Estimation of curvature from volume fractions using parabolic reconstruction on two-dimensional unstructured meshes

Fabien Evrard, Fabian Denner, Berend van Wachem

Imperial College London, Department of Mechanical Engineering, Exhibition Road, London, SW7 2AZ, United Kingdom

Abstract

This paper proposes a method to estimate the curvature of an interface represented implicitly by discrete volume fractions on an unstructured two-dimensional mesh. The method relies on the computation of local parabolic reconstructions of the interface. The parabolic reconstruction of the interface in a given computational cell is obtained by solving a local non-linear minimisation problem, and only requires additional information from two neighbouring cells. This compactness ensures a robust behaviour on poorly-resolved interfaces. The proposed method is proven to be analogous to the height-function method for Cartesian configurations with consistent heights, and can be interpreted as a generalisation of the height-function method to meshes of any type. Tests are conducted on a range of interfaces with known curvature. The method is shown to converge with mesh refinement with the same order of accuracy as the height-function method for all three types of meshes tested, *i.e.* Cartesian, triangular, and polygonal.

Keywords: Curvature, Volume fraction, Unstructured mesh, Volume-of-fluid, Height-function

1. Introduction

The modelling of interfacial flows with surface tension represents a numerical challenge. In the context of the so-called 'volume-tracking' or 'volume-of-fluid' (VOF) methods [1–5], this challenge predominantly lies in the accurate transport of the fluid/fluid interface, and in the evaluation of its curvature. An accurate estimation of the interface curvature is indeed needed in order to avoid the generation of parasitic and/or spurious flow currents [6, 7]. VOF frameworks rely on an implicit definition of the interface between the two immiscible fluids a and b. A discontinuous indicator function $\chi : \mathbb{R}^d \to \mathbb{R}$ characterises the type of fluid present at each location of the d-dimension space, *i.e.*

$$\chi(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \in \text{fluid } a \\ 0 & \text{if } \boldsymbol{x} \in \text{fluid } b \end{cases} . \tag{1}$$

In each control volume (or computational cell) K of the simulation mesh $\mathcal{M}$, the local fraction of volume occupied by the fluid a is given by

$$\gamma_K = \frac{1}{\ell(K)} \int_K \chi(\boldsymbol{x})\, d\boldsymbol{x}, \tag{2}$$

where $\ell(K) = \int_K d\boldsymbol{x}$ is the Lebesgue measure[1] of K. This discrete field γ is referred to as the 'volume fraction' field (illustrated in figure 1). In classic VOF frameworks, it is the sole piece of information from which the geometric properties of the interface are to be extracted. In other words, evaluating local interface normals and curvatures implies to compute the first and second derivatives of the indicator function χ, when the only quantities which are known are the integrals of χ in the computational cells. Early-days VOF curvature evaluation methods rely on the mollification of the volume fraction field (*e.g.* by means of convolution with a smooth kernel function) [8–12]. The volume fraction field having been made 'more differentiable', the local normals and curvatures can then be obtained by differentiating the mollified field once and twice, respectively. The VOF method has also been coupled with level-set (LS) representations of the interface. High-order methods can then be applied onto the reconstructed-distance-function (RDF) in order to extract curvature [11, 13, 14]. It has been shown that both of these families of methods exhibit poor or no convergence at all under mesh refinement [11, 15, 16], making them badly

[1]For the rest of this paper, $\ell(\cdot)$ refers to the Lebesgue measure

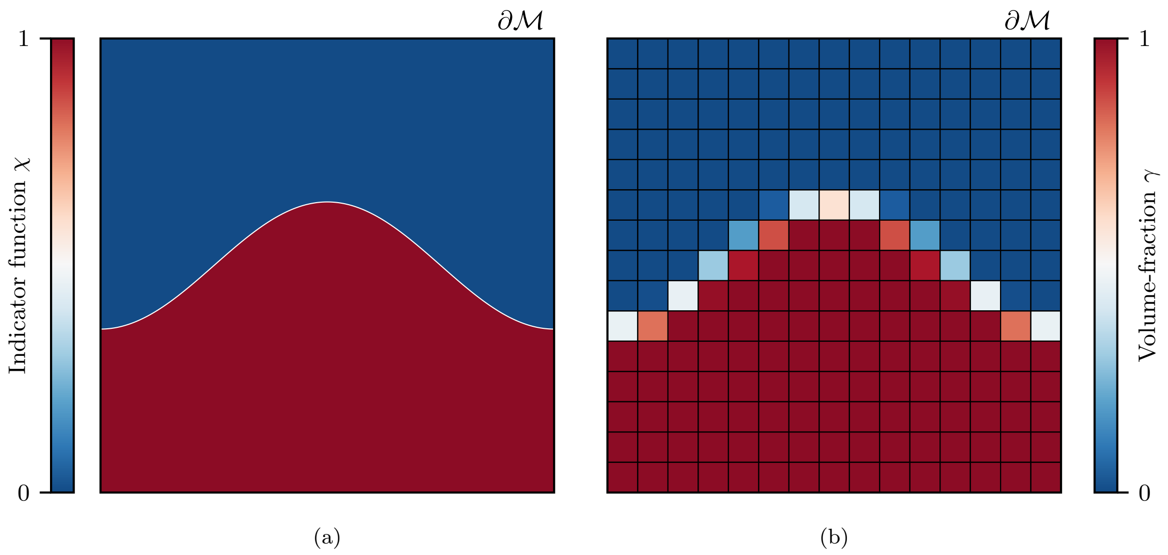


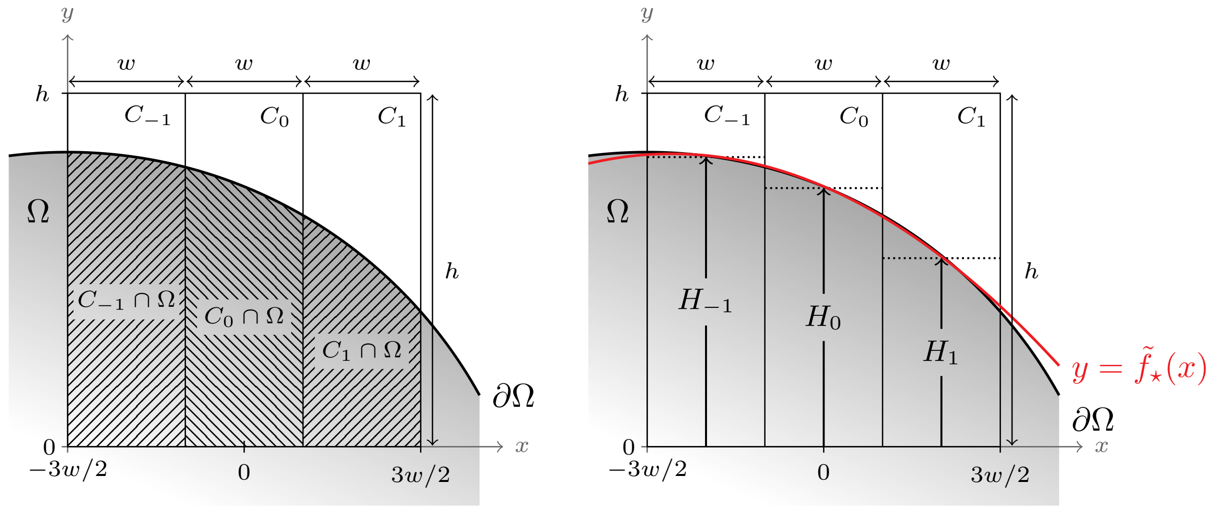
Figure 1: Illustration of the VOF formalism. (a) Indicator function field of a fluid body: the indicator function $\chi(\boldsymbol{x})$ is defined at all positions $\boldsymbol{x}$ in the simulation domain, and is equal to either 0 or 1; (b) Associated volume-fraction field γ defined for all control volumes of mesh $\mathcal{M}$, and for which $0 \leq \gamma_K \leq 1, \forall K \in \mathcal{M}$. The boundary $\delta\mathcal{M}$ of the simulation domain is represented by a thick black line, while the control volumes $K \in \mathcal{M}$ are separated by thin black lines.

suited for surface tension dominated flows. The only VOF curvature evaluation method which has been shown – both analytically [17, 18] and numerically [11, 19, 20] – to convergence with mesh refinement is the height-function (HF) method. The HF curvature evaluation method, in its original form, is limited to Cartesian grids and requires to be coupled with an alternative method when consistent heights can not be obtained (*i.e.* for high-curvature regions of the interface). It converges with second-order accuracy [11, 18–20]. An attempt to extend the method to unstructured two-dimensional meshes was conducted by Ito et al. [16], but the method shows less than first-order accuracy. Variants of the HF method with embedded (or mesh-decoupled) stencils have recently been developed, suggesting that this could lead to converging curvature evaluation on unstructured meshes [20, 21]. The mesh-decoupled height-function method of Owkes and Desjardins [20] shows second-order accuracy on Cartesian grids with smaller errors at low resolutions than classic or mixed HF methods. The embedded height-function (EHF) method of Ivey and Moin [21] shows monotically decreasing curvature errors with refinement on unstructured meshes – their convergence rate dropping at high-resolutions due to errors in the interpolation of the volume fractions on the embedded stencils. Finally, the approach of Renardy and Renardy [22] proposes to evaluate curvature from an optimal fit for a quadratric approximation of the interface over a local stencil. The method, limited to Cartesian grids, showed a great reduction of the curvature-error-induced parasitic currents.
To this date, no method yields height-function-like convergence rates on unstructured meshes, making surface tension dominated flows difficult to simulate in complex geometries using the VOF method [7]. This paper proposes a method to compute local parabolic reconstructions of the interface, from which curvature is then extracted. This curvature evaluation method, equivalent to the HF method for well-posed Cartesian configurations, is shown to converge on all types of meshes with the same order of accuracy as the HF method.

2. Analogy between the parabolic reconstruction problem and the height-function method

Consider a fluid body $\Omega \subset \mathbb{R}^2$ with its boundary $\partial\Omega$ crossing three rectangular cells C_{-1}, C_0, and C_1, of height h and width w. The three cells are adjacent to each other, forming a $3w \times h$ rectangular stencil, and the origin of the local coordinate system is located at the middle of the central bottom edge. This setup is summarised in figure 2a. It is assumed that, locally, $\partial\Omega$ can be written in an explicit form $y = f(x)$, and that $0 \leq f(x) \leq h, \forall x \in [-3w/2, 3w/2]$. Under these assumptions, the area A_i of $C_i \cap \Omega$ (*i.e.* the 'volume of fluid' in each cell) is given by

$$A_i = \int_{(i-1/2)w}^{(i+1/2)w} f(x)\,dx, \quad i \in \{-1, 0, 1\}. \tag{3}$$



(a) A fluid body $\Omega \subset \mathbb{R}^2$ intersects three rectangular cells C_{-1}, C_0, and C_1

(b) Heights associated with the volumes of fluids in each cell, and the parabolic reconstruction $y = \tilde{f}_\star(x)$

Figure 2: Illustration of the analogy between the parabolic reconstruction problem and the height-function method.

The 'height of fluid' in each column (see figure 2b) follows as

$$H_i = \frac{A_i}{w}, \quad i \in \{-1, 0, 1\}. \tag{4}$$

Let us now consider the quadratic height function

$$\tilde{f}(x) = \mathrm{a}\, x^2 + \mathrm{b}\, x + \mathrm{c}, \tag{5}$$

with coefficients $(\mathrm{a}, \mathrm{b}, \mathrm{c}) \in \mathbb{R}^3$. The quantity $\tilde{A}_i$ is the integral of $\tilde{f}(x)$ in column C_i, and $\tilde{H}_i$ is the associated height, *i.e.*

$$\tilde{H}_i = \frac{1}{w} \int_{(i-1/2)w}^{(i+1/2)w} \tilde{f}(x)\, dx, \quad i \in \{-1, 0, 1\}. \tag{6}$$

Equating H_i with $\tilde{H}_i$ (or equivalently A_i with $\tilde{A}_i$) for all $i \in \{-1, 0, 1\}$ reduces to solving the linear system of equations

$$H_i = \frac{1}{w} \left[\mathrm{a}\, \frac{x^3}{3} + \mathrm{b}\, \frac{x^2}{2} + \mathrm{c}\, x \right]_{(i-1/2)w}^{(i+1/2)w}, \quad i \in \{-1, 0, 1\}, \tag{7}$$

which, in its matricial form, writes

$$M \begin{bmatrix} \mathrm{a} \\ \mathrm{b} \\ \mathrm{c} \end{bmatrix} = \begin{bmatrix} H_{-1} \\ H_0 \\ H_1 \end{bmatrix}, \tag{8}$$

with

$$M = \begin{bmatrix} 13w^2/12 & -w & 1 \\ w^2/12 & 0 & 1 \\ 13w^2/12 & w & 1 \end{bmatrix}. \tag{9}$$

Provided that $w \neq 0$, M^{-1} is defined, and it is given by

$$M^{-1} = \begin{bmatrix} 1/2w^2 & -1/w^2 & 1/2w^2 \\ -1/2w & 0 & 1/2w \\ -1/24 & 13/12 & -1/24 \end{bmatrix}. \tag{10}$$

This means that for the configuration considered in this example, there exists a unique parabola $y = \tilde{f}_\star(x)$ whose coefficients satisfy the system of equations (7). Illustrated in figure 2b, $y = \tilde{f}_\star(x)$ is the constrained parabola which produces column heights $\tilde{H}_i$ equal to the original arbitrary heights H_i. Note that $\tilde{f}_\star(x)$ is not necessarily bounded to $[0, h]$ for $x \in [-3w/2, 3w/2]$, and that one does not necessarily have

$\tilde{f}_\star(iw) = H_i, \forall i \in \{-1, 0, 1\}$ either. From equation (10), it results that the coefficients of $\tilde{f}_\star(x)$ are

$$\mathrm{a} = \frac{H_{-1} - 2H_0 + H_1}{2w^2}, \tag{11}$$

$$\mathrm{b} = \frac{H_1 - H_{-1}}{2w}, \tag{12}$$

$$\mathrm{c} = -\frac{H_{-1} - 26H_0 + H_1}{24}. \tag{13}$$

The first derivative of $\tilde{f}_\star(x)$ at $x = 0$ is given by

$$\tilde{f}'_\star(0) = \mathrm{b} = \frac{H_1 - H_{-1}}{2w}, \tag{14}$$

and the second derivative by

$$\tilde{f}''_\star(0) = 2\mathrm{a} = \frac{H_{-1} - 2H_0 + H_1}{w^2}. \tag{15}$$

The formulas (14) and (15) are in fact the central-differences formulas which are employed in the widely-used height-function (HF) curvature evaluation method [11]. From this observation, one deduces the following proposition:

Proposition 1. *Calculating the derivatives $\tilde{f}'_\star(0)$ and $\tilde{f}''_\star(0)$ of the quadratic polynomial $\tilde{f}_\star(x)$ whose column heights satisfy $\tilde{H}_i = H_i, \forall i \in \{-1, 0, 1\}$, is equivalent to using central-differences on the heights H_i.*

The major outcome of this proposition is that the HF method relates to the volume-fraction-based parabolic fit:

$$\text{Find } (\mathrm{a}, \mathrm{b}, \mathrm{c}) \text{ such that } \tilde{H}_i = H_i, \quad \forall i \in \{-1, 0, 1\}, \tag{$\mathcal{P}_{\mathrm{vf}}$}$$

which goes against the common misconception that the HF method is linked to the point-based parabolic fit:

$$\text{Find } (\mathrm{a}, \mathrm{b}, \mathrm{c}) \text{ such that } \tilde{f}(iw) = H_i, \quad \forall i \in \{-1, 0, 1\}. \tag{$\mathcal{P}_{\mathrm{pt}}$}$$

The correct approach to evaluate curvature from a parabolic reconstruction, hence, is to consider the volume-fraction-based parabolic fit ($\mathcal{P}_{\mathrm{vf}}$) rather than the point-based parabolic fit ($\mathcal{P}_{\mathrm{pt}}$).
These results, although seemingly anecdotic, are the foundation of this paper. Having shown that the volume-fraction-based parabolic reconstruction problem described in this section is equivalent to the HF method with regards to the evaluation of the derivatives (and thus curvature), this paper proposes a method for its resolution for arbitrary discretisations – where the HF method is not applicable anymore – with the aim of reaching HF-like convergence rates for the evaluation of curvature on unstructured meshes. The rest of the paper is organised as follow: section 3 defines the parabolic reconstruction problem in a general arbitrary case and discusses the evaluation of relevant quantities for its resolution; section 4 describes the general curvature evaluation algorithm, and a convergence study of the method is conducted in section 5. Finally, conclusions are drawn in section 6.

3. Resolution of the parabolic reconstruction problem in a general arbitrary case

3.1. Definition of the local problem

Consider N arbitrary polygons P_i, which are not necessarily convex and may or may not have vertices in common, and a fluid body $\Omega \subset \mathbb{R}^2$ with boundary $\partial\Omega$. The subset Ω is characterised by a discontinuous indicator function $\chi : \mathbb{R}^2 \to \mathbb{R}$ defined as

$$\chi(x, y) = \left\{ \begin{array}{ll} 1 & \text{if } (x, y) \in \Omega \\ 0 & \text{if } (x, y) \notin \Omega \end{array} \right. . \tag{16}$$

To each polygon P_i is associated a 'volume fraction' γ_i given by

$$\gamma_i = \frac{1}{\ell(P_i)} \iint_{P_i} \chi(x, y)\, dxdy. \tag{17}$$

The area $A_i = \ell(P_i \cap \Omega), i \in \{1, \ldots, N\}$ relates to the volume fraction γ_i following

$$A_i = \ell(P_i)\gamma_i. \tag{18}$$

An example of such a setup with $N = 3$ is given in figure 3.

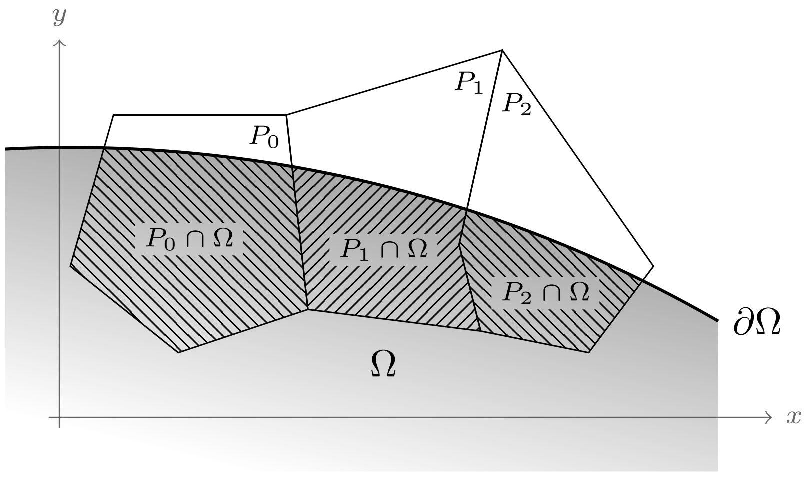


Figure 3: A fluid body $\Omega \subset \mathbb{R}^2$ intersects three arbitrary polygonal cells P_0, P_1, and P_2.

It is assumed that the interface $\partial\Omega$ can be locally approximated by a parabola $y = \tilde{f}(x)$ with

$$\tilde{f}(x) = \mathrm{a}\,x^2 + \mathrm{b}\,x + \mathrm{c}. \tag{19}$$

Let $\Lambda \subset \mathbb{R}^2$ be the subset $\{y \leq \tilde{f}(x), \forall x \in \mathbb{R}\}$. The volume fraction associated with Λ is given by

$$\tilde{\gamma}_i = \frac{1}{\ell(P_i)} \iint_{P_i} \tilde{\chi}(x, y)\, dxdy, \tag{20}$$

where $\tilde{\chi}$ is the parabola indicator function, *i.e.*

$$\tilde{\chi}(x, y) = \left\{ \begin{array}{lll} 1 & \text{if } (x, y) \in \Lambda & (\Leftrightarrow y \leq \tilde{f}(x)) \\ 0 & \text{if } (x, y) \notin \Lambda & (\Leftrightarrow y > \tilde{f}(x)) \end{array} \right. . \tag{21}$$

The area $\tilde{A}_i = \ell(P_i \cap \Lambda)$ relates to $\tilde{\gamma}_i$ following

$$\tilde{A}_i = \ell(P_i)\tilde{\gamma}_i. \tag{22}$$

In order to minimise the difference between the $\tilde{\gamma}_i$ and the given volume fractions γ_i, the coefficients of the parabola are chosen to be the solutions of the minimisation problem

$$\min_{(\mathrm{a},\mathrm{b},\mathrm{c}) \in \mathbb{R}^3} J(\mathrm{a}, \mathrm{b}, \mathrm{c}), \tag{$\mathcal{P}$}$$

where

$$J(\mathrm{a}, \mathrm{b}, \mathrm{c}) = \frac{1}{2} \sum_{i=1}^{N} (\tilde{\gamma}_i - \gamma_i)^2. \tag{23}$$

In the example presented in section 2, $\tilde{\gamma}$ was varying linearly with the coefficients $(\mathrm{a}, \mathrm{b}, \mathrm{c})$, which allowed for the quick and direct resolution of problem ($\mathcal{P}_{\mathrm{vf}}$). In the general case, however, $\tilde{\gamma}$ does not vary linearly with the coefficients $(\mathrm{a}, \mathrm{b}, \mathrm{c})$, and its computation is not straightforward. The problem ($\mathcal{P}$) thus needs to be solved iteratively using, for instance, a quasi-Newton or Newton minimisation method. Although not trivial, analytical expressions can be derived for the necessary calculations of the gradient and Hessian matrix of J, and are given in the following section.

3.2. Computation of the cost function and its derivatives

Solving ($\mathcal{P}$) using a Newton method requires to evaluate $\tilde{\gamma}$ efficiently and accurately, as well as its gradient and Hessian matrix with regards to the coefficients $(\mathrm{a}, \mathrm{b}, \mathrm{c})$,

$$\nabla \tilde{\gamma} = \begin{bmatrix} \dfrac{\partial \tilde{\gamma}}{\partial \mathrm{a}} \\ \dfrac{\partial \tilde{\gamma}}{\partial \mathrm{b}} \\ \dfrac{\partial \tilde{\gamma}}{\partial \mathrm{c}} \end{bmatrix}, \qquad \mathrm{H}_{\tilde{\gamma}} = \nabla \left(\nabla \tilde{\gamma} \right)^T = \begin{bmatrix} \dfrac{\partial^2 \tilde{\gamma}}{\partial \mathrm{a}^2} & \dfrac{\partial^2 \tilde{\gamma}}{\partial \mathrm{a}\, \partial \mathrm{b}} & \dfrac{\partial^2 \tilde{\gamma}}{\partial \mathrm{a}\, \partial \mathrm{c}} \\ \dfrac{\partial^2 \tilde{\gamma}}{\partial \mathrm{b}\, \partial \mathrm{a}} & \dfrac{\partial^2 \tilde{\gamma}}{\partial \mathrm{b}^2} & \dfrac{\partial^2 \tilde{\gamma}}{\partial \mathrm{b}\, \partial \mathrm{c}} \\ \dfrac{\partial^2 \tilde{\gamma}}{\partial \mathrm{c}\, \partial \mathrm{a}} & \dfrac{\partial^2 \tilde{\gamma}}{\partial \mathrm{c}\, \partial \mathrm{b}} & \dfrac{\partial^2 \tilde{\gamma}}{\partial \mathrm{c}^2} \end{bmatrix}. \tag{24}$$

From there, the cost function J, its gradient, and its Hessian matrix directly follow as

$$J = \frac{1}{2} \sum_{i=1}^{N} (\tilde{\gamma}_i - \gamma_i)^2, \tag{25}$$

$$\nabla J = \sum_{i=1}^{N} \nabla \tilde{\gamma}_i (\tilde{\gamma}_i - \gamma_i), \tag{26}$$

$$\mathrm{H}_J = \sum_{i=1}^{N} \mathrm{H}_{\tilde{\gamma}_i} (\tilde{\gamma}_i - \gamma_i) + \nabla \tilde{\gamma}_i \left(\nabla \tilde{\gamma}_i \right)^T. \tag{27}$$

In order to calculate $\nabla \tilde{\gamma}$ and $\mathrm{H}_{\tilde{\gamma}}$, the volume fraction $\tilde{\gamma}$ needs to be written as a function of the parabola coefficients $(\mathrm{a}, \mathrm{b}, \mathrm{c})$. Doing so requires to define a robust way of expressing $\tilde{\gamma}$ for any polygonal cell. This process is described here:

Consider an arbitrary polygon P with n vertices $\boldsymbol{x}_j$ which have been ordered anti-clockwise. The area of P is obtained using Gauss' area formula

$$\ell(P) = \frac{1}{2} \left[\left(\sum_{j=1}^{n-1} x_j y_{j+1} \right) - \left(\sum_{j=1}^{n-1} x_{j+1} y_j \right) + x_n y_1 - x_1 y_n \right]. \tag{28}$$

The intersections between P and $\partial \Lambda$ are referred to as the points $\boldsymbol{i}_k, k \in \{1, \ldots, m\}$. If $0 < \tilde{\gamma} < 1$, then necessarily $m \geq 2$. The computation of these intersections is detailed in Appendix A. A main algorithmic difficulty for the calculation of $\tilde{\gamma}$ lies in the fact that $P \cap \Lambda$ can be formed of several distinct subsets (even if P is convex). To make sure that these distinct subsets are rightly accounted for, a modified Weiler-Atherton polygon clipping algorithm is employed [23]. This algorithm is described in Appendix B; it returns a list of q polygons Q_l which approximate the distinct subsets of $P \cap \Lambda$. The edges of the polygons Q_l which are not on ∂P are representative of the sections of the parabola $\partial \Lambda$ which are inside P. Once $P \cap \Lambda$ has been clipped accordingly, the area $\tilde{A} = \ell(P \cap \Lambda)$ is divided into three components for its calculation, as illustrated in figure 4:

1. The area $\tilde{A}_{\text{poly}}$ of the union of the clipped polygons Q_l. The area of each polygon is calculated using Gauss' area formula (28), leading to

$$\tilde{A}_{\text{poly}} = \frac{1}{2} \sum_{l=1}^{q} \left[\left(\sum_{j=1}^{n_{Q_l}-1} x_j y_{j+1} \right) - \left(\sum_{j=1}^{n_{Q_l}-1} x_{j+1} y_j \right) + x_{n_{Q_l}} y_1 - x_1 y_{n_{Q_l}} \right], \tag{29}$$

2. The area $\tilde{A}_{\text{int}}$ under the segments of the parabola $\partial \Lambda$ which are inside P

$$\tilde{A}_{\text{int}} = \sum_{l=1}^{q} \sum_{\substack{\text{edges} \\ \in \partial Q_l \\ \notin \partial P}} \int_{\min(x_{\text{edge}})}^{\max(x_{\text{edge}})} \tilde{f}(x)\, dx, \tag{30}$$

3. The area $\tilde{A}_{\text{mid}}$, approximation of $\tilde{A}_{\text{int}}$ using the midpoint rule

$$\tilde{A}_{\text{mid}} = \sum_{l=1}^{q} \sum_{\substack{\text{edges} \\ \in \partial Q_l \\ \notin \partial P}} \left[\frac{\tilde{f}(\min(x_{\text{edge}})) + \tilde{f}(\max(x_{\text{edge}}))}{2} \left(\max(x_{\text{edge}}) - \min(x_{\text{edge}})\right) \right]. \tag{31}$$

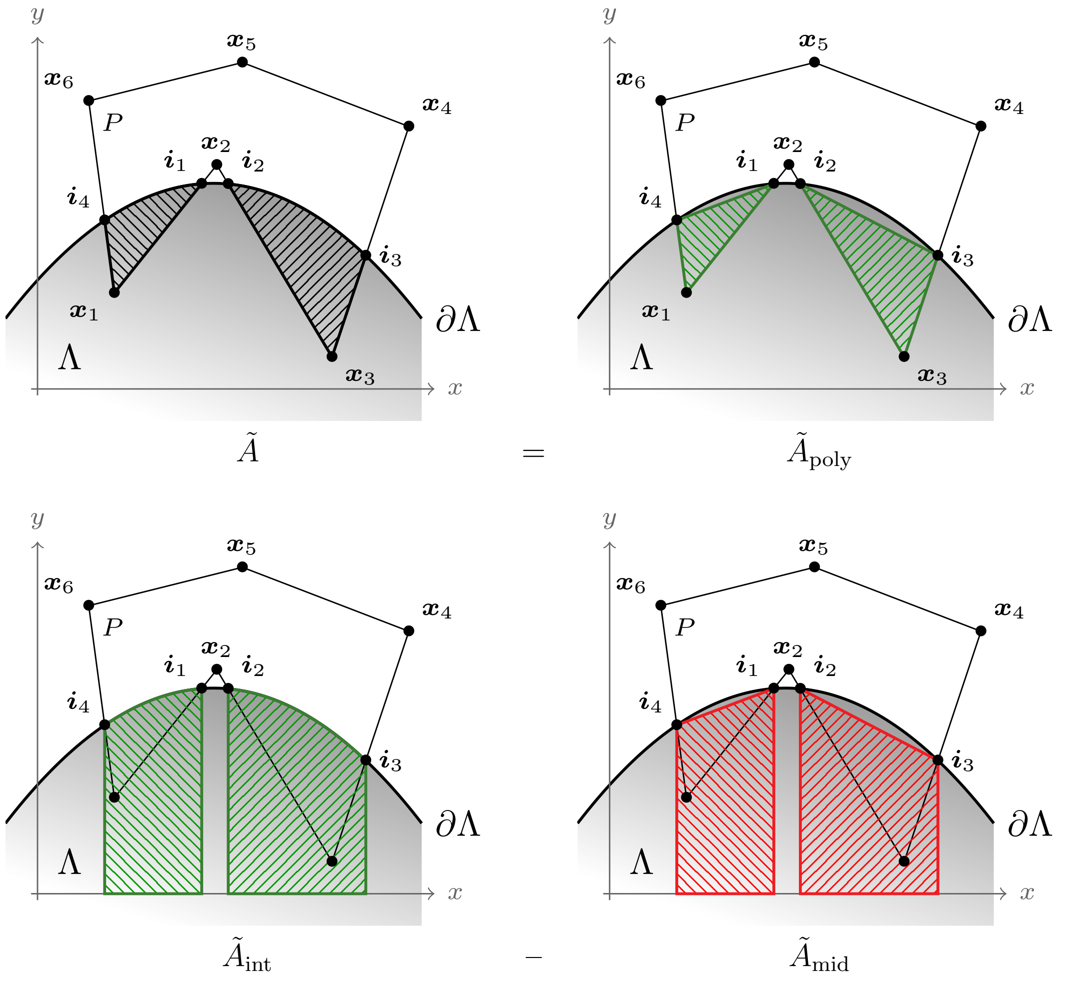


Figure 4: 3-step computation of the area $\tilde{A} = \tilde{A}_{\text{poly}} + \tilde{A}_{\text{int}} - \tilde{A}_{\text{mid}} = \ell(P \cap \Lambda)$, where P is an arbitrary polygon, and Λ the subset $\{y \leq \tilde{f}(x), \forall x \in \mathbb{R}\}$ with $\tilde{f}(x)$ a quadratic polynomial.

With these definitions, the volume fraction $\tilde{\gamma}$ is given by

$$\tilde{\gamma} = \frac{\tilde{A}_{\text{poly}} + \tilde{A}_{\text{int}} - \tilde{A}_{\text{mid}}}{\ell(P)}. \tag{32}$$

The calculation of $\nabla\tilde{\gamma}$ and $\mathrm{H}_{\tilde{\gamma}}$ thus implies to differentiate the terms $\tilde{A}_{\text{poly}}$, $\tilde{A}_{\text{int}}$, and $\tilde{A}_{\text{mid}}$, with regards to the coefficients of the parabola. The key-steps of this derivation are described in Appendix C.
Note that a generalisation of the present approach to arbitrary implicit curves $f(x, y) = 0$ is used to initialise the volume fraction fields considered in the examples and test-cases of this paper. The main difference between the approach presented in this section and its generalisation to arbitrary functions lies in the calculation of the term $(\tilde{A}_{\text{int}} - \tilde{A}_{\text{mid}})$, as the integral in equation (30) may not have an obvious analytical solution in the latter case. The details of this modification are provided in Appendix D.

3.3. Resolution of the minimisation problem ($\mathcal{P}$)

A Newton method is employed for the minimisation of J (see algorithm 1).

Algorithm 1: Newton minimisation of J

```
$k$                              $\longleftarrow$  1
$k_{\max}$                       $\longleftarrow$  Maximum number of iterations
$\epsilon, \partial$             $\longleftarrow$  Small positive constants
$\mathbf{a}_k$                   $\longleftarrow$  Initial guess $\mathbf{a}_0$
$J_k, \nabla J_k, \mathrm{H}_{J_k}$  $\longleftarrow$  ComputeCostF&Derivatives($\mathbf{a}_0$)

while $|J_k| < \epsilon$ and $|\nabla J_k| < \partial$ and $k < k_{\max}$ do
    $\boldsymbol{p}_k \longleftarrow -\mathrm{H}_{J_k}^{-1} \nabla J_k$    Compute descent direction
    $\alpha_k \longleftarrow$ ComputeStep($\mathbf{a}_k, \boldsymbol{p}_k$)    Compute step
    $\mathbf{a}_{k+1} \longleftarrow \mathbf{a}_k + \alpha_k \boldsymbol{p}_k$    Update coefficients
    $J_{k+1}, \nabla J_{k+1}, \mathrm{H}_{J_{k+1}} \longleftarrow$ ComputeCostF&Derivatives($\mathbf{a}_{k+1}$)    Update cost function and derivatives
    $k \longleftarrow k + 1$    Update counter
```

Let $\mathbf{a}$ be the vector of coefficients of the parabola,

$$\mathbf{a} = \begin{bmatrix} \mathrm{a} & \mathrm{b} & \mathrm{c} \end{bmatrix}^T. \tag{33}$$

The Newton minimisation method is based on the second-order Taylor series approximation to J, *i.e.*

$$J(\mathbf{a} + \boldsymbol{p}) \simeq J(\mathbf{a}) + \boldsymbol{p}^T \nabla J + \frac{1}{2} \boldsymbol{p}^T \mathrm{H}_J \boldsymbol{p}, \tag{34}$$

with $|\boldsymbol{p}| \ll |\mathbf{a}|$. This quadratic approximation of J near $\mathbf{a}$ will reach an extremum for

$$\nabla J + \mathrm{H}_J \boldsymbol{p} = \mathbf{0}. \tag{35}$$

The Newton direction of minimisation $\boldsymbol{p}$ is thus solution of the equation (35). Here, H_J is a 3×3 matrix so it can easily be inverted directly. The vector $\boldsymbol{p}$, however, is a descent direction provided that H_J is positive definite. This may not be the case, especially if $\mathbf{a}$ is far from a local minimiser $\mathbf{a}_\star$. For this reason, equation (35) is solved using a modified Cholesky decomposition [24, 25] that guarantees $\boldsymbol{p}$ to be a descent direction. Special attention must also be paid while updating the coefficients of the parabola in algorithm 1. If the updated parabola leaves one of the polygonal cells, this cell will not contribute to ∇J anymore, making it very unlikely that the parabola re-enters the cell during a future iteration. This, obviously, can prevent the problem from converging towards a local minimum. In order to avoid such situations, the update of the parabola coefficients follows

$$\mathbf{a}_{k+1} = \mathbf{a}_k + \alpha_k \boldsymbol{p}_k, \tag{36}$$

where α_k is taken as

$$\alpha_k = \min(1, \alpha_{\text{out}}), \tag{37}$$

α_{out} being the smallest positive step for which $\mathbf{a}_{k+1}$ leaves one of the polygonal cells.
Although it is possible to prove the existence and uniqueness of a zero minimum for J in the example provided in section 2, such a proof is not possible in the general case. In practice, using algorithm 1 for the resolution of ($\mathcal{P}$) with $N = 3$ will, in the vast majority of cases, converge towards a zero minimum.

4. Evaluation of curvature from local parabolic reconstructions

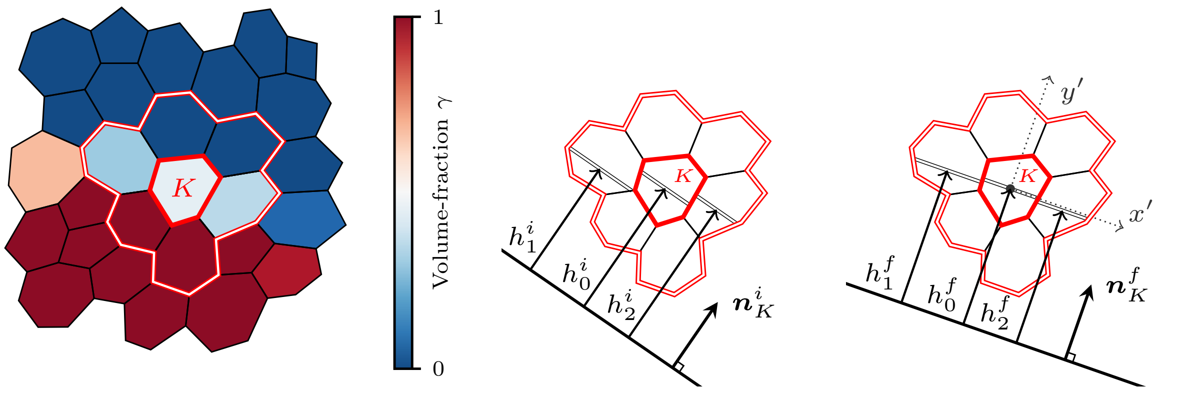
Before entering the details of the present curvature evaluation method, the following denominations are introduced:

- An "interfacial cell" is a cell whose volume fraction is strictly enclosed between 0 and 1. For the rest of this paper, a cell is considered to be interfacial if its volume fraction satisfies the inequalities $10^{-6} < \gamma < 1 - 10^{-6}$.
- The "nearest neighbourhood" of a cell K is defined as the ensemble of cells which share at least an edge or a vertex with cell K, as illustrated in figure 5a.

- The "direct neighbourhood" of a cell K is defined as the ensemble of cells which share at least an edge with cell K, and is a subset of the nearest neighbourhood of K.

4.1. Computation of the interface normals

The present method computes curvature from local parabolic reconstructions. Prior to the computation of these reconstructions, interface normals are to be evaluated as they determine the orientation of the local coordinate systems in which to solve the parabolic reconstruction problems. A first approximation of the normals is obtained from the gradient of the volume fraction field, that is $\boldsymbol{n}^i = \nabla\gamma/|\nabla\gamma|$ (the subscripts i and f are used to label the initial and final normal estimates). These normals are then corrected using the second-order iterative method of Ito et al. [16]. This procedure relies on the definition of local heights which are used to adjust the orientation of the interface normals. Consider, for instance, an interfacial cell K with an initial normal guess $\boldsymbol{n}^i_K$. The normal $\boldsymbol{n}^i_K$ is used to compute piecewise-linear approximations of the interface in K and in the interfacial cells of its nearest neighbourhood, based on the local values of volume fraction. Having defined a baseline reference orthogonal to $\boldsymbol{n}^i_K$, the height of fluid in each of these cells can be taken as the signed distance between the linear reconstruction and the reference line, as illustrated in figure 5b. If h_0 is the height associated with the cell K and $h_j, j \in \{1, n\}$, the heights associated with the n interfacial cells in the nearest neighbourhood of K, the normal $\boldsymbol{n}^i_K$ is rotated as to minimise the absolute differences $|h_0 - h_j|$. The procedure is repeated until convergence is reached. For details on how to calculate the amount of rotation needed at each step of the minimisation process, the reader is referred to the work of Ito et al. [16]. The final converged result $\boldsymbol{n}^f_K$ obtained from the initial setup of figure 5b is shown in figure 5c.



(a) Cell K and its nearest neighbourhood, delimited by a red and white double line

(b) Local interface heights in the reference frame of the initial normal guess $\boldsymbol{n}^i_K$

(c) Local interface heights in the reference frame of the final normal $\boldsymbol{n}^f_K$ after the iterative correction procedure [16]

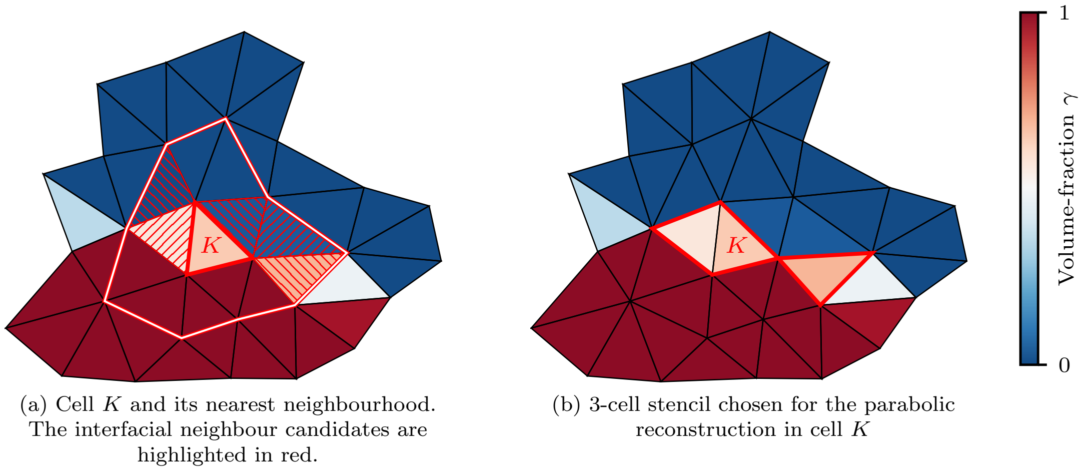
Figure 5: Illustration of the nearest neighbourhood of a given cell K and initial/final setups associated with the correction of the interface normal in K. The local piecewise-linear reconstructions in the interfacial cells are represented by the black and white double lines.

4.2. Choice of a local stencil for the parabolic reconstruction

Once the interface normals have been computed, two neighbours are chosen within the nearest neighbourhood of each interfacial cell in order to form a 3-cell stencil in which the parabolic reconstruction problem is to be solved. Three configurations can then be encountered, as the nearest neighbourhood of an interfacial cell K can contain:

(a) less than two interfacial cells,
(b) exactly two interfacial cells,
(c) more than two interfacial cells.

Case (a) is peculiar and should, in principal, almost never occur. It typically indicates that the interface is locally very poorly resolved. In such a case, the parabolic reconstruction is not computed for cell K, and the curvature is interpolated from neighbouring cells (as explained in section 4.4). Case (b) leads to an obvious and simple choice: the two neighbour interfacial cells which, with the cell K under consideration, form the 3-cell stencil. Case (c), finally, requires to select the 'best' two candidates within the set of neighbouring interfacial cells, which will be used to form the 3-cell stencil. An exemple of such a case is given in figure 6a. The best two candidates are chosen to be the neighbours whose volume fractions are closest to 0.5, as shown in figure 6b.



(a) Cell K and its nearest neighbourhood. The interfacial neighbour candidates are highlighted in red.

(b) 3-cell stencil chosen for the parabolic reconstruction in cell K

Figure 6: Choice of a local stencil for the parabolic reconstruction in cell K: the available interfacial candidates are highlighted in (a), and the final 3-cell stencil is shown in (b).

The 3-cell stencil, in which the parabolic reconstruction problem is solved, is written in a local orthogonal coordinate system (x', y') whose origin is the center of cell K (see exemple in figure 5c). The local coordinate system is scaled so that the area of the smallest stencil cell is equal to unity, and it is oriented such that the local interface normal is given by $\boldsymbol{n'} = [\,0\;1\,]^T$ in this new frame of reference. Note that the initial guess $\boldsymbol{a}_0$ in algorithm 1 is taken as the parabola that goes through the centres of the piecewise-linear interface reconstruction segments in each stencil cell.

4.3. Convergence and cell merging

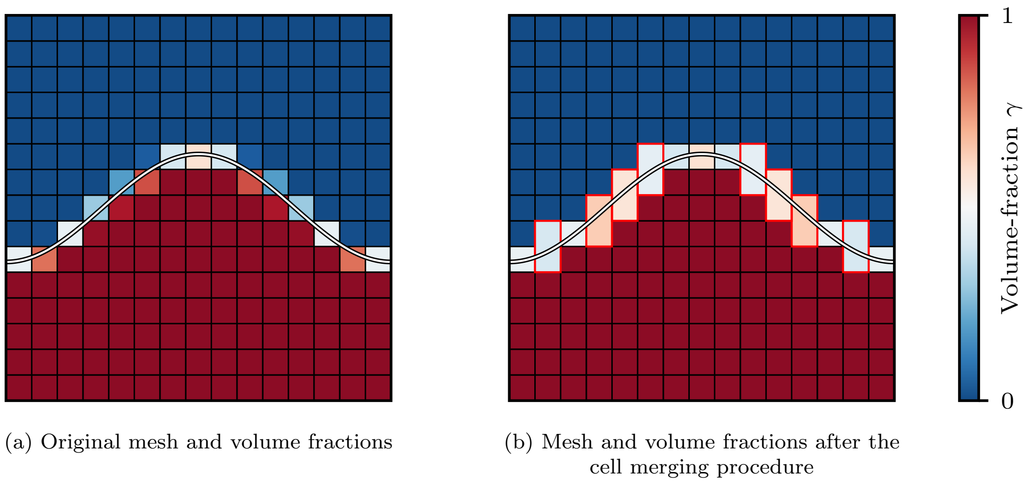
In practice, the convergence of the local minimisation problem ($\mathcal{P}$) largely depends on the values of the volume fractions in the stencil cells. If γ is very close to 0 or 1 (*i.e.* the cell is almost 'empty' or almost 'full') in some of the stencil cells, then algorithm 1 may struggle to converge, mainly due to numerical approximations. In order to improve convergence and avoid such cases, interfacial cells which are almost 'empty' are merged with cells which are almost or completely 'full', and vice-versa, in an attempt to bring the volume fractions of interfacial cells as close as possible to 0.5. This merging procedure is achieved in the following manner:

1. In a first step, a "merging candidate" C is attributed to each cell K of the computational mesh.
   - If K is an interfacial cell and its volume fraction is such that $\gamma_K < 0.1$ or $\gamma_K > 0.9$ (the cell is almost empty or almost full), then a potential candidate for merging is searched within its direct neighbourhood. Merging K with a direct neighbour N results in the volume fraction

     $$\gamma_{K\cup N} = \frac{\gamma_K \ell(K) + \gamma_N \ell(N)}{\ell(K \cup N)} \,. \tag{38}$$

     A direct neighbour N is considered a suitable candidate if $|0.5 - \gamma_K| > |0.5 - \gamma_{K\cup N}|$ and $|0.5 - \gamma_N| > |0.5 - \gamma_{K\cup N}|$, *i.e.* if the merging of N and K results in a volume fraction that is closer to 0.5 than it is in both K and N. Of all suitable candidates in the direct neighbourhood of K, the merging candidate C is chosen to be the one resulting in the smallest value of $|0.5 - \gamma_{K\cup N}|$.
   - If K is not an interfacial cell, or no suitable direct neighbours can be found, then C is chosen to be the cell K itself (*i.e.* K does not need to be changed).
2. In a second step, it is decided whether to merge cells with the attributed candidates or not. If the candidate C associated with a given cell K is the cell K itself, then nothing is done. If C is different than K, and the candidate associated with C is the cell K or C itself, then both cells are merged. In all other cases, the cells are not merged.

The application of the merging procedure is illustrated in figure 7.



(a) Original mesh and volume fractions

(b) Mesh and volume fractions after the cell merging procedure

Figure 7: Cell merging procedure: interfacial cells which are almost 'empty' are merged with cells which are almost or completely 'full', and interfacial cells which are almost 'full' are merged with cells which are almost or completely 'empty'. Original cells which have been merged together are highlighted in red in figure (b). The exact interface from which the original volume fractions have been computed is represented by the double black and white line.

4.4. Curvature integration

Once the local parabolic reconstruction has been computed for a given cell K, curvature is integrated on the portions of the parabola inside K using a Legendre-Gauss quadrature rule. The details of this procedure can be found in Appendix D. The integration of curvature using (D.2) is preferred to its computation from a single point on the parabola or directly from the parabola coefficients. It is robust as it naturally deals with complex configurations, such as cells containing multiple distinct portions of parabola, and it is also consistent with the VOF framework, because the integrated curvature value is coherent with the finite-volume formalism and in phase with the essence of volumetric surface-tension treatment (*e.g.* the Continuum Surface Force model [8]).
If the parabolic reconstruction problem has not converged in cell K, then it is checked whether the parabolas issued from the neighbouring cells in which convergence has been reached, intersect with cell K. If that is the case, then curvature is averaged from the integrals on all intersecting neighbouring parabolas. If all of these attempts have failed, curvature is interpolated from the neighbours. In practice, the method very rarely resorts to this last options. The few occasions for which this is the case are linked to poorly-resolved interfaces where the local parabolic reconstruction problem sometimes fails to converge. Overall, in the test-cases presented in this paper, about 0.6% of the interfacial cells where $1/\kappa\Delta x < 4$ (poorly-resolved interfaces) see the parabolic reconstruction fail and can not find an intersecting parabola from their neighours, while this is the case for 0.0006% of the interfacial cells with $1/\kappa\Delta x \geq 4$.

4.5. Algorithm summary

The present curvature evaluation method is summarised in algorithm 2. Figure 8 shows the curvature fields of an ellipse with maximum curvature κ_{max}, obtained on three different mesh types (Cartesian, triangular, and polygonal mesh), and various mesh resolutions ($1/\kappa_{\text{max}}\Delta x \simeq 1$, $1/\kappa_{\text{max}}\Delta x \simeq 2.5$, and $1/\kappa_{\text{max}}\Delta x \simeq 5$). The local parabolic reconstructions are shown in red. Noticeably, one can see that the method performs well at very low resolution ($1/\kappa_{\text{max}}\Delta x \simeq 1$). In fact, an additional method is not required to compute the curvature of under-resolved interfaces (as it is the case for the classic height-function method).

5. Analysis of curvature errors and computational cost of the method

The errors associated with the present curvature evaluation method are studied in order to assess its performance against existing methods as well as its convergence behaviour. Three interface shapes are considered in this study:

(a) A circle with radius r whose exact curvature is $\kappa_{\text{ref}} = 1/r$ (figure 9a).
(b) An ellipse with semi-minor axis a and semi-major axis b whose maximum curvature is given by $\kappa_{\text{ref}} = b/a^2$ (figure 9b).

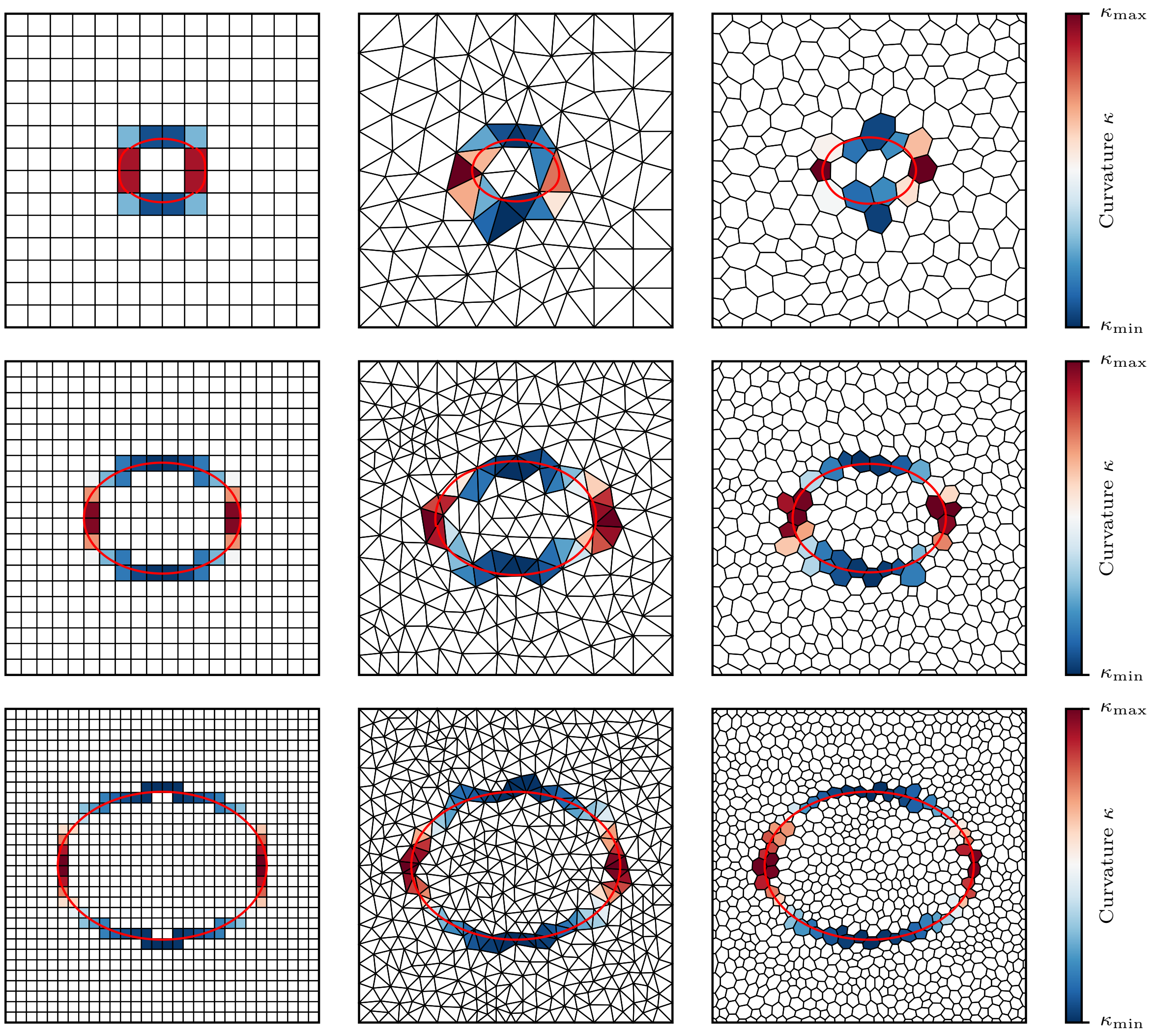
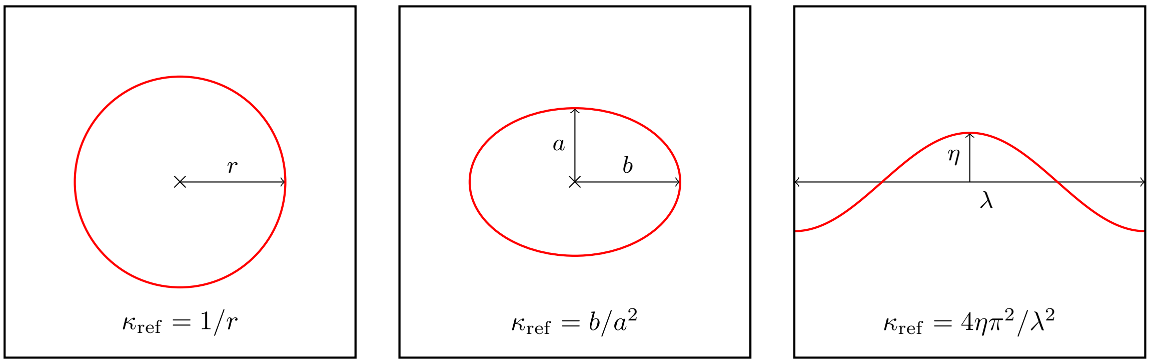


Figure 8: Curvature fields of an ellipse using the present method on: (left column) a Cartesian grid; (middle column) a triangular mesh; (right column) a polygonal mesh; and with: (top row) $1/(\kappa_{\max}\Delta x) \simeq 1$; (middle row) $1/(\kappa_{\max}\Delta x) \simeq 2.5$; (bottom row) $1/(\kappa_{\max}\Delta x) \simeq 5$. The local parabolic reconstructions are shown in red.

Algorithm 2: Curvature evaluation from parabolic reconstructions

```
Compute interface normals
Merge problematic cells
for each interfacial cell do
    Generate local stencil
    Solve parabolic reconstruction problem in local coordinate system
for each interfacial cell do
    if parabolic reconstruction has converged then
        Integrate curvature from parabola
    else
        if parabolas from neighbours intersect current cell then
            Integrate curvature from neighbour parabolas
for each interfacial cell do
    if curvature evaluation has failed then
        Interpolate curvature from neighbours
```

(c) A sine wave with amplitude η and wavelength λ whose maximum absolute curvature is given by $\kappa_{\text{ref}} = 4\eta\pi^2/\lambda^2$ (figure 9c). The ratios $\eta/\lambda = 0.25$ and $\eta/\lambda = 0.01$ are used.



r
$\kappa_{\text{ref}} = 1/r$
a
b
$\kappa_{\text{ref}} = b/a^2$
η
λ
$\kappa_{\text{ref}} = 4\eta\pi^2/\lambda^2$

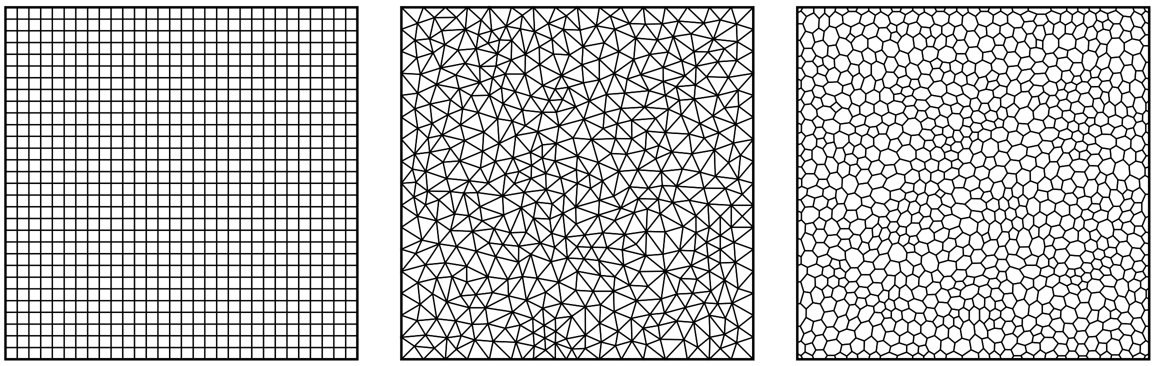
(a) Circle (b) Ellipse (c) Sine wave

Figure 9: Interfaces considered in the convergence study, and their reference curvature κ_{ref}.

All tests are conducted on three different types of meshes:

(a) A Cartesian grid with constant mesh spacing (figure 10a).
(b) A triangular mesh generated using the `Triangle` mesh library of Shewchuk [26] (figure 10b).
(c) A polygonal mesh generated using the dual polygonal mesh generation method of Balafas [27] (figure 10c).

The domain considered for all cases is a 1×1 square, and the mesh spacing is defined as $\Delta x = \sqrt{1/N}$, where N is the number of cells in the mesh.



(a) Cartesian grid (b) Triangular mesh (c) Polygonal mesh

Figure 10: Types of meshes considered for the convergence study.

In all numerical experiments, the volume fraction field is initialised using a modified version of the method described in section 3.2, for which $\tilde{A}_{\text{int}}$ is evaluated using a 64-point Legendre-Gauss quadrature rule, as explained in Appendix D. The total amount of interfacial cells is N_c. For all interfacial cells, the exact curvature κ_{exact} is obtained by integrating the curvature of the portions of interface contained within the cell, again using a 64-point Legendre-Gauss quadrature rule. Two absolute curvature error norms are considered:

$$\|\kappa - \kappa_{\text{exact}}\|_\infty = \max_{s\in\{1,\ldots,N_s\}} \max_{i\in\{1,\ldots,N_c\}} |\kappa_{i,s} - \kappa_{\text{exact},i,s}|, \tag{39}$$

$$\|\kappa - \kappa_{\text{exact}}\|_2 = \frac{1}{N_s}\sum_{s=1}^{N_s}\sqrt{\frac{\sum_{i=1}^{N_c}\left(\kappa_{i,s} - \kappa_{\text{exact},i,s}\right)^2}{N_c}}, \tag{40}$$

where N_s is the number of simulations performed. Each simulation is executed with random center position for the case of the circle and ellipse, and random vertical position for the sine wave. All curvature

errors are normalised by the reference curvature κ_{ref}, and are the results of $N_{\mathrm{s}} = 100$ tests for each resolution. The level of refinement of the interface is quantified by $1/(\kappa_{\mathrm{ref}}\Delta x)$ which accounts for the number of cells along the smallest radius of curvature of the interface. The present method is compared against least-square differentiation on a convoluted volume fraction field following the method of Denner and van Wachem [12] (CV method), a reconstructed-distance-function method based on Cummins et al. [11] and its unstructured extension from Ito et al. [14] (RDF method), and an in-house implementation of the mixed height-function method of Popinet [19] (HF method). Finally, in all error plots, the marks are representative of the type of mesh which is employed: -■- Squares – the mesh is a Cartesian grid, -▲- Triangles – the mesh is made of triangular cells, -●- Circles – the mesh is made of arbitrary polygonal cells. Figures 11 and 12 respectively show the $\| \cdot \|_\infty$ and $\| \cdot \|_2$ norms of the curvature errors for a circle using the CV, RDF, HF, and present methods. Both the CV and RDF methods diverge with order one, as shown by Raessi et al. [15], while the HF method converges with second-order accuracy, as expected from theory [18] and shown in numerous numerical experiments [11, 19, 20]. The present method also converges with second-order accuracy for both the $\| \cdot \|_\infty$ and $\| \cdot \|_2$ norms, and that is for all types of meshes (Cartesian, triangular, polygonal). The errors associated with unstructured meshes (triangular and polygonal) are sensibly identical to those obtained on the Cartesian grid, which themselves are of the same order of magnitude as the HF method errors. Figures 13 and 14 show the curvature errors on an ellipse using the CV, RDF, HF, and present methods. Both the $\| \cdot \|_\infty$ and $\| \cdot \|_2$ norms of the errors for the HF and present methods converge with first-order accuracy, which is to be expected in this case. Indeed, the HF method is proven to be second-order accurate with regards to the curvature at the point of intersection between the interface and the symmetry axis of the heights stencil. This quantity, itself, is a first-order approximation of the integral of curvature in the middle column. Second-order convergence is achieve for the circle case because curvature is then constant, but as soon as the gradient of the exact curvature is different than zero, one should expect first-order convergence. Figures 15, 16 and 17, 18 show the errors for the sine wave cases with $\eta/\lambda = 0.25$ and $\eta/\lambda = 0.01$, respectively. Here again, the present method shows first-order convergence with mesh refinement on all mesh types, and so does the HF method. The case with $\eta/\lambda = 0.01$, in figures 17 and 18, is the only case for which the triangular and polygonal meshes perform significantly worse than the Cartesian grid. With the ratio $\eta/\lambda = 0.01$, the interface is almost horizontal which, on a Cartesian grid, means that the parabolic reconstruction problem is essentially well-posed everywhere (in the sense defined in section 2). It then makes sense that the Cartesian grid performs somewhat better than the unstructured meshes in this example. Figure 19 shows the curvature fields obtained for the $\eta/\lambda = 0.01$ case on all mesh types, for a resolution $1/\kappa_{\mathrm{ref}}\Delta x \simeq 75$, and using the present, RDF, and CV methods. At this relatively coarse resolution ($\lambda \simeq 30\Delta x$), the CV and RDF methods already show large visible errors.
Finally, the computational cost of the proposed method is assessed on a specific test case: a circular interface with $1/(\kappa_{\mathrm{ref}}\Delta x) = 100$. Table 1 shows the timings (in μs per interfacial cell) associated with the present and HF methods, as well as the average number of iterations required to reach convergence when computing the local parabolic reconstructions. The timings of the present method are broken down into three components for: the merging procedure, the computation of the interface normals following Ito et al. [16], and the curvature evaluation itself (*i.e.* the computation of the parabolic reconstructions and the integration of curvature in each interfacial cell). These results reveal two things. First, the average number of iterations required to reach convergence is low and similar on all types of meshes ($\simeq 3$). This means that reaching convergence for the parabolic fit is not more difficult on the unstructured meshes than it is on a Cartesian grid. Second, the timings show that the present method has a computational cost which is about two orders of magnitude greater than the HF method, known for its simplicity and low cost. All in all, this ratio is relatively good when one considers the iterative nature of the present method, and its performance on unstructured meshes. In practice, the computational expense associated with the present method is several orders of magnitude smaller than what is typically required to solve the Navier-Stokes equations on a given mesh, making it a viable option for the evaluation of curvature in interfacial flow simulations.

6. Conclusion

This paper proposes a method for the estimation of curvature from volume fractions on two-dimensional meshes of any type, which is based on local parabolic reconstructions. Each parabolic reconstruction requires to solve a local minimisation problem using an iterative Newton method. The proposed curvature evaluation method is proven to be equivalent to the height-function method [11] for Cartesian configurations with consistent heights. Numerical experiments show that it converges with mesh refinement with

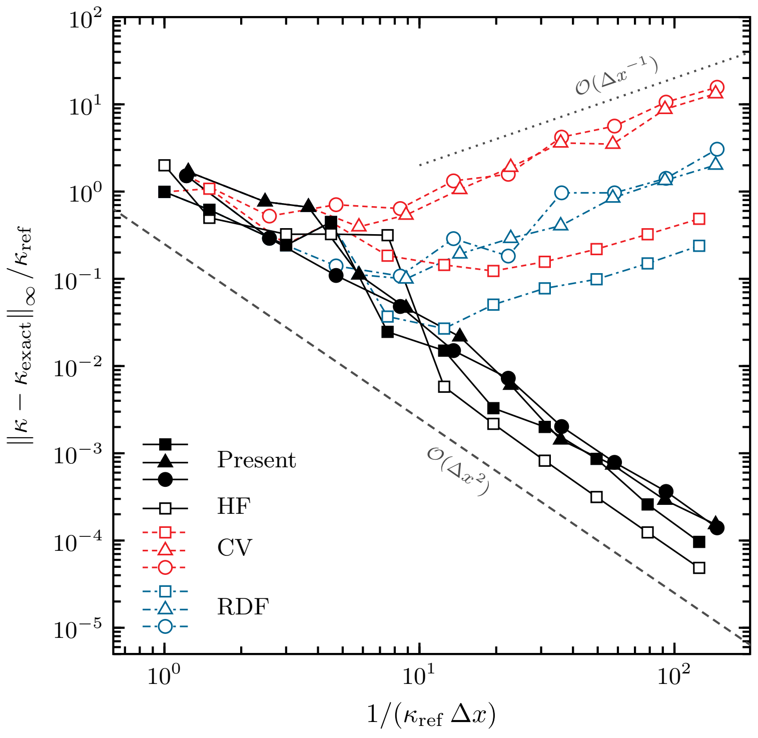


Figure 11: Maximum curvature errors for the circle case as a function of mesh refinement following the convoluted volume fraction method (CV), the reconstructed-distance-function method (RDF), the height-function method (HF), and the method proposed in this paper (Present). The values are normalised by $\kappa_{\mathrm{ref}} = 1/r$. For each mesh resolution, the errors correspond to the maximum values reached over 100 simulations with random center position. Squares, triangles, or circles, respectively relate to Cartesian, triangular, or polygonal meshes.

Mesh type	**HF method**	**Present method**				
		Merging procedure	Normals evaluation	Curvature evaluation	Total time	Average # of iterations
□	2.1	6.2	61.1	36.5	103.8	2.96
△	–	6.0	113.9	37.4	157.3	3.17
○	–	7.9	126.1	55.2	189.2	3.05

Table 1: Computational time spent for each interfacial cell (in μs), and average number of iterations required to solve the local parabolic reconstruction problems, for the circle test case with $1/(\kappa_{\mathrm{ref}}\Delta x) = 100$, using the present method on all three types of meshes, and the HF method on a Cartesian mesh.

the same order of accuracy as the height-function method, both on structured and unstructured meshes. Furthermore, the errors obtained on unstructured meshes are, in most cases, comparable to the errors obtained on Cartesian grids with similar resolutions. Such convergence properties on unstructured meshes, which have not yet been reached in the literature, open a new world of possibilities for the simulation of interfacial flows in complex geometries using the VOF method. In particular, one can finally consider using triangular or polygonal adaptive-mesh-refinement for such applications without being restricted by crippling parasitic flow currents at the interface. The proposed method can be extended to three-dimensional meshes based on the analogy shown in this paper between the height-function method and the parabolic reconstruction problem. This raises, however, additional challenges due to the increased complexity of the minimisation problem.

Appendix A. Intersections computation

Consider an edge $[\boldsymbol{x}_0\boldsymbol{x}_1]$ and a parabola $\partial\Lambda$ defined by $y = \tilde{f}(x) = \mathrm{a}\,x^2 + \mathrm{b}\,x + \mathrm{c}$. Two cases are to distinguish for the computation of the intersections:

- If $x_0 = x_1$ (*i.e.* $[\boldsymbol{x}_0\boldsymbol{x}_1]$ is vertical), then there exists a unique intersection between $\partial\Lambda$ and the line $(\boldsymbol{x}_0\boldsymbol{x}_1)$ which is given by

$$\boldsymbol{i}_0 = \begin{bmatrix} x_0 & \tilde{f}(x_0) \end{bmatrix}^T \tag{A.1}$$

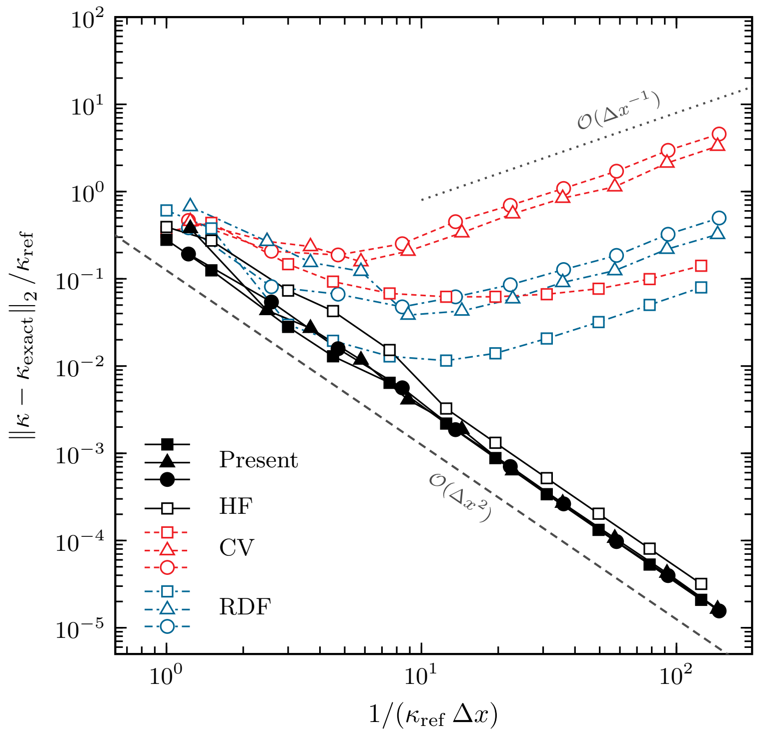


Figure 12: Root mean square of the curvature errors for the circle case as a function of mesh refinement following the convoluted volume fraction method (CV), the reconstructed-distance-function method (RDF), the height-function method (HF), and the method proposed in this paper (Present). The values are normalised by $\kappa_{\mathrm{ref}} = 1/r$. For each mesh resolution, the errors have been averaged from 100 simulations with random center position. Squares, triangles, or circles, respectively relate to Cartesian, triangular, or polygonal meshes.

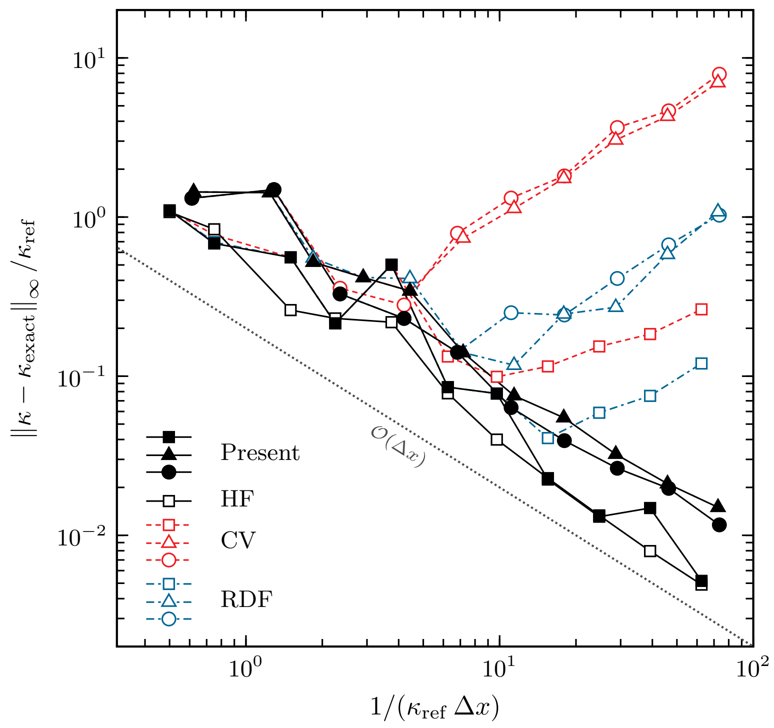


Figure 13: Maximum curvature errors for the ellipse case as a function of mesh refinement following the convoluted volume fraction method (CV), the reconstructed-distance-function method (RDF), the height-function method (HF), and the method proposed in this paper (Present). The values are normalised by $\kappa_{\mathrm{ref}} = b/a^2$. For each mesh resolution, the errors correspond to the maximum values reached over 100 simulations with random center position. Squares, triangles, or circles, respectively relate to Cartesian, triangular, or polygonal meshes.

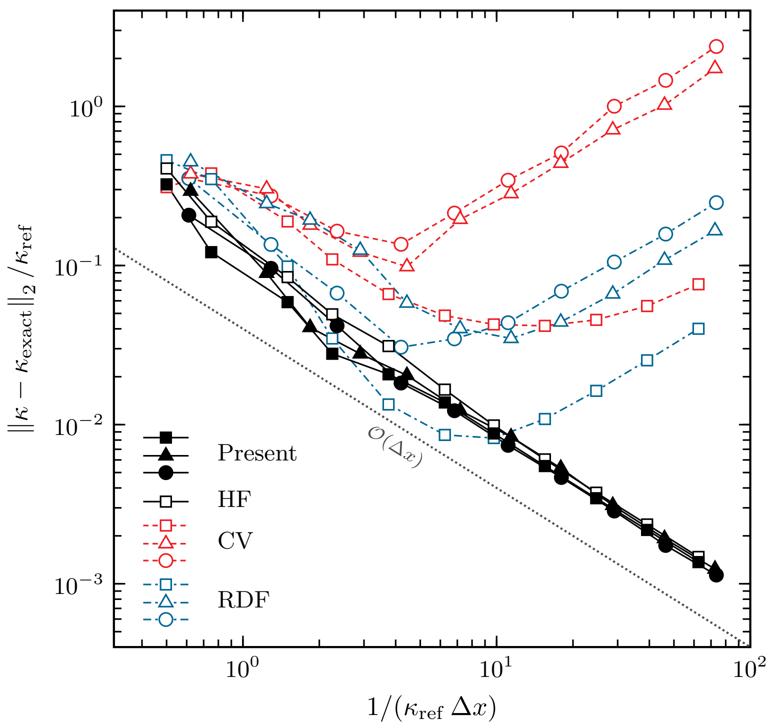


Figure 14: Root mean square of the curvature errors for the ellipse case as a function of mesh refinement following the convoluted volume fraction method (CV), the reconstructed-distance-function method (RDF), the height-function method (HF), and the method proposed in this paper (Present). The values are normalised by $\kappa_{\mathrm{ref}} = b/a^2$. For each mesh resolution, the errors have been averaged from 100 simulations with random center position. Squares, triangles, or circles, respectively relate to Cartesian, triangular, or polygonal meshes.

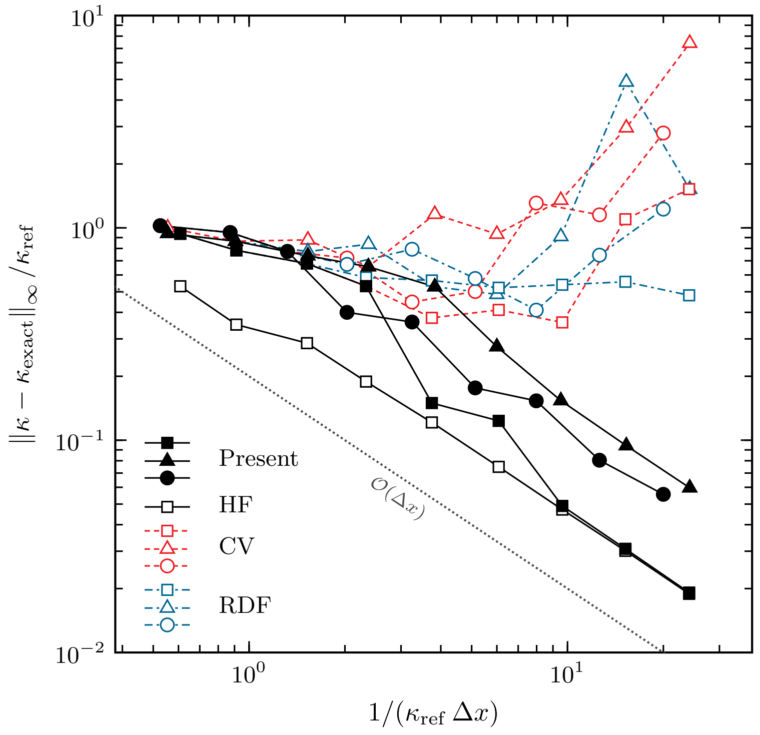


Figure 15: Maximum curvature errors for the sine wave case with $\eta/\lambda = 0.25$ as a function of mesh refinement following the convoluted volume fraction method (CV), the reconstructed-distance-function method (RDF), the height-function method (HF), and the method proposed in this paper (Present). The values are normalised by $\kappa_{\mathrm{ref}} = 4\eta\pi^2/\lambda^2$. For each mesh resolution, the errors correspond to the maximum values reached over 100 simulations with random vertical position. Squares, triangles, or circles, respectively relate to Cartesian, triangular, or polygonal meshes.

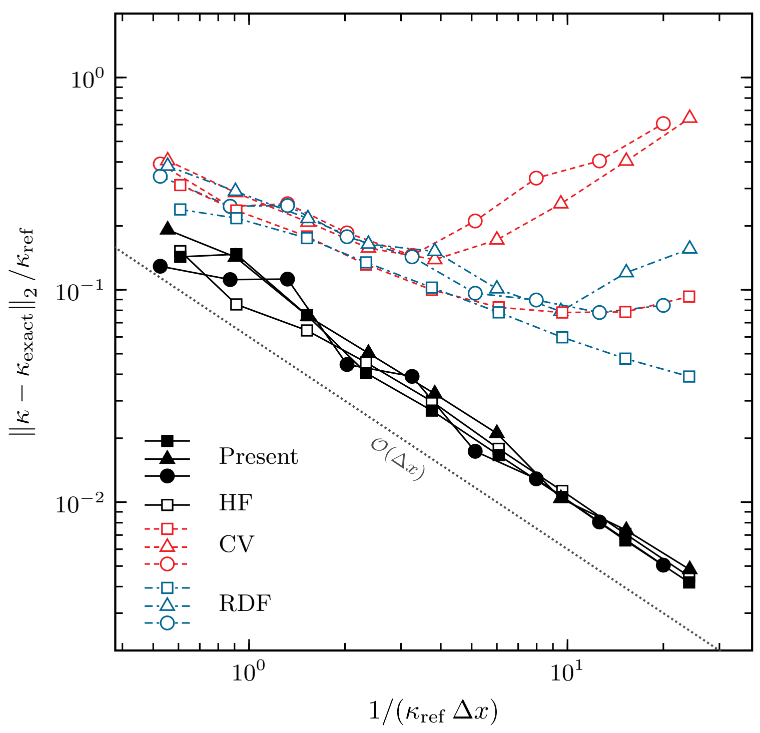


Figure 16: Root mean square of the curvature errors for the sine wave case with $\eta/\lambda = 0.25$ as a function of mesh refinement following the convoluted volume fraction method (CV), the reconstructed-distance-function method (RDF), the height-function method (HF), and the method proposed in this paper (Present). The values are normalised by $\kappa_{\text{ref}} = 4\eta\pi^2/\lambda^2$. For each mesh resolution, the errors have been averaged from 100 simulations with random vertical position. Squares, triangles, or circles, respectively relate to Cartesian, triangular, or polygonal meshes.

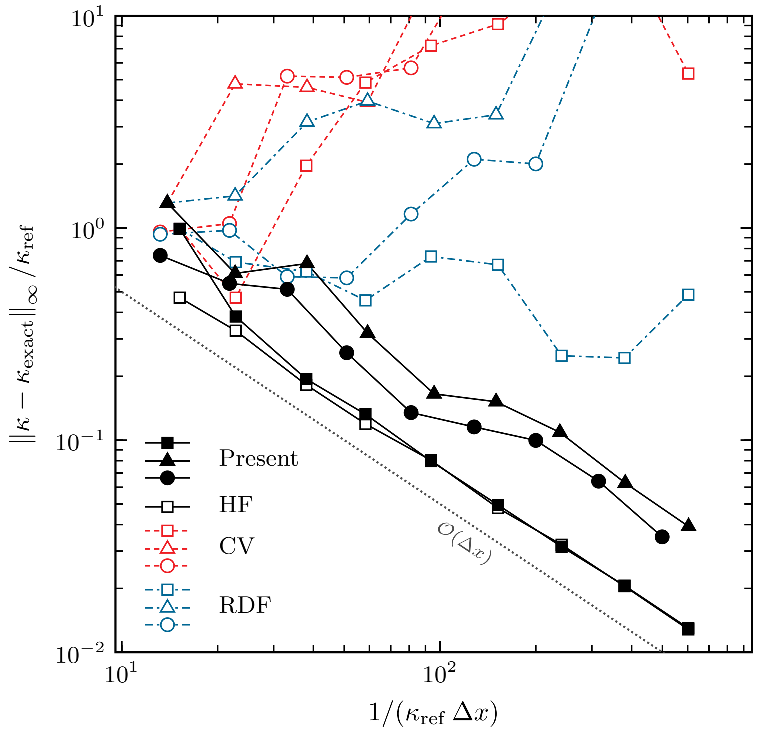


Figure 17: Maximum curvature errors for the sine wave case with $\eta/\lambda = 0.01$ as a function of mesh refinement following the convoluted volume fraction method (CV), the reconstructed-distance-function method (RDF), the height-function method (HF), and the method proposed in this paper (Present). The values are normalised by $\kappa_{\text{ref}} = 4\eta\pi^2/\lambda^2$. For each mesh resolution, the errors correspond to the maximum values reached over 100 simulations with random vertical position. Squares, triangles, or circles, respectively relate to Cartesian, triangular, or polygonal meshes.

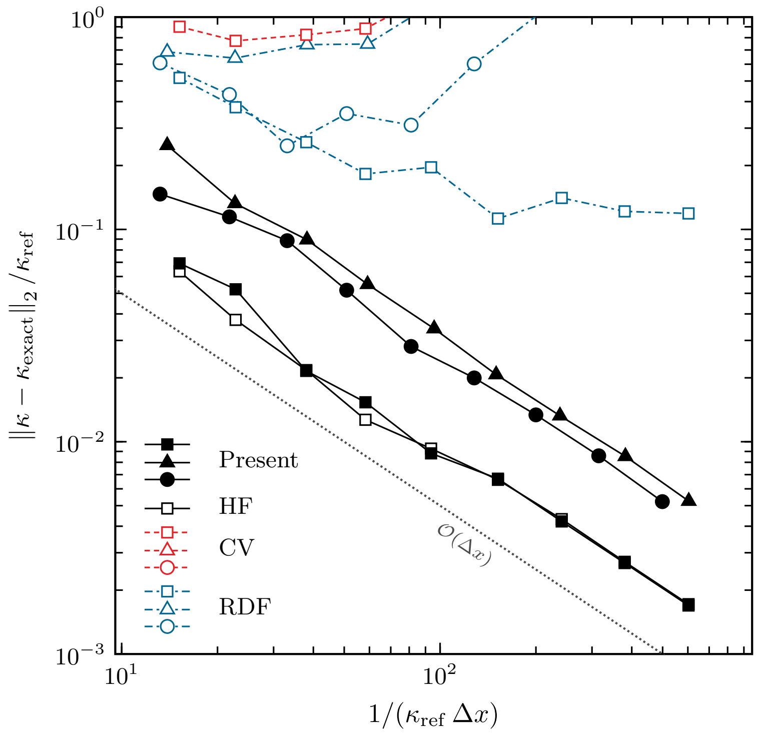


Figure 18: Root mean square of the curvature errors for the sine wave case with $\eta/\lambda = 0.01$ as a function of mesh refinement following the convoluted volume fraction method (CV), the reconstructed-distance-function method (RDF), the height-function method (HF), and the method proposed in this paper (Present). The values are normalised by $\kappa_{\mathrm{ref}} = 4\eta\pi^2/\lambda^2$. For each mesh resolution, the errors have been averaged from 100 simulations with random vertical position. Squares, triangles, or circles, respectively relate to Cartesian, triangular, or polygonal meshes.

Provided that $\min(y_0, y_1) \leq \tilde{f}(x_0) \leq \max(y_0, y_1)$, then $\boldsymbol{i}_0$ belongs to the edge $[\boldsymbol{x}_0\boldsymbol{x}_1]$ and its gradients and Hessian matrices with regards to the coefficients $(\mathrm{a}, \mathrm{b}, \mathrm{c})$ are given by

$$\nabla \boldsymbol{i}_{0x} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T, \quad \nabla \boldsymbol{i}_{0y} = \begin{bmatrix} x_0^2 & x_0 & 1 \end{bmatrix}^T, \quad \mathrm{H}_{\boldsymbol{i}_{0x}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathrm{H}_{\boldsymbol{i}_{0y}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{A.2}$$

- If $x_0 \neq x_1$ (*i.e.* $[\boldsymbol{x}_0\boldsymbol{x}_1]$ is not vertical), then $\partial\Lambda$ intersects the line $(\boldsymbol{x}_0\boldsymbol{x}_1)$ twice, once, or not at all. Let D and E be the coefficient of the line $(\boldsymbol{x}_0\boldsymbol{x}_1)$, that is

$$D = \frac{y_1 - y_0}{x_1 - x_0}, \tag{A.3}$$

$$E = y_0 - D\,x_0. \tag{A.4}$$

  - If $\mathrm{a} = 0$ (*i.e.* $\partial\Lambda$ is a line):
    * If $\mathrm{b} \neq D$ (*i.e.* $\partial\Lambda$ and $(\boldsymbol{x}_0\boldsymbol{x}_1)$ are not parallel), then $\partial\Lambda$ intersects $(\boldsymbol{x}_0\boldsymbol{x}_1)$ once and the intersection is given by

$$\boldsymbol{i}_0 = \begin{bmatrix} \dfrac{E - \mathrm{c}}{\mathrm{b} - D} & D\dfrac{E - \mathrm{c}}{\mathrm{b} - D} + E \end{bmatrix}^T \tag{A.5}$$

      Its gradients and Hessian matrices are given by

$$\nabla \boldsymbol{i}_{0x} = \begin{bmatrix} 0 & \dfrac{\mathrm{c} - E}{(D - \mathrm{b})^2} & \dfrac{1}{(D - \mathrm{b})} \end{bmatrix}^T, \quad \nabla \boldsymbol{i}_{0y} = D\,\nabla \boldsymbol{i}_{0x} \tag{A.6}$$

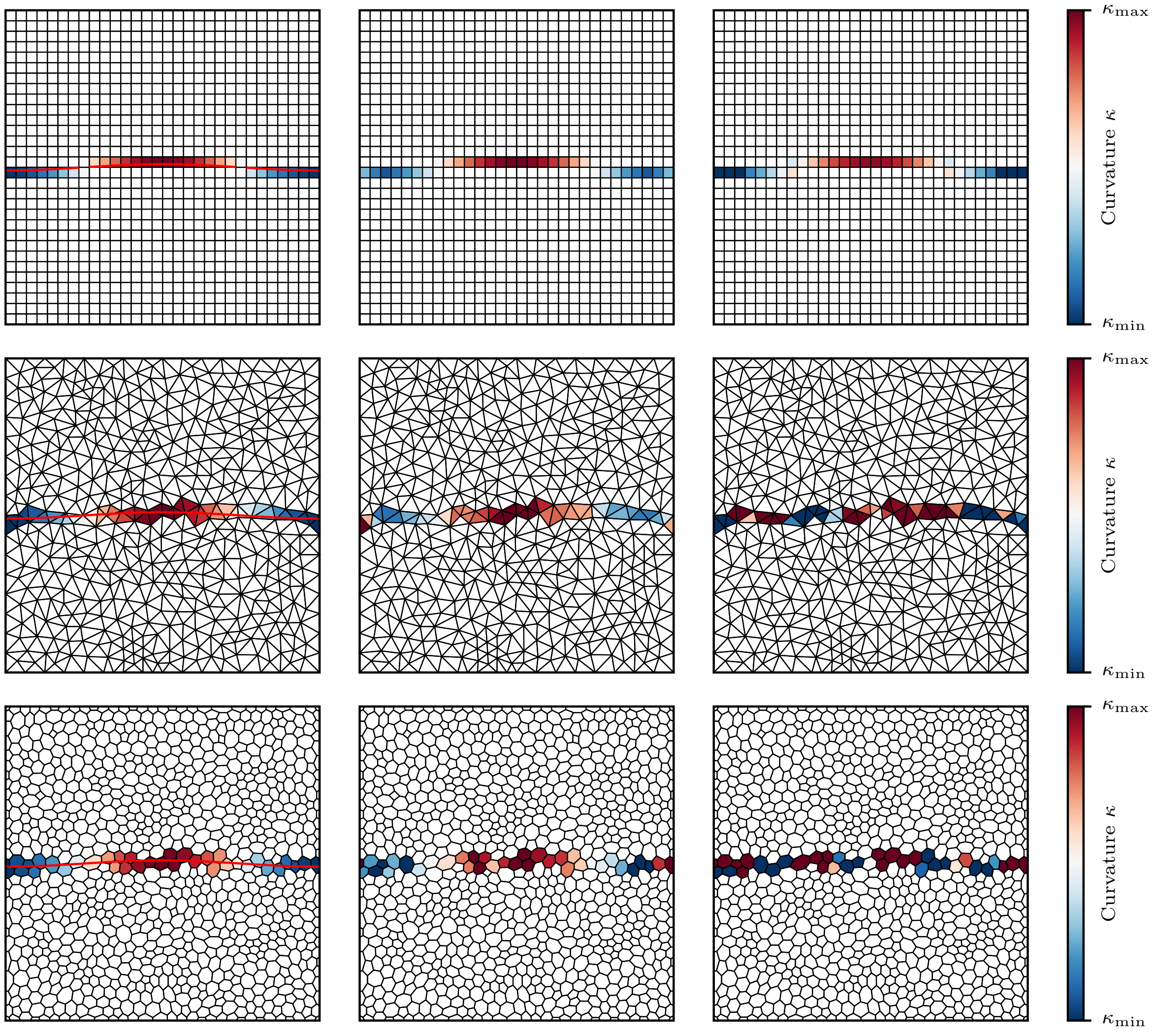


Figure 19: Curvature fields on a sine wave with $\eta/\lambda = 0.01$ and $1/\kappa_{\text{ref}}\Delta x \simeq 75$. The methods employed are: (left column): Present method; (middle column) RDF method; (right column): CV method. At this relatively coarse resolution, large errors are already noticeable for the CV and RDF methods.

$$\mathrm{H}_{\boldsymbol{i}_{0x}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \dfrac{E-\mathrm{c}}{2(D-\mathrm{b})^3} & \dfrac{1}{(D-\mathrm{b})^2} \\ 0 & \dfrac{1}{(D-\mathrm{b})^2} & 0 \end{bmatrix}, \quad \mathrm{H}_{\boldsymbol{i}_{0y}} = D\,\mathrm{H}_{\boldsymbol{i}_{0x}} \tag{A.7}$$

– If $\mathrm{a} \neq 0$ (*i.e.* $\partial\Lambda$ is a parabola): Let A, B, and C, be the coefficients of the quadratic equation one needs to solve to compute the intersection(s), and Δ the discriminant of this equation

$$A = \mathrm{a}, \tag{A.8}$$
$$B = \mathrm{b} - D, \tag{A.9}$$
$$C = \mathrm{c} - E, \tag{A.10}$$
$$\Delta = B^2 - 4AC. \tag{A.11}$$

* If $\Delta < 0$, then $\partial\Lambda$ does not intersect $(\boldsymbol{x}_0\boldsymbol{x}_1)$
* If $\Delta = 0$, then $\partial\Lambda$ intersects $(\boldsymbol{x}_0\boldsymbol{x}_1)$ once and thus the case is irrelevant for the purpose of this paper

* If $\Delta > 0$, then $\partial\Lambda$ intersects $(\boldsymbol{x}_0\boldsymbol{x}_1)$ twice and the intersections are given by

$$\boldsymbol{i}_0 = \begin{bmatrix} \frac{-B-\sqrt{\Delta}}{2A} & D\frac{-B-\sqrt{\Delta}}{2A}+E \end{bmatrix}^T, \quad \boldsymbol{i}_1 = \begin{bmatrix} \frac{-B+\sqrt{\Delta}}{2A} & D\frac{-B+\sqrt{\Delta}}{2A}+E \end{bmatrix}^T \tag{A.12}$$

Their gradients are given by

$$\nabla\boldsymbol{i}_{0x} = \begin{bmatrix} \frac{C}{A\sqrt{\Delta}} - \frac{-B-\sqrt{\Delta}}{2A^2} & \frac{-1-B/\sqrt{\Delta}}{2A} & \frac{1}{\sqrt{\Delta}} \end{bmatrix}^T, \quad \nabla\boldsymbol{i}_{0y} = D\,\nabla\boldsymbol{i}_{0x} \tag{A.13}$$

$$\nabla\boldsymbol{i}_{1x} = \begin{bmatrix} \frac{-C}{A\sqrt{\Delta}} - \frac{-B+\sqrt{\Delta}}{2A^2} & \frac{-1+B/\sqrt{\Delta}}{2A} & -\frac{1}{\sqrt{\Delta}} \end{bmatrix}^T, \quad \nabla\boldsymbol{i}_{1y} = D\,\nabla\boldsymbol{i}_{1x} \tag{A.14}$$

and Hessian matrices by

$$\mathrm{H}_{\boldsymbol{i}_{0x}} = \frac{1}{\Delta^{3/2}} \begin{bmatrix} \frac{F-B\Delta^{3/2}}{A^3} & \frac{\Delta^{3/2}+G}{2A^2} & 2C \\ \frac{\Delta^{3/2}+G}{2A^2} & 2C & -B \\ 2C & -B & 2A \end{bmatrix}, \quad \mathrm{H}_{\boldsymbol{i}_{0y}} = D\,\mathrm{H}_{\boldsymbol{i}_{0x}} \tag{A.15}$$

$$\mathrm{H}_{\boldsymbol{i}_{1x}} = \frac{1}{\Delta^{3/2}} \begin{bmatrix} \frac{-F-B\Delta^{3/2}}{A^3} & \frac{\Delta^{3/2}-G}{2A^2} & -2C \\ \frac{\Delta^{3/2}-G}{2A^2} & -2C & B \\ -2C & B & -2A \end{bmatrix}, \quad \mathrm{H}_{\boldsymbol{i}_{1y}} = D\,\mathrm{H}_{\boldsymbol{i}_{1x}} \tag{A.16}$$

with

$$F = 6AC(B^2 - AC) - B^4 \tag{A.17}$$

and

$$G = B\Delta - 2ABC \tag{A.18}$$

Appendix B. Modified Weiler-Atherton polygon clipping algorithm

Consider a polygon P with n vertices $\boldsymbol{x}_j$ which have been ordered anti-clockwise, and a parabola $y = \tilde{f}(x)$. The intersections between P and the parabola are computed by looping over the polygonal chain $\boldsymbol{x}_j, j \in \{1, \dots, n\}$. They are represented by the points $\boldsymbol{i}_k, k \in \{1, \dots, m\}$ in figure B.20a.

Algorithm 3: Weiler-Atherton polygon clipping algorithm

```
while intersection list is not empty do
    Start on polygon to clip
    Start at ‘entry’ intersection
    Add point to clipped polygon list
    Remove point from intersection list
    while next point is not starting point do
        Add next point to clipped polygon list
        if next point is an intersection then
            Remove from intersection list
            Switch to other polygon
```

In order for the Weiler-Atherton polygon clipping algorithm [23] to be employed, one needs to define two polygons: the polygon to clip, and the clipping region. The polygon to clip is given by P to which the intersections have been added. In the example of figure B.20a, it is the chain $\{\boldsymbol{x}_1, \boldsymbol{i}_1, \boldsymbol{x}_2, \boldsymbol{i}_2, \boldsymbol{x}_3, \boldsymbol{i}_3, \boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6, \boldsymbol{i}_4\}$.

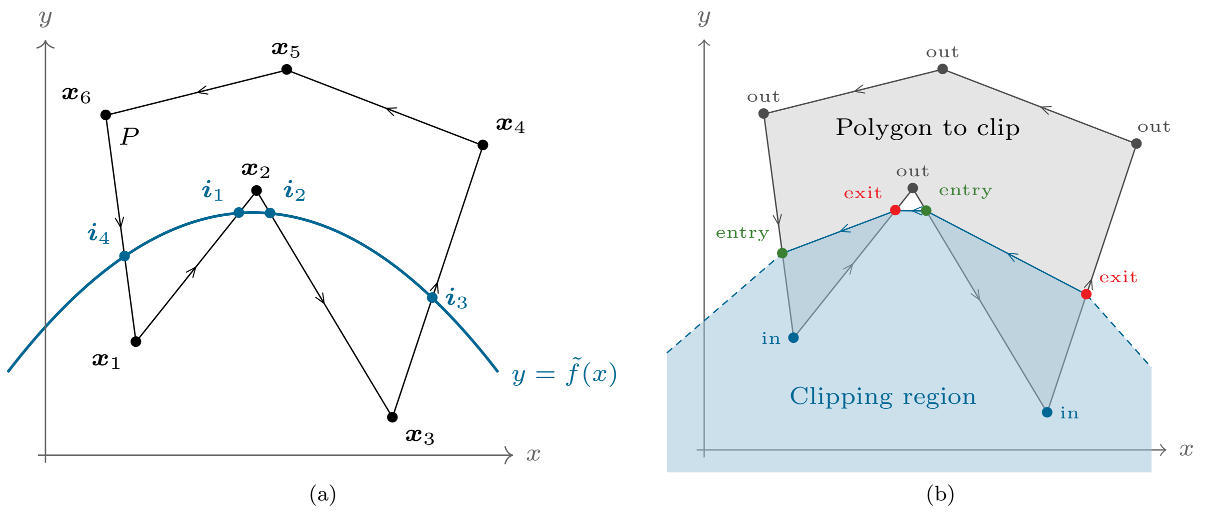


Figure B.20: Illustration of the polygon clipping process: (a) Original polygonal cell P and the parabola $y = \tilde{f}(x)$, with their intersections $\boldsymbol{i}_k$. (b) Polygon to clip and clipping region for the polygon clipping algorithm; the points have been given the flags 'in', 'out', 'entry', and 'exit'.

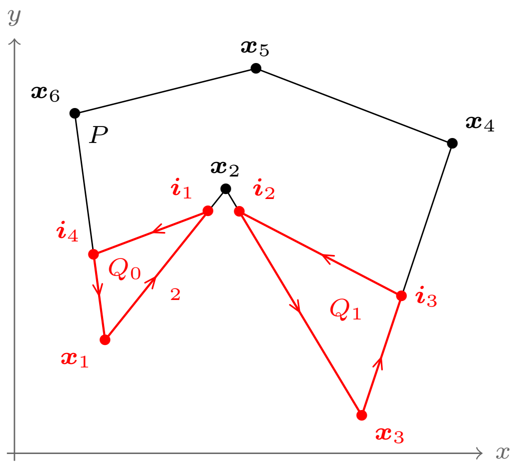


Figure B.21: Clipped polygons Q_0 and Q_1 resulting from the application of the polygon clipping algorithm to the example presented in figure B.20a.

The clipping region is the polygonal approximation of the region below the parabola $y = \tilde{f}(x)$, which is obtained from the list of intersections reordered from the largest to the smallest x-coordinate. In the example of figure B.20a, it is given by the chain $\{\boldsymbol{i}_3, \boldsymbol{i}_2, \boldsymbol{i}_1, \boldsymbol{i}_4\}$. This definition of the clipping region slightly differs from classic Weiler-Atherton formulations when both the polygon to clip and clipping region are closed polygons. Here, the clipping region is open and only the portion that is of interest for our purpose is explicitly defined. The points $\boldsymbol{x}_j$ are given an 'in' or 'out' flag which shows whether they lie below or above the parabola, respectively. The points $\boldsymbol{i}_k$ are given an 'entry' or 'exit' flag: if the edge of P they lie on goes from 'out' to 'in', then it is an 'entry', else it is an 'exit' (see figure B.20b). The polygon clipping algorithm then follows algorithm 3. Applied to the example in figure B.20a, the algorithm 3 generate two clipped polygon Q_0 and Q_1 shown in figure B.21.

Appendix C. Volume fraction gradient and Hessian matrix computation

Consider a polygon P which has been clipped into q polygons Q_l using the algorithm described in Appendix B. The volume fraction $\tilde{\gamma}$ associated with the parabola $y = \tilde{f}(x)$ is given by

$$\tilde{\gamma} = \frac{\tilde{A}_{\text{poly}} + \tilde{A}_{\text{int}} - \tilde{A}_{\text{mid}}}{\ell(P)}, \tag{C.1}$$

with $\tilde{A}_{\text{poly}}$, $\tilde{A}_{\text{int}}$, and $\tilde{A}_{\text{mid}}$, given by (29), (30), and (31). It then directly follows that

$$\nabla\tilde{\gamma} = \frac{1}{\ell(P)}\left(\nabla\tilde{A}_{\text{poly}} + \nabla\tilde{A}_{\text{int}} - \nabla\tilde{A}_{\text{mid}}\right), \tag{C.2}$$

and

$$\mathrm{H}_{\tilde{\gamma}} = \frac{1}{\ell(P)}\left(\mathrm{H}_{\tilde{A}_{\text{poly}}} + \mathrm{H}_{\tilde{A}_{\text{int}}} - \mathrm{H}_{\tilde{A}_{\text{mid}}}\right). \tag{C.3}$$

The vertices of the polygons Q_l are either issued from the vertices $\boldsymbol{x}_j$ of P, or from the intersections $\boldsymbol{i}_k$ between P and the parabola. Because the mesh vertices are fixed, one necessarily has

$$\nabla\boldsymbol{x}_j = \boldsymbol{0} \text{ and } \mathrm{H}_{\boldsymbol{x}_j} = \boldsymbol{0}, \quad \forall j \in \{1,\ldots,n\}. \tag{C.4}$$

The position of the intersections, however, varies with the coefficients $(\mathrm{a},\mathrm{b},\mathrm{c})$ of the parabola. Their gradients and Hessian matrices are given in Appendix A.
All in all, calculating the right-hand sides of (C.2) and (C.3) requires to be able to compute the gradient and Hessian matrices of the terms

$$T_1 = x_0 y_1, \tag{C.5}$$

$$T_2 = \int_{x_2}^{x_3} \tilde{f}(x)\,dx, \tag{C.6}$$

where $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ could be vertices of P and/or intersections between P and the parabola, while $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ are necessarily intersections. These quantities are given in the following equations

$$\nabla T_1 = y_1\,\nabla x_0 + x_0\,\nabla y_1 \tag{C.7}$$

$$\nabla T_2 = \begin{bmatrix}\frac{x_3^3}{3} & \frac{x_3^2}{2} & x_3\end{bmatrix}^T + y_3\,\nabla x_3 - \begin{bmatrix}\frac{x_2^3}{3} & \frac{x_2^2}{2} & x_2\end{bmatrix}^T - y_2\,\nabla x_2 \tag{C.8}$$

$$\mathrm{H}_{T_1} = y_1\,\mathrm{H}_{x_0} + x_0\mathrm{H}_{y_1} + \nabla x_0(\nabla y_1)^T + \nabla y_1(\nabla x_0)^T \tag{C.9}$$

$$\mathrm{H}_{T_2} = \nabla x_3\,\begin{bmatrix}x_3^2 & x_3 & 1\end{bmatrix}^T + y_3\,\mathrm{H}_{x_3} + \nabla x_3(\nabla y_3)^T - \nabla x_2\,\begin{bmatrix}x_2^2 & x_2 & 1\end{bmatrix}^T - y_2\,\mathrm{H}_{x_2} - \nabla x_2(\nabla y_2)^T \tag{C.10}$$

Appendix D. Volume fraction calculation from arbitrary implicit interfaces

The generalisation of the method presented in section 3.2 to arbitrary implicit interfaces $f(x,y) = 0$ requires to change the way the term $(\tilde{A}_{\text{int}} - \tilde{A}_{\text{mid}})$ is computed, as the integral in equation (30) most probably does not have an obvious solution. The area $\tilde{A} = \ell(P \cap \Lambda)$ is then divided into two terms $\tilde{A}_{\text{poly}}$ and $\tilde{A}_{\text{quad}}$, as illustrated in figure D.22. The term $\tilde{A}_{\text{poly}}$ is calculated using (29), while the area $\tilde{A}_{\text{quad}}$ is approximated using a Legendre-Gauss quadrature rule.

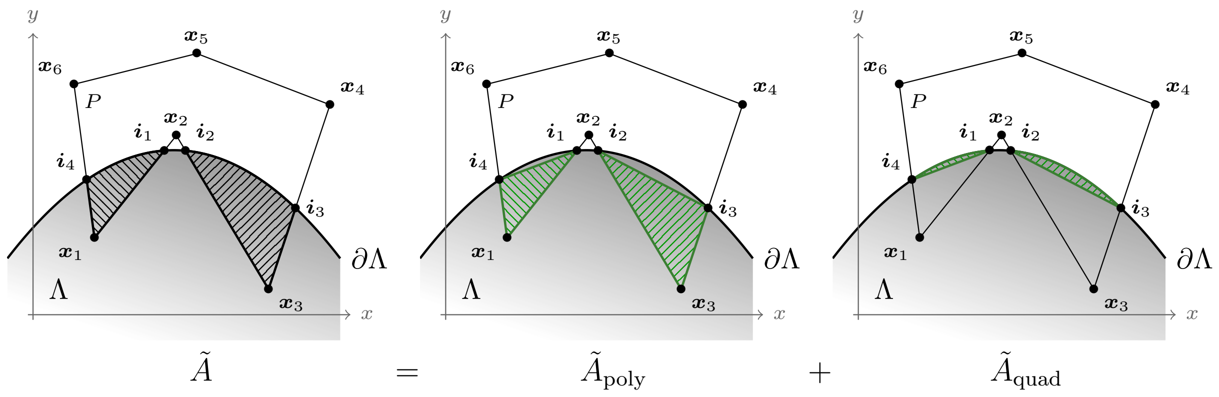


Figure D.22: 2-step computation of the area $\tilde{A} = \tilde{A}_{\text{poly}} + \tilde{A}_{\text{quad}} = \ell(P \cap \Lambda)$, where P is an arbitrary polygon, and Λ the subset $\{f(x,y) \leq 0, (x,y) \in \mathbb{R}^2\}$ with $f(x,y) = 0$ an arbitrary implicit curve.

The construction of the quadrature points used for the calculation of $\tilde{A}_{\text{quad}}$ is illustrated in figure D.23. Each portion of interface located inside the polygon P is considered in a local orthonormal coordinate system (x', y'), in which both end-points of the curve are located on the x' axis. Quadrature points $\boldsymbol{q}_i, i \in \{0, n\}$, are positioned on the interface so that their x' coordinates respect the chosen quadrature rule. The area of each portion of interface can then be approximated by numerically integrating the heights h_i of the quadrature points $\boldsymbol{q}_i$. Using the notations introduced in section 3.2, one can write that

$$\tilde{A}_{\text{quad}} = \sum_{l=1}^{q} \sum_{\substack{\text{edges} \\ \in \partial Q_l \\ \notin \partial P}} \left[\frac{\ell(\text{edge})}{2} \sum_{i=0}^{n} w_i \, h_i \right] , \tag{D.1}$$

where w_i are the quadrature weights associated with n quadrature points (and $\sum_0^n w_i = 2$).

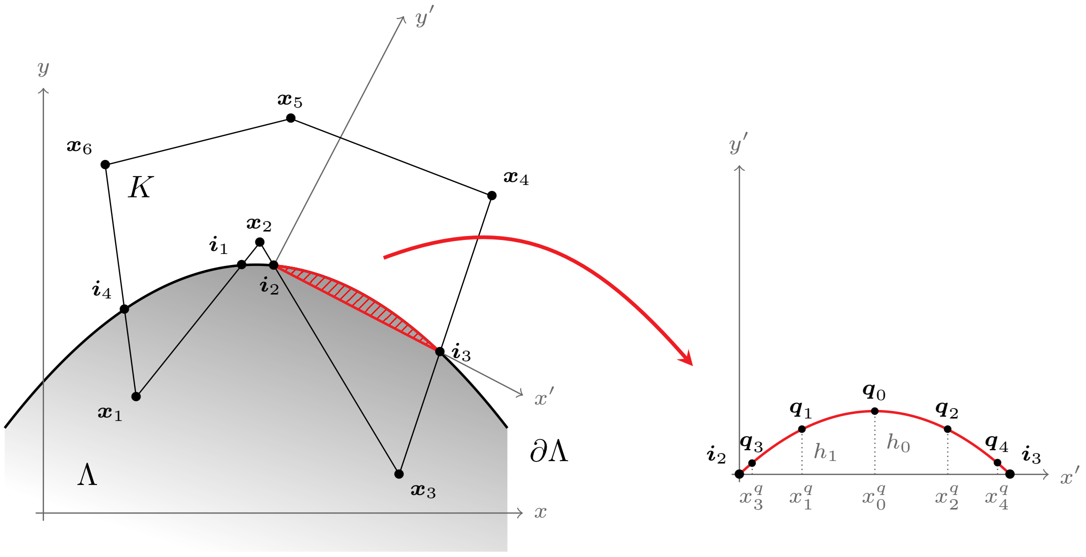


Figure D.23: Construction of the quadrature points $\boldsymbol{q}_i$, for the computation of the area highlighted in red. The area is approximated by numerically integrating the heights h_i. A 5-point quadrature rule has been chosen for this example.

This formalism is also employed to integrate the curvature of the interface inside the polygon P, which follows as

$$\kappa_P = \frac{1}{L_P} \sum_{l=1}^{q} \sum_{\substack{\text{edges} \\ \in \partial Q_l \\ \notin \partial P}} \left[\frac{\ell(\text{edge})}{2} \sum_{i=0}^{n} w_i \, \kappa(\boldsymbol{q}_i) \right] , \tag{D.2}$$

where

$$L_P = \sum_{l=1}^{q} \sum_{\substack{\text{edges} \\ \in \partial Q_l \\ \notin \partial P}} \ell(\text{edge}) \, . \tag{D.3}$$

In this paper, a 64-point Legendre-Gauss quadrature rule is chosen to initialise the volume fraction fields used in the examples and test-cases. This guarantees that the relative volume error associated with the initialised volume fraction field typically lies within a few orders of magnitude of the floating point machine accuracy ($\simeq 10^{-15}$). For the integration of curvature, a 5-point quadrature rule is deemed sufficient.

Acknowledgements

The authors are grateful for the financial support from Petrobras and the EPSRC (grant EP/M021556/1).

References

[1] R. B. DeBar, Fundamentals of the KRAKEN Code. [Eulerian Hydrodynamics Code for Compressible Nonviscous Flow of Several Fluids in Two-Dimensional (Axially Symmetric) Region], Technical Report, California University, Livermore (USA). Lawrence Livermore Lab, 1974.

[2] W. F. Noh, P. Woodward, SLIC (Simple Line Interface Calculation), in: A. I. van de Vooren, P. J. Zandbergen (Eds.), Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics June 28 – July 2, 1976 Twente University, Enschede, Springer Berlin Heidelberg, Berlin, Heidelberg, 1976, pp. 330–340.

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

[4] W. J. Rider, D. B. Kothe, Reconstructing Volume Tracking, Journal of Computational Physics 141 (1998) 112–152.

[5] R. Scardovelli, S. Zaleski, Direct numerical simulation of free-surface and interfacial flow, Annual Review of Fluid Mechanics 31 (1999) 567–603.

[6] T. Abadie, J. Aubin, D. Legendre, On the combined effects of surface tension force calculation and interface advection on spurious currents within Volume of Fluid and Level Set frameworks, Journal of Computational Physics 297 (2015) 611–636.

[7] S. Popinet, Numerical models of surface tension, Annual Review of Fluid Mechanics 50 (2018).

[8] J. Brackbill, D. Kothe, C. Zemach, Continuum Method for Modeling Surface Tension, Journal of Computational Physics 100 (1992) 335–354.

[9] B. Lafaurie, C. Nardone, R. Scardovelli, S. Zaleski, G. Zanetti, Modelling merging and fragmentation in multiphase flows with SURFER, Journal of Computational Physics 113 (1994) 134–147.

[10] M. Williams, D. Kothe, E. Puckett, Accuracy and convergence of continuum surface-tension models, in: W. Shyy, R. Narayanan (Eds.), Fluid Dynamics at Interfaces, Cambridge University Press, Cambridge, 1999, pp. 294–305.

[11] S. Cummins, M. Francois, D. Kothe, Estimating curvature from volume fractions, Computers & Structures 83 (2005) 425–434.

[12] F. Denner, B. 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 (2014) 218–255.

[13] M. Sussman, E. 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) 301–337.

[14] K. Ito, T. Kunugi, H. Ohshima, T. Kawamura, A volume-conservative PLIC algorithm on three-dimensional fully unstructured meshes, Computers & Fluids 88 (2013) 250 – 261.

[15] M. Raessi, J. Mostaghimi, M. Bussmann, Advecting normal vectors: A new method for calculating interface normals and curvatures when modeling two-phase flows, Journal of Computational Physics 226 (2007) 774–797.

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

[17] M. Sussman, K. Smith, M. Hussaini, M. Ohta, R. Zhiwei, A sharp interface method for incompressible two-phase flows, Journal of Computational Physics 221 (2007) 469–505.

[18] G. Bornia, a. Cervone, S. Manservisi, R. Scardovelli, S. Zaleski, On the properties and limitations of the height function method in two-dimensional Cartesian geometry, Journal of Computational Physics 230 (2011) 851–862.

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

[20] M. Owkes, O. Desjardins, A mesh-decoupled height function method for computing interface curvature, Journal of Computational Physics 281 (2014) 285–300.

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

[22] Y. Renardy, M. Renardy, PROST: A Parabolic Reconstruction of Surface Tension for the Volume-of-Fluid Method, Journal of Computational Physics 183 (2002) 400–421.

[23] K. Weiler, P. Atherton, Hidden Surface Removal Using Polygon Area Sorting, SIGGRAPH Comput. Graph. 11 (1977) 214–222.

[24] P. E. Gill, W. Murray, M. H. Wright, Practical Optimization, Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1981.

[25] J. Nocedal, S. J. Wright, Numerical Optimization, Springer series in operations research and financial engineering, Springer, New York, NY, USA, 2nd edition, 2006.

[26] J. R. Shewchuk, Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator, Applied Computational Geometry: Towards Geometric Engineering 1148 (1996) 203–222.

[27] G. Balafas, Polyhedral Mesh Generation for CFD-Analysis of Complex Structures, Msc thesis, Technische Universität München, 2014.