3DPVT 2004*, 2004.

[50] Anshuman Razdan and Myungsoo Bae. Curvature estimation scheme for triangle meshes using biquadratic Bézier patches. *CAD Computer Aided Design*, 37(14):1481–1491, 2005.

[51] P Yang and X Qian. Direct Computing of Surface Curvatures for Point-Set Surfaces. *Symposium on Point-Based Graphics*, 2007.

[52] Jack Goldfeather and Victoria Interrante. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Transactions on Graphics*, 23(1):45–63, 2004.

[53] Alfred Gray. *Modern Differential Geometry of Curves and Surfaces with Mathematica*. CRC Press, Inc., USA, 1st edition, 1996.

[54] http://www.termofluids.com/. Termo Fluids S.L.

[55] Ahmad Amani, Néstor Balcázar, Alireza Naseri, and Joaquim Rigola. A numerical approach for non-Newtonian two-phase flows using a conservative level-set method. *Chemical Engineering Journal*, 385(December 2019):123896, 2020.

[56] Lluís Jofre, Oriol Lehmkuhl, Jesús Castro, and Assensi Oliva. A 3-D Volume-of-Fluid advection method based on cell-vertex velocities for unstructured meshes. *Computers and Fluids*, 94:14–29, 2014.

[57] E Gutiérrez, F Favre, N Balcázar, A Amani, and J Rigola. Numerical approach to study bubbles and drops evolving through complex geometries by using a level set-Moving mesh-Immersed boundary method. *Chemical Engineering Journal*, 349(February):662–682, 2018.

[58] Ahmad Amani, Nestor Balcazar, Alireza Naseri, and Asensi Oliva. A Study on Binary Collision of GNF Droplets Using a Conservative Level-Set Method. In *6th European Conference on Computational Mechanics (ECCM 6)- 7th European Conference on Computational Fluid Dynamics (ECFD 7)*, Glasgow, UK, 2018.

[59] Ahmad Amani, Nestor Balcazar, Enrique Gutiérrez, and Asensi Oliva. DNS of un-equal size droplets collision using a moving-mesh/level-set method. In *ERCOFTAC workshop direct and large eddy simulation 12 (DLES 12)*, Madrid, Spain,

2019.

List of Figures

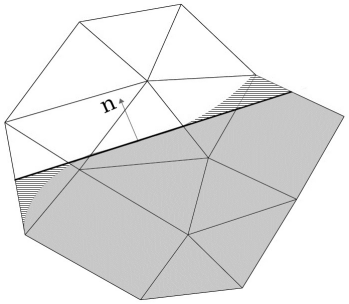

Figure 1: LVIRA error function for a 2D unstructured mesh stencil. Error e($\mathbf{n}$) for the central cell c is calculated by extending its reconstructed interface to all the neighbor cells and summing all the hashed area representing $\alpha_{kF} - \beta_{kF}$ in each neighbour cell

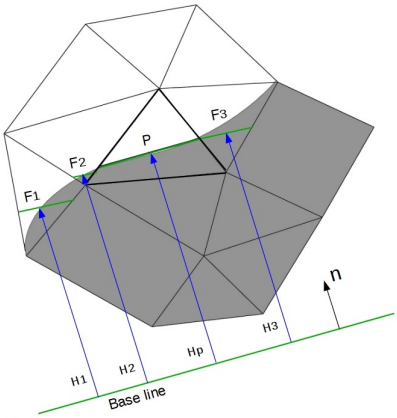

Figure 2: Definition of HF, using the predicted normal vector of $\mathbf{n}$ in reference cell $\mathbf{P}$.

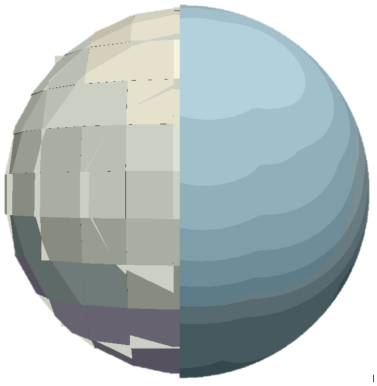

Figure 3: A comparison of PLIC interface (left) with grid spacing of $h = 0.1d$ (with d as droplet diameter) Vs. exact spherical interface (right).

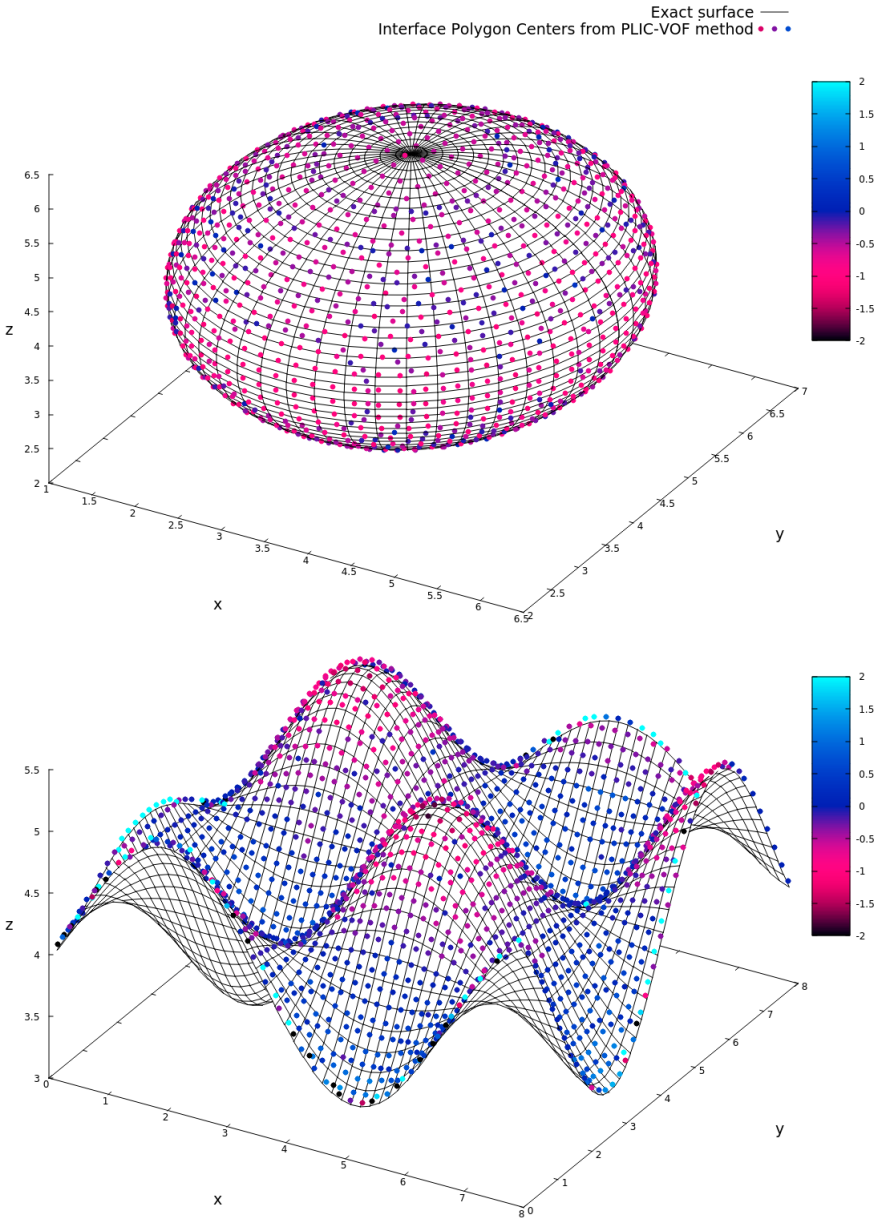

Figure 4: Extracted center points (**Pi**) of interface polygons in PLIC-VOF method compared with the exact surface of the associated oblate spheroid (top), and wave function (bottom). The color palette represents the error $E = (\mathbf{P}_i - \mathbf{P}) \cdot \mathbf{n}_P / h \times 100$, where $\mathbf{P}$ is the point on the exact surface closest to **Pi**, and h=0.2 is characteristic size of grid cell.

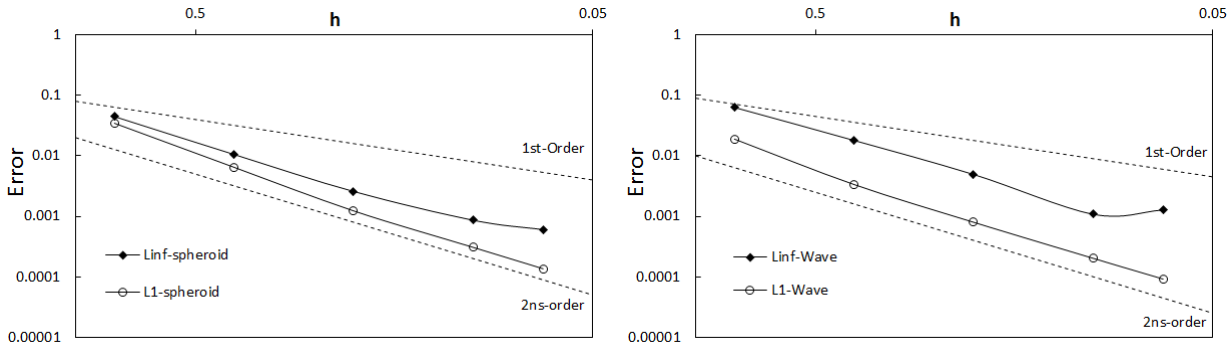

Figure 5: L_1 and L_{inf} spatial convergence for $error = (\mathbf{P}_i - \mathbf{P}) \cdot \mathbf{n}$ where $\mathbf{P}_i$ is the interface polygon center in PLIC-VOF method and P is the point on the surface of spheroid (left) and wave function (right) closest to $\mathbf{P}_i$.

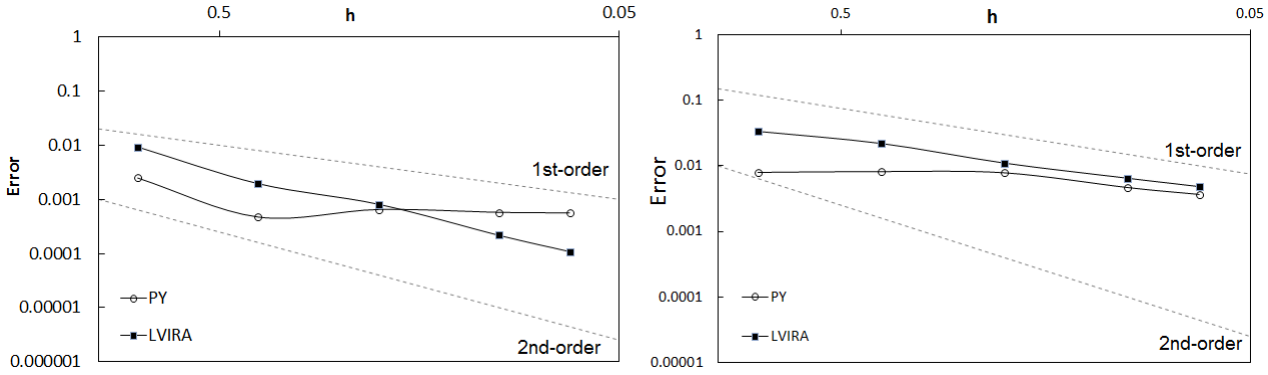

Figure 6: L_1 and L_{inf} of spatial convergence of error in evaluating surface normal vector n as defined in equation 26 using PY method compared with LVIRA for a spheroid interface. Left: L_1, right: L_{inf}

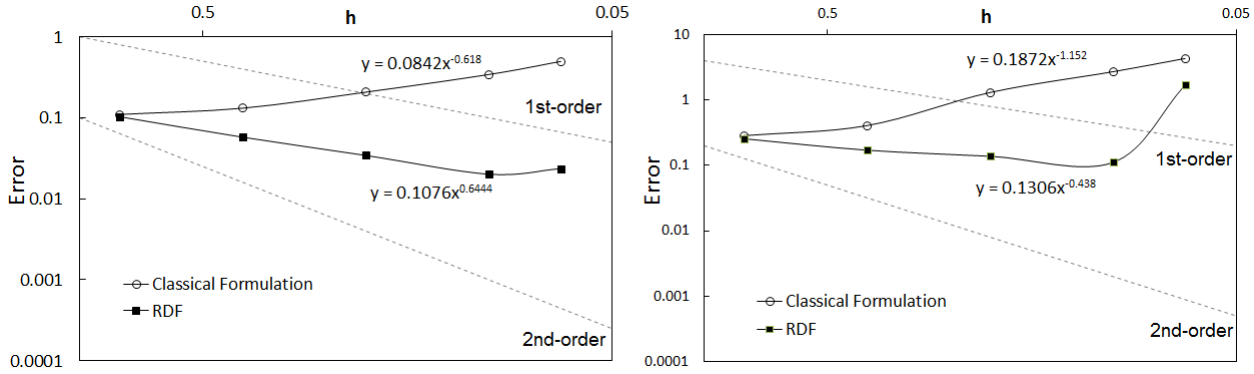

Figure 7: L_1 and L_{inf} on spatial convergence of error in evaluating surface curvature κ as defined in equation 26 using classical formulation method as $\kappa = \nabla \cdot \left(\frac{\nabla \alpha}{|\nabla \alpha|}\right)$ compared with RDF method as $\kappa = \nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|}\right)$ with ϕ as distance function for a spheroid interface. Left: L_1, right: L_{inf}

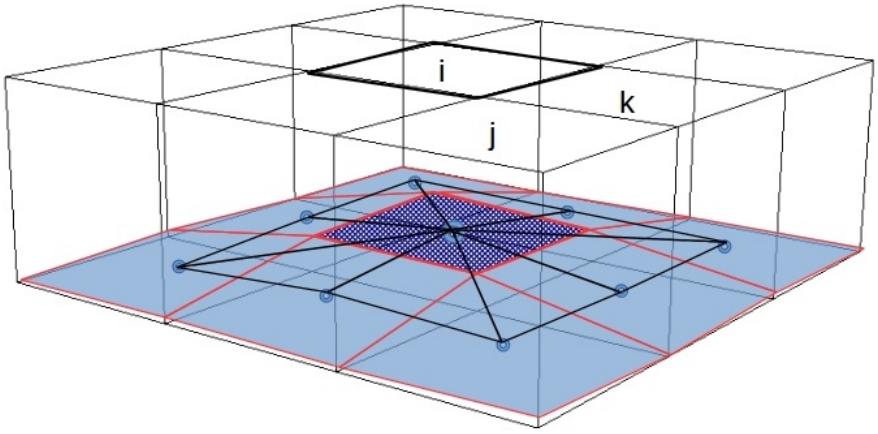

Figure 8: Interface polygons (area with red-line edges) along with interface polygon centers (blue points) and constructed triangular meshes obtained by connecting the point $\mathbf{P}_i$ to its neighbours $\mathbf{P}_j$ and $\mathbf{P}_k$. (please note that for the sake of clarity of viewing this figure, the plotted interface polygons are not necessarily planar).

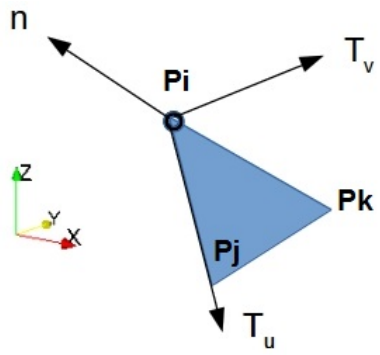

Figure 9: Transformation of each triangle into a local coordinate system.

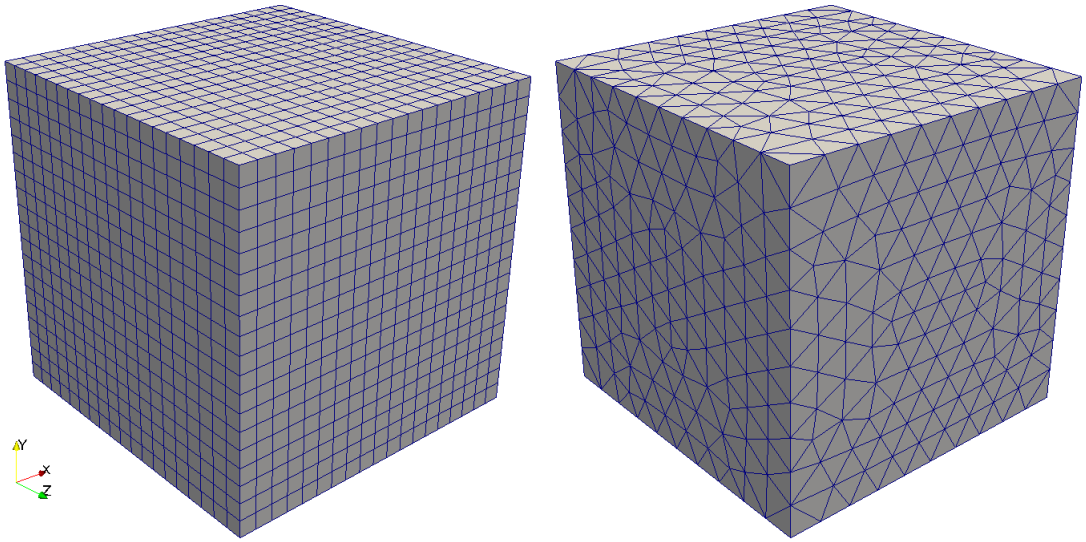

Figure 10: Two different computational grids with structured (left) and unstructured (right) mesh types

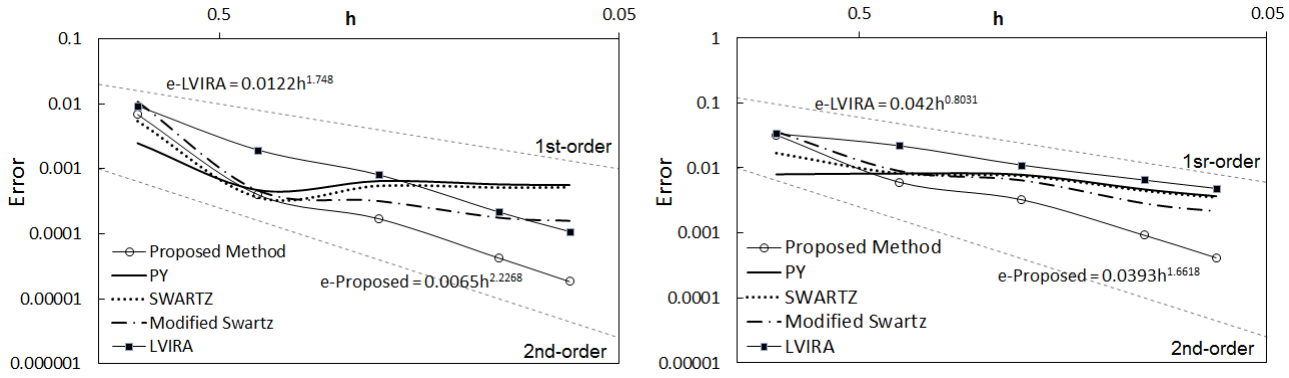

Figure 11: L_1 (left) and L_{inf} (right) of spatial convergence of error in evaluating surface normal vector n as defined in equation 26 using PY, Swartz, Modified Swartz, and LVIRA methods compared with the method proposed in this paper for spheroid test case.

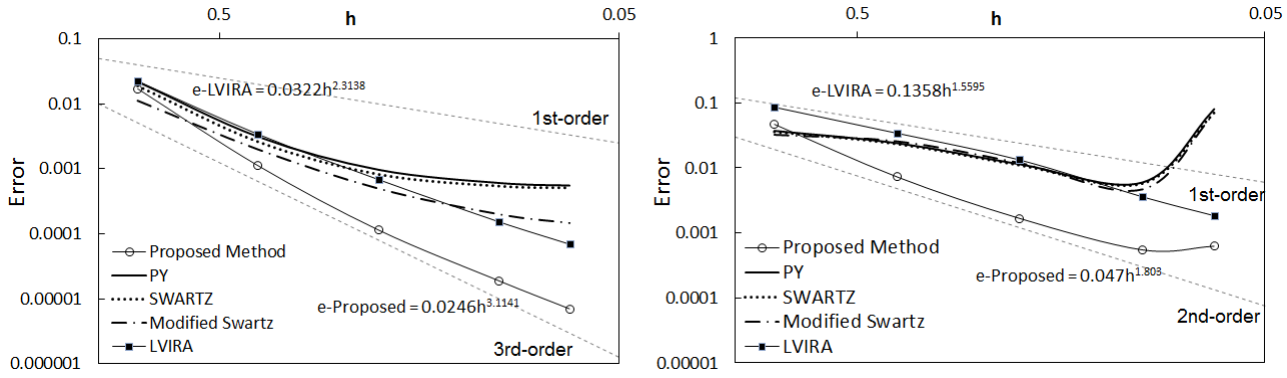

Figure 12: L_1 (left) and L_{inf} (right) of spatial convergence of error in evaluating surface normal vector n as defined in equation 26 using PY, Swartz, Modified Swartz, and LVIRA methods compared with the method proposed in this paper for the 3D wave test case.

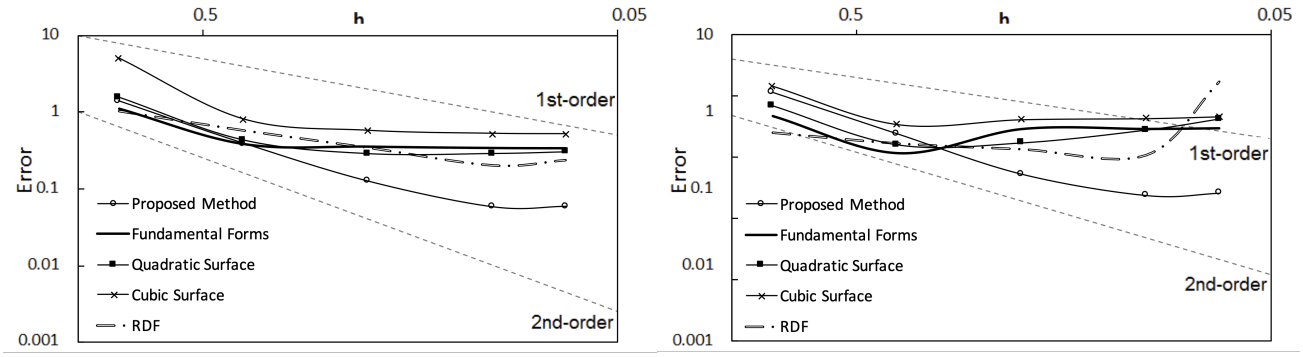

Figure 13: L_1 (left) and L_{inf} (right) of spatial convergence of error in evaluating surface curvature κ as defined in equation 26 using methods presented in Extracting Fundamental Forms and Surface fitting sections (items 1,2) compared with the method proposed in Surface fitting section (item 3) for spheroid test case.

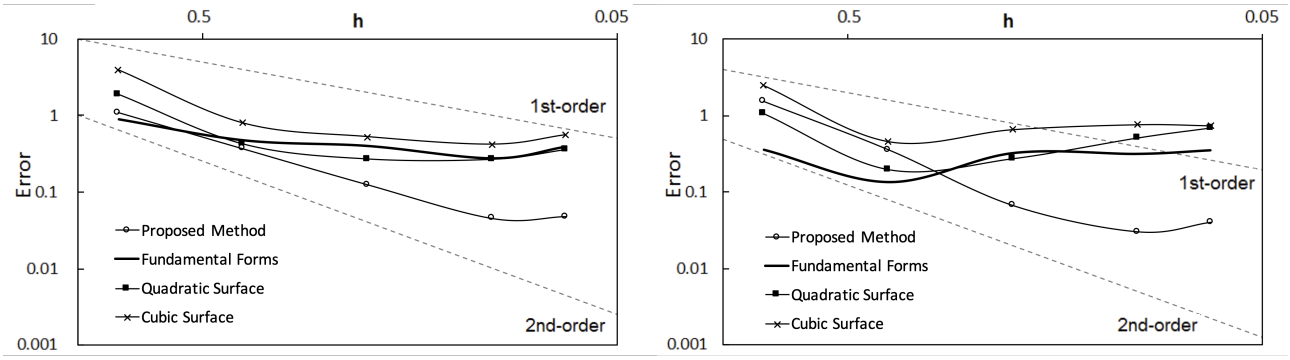

Figure 14: L_1 (left) and L_{inf} (right) of spatial convergence of error in evaluating surface curvature κ as defined in equation 26 using methods presented in Extracting Fundamental Forms and Surface fitting section (items 1,2) compared with the method proposed in Surface fitting section (item 3) for 3D wave test case.

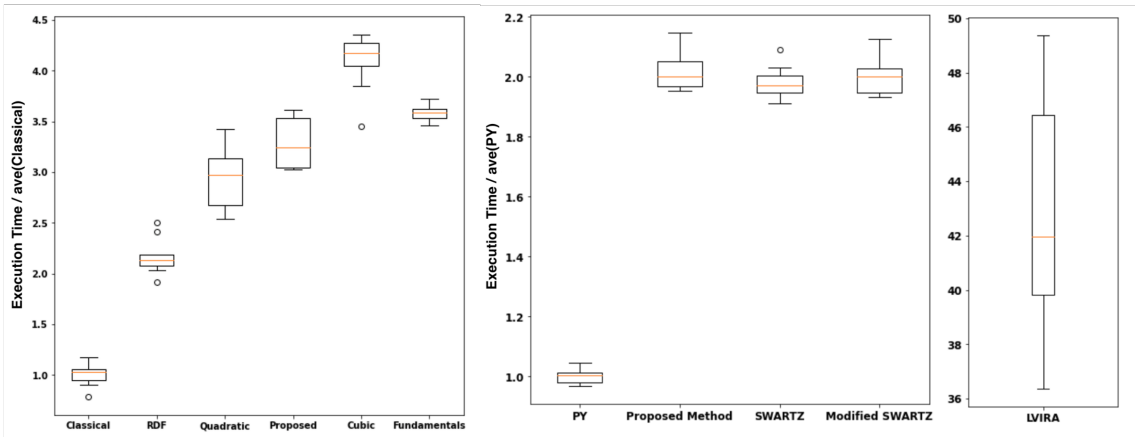

Figure 15: Execution time distribution of different approaches for 100-times repeated spheroid test case. Left: Execution time of different curvature estimation approaches normalized by the average execution time of Classical formulation tests. Right: Execution time of different normal vector estimation approaches normalized by the average execution time of PY method tests.