Journal of Computational Physics 365 (2018) 37–55

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Importance of curvature evaluation scale for predictive simulations of dynamic gas–liquid interfaces

Mark Owkes *, Eric Cauble, Jacob Senecal, Robert A. Currie

Mechanical and Industrial Engineering, Montana State University, Bozeman, MT, 59717, USA

ARTICLE INFO

Article history:
Received 22 May 2017
Received in revised form 6 March 2018
Accepted 8 March 2018
Available online xxxx

Keywords:
Adjustable Curvature Evaluation Scale (ACES)
Surface tension
Volume of fluid
Marker particle
Height function
Discretization scale

ABSTRACT

The effect of the scale used to compute the interfacial curvature on the prediction of dynamic gas–liquid interfaces is investigated. A new interface curvature calculation methodology referred to herein as the Adjustable Curvature Evaluation Scale (ACES) is proposed. ACES leverages a weighted least squares regression to fit a polynomial through points computed on the volume-of-fluid representation of the gas–liquid interface. The interface curvature is evaluated from this polynomial. Varying the least squares weight with distance from the location where the curvature is being computed, adjusts the scale the curvature is evaluated on. ACES is verified using canonical static test cases and compared against second- and fourth-order height function methods.

Simulations of dynamic interfaces, including a standing wave and oscillating droplet, are performed to assess the impact of the curvature evaluation scale for predicting interface motions. ACES and the height function methods are combined with two different unsplit geometric volume-of-fluid (VoF) schemes that define the interface on meshes with different levels of refinement. We find that the results depend significantly on curvature evaluation scale. Particularly, the ACES scheme with a properly chosen weight function is accurate, but fails when the scale is too small or large. Surprisingly, the second-order height function method is more accurate than the fourth-order variant for the dynamic tests even though the fourth-order method performs better for static interfaces. Comparing the curvature evaluation scale of the second- and fourth-order height function methods, we find the second-order method is closer to the optimum scale identified with ACES. This result suggests that the curvature scale is driving the accuracy of the dynamics. This work highlights the importance of studying numerical methods with realistic (dynamic) test cases and that the interactions of the various discretizations is as important as the accuracy of one part of the discretization.

1. Introduction

Many gas–liquid flows are controlled by the dynamics at the phase interface, particularly the surface tension force. For example, in the atomization of a liquid fuel into droplets, the surface tension force controls the growth of interfacial instabilities that break apart the liquid core, forming ligaments and droplets that may again break apart if the flow inertia

* Corresponding author.
E-mail address: mark.owkes@montana.edu (M. Owkes).

https://doi.org/10.1016/j.jcp.2018.03.018
0021-9991/

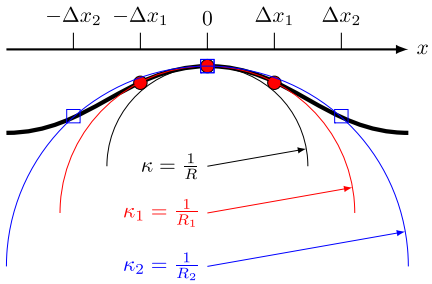


Fig. 1. Gas–liquid interface shown with thick line. Exact curvature shown with κ. Discrete curvatures κ_1 and κ_2 are computed on small and large scales, respectively.

is larger than the surface tension force. For predictive simulations, of this and other gas–liquid flows, the surface tension force needs to be accurate and should converge with mesh refinement.

The surface tension force $\boldsymbol{f}_\sigma = \sigma\kappa\boldsymbol{n}$ depends on the surface tension coefficient σ (assumed constant), curvature of the gas–liquid interface κ, and the interface normal vector $\boldsymbol{n}$ [1]. In this work we focus on the problem of computing an accurate interfacial curvature. Many methods have been developed to compute curvatures and the approach depends on the underlying methodology used to track or capture the phase interface. A variety of interface tracking methods exist such as deforming meshes [2] and Lagrangian marker particle [3]. Interface capturing schemes use an implicit representation of the interface and include level set [3] and volume-of-fluid schemes [4]. Of all these methods that provide the interface location, the volume-of-fluid (VoF) method is very common. The VoF approach captures the interface by storing the ratio of liquid volume to cell volume, a quantity known as the liquid volume fraction. This work on assessing the curvature evaluation scale and the proposed ACES method use geometric VoF schemes as the interface capturing method, although the result may be applicable to other methodologies.

Computing curvatures from VoF methods can be challenging since VoF methods only store the liquid volume fraction and computing derivatives of this discontinuous quantity leads to large discretization errors [1]. As a result, more advanced numerical methods have been developed, such as smoothing the discontinuous liquid volume fraction function [5]. However, even with smoothing, the methods fail to converge [6]. An alternative and popular approach is the height function method [7–9]. This method integrates the liquid volume fraction along columns aligned with the computational mesh to remove the discontinuity and allows for standard finite difference operators to be used to compute the curvature. The method has been shown to converge under mesh refinement for test cases with exact liquid volume fraction fields [10].

While the height function method is simple in principle, the columns used to define the heights often need to be modified in realistic engineering flows, and several modifications have been proposed including: 1) changing the number of cells used in the columns [11–13,10], 2) using a combination of heights and widths for interfaces with high curvatures [14], 3) decoupling the columns from the computational mesh [15], and even applying the approach to level sets [16]. ACES avoids many of these challenges since the least squares regression provides more flexibility than the height function method.

The curvature evaluation scale refers to the size of the computational stencil used by the discretization to compute the curvature. Fig. 1 provides an example of a continuous function representing an interface and two numerical discretizations of the function and their associated curvatures. The two discretizations use three function values with different spacing ($\Delta x_1 < \Delta x_2$) and thus scales. It is clear that the curvature evaluation scales can drastically impact the computed curvature. Curvature evaluation scales are limited when using the height function method. Stencil size can be varied leading to second- and fourth-order formulations [10]. The fourth-order scheme has a larger curvature evaluation scale, but the schemes are limited to these scales, unless more development occurs.

The proposed ACES method provides an alternative to the height function method and allows for the curvature scale to be easily adjusted. ACES computes the interface curvature by fitting a polynomial to interfacial points computed from the VoF interface representation. The fit is performed with a weighted least squares regression. The ACES method is similar to techniques used in interface tracking schemes [17–19] as ACES fits the interface represented by interfacial points. The weighting of interfacial points in the weighted least squares fit is similar to the kernel used in convolution methods that smooth the liquid volume fraction by weighting the volume fractions by their distance from the location the curvature is being computed [20]. Applying the polynomial fit to interfacial points computed from a VoF implementation is a novel contribution of this work. Details of the ACES methodology are provided in Section 2. ACES and our implementations of the height function method (Section 3) are verified with static interfaces and results are provided in Section 4.

There has been a lot of work on developing and testing interface curvature schemes [9,11–15]. However, the authors are not aware of any work that studies the effect of the curvature evaluation scale on dynamic interfaces. Some notable studies compare the second- and fourth-order height function methods [21,22,10]. Sussman and Ohta [21] compare the second- and fourth-order height function methods using the parasitic currents test case, which tests the coupling of the curvature calculation with a Navier–Stokes solver, but only for very small interface perturbations. Zhang and Fogelson [23] compute the curvature of a transported interface and vary the scale the curvature is computed on, but the transport is not coupled with a Navier–Stokes solver. In this work, we perform dynamic simulations with different curvature scales and show that

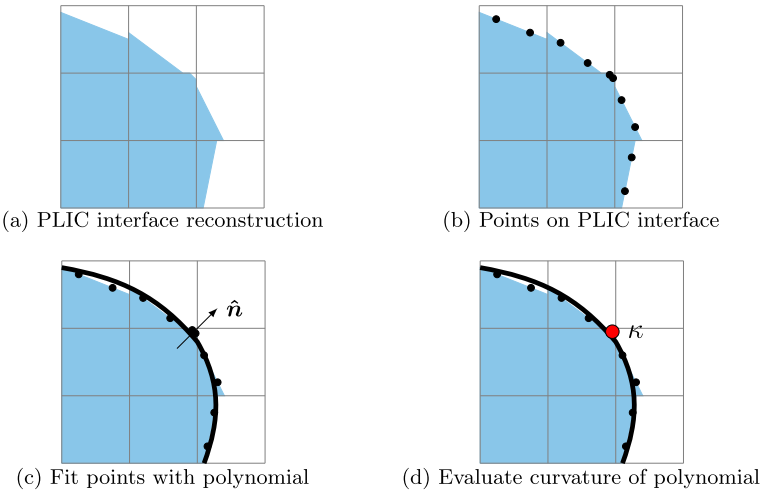


Fig. 2. Methodology to compute interface curvature using the proposed Poly-curv method.

improving the accuracy of curvature schemes does not affect the results as much as matching the curvature evaluation scale to the scale of perturbations created by the velocity field. Dynamic simulations are performed using two variants of unsplit geometric VoF schemes and curvatures computed with ACES and second- and fourth-order height function methods. Details of the Navier–Stokes solution methodology are provided in Section 5 followed by results from simulations of dynamic interfaces in Section 6.

2. ACES methodology

ACES leverages the versatility of a least-squares regression and fits points that are computed on the phase interface with a polynomial function. This proposed method is similar to schemes employed by marker particle interface tracking schemes [17–19], but is novelly applied in the context of VoF and with the goal of varying the curvature evaluation scale.

We begin with an overview of the proposed method and then provide details in the subsequent subsections. In a VoF interface capturing framework the liquid volume fraction is stored within each cell. From the liquid volume fraction, the interface is commonly reconstructed with a plane within each computational cell, known as a piecewise linear interface calculation (PLIC), see Fig. 2a. To compute the curvature of this interface, interfacial points are created on the PLIC interface reconstruction. For example, in Fig. 2b two points are computed on each PLIC. Next, a polynomial is fit to the points using a weighted least-squares regression in a coordinate system oriented with respect to the interface normal vector, Fig. 2c. The curvature is computed directly from the polynomial function, Fig. 2d. The *weighted* least squares allows for the curvature evaluation scale to be easily varied. Details of each of these steps are provided below.

2.1. Reconstruction interface

Reconstructing the interface is a common step in geometric VoF schemes and is used during the transport step where an explicit location of the interface is needed. PLIC is a popular interface reconstruction, and it represents the interface with a plane within each computational cell [24–26]. The PLIC is constrained by the liquid volume fraction α and the orientation of the plane defined with the interface normal vector $\hat{\boldsymbol{n}}$. In this work, the liquid volume fraction, defined as the ratio of liquid volume to cell volume, will be computed 'exactly' or with second-order accuracy for the static tests and with a VoF method for the dynamic tests [27]. The 'exact' volume fractions are either computed with analytic integrations for the two-dimensional tests or with a numerical integration technique that performs octree refinement followed by integration under a linear approximation of the interface which ensures the volume fraction error is less than 1×10^{-10}. Second-order liquid volume fractions are computed using the trapezoidal integration rule. The interface normal will be either specified using an exact solution or computed using the ELVIRA method [28]. Other algorithms to compute the interface normal could be used, i.e., [29,10,1]. We tested ACES with normals computed using a second-order height function method and found only a small modification to the results at the smallest curvature evaluation scales. Therefore, provided the interface normal is computed with second-order accuracy the results and conclusions provide herein will remain the same.

2.2. Compute interfacial points

The next step in the algorithm is to compute interfacial points that are defined on the PLIC interface reconstruction. The number of points and their distribution on the reconstruction are free variables. The points provide information on the

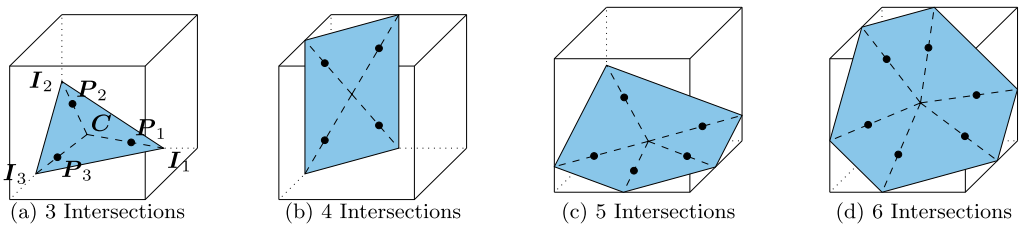


Fig. 3. Calculation of interfacial points from PLIC reconstructions with varying number of intersections with cell edges. PLIC shown with blue, dashed lines connect intersections $\boldsymbol{I}_{p,i,j,k}$, the barycenter $\boldsymbol{C}_{i,j,k}$, and interfacial points $\boldsymbol{P}_{p,i,j,k}$. All points are labeled on part (a). Note that i, j, k is omitted from all subscripts. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

interface shape within the cell and can only provide as much information as the reconstruction they represent. Therefore, for the linear function provided by the PLIC, only a few points are required to represent the shape.

There are various ways that a PLIC interface can intersect a computational cell and our approach is to add one interfacial point for each intersection of the PLIC with a cell edge. Fig. 3 shows how points are computed for a PLIC with 3, 4, 5, and 6 intersections. Intersections between the PLIC interface reconstruction and cell edges are computed and denoted $\boldsymbol{I}_{p,i,j,k}$ for $p = 1, \ldots, N_P$ where N_P is the number of intersections (and interfacial points). In the previous equation, i, j, and k are x, y, and z indices for the computational cells on the underlying Cartesian grid. Computing the intersection points is performed using computational geometry routines (e.g., Ref. [30]). The average of the intersections is used to define the center of the PLIC reconstruction $\boldsymbol{C}_{i,j,k} = \frac{1}{N_P}\sum_{p=1}^{N_P} \boldsymbol{I}_{p,i,j,k}$. Interfacial points are computed along lines between the intersections $\boldsymbol{I}_{p,i,j,k}$ and PLIC center $\boldsymbol{C}_{i,j,k}$. The location of the points along these lines is chosen to match the abscissas of a second-order Gauss–Legendre quadrature and are computed using

$$\boldsymbol{P}_{p,i,j,k} = \boldsymbol{C}_{i,j,k} + \sqrt{\frac{1}{3}}\left(\boldsymbol{I}_{p,i,j,k} - \boldsymbol{C}_{i,j,k}\right) \tag{1}$$

for $p = 1 \ldots N_P$. Other methodologies could be used to place interfacial points on the PLIC interface reconstruction. However, this procedure was found to work well and is used to define interfacial points within each computational cell that contains the gas–liquid interface.

2.3. Fit polynomial to interfacial points

The curvature is required at all cells that contain the gas–liquid interface. At each of these cells, the interfacial points are combined with points from neighboring cells to form a large collection of interfacial points

$$\mathcal{P}_{i,j,k} = \bigcup_{i'=i-N_N}^{i+N_N} \bigcup_{j'=j-N_N}^{j+N_N} \bigcup_{k'=k-N_N}^{k+N_N} \bigcup_{p=1}^{N_{P,i',j',k'}} \boldsymbol{P}_{p,i',j',k'}. \tag{2}$$

We define $N_{\mathcal{P}}$ to be the number of interfacial points in the collection $\mathcal{P}_{i,j,k}$. In the previous equation, N_N is number of neighbors used in the stencil and its influence on the curvature calculation will be analyzed in the results section.

A polynomial is fit to this collection of interfacial points using a weighted least squares regression. For example, a second-order polynomial can be written as

$$f(t_1, t_2) = a_{0,0} + a_{1,0}t_1 + a_{2,0}t_1^2 + a_{0,1}t_2 + a_{1,1}t_1t_2 + a_{0,2}t_2^2. \tag{3}$$

In general, the polynomial can be written as

$$f(t_1, t_2) = \sum_{n=0}^{\mathcal{O}} \sum_{m=0}^{\mathcal{O}-n} a_{m,n} t_1^m t_2^n, \tag{4}$$

where $\mathcal{O}$ is the polynomial order and the impact of $\mathcal{O}$ on the curvature is tested in the results section.

The polynomial is defined in a local coordinate system constructed with the interface unit normal vector $\hat{\boldsymbol{n}}$ and interface unit tangential vectors $\hat{\boldsymbol{t}}_1$ and $\hat{\boldsymbol{t}}_2$. A set of orthonormal tangential vectors is found using the algorithm in Appendix A. To deal with the change in coordinate systems from $[\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}]$ to $[\hat{\boldsymbol{n}}, \hat{\boldsymbol{t}}_1, \hat{\boldsymbol{t}}_2]$, the rotation matrix is used and defined as

$$\boldsymbol{R} = \begin{bmatrix} \hat{\boldsymbol{n}}^{\mathsf{T}} \\ \hat{\boldsymbol{t}}_1^{\mathsf{T}} \\ \hat{\boldsymbol{t}}_2^{\mathsf{T}} \end{bmatrix} = \begin{bmatrix} n_x & n_y & n_z \\ t_{1,x} & t_{1,y} & t_{1,z} \\ t_{2,x} & t_{2,y} & t_{2,z} \end{bmatrix}. \tag{5}$$

Additionally, the interfacial points are translated such that the origin is placed at the center of the PLIC reconstruction in the computational cell of interest, denoted $\boldsymbol{C}_{i,j,k}$. Combining the translation and rotation, the coordinate transformation is performed using

$$\boldsymbol{\mathcal{P}}'_{i,j,k} = \boldsymbol{R}\left(\boldsymbol{\mathcal{P}}_{i,j,k} - \boldsymbol{C}_{i,j,k}\right), \tag{6}$$

where $\boldsymbol{\mathcal{P}}'_{i,j,k}$ are the interfacial points in the $[\hat{\boldsymbol{n}}, \hat{\boldsymbol{t}}_1, \hat{\boldsymbol{t}}_2]$ coordinate system.

In the rotated and translated coordinate system, a weighted least-squares regression is used to fit the polynomial to the interfacial points. The regression is performed by minimizing the weighted sum of the square of the residual, i.e.,

$$R^2 = \sum_{p=1}^{N_{\mathcal{P}}} W_{p,i,j,k}\left[\mathcal{P}'_{n,p,i,j,k} - f(\mathcal{P}'_{t_1,p,i,j,k}, \mathcal{P}'_{t_2,p,i,j,k})\right]^2, \tag{7}$$

where $\mathcal{P}'_n$, $\mathcal{P}'_{t_1}$, and $\mathcal{P}'_{t_2}$ are the normal and tangential components of $\boldsymbol{\mathcal{P}}'$, respectively. The residual is minimized by setting the derivatives with respect to the unknown a's in $f(t_1, t_2)$ equal to zero, i.e., $\frac{\partial R^2}{\partial a_m} = 0$ for $m = 1, \ldots, N_t$, and solving the system of equations for the unknowns $[a_1, a_2, \ldots, a_{N_t}]$. In these equations, N_t is the number of terms in the polynomial.

The weight associated with the pth interfacial point $W_{p,i,j,k}$ is computed using two properties of the interfacial point, namely 1) the area of the PLIC interface represented by the point and 2) the distance the point is from the location where the curvature is being evaluated ($\boldsymbol{C}_{i,j,k}$). The weight is the product of the area weight $W^A_{p,i,j,k}$ and the distance weight $W^D_{p,i,j,k}$ and can be written as $W_{p,i,j,k} = W^{\mathrm{A}}_{p,i,j,k} W^{\mathrm{D}}_{p,i,j,k}$.

The area weight is computed using

$$W^A_{p,i,j,k} = \frac{A_p}{\Delta^2 N_P}, \tag{8}$$

where A_p is the area of the PLIC in the computational cell associated with the pth interfacial point. This area is computed using an analytic relation for the area of the PLIC, which is a polygon with vertices $I_{p,i,j,k}$ [31]. In the previous equation Δ is the characteristic size of the computational cell defined as $\Delta = (\Delta x\, \Delta y\, \Delta z)^{1/3}$.

The distance weight is defined using the Gaussian distribution

$$W^D_{p,i,j,k} = \frac{1}{\sigma_W \Delta \sqrt{2\pi}} \exp\left[-\frac{\|\boldsymbol{\mathcal{P}}_{p,i,j,k} - \boldsymbol{C}_{i,j,k}\|^2}{2(\sigma_W \Delta)^2}\right], \tag{9}$$

where σ_W is the standard deviation normalized by the characteristic mesh size. σ_W is the parameter that allows for the curvature evaluation scale to be varied and the effect on static and dynamic interface accuracy will be shown in the results sections.

2.4. Evaluate curvature

The final step in the proposed method is to evaluate the curvature

$$\kappa = -\nabla \cdot \hat{\boldsymbol{n}}, \tag{10}$$

where

$$\hat{\boldsymbol{n}} = \left[\frac{1}{\sqrt{1 + f_{t_1}^2 + f_{t_2}^2}}, -\frac{f_{t_1}}{\sqrt{1 + f_{t_1}^2 + f_{t_2}^2}}, -\frac{f_{t_2}}{\sqrt{1 + f_{t_1}^2 + f_{t_2}^2}}\right]^{\mathsf{T}} \tag{11}$$

and the subscripts indicate derivatives. These equations can be simplified to the expression

$$\kappa = -\frac{f_{t_1 t_1} + f_{t_2 t_2} + f_{t_1 t_1} f_{t_2}^2 + f_{t_2 t_2} f_{t_1}^2 - 2 f_{t_1 t_2} f_{t_1} f_{t_2}}{\left(1 + f_{t_1}^2 + f_{t_2}^2,\right)^{3/2}}, \tag{12}$$

that is evaluated at $(t_1, t_2) = (0, 0)$, which due to the coordinate translation, Eq. (6), is the center of the PLIC reconstruction $\boldsymbol{C}_{i,j,k}$.

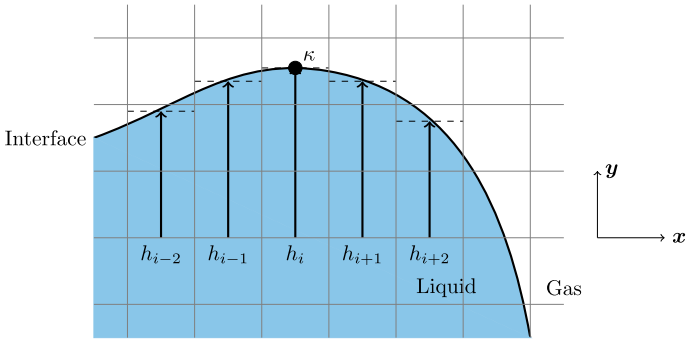


Fig. 4. Example of two-dimensional vertical heights used in the second- and fourth-order height function methods.

2.5. Summary of ACES

In summary, the proposed interfacial curvature calculation procedure has the following steps:

1. compute interfacial points on the PLIC interface reconstruction,
2. for each computational cell that contains an interface, form a collection of points by combining points within the computational stencil,
3. rotate and translate the interfacial points into a local coordinate system orientated with respect to the interface normal vector,
4. fit a polynomial to the rotated and translated interfacial points using a weighted least squares regression, and
5. evaluate the curvature of the polynomial.

3. Height function methodology

Height function methods are a common approach to compute curvatures of VoF interface representations and may be considered the state-of-the-art approach [7–9]. Therefore, the proposed ACES method is compared to second- and fourth-order height function methods. In this section, we provide an overview of the height function methods used herein. The implementations use the second-order mesh-decoupled height function method by Owkes and Desjardins [15] and the fourth-order height function method described by Sussman and Ohta [21].

In its simplest form, the height method computes the interface curvature using finite-difference operators applied to heights. The heights are evaluated by integrating under the interface in columns aligned with the computational mesh as shown in Fig. 4. For an interface that is oriented more horizontally than vertically, the integration can be performed by summing the liquid volume fractions within columns using

$$H_{i',k'} = \sum_{j'=j-N_H}^{j+N_H} \alpha_{i'j'k'} \Delta y \quad \text{for} \quad \begin{cases} i' = i - N_N, \ldots, i + N_N \\ k' = k - N_N, \ldots, k + N_N \end{cases}, \tag{13}$$

where these are the heights used to compute the curvature in the i, j, k cells, N_N is the number of neighbors in the horizontal directions (x and z) and N_H is the number of neighbors in the vertical direction (y) within the stencil. The second-order scheme uses $N_N = 1$ and $N_H = 3$ for a stencil size of $3 \times 3 \times 7$ and the fourth-order scheme uses $N_N = 2$ and $N_H = 6$ for a stencil size of $5 \times 5 \times 13$ [21]. This very large stencil for the fourth-order scheme is needed to have well-defined heights that contain a completely liquid and completely gas cell. But, the large stencil adds significant communication costs for parallel implementations.

Interface curvature is computed using Eq. (12) with approximations for the derivatives that use the heights. Different approximations are used in the second- and fourth-order variants of the height function method as described below.

3.1. Second-order height function method

The second-order method uses the following finite-difference operators to approximate the needed derivatives with second-order accuracy:

$$h_x \approx \frac{H_{i+1,k} - H_{i-1,k}}{2\Delta x}, \tag{14a}$$

$$h_z \approx \frac{H_{i,k+1} - H_{i,k-1}}{2\Delta z}, \tag{14b}$$

$$h_{xx} \approx \frac{H_{i+1,k} - 2H_{i,k} + H_{i-1,k}}{\Delta x^2}, \tag{14c}$$

$$h_{zz} \approx \frac{H_{i,k+1} - 2H_{i,k} + H_{i,k-1}}{\Delta z^2}, \quad \text{and} \tag{14d}$$

$$h_{xz} \approx \frac{H_{i+1,k+1} - H_{i+1,k-1} - H_{i-1,k+1} + H_{i-1,k-1}}{2\Delta x\, 2\Delta z}. \tag{14e}$$

Our implementation extends the second-order height function by employing the mesh-decoupled height function method [15]. This method computes heights within columns not aligned with the computational mesh and improves the curvature calculation for underresolved interfaces. For resolved interfaces, with well-defined mesh-aligned heights, the standard second-order height function method is used. Additional details on the implementation are provided by Owkes and Desjardins [15].

3.2. Fourth-order height function method

Simply using fourth-order finite-difference operators for the derivatives in Eq. (12) is not sufficient for a fourth-order height function scheme, and as shown by Bornia et al. [10] introduces a second-order error. This error results from the fact that the heights are integral quantities with respect to the columns they are defined on and provide only a second-order approximation of the interface location. To develop a three-dimensional fourth-order method we follow the ideas of Sussman and Ohta and derive fourth-order finite difference operators using the height functions (interpreted as integral quantities) as the input [21]. The procedure is detailed in Appendix B.

4. Verification of ACES

4.1. Curvature of a cylindrical interface

The proposed ACES methodology and our height function implementations are tested by computing the curvature of a cylindrical interface, and then comparing the result against the exact solution. The problem consists of a unit square domain $[-0.5, 0.5]^2$ with a cylinder of diameter $D = 0.2$ located near the center of the domain. The center of the cylinder is varied to avoid mesh alignments. Each reported data point is the average of 50 simulations that are conducted with different cylinder centers (x_o, y_o) computed with

$$(x_o, y_o) = (\Delta x\, \mathcal{U}_1, \Delta y\, \mathcal{U}_2)$$

where $\mathcal{U}_n$ is a realization from the uniform distribution defined on the interval $[-1, 1]$.

Two errors are used to assess the accuracy and convergence rate of the proposed scheme. The errors compare the computed curvature κ and the exact curvature $\kappa_e = \frac{1}{R} = \frac{2}{D}$ and are defined using

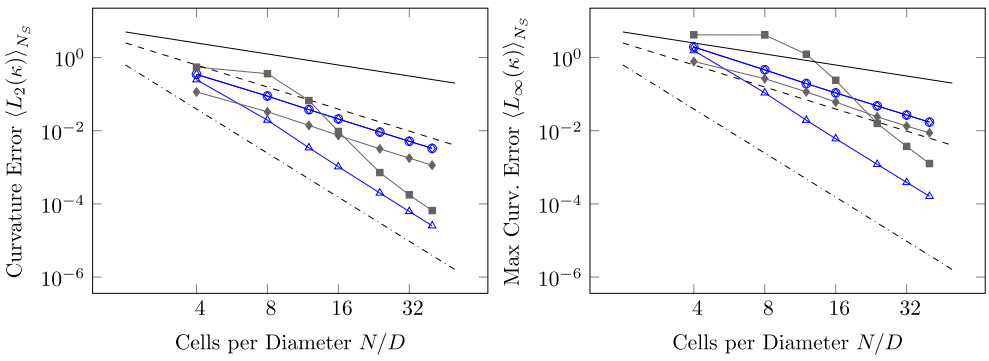
$$L_2(\kappa) = \frac{\sqrt{\sum_{n=1}^{N_{\text{cells}}} \left(\kappa_n - \kappa_{e,n}\right)^2}}{\sqrt{\sum_{n=1}^{N_{\text{cells}}} \kappa_{e,n}^2}}, \quad \text{and} \tag{15}$$

$$L_\infty(\kappa) = \max_{n=1\ldots N_{\text{cells}}} \left| \frac{\kappa_n - \kappa_{e,n}}{\kappa_{e,n}} \right|. \tag{16}$$

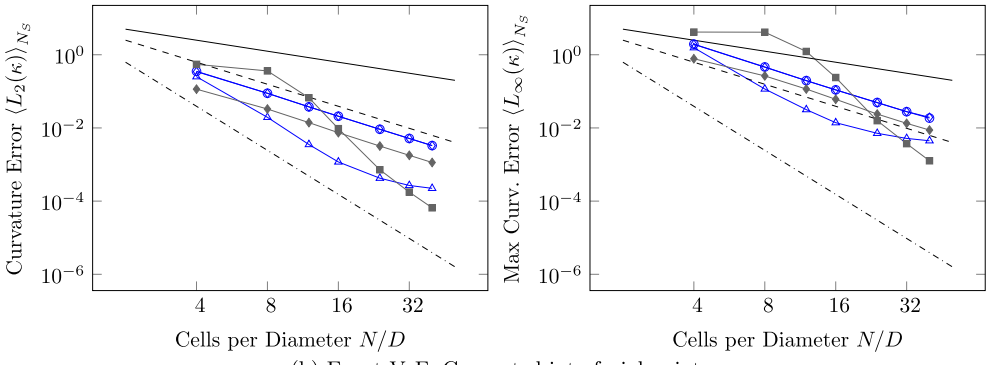
Note that only computational cells that contain the gas–liquid interface are included in the summation and maximum operators in the previous equations, as the curvature is only defined at the interface. These errors are averaged over the $N_S = 50$ simulations and the average errors are denoted $\langle L_2(\kappa)\rangle_{N_S}$ and $\langle L_\infty(\kappa)\rangle_{N_S}$, respectively.

Various parameters in the proposed scheme are tested and the results are compared against the commonly used height function method [10,15]. The first parameter is the order of the polynomial fit to the interfacial points. Orders of $\mathcal{O} = 2$, 3, and 4 are tested and the results are shown in Fig. 5, wherein the curvature errors are plotted versus the number of cells across the diameter of the droplet N/D. In addition to varying the polynomial order, different interface representations are used. In Fig. 5a the liquid volume fractions are exact and the interfacial points exactly fall on the circle (the PLIC interface is not used to compute the points). This scenario provides optimal inputs to both the height function method and ACES. In Fig. 5b the liquid volume fraction is exact, but interfacial points are computed from the PLIC interface representation. Finally, in Fig. 5c liquid volume fractions are defined with second-order accuracy and interfacial points are computed from the PLIC interface reconstruction. This scenario mimics realistic conditions by introducing second-order errors into the initialized VoF field to simulate errors from a second-order accurate VoF advection method.

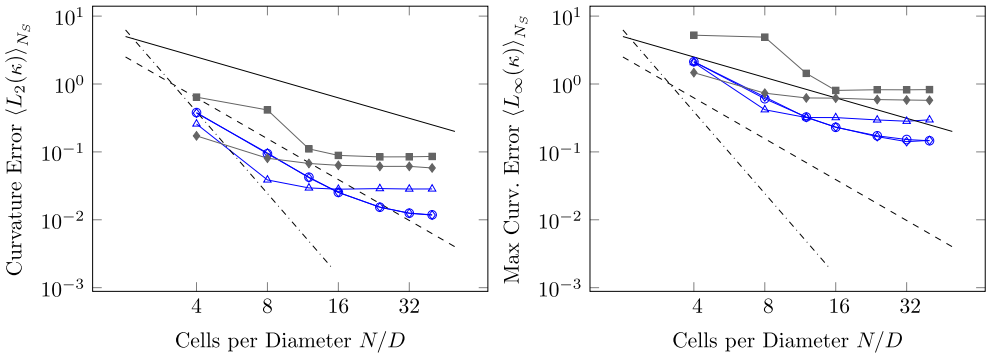
The results show that the proposed method with second- and fourth-order polynomials computes second- and fourth-order accurate curvatures and performs similarly to the height function methods. On finer meshes the proposed scheme computes more accurate curvatures than the height function approach when second-order volume fractions are used and the height function method fails to converge (Fig. 5c). This lack of convergence is consistent with previous studies [32,15,33]



(a) Exact VoF, Exact interfacial points



(b) Exact VoF, Computed interfacial points

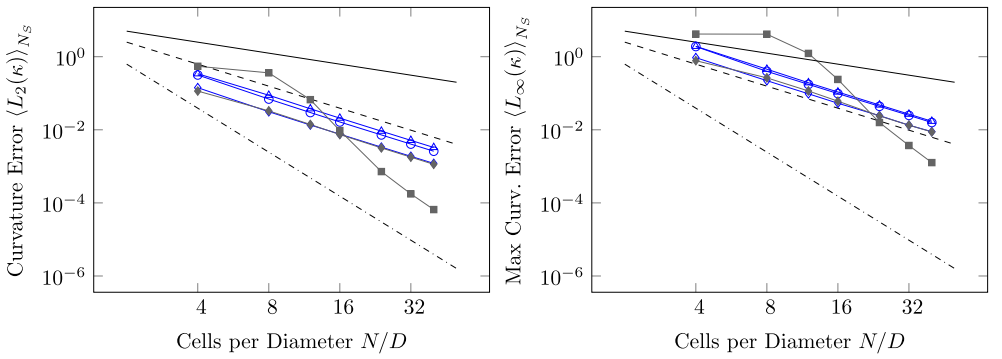


(c) 2nd-order VoF, Computed interfacial points

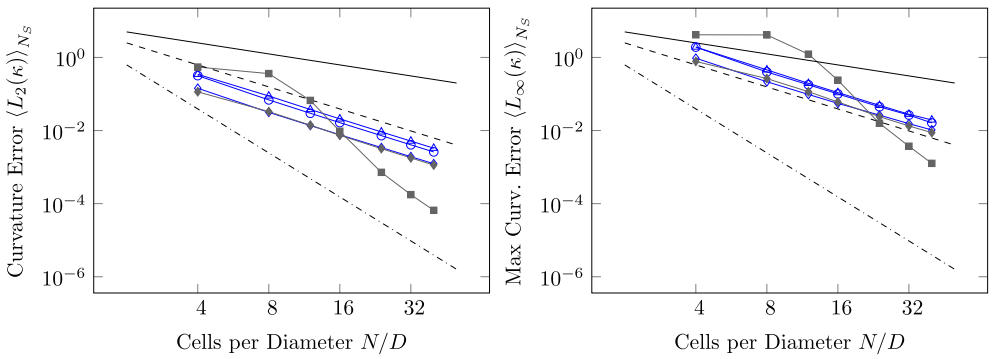
Fig. 5. Effect of polynomial order $\mathcal{O}$ on $L_2(\kappa)$ and $L_\infty(\kappa)$ errors for cylindrical test case. Errors for ACES scheme with $N_N = 3$, $\sigma_W = 1$, and $\mathcal{O} = 2$ (—◇—), 3 (—○—), and 4 (—△—). Errors for second- and fourth-order height function method shown with (—◆—) and (—■—), respectively. First (——), second (- - -), and fourth (-·-·-) order convergence rates shown for reference.

and will hinder the performance of the height function method for realistic applications when the method is coupled with second-order VoF transport schemes. The proposed scheme converges for a wider range of mesh refinement levels. When the method stops converging the error is almost an order of magnitude smaller than the error of the curvature computed with the height function method.

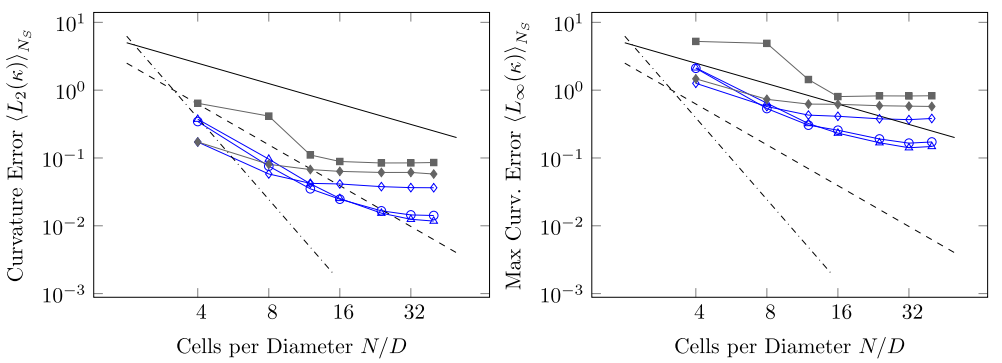
Fig. 6 shows the impact of the number of neighbors used in the computational stencil N_N on the convergence properties of the proposed scheme. Results are displayed in the same format as Fig. 5. When exact VoF and interfacial points are used (Fig. 6a) the height function method shows the expected second- and fourth-order convergence. On the coarsest meshes the fourth-order height function method suffers from ill-defined heights within the large stencil that is required. The proposed scheme shows second-order convergence and the errors are the smallest when the least number of neighbors are used and $N_N = 1$. This is because, when using less neighbors the polynomial fits the interface shape near the computational cell of interest more closely. However, when the interfacial points are computed from the PLIC (Fig. 6b) and errors in their location exist, including more neighbors improves the calculation. When a second-order VoF field is used (Fig. 6c) the best results are computed when the largest number of neighbors is used ($N_N = 3$).



(a) Exact VoF, Exact interfacial points



(b) Exact VoF, Computed interfacial points



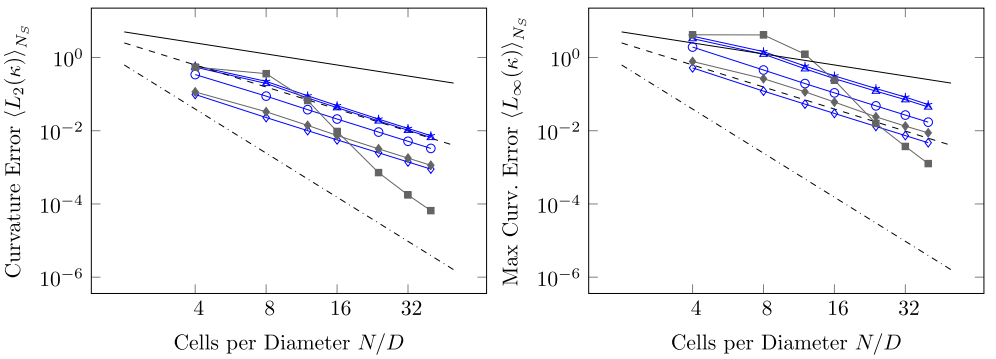
(c) 2nd-order VoF, Computed interfacial points

Fig. 6. Effect of stencil size N_N on $L_2(\kappa)$ and $L_\infty(\kappa)$ errors for cylindrical test case. Errors for ACES scheme with $\mathcal{O}=2$ and $\sigma_W=1$, and $N_N=1$ (—◇—), 2 (—○—), and 3 (—△—). Errors for second- and fourth-order height function method shown with (—♦—) and (—■—), respectively. First (——), second (- - -), and fourth (-·-·-) order convergence rates shown for reference.

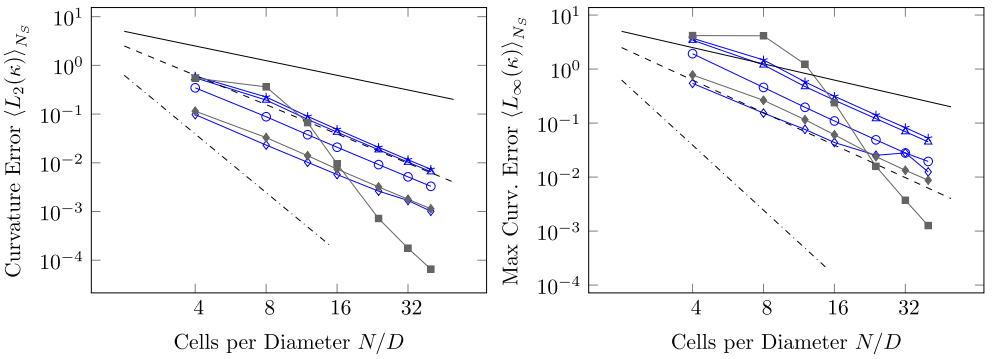
A computational stencil defined with $N_N=3$ uses $7\times7\times7$ grid cells to compute the curvature in each cell containing an interface. While this stencil is large, it has the same inter-processor communication requirements as the second-order height function method, and it requires less communication than the fourth-order height function method. Additionally, the height function method requires this stencil since truncating the stencil results in ill-defined heights that severely hinder accuracy [15]. Contrarily, with the proposed scheme, the stencil size can be reduced with only a small decrease in accuracy as shown in Fig. 6.

The width of the Gaussian weighting function, controlled by σ_W in Eq. (9), is tested and the results shown in Fig. 7. When exact interfacial points are used (Fig. 7a), the smallest σ_W produces the best results. However, with a second-order VoF field, all the σ_W values perform similarly (Fig. 7c). However, these results only assess the curvature and not the interaction of the curvature with the flow solver. Particularly, the interaction of interface perturbations created on the velocity scale to the curvature characterization of these perturbations on the curvature evaluation scale is not assessed by static tests. Therefore, additional results are provided with the dynamic interface tests in Section 6.

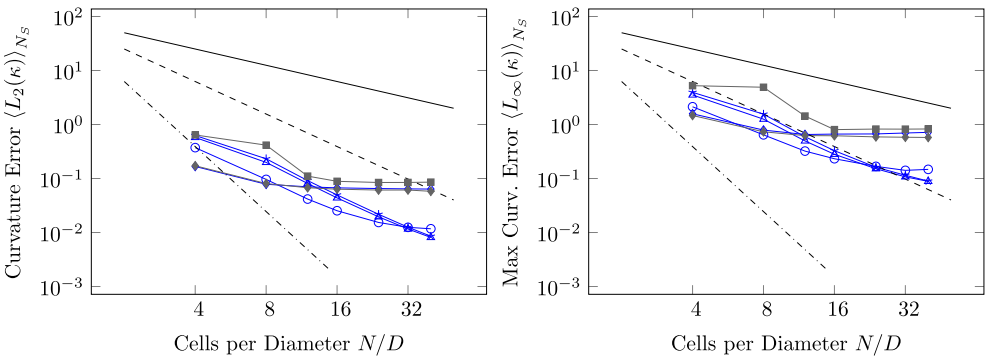
For this test and the following static test cases only one independent parameter ($\mathcal{O}$, N_N, or σ_W) is varied in each plot. The best combination of all the tested values is with a fourth-order polynomial ($\mathcal{O}=4$), 3 neighbors ($N_N=3$), and the



(a) Exact VoF, Exact interfacial points



(b) Exact VoF, Computed interfacial points



(c) 2nd-order VoF, Computed interfacial points

Fig. 7. Effect of curvature evaluation scale characterized by the distance weight σ_W on $L_2(\kappa)$ and $L_\infty(\kappa)$ errors for cylindrical test case. Errors for ACES scheme with $\mathcal{O}=2$ and $N_N=3$, and $\sigma_W=0.5$ (—◆—), 1 (—○—), 2 (—▲—), and 3 (—★—). Errors for second- and fourth-order height function method shown with (—◆—) and (—■—), respectively. First (——), second (- - -), and fourth (-·-·-) order convergence rates shown for reference.

Gaussian width set by $\sigma_W=2$. However, this combination does not work best for the dynamic test cases, where the best set of parameters is $\mathcal{O}=2$, $N_N=3$, and $\sigma_W\approx 1$. Therefore, in these figures, we set the parameters not being varied to the values that work well in the dynamic simulations, giving a realistic overview of how the method performs.

The computational cost of the proposed method was also tested and the values compared to the mesh-decoupled height function method [15]. Fig. 8 shows the computational time required to compute the curvature of the cylindrical interface. The calculations are performed with 1 processor on a 2.8 GHz Intel Core i7. For underresolved droplets, the standard height function method becomes inaccurate and the algorithm switches to the mesh-decoupled formulation, which adds computational cost. The proposed ACES method is less expensive than the mesh-decoupled algorithm, but more expensive than the standard height function. Additional timing results were computed for the dynamic standing wave test case described in Section 6.1. For this case with a mesh resolution of $N_x=64$ timing data is shown in Table 1. The computational cost has been split between the curvature calculation, the unsplit, geometric VoF advection and PLIC reconstruction (described in Section 5.2 [30,27]), the pressure Poisson solver (black box multigrid [34]), and the rest of the code. The ACES method is more expensive than the standard height function, but remains a small fraction (5%) of the total computational cost.

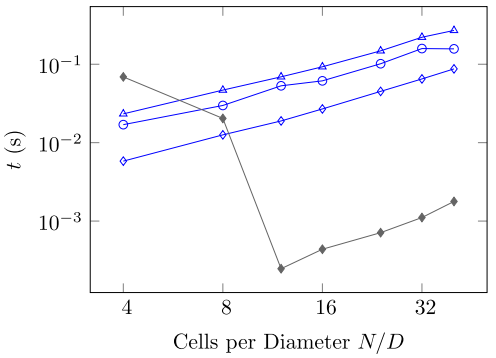


Fig. 8. Computational time required to compute interface curvature for the cylindrical interface test case. Results computed with the ACES scheme with $\mathcal{O} = 2$ (—◆—), 3 (—○—), and 4 (—△—) and the second-order mesh-decoupled height function method (—◆—).

Table 1
Percent of computational time spent in different parts of code for the ACES method ($\mathcal{O} = 2$, $N_N = 3$, and $\sigma_W = 1$) and the second-order height function method.

	Curvature	VoF Adv. & PLIC	Pressure	Rest
ACES	5%	47%	37%	11%
Height function	1%	48%	40%	11%

4.2. Curvature of a sinusoid interface

This test case uses a two-dimensional sinusoidal interface defined as $y(x) = y_o + A\sin(2\pi(x - x_o))$ with amplitude $A = 0.1$. Random perturbations to the initial condition are included to avoid mesh alignment by setting $x_o = \Delta x \mathcal{U}_1$ and $y_o = \Delta y \mathcal{U}_2$. The interface is defined on a unit square domain $[-0.5, 0, 5]^2$. The sinusoid with this amplitude is more horizontal than vertical at every x location and allows the second- and fourth-order height function methods to always have well-defined heights.

Similarly to the cylinder test case, $N_S = 50$ simulations are performed with random interface locations and the $\langle L_2(\kappa)\rangle_{N_S}$ and $\langle L_\infty(\kappa)\rangle_{N_S}$ errors are computed. The results are shown in Fig. 9 using the same format as the cylinder results with the parameters identified to perform well in the dynamic tests, namely $\mathcal{O} = 2$, $N_N = 3$, and $\sigma_W = 1$. When an exact VoF field is used (Figs. 9a and 9b) the height function methods perform excellently due to the fact that the heights are always well-defined. ACES provides a second-order accurate curvature when an exact VoF field is used although the convergence rate decreases when the interfacial points are computed from the PLIC on the finest meshes (Fig. 9b). When a second-order VoF field is used the height function methods fail to converge on even the coarsest meshes (Fig. 9c). The ACES method converges on the coarser meshes but the convergence stops on the finest meshes where the magnitude of the error is almost an order of magnitude smaller than the height function methods' errors.

4.3. Curvature of a spherical interface

The spherical test case evaluates the performance of the proposed scheme in three-dimensions. The test uses a sphere of diameter $D = 0.4$ within a unit cubic domain centered on the origin $[-0.5, 0.5]^3$. The center of the sphere is placed randomly near the center of the domain using a similar procedure as is used for the cylindrical test case, i.e., the center is computed with $(x_o, y_o, z_o) = (\Delta x\,\mathcal{U}_1, \Delta y\,\mathcal{U}_2, \Delta z\,\mathcal{U}_3)$. Each reported data point is the average of $N_S = 10$ random sphere positions. Fig. 10 shows the curvature errors and similar trends are observed in three-dimensions as were shown for the two-dimensional cylindrical test. The largest difference in three-dimensions is with the height function method, wherein there are large errors on the coarser meshes and a lack of convergence in the L_∞ error norm. These errors are due to the difficulty of computing well-defined heights aligned with the computational mesh in three-dimensions. The ACES method performs well even when a second-order VoF field is used and shows between second- and first-order convergence.

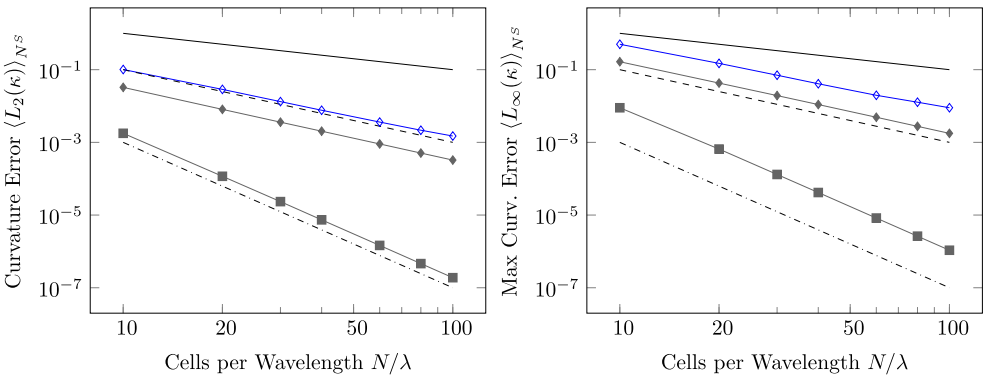
5. Navier–Stokes solver methodology

5.1. Governing equations

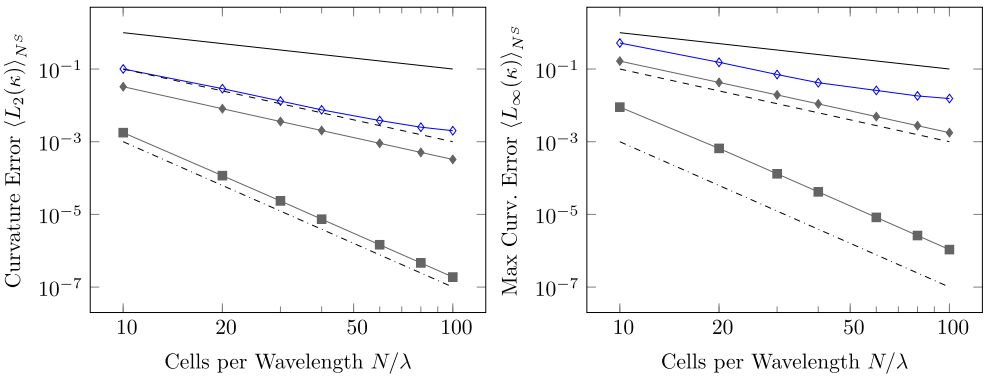
Conservation of mass and momentum for a low Mach number, variable density flow is described by

$$\frac{\partial \rho_\phi}{\partial t} + \nabla\cdot\left(\rho_\phi \boldsymbol{u}_\phi\right) = 0 \quad \text{and} \tag{17}$$

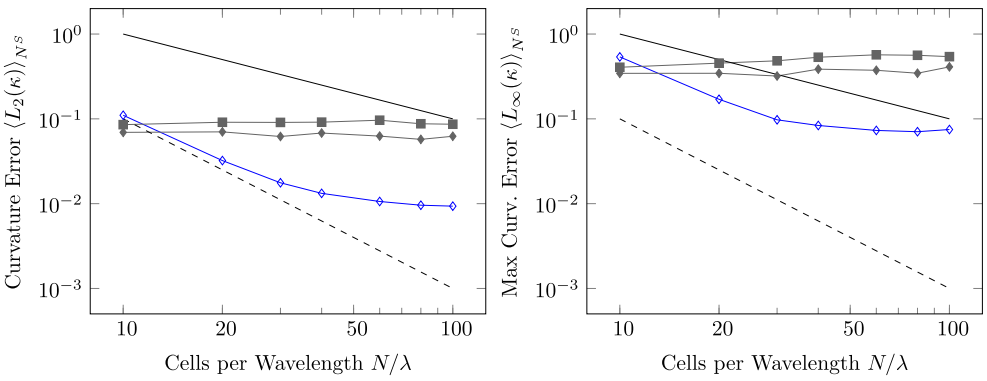
$$\frac{\partial \rho_\phi \boldsymbol{u}_\phi}{\partial t} + \nabla\cdot\left(\rho_\phi \boldsymbol{u}_\phi \otimes \boldsymbol{u}_\phi\right) = -\nabla p_\phi + \nabla\cdot\left(\mu_\phi\left[\nabla \boldsymbol{u}_\phi + \nabla \boldsymbol{u}_\phi^{\mathsf{T}}\right]\right) + \rho_\phi \boldsymbol{g} \tag{18}$$



(a) Exact VOF, Exact interfacial points



(b) Exact VOF, Computed interfacial points



(c) 2nd-order VOF, Computed interfacial points

Fig. 9. Sinusoid test case $L_2(\kappa)$ and $L_\infty(\kappa)$ errors. Errors for ACES with $\mathcal{O}=2$, $N_N=3$ and $\sigma_W=1$ (—◇—) and second- and fourth order height function method shown with (—◆—) and (—■—), respectively.proposed and height function methods shown with diamonds and squares, respectively. First (——), second (- - -), and fourth (-·-·-) order convergence rates shown for reference.

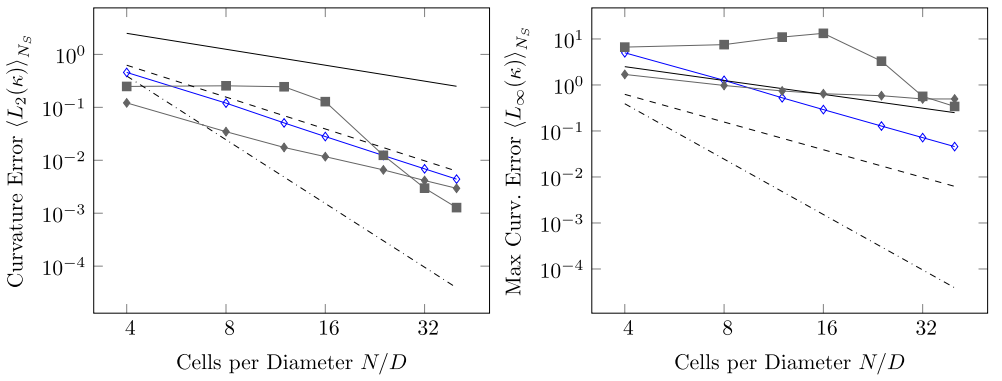
where ρ_ϕ is the density, $\boldsymbol{u}_\phi=[u,v,w]_\phi$ is the velocity field vector, t is time, p_ϕ is the hydrodynamic pressure, μ_ϕ is the dynamic viscosity, and $\boldsymbol{g}$ is the gravitational acceleration. The subscript ϕ indicates the phase and takes values of $\phi=g$ or $\phi=l$ in the gas or liquid phase, respectively.

These equations have been written in both the gas and liquid phases and are connected through jump conditions at the phase interface. For example, the jumps in density and viscosity at the interface Γ are written as

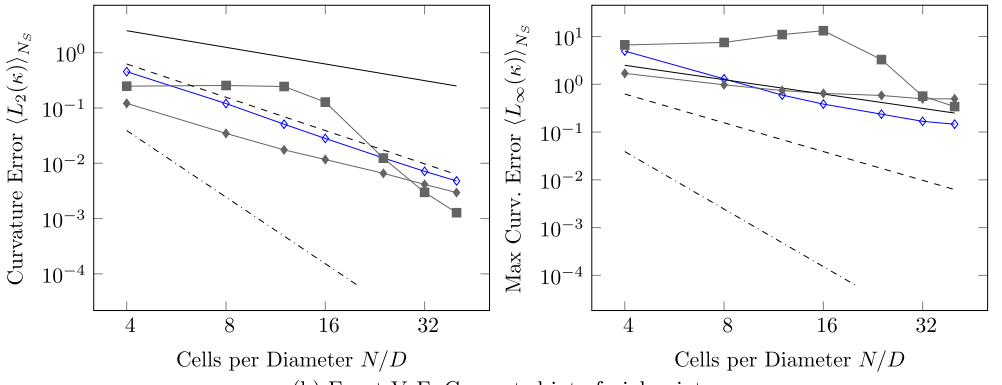
$$[\rho]_\Gamma=\rho_l-\rho_g \quad \text{and} \tag{19}$$

$$[\mu]_\Gamma=\mu_l-\mu_g. \tag{20}$$

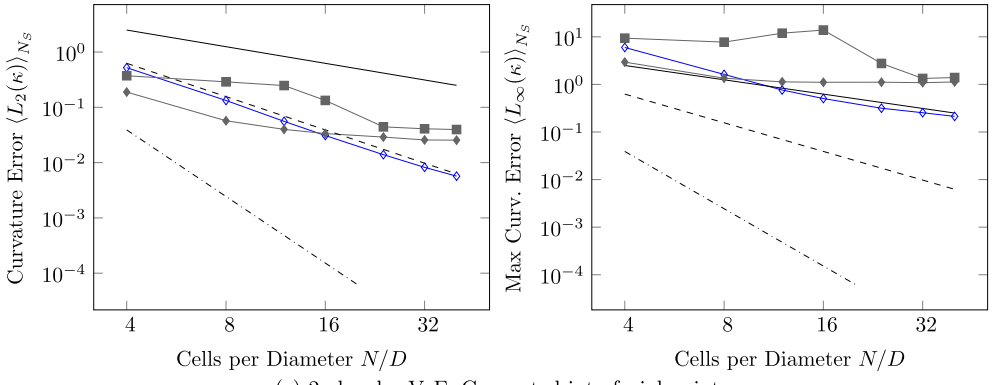
In the absence of a phase change, the velocity field is continuous in the interface normal direction, i.e., $[\boldsymbol{u}\cdot\boldsymbol{n}]_\Gamma=0$. The tangential velocity is assumed to be continuous, $[\boldsymbol{u}\cdot\boldsymbol{t}_d]_\Gamma=0$, for $d=1,2$, which is analogous to the no-slip condition, commonly applied near walls. Together, these conditions provide continuity of the velocity at the phase interface, i.e.,



(a) Exact VoF, Exact interfacial points



(b) Exact VoF, Computed interfacial points



(c) 2nd-order VoF, Computed interfacial points

Fig. 10. Sphere test case $L_2(\kappa)$ and $L_\infty(\kappa)$ errors for proposed (—◇—) and second-order (—♦—) and fourth-order (—■—) height function methods. $\mathcal{O}=2$, $N_N=3$ and $\sigma_W=1$. First (——), second (- - -), and fourth (-·-·-) order convergence rates shown for reference.

$$[\boldsymbol{u}]_\Gamma = 0. \tag{21}$$

The pressure is discontinuous due to contributions from surface tension and the normal component of the viscous stress, i.e.,

$$[p]_\Gamma = \sigma\kappa + 2[\mu]_\Gamma \boldsymbol{n}^{\mathsf{T}} \cdot \nabla \boldsymbol{u} \cdot \boldsymbol{n}, \tag{22}$$

where σ is the surface tension coefficient and κ is the interface curvature. This is the curvature that the proposed method provides.

5.2. Solution methodology

The governing equations are solved with the NGA computational platform [35]. NGA uses a staggered Cartesian mesh with pressure and other scalars located at the center of each computational cell and velocity components located at the faces

of the computational cell. Time discretization is performed using an iterative second-order Crank–Nicolson formulation that incorporates a semi-implicit correction on each sub-iteration, similar to the method of Choi and Moin [36]. Away from the phase interface, NGA uses arbitrarily high-order finite difference operators that conservatively transport mass, momentum, and any other scalars [37]. These operators are well suited for simulations of turbulent flows. Near the phase interface the finite difference operators are inappropriate due to discontinuities. Alternatively, an unsplit geometric semi-Lagrangian VoF method is leveraged.

Two semi-Lagrangian VoF methods are used in this study and are referred herein as 'standard VoF' and 'refined VoF'. The first method uses semi-Lagrangian fluxes of the liquid volume fraction, density, and momentum on the standard flow solver mesh and is based on Ref. [30]. The second computes liquid volume fraction fluxes on a mesh that is twice as fine as the flow solver mesh, to gain discrete conservation of mass and momentum [27]. Both methods are tested because the standard VoF scheme uses the same interface and velocity scales and is similar to many schemes in use today. The refined scheme allows interface perturbations to exist on a smaller scale. Together the schemes demonstrate the effect of curvature scale over a range of interface capturing scales.

6. Dynamic interface tests

The static test cases presented in Section 4 demonstrated that the ACES and height function methods provide accurate curvature. However, how those curvatures interact with the rest of the Navier Stokes solver is also important. The effect of the curvature evaluation scale is studied with the standing-wave and oscillating droplet test cases.

6.1. Standing wave

This test case consists of the temporal evolution of a sinusoidal gas–liquid interface [38]. The computational domain is a two-dimensional rectangle of size $[-\pi, \pi] \times [-2\pi, 2\pi]$. Periodic boundary conditions are used in the x direction and slip conditions in the y direction. Two immiscible fluids are placed in the domain and separated by an interface initialized to a sinusoidal wave. The height of the wave is given by $y = A_o \cos(2\pi x/\lambda)$ where the wavelength is set to $\lambda = 2\pi$ the initial amplitude is $A_o = \lambda/100$. Prosperetti derived an analytical solution that provides the wave amplitude versus time when the kinematic viscosity of both phases is equal.

Parameters are non-dimensionalized and chosen following previous studies [39,37,16]. The density ratio is set to unity, with the fluid density set to $\rho_l = \rho_g = 1$. The surface tension coefficient is set to $\sigma_W = 2$ and the dynamic viscosity of both fluids is set to $\mu = 0.064720863$. Time is non-dimensionalized using the inviscid oscillation frequency

$$\omega_o = \sqrt{\frac{\sigma}{\rho_l + \rho_g}}. \tag{23}$$

Simulations are performed to a non-dimensional time of $\omega_o t = 20$.

Interface dynamics are evaluated by comparing the wave amplitude to the analytic solution. The amplitude $A(t)$ is computed by fitting the interface with a sinusoid $y(t) = A(t)\cos(2\pi x/\lambda)$ at each timestep. An L_2 error is computed to give an average amplitude over time using

$$L_2(A) = \frac{\sqrt{\sum_{n=1}^{N_{\text{time}}} \left(A(t_n) - A_e(t_n)\right)^2}}{\sqrt{\sum_{n=1}^{N_{\text{cells}}} A_e(t_n)^2}}, \tag{24}$$

where $A_e(t)$ is the amplitude at time t computed with the analytic solution.

In Fig. 11 the L_2 error is shown for the tested mesh resolutions ($N_x = 8$, 16, 32, 64, and 128). This figure encompasses the idea of the paper which is that the curvature evaluation scale, characterized by σ_W, significantly controls the accuracy of the interface dynamics. When the curvature evaluation scale is too small, the curvature calculation is prone to errors due to the small stencil used to compute the curvature. Contrarily, when the scale is too large, small scale interface perturbations are averaged out and only the larger interface features are captured by the calculation. The optimal curvature evaluation scale is roughly $\sigma_W = 0.6$ independent of mesh resolution. This result is encouraging as it indicates that structures represented on different meshes or different size structures on a given mesh should all be characterized using the same curvature evaluation scale. Stated another way, it is more important for the curvature evaluation method to capture interface perturbations on the velocity scale and less important to capture the large scale interface shape. The standard and refined VoF methods produce comparable results and both have an optimum value of $\sigma_W \approx 0.6$.

The height function (both second- and fourth-order) performs well. Surprisingly, the fourth-order height function method is not more accurate than the second-order version in predicting the interface dynamics. This could be due to the larger curvature evaluation scale of the fourth-order method compared to the second-order method. Note that the curvature evaluation scale characterized by σ_W for the height function methods is computed by fitting the Gaussian distribution in Eq. (9) to the stencil used in the height function method.

The convergence of the methods with mesh resolution is shown in Fig. 12, which provides results for the second- and fourth-order height function methods and the best variant of ACES with $\sigma_W = 0.6$. The three methods perform similarly and

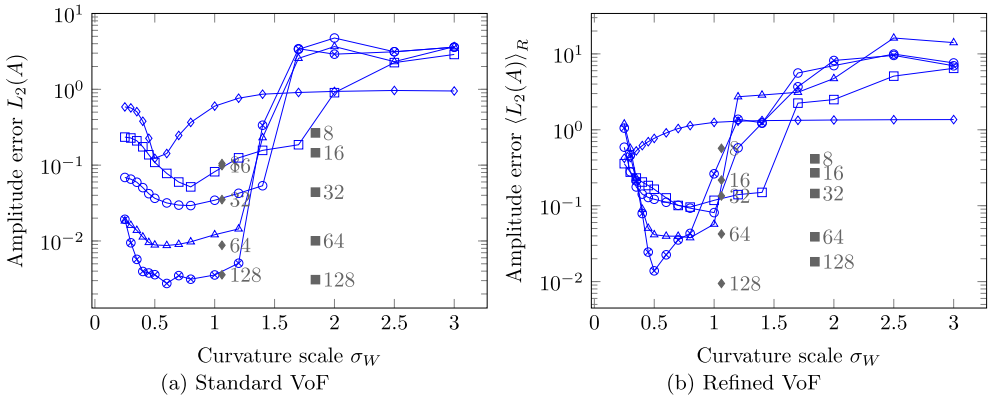


Fig. 11. Error in standing wave amplitude A plotted versus the curvature evaluation scale characterized by σ_W. Results computed with ACES ($\mathcal{O} = 2$ and $N_N = 3$) and resolutions of $N/\lambda = 8$, 16, 32, 64, and 128 shown with (—◇—), (—□—), (—○—), (—△—), (—⊗—), respectively. Second- and fourth-order height function shown with (—♦—) and (—■—), respectively with mesh resolutions indicated by numbers on the figures.

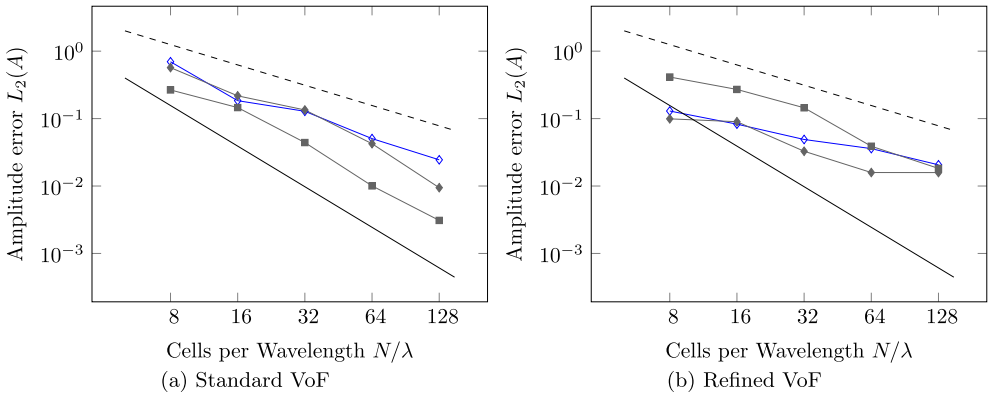


Fig. 12. Error in standing wave amplitude A plotted versus the mesh resolution per wavelength.. Results computed with ACES ($\mathcal{O} = 2$, $N_N = 3$, and $\sigma_W = 0.6$) shown with (—◇—) and second- and fourth-order height function with (—♦—) and (—■—), respectively. First-order (- - -) and second-order (——) convergence rates shown for reference.

on the finer meshes the second- and fourth-order height function methods perform almost the same. This results highlights that the interaction of the curvature calculation with the velocity field is more important than the accuracy of the curvature calculation.

6.2. Oscillating droplet

This test consists of the dynamics of a two-dimensional droplet initialized with an elliptic interface shape defined by

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1. \tag{25}$$

The surface tension force causes the droplet to oscillate with the semi-major axis alternating from being aligned with the x and y axes. The exact period of the oscillation T_e can be computed analytically and is described by [40,41]

$$T_e = 2\pi \sqrt{\frac{\left(\rho_l + \rho_g\right) R^3}{6\sigma}} \tag{26}$$

where $R = \sqrt{ab}$ is the equivalent circular radius.

The simulation consists of an elliptic droplet placed in the center of a unit square domain $[-0.5, 0.5]^2$. The ellipse is specified with $a = 0.1$ and $b = 0.12$. Densities are set to $\rho_l = 1000$ and $\rho_g = 1.3$ and the surface tension coefficient is set to $\sigma = 0.0728$. Simulations are run to a non-dimensional time of $t/T_e = 2$.

Fig. 13 shows the kinetic energy versus non-dimensional time. Interestingly, all the simulations were stable for the time simulated, however, the kinetic energy in the simulation which indicates the droplet oscillation amplitude is effected by the curvature evaluation scale. This suggests that the growth or decay rate of interface perturbations is impacted by the curvature evaluation scale.

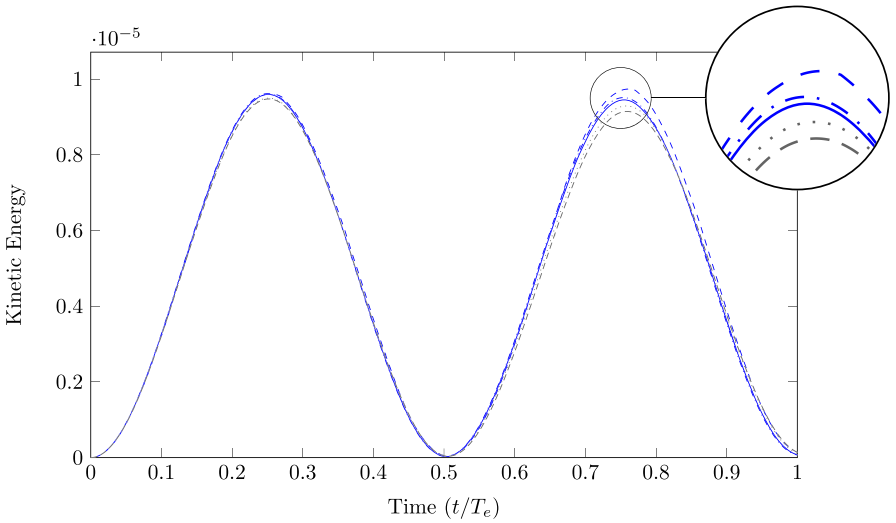


Fig. 13. Kinetic energy versus time for the oscillating droplet test case. ACES results shown with $\sigma_W = 0.7$ (- - -), $\sigma_W = 1.0$ (——), and $\sigma_W = 1.2$ (- - -). Second- and fourth-order height function shown with (-·-·-) and (- - -), respectively.

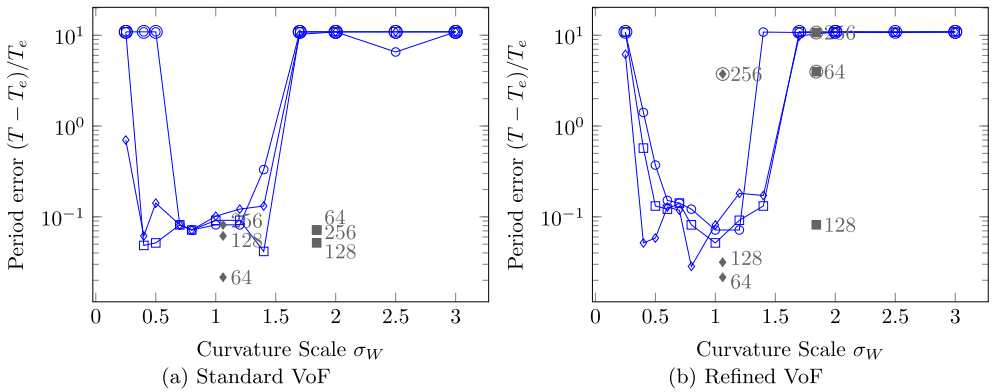


Fig. 14. Error in oscillating droplet period versus the curvature evaluation scale characterized by σ_W. Results computed with ACES ($\mathcal{O} = 2$ and $N_N = 3$) with mesh resolutions of $N = 64$, 128, 256 shown with (—◇—), (—□—), and (—○—), respectively. Second- and fourth-order height function results indicated with (—◆—) and (—■—) and mesh resolutions indicated by numbers on figure. Circled data points indicate an unstable simulation.

The oscillation period is computed from the kinetic energy by finding the time between peaks in the cross-correlation function of kinetic energy. Fig. 14 shows the error in droplet period denoted $(T - T_e)/T_e$ for the tested mesh resolutions ($N_x = 64$, 128, and 256). These resolutions are used to ensure the fourth-order height function method remains well-defined. This test case proved to be more challenging to simulate than the standing wave and many of the simulations are unstable if the curvature evaluation scale is too small or large. In Fig. 14 the circled data points indicate unstable simulations. Note that for these simulations the period was set to zero to allow for a sensible error to be computed. Similarly to the standing wave test case an optimum curvature evaluation scale is identified that corresponds to $\sigma_W \approx 1$.

7. Conclusions

The scale used to compute the interface curvature is found to significantly control the accuracy of dynamic interfaces. Using the proposed ACES curvature calculation method, which allows for the curvature evaluation scale to be varied, it is demonstrated that an optimum scale exists. ACES is compared to the commonly used height function method, and the methods are tested with both static and dynamic interfaces. The static interface provides insight into how accurate the computed curvatures are. The dynamic test cases highlight that curvature accuracy is not the most important feature, and that the scale on which method computes the curvature has a larger impact on the growth or decay of interface perturbations. It is found that accurate dynamic simulations require matching the curvature evaluation scale to the scale the velocity (and interface perturbations) exist on. These dynamic tests have not been performed previously, and emphasize the importance of testing the coupling of numerical methods in addition to testing each component in a discretization strategy.

8. Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1511325. Computational efforts were performed on the Hyalite High-Performance Computing System, operated and supported by University Information Technology Research Cyberinfrastructure at Montana State University.

Appendix A. Algorithm to form local coordinate system

The local orthonormal coordinate system $\left[\hat{\boldsymbol{n}}, \hat{\boldsymbol{t}}_1, \hat{\boldsymbol{t}}_2\right]$ is constructed with the following algorithm for the computational cell with index i, j, k.

Algorithm 1 Pseudocode used to construct local orthonormal coordinate system.

input i ▷ x-index of computational cell
input j ▷ y-index of computational cell
input k ▷ z-index of computational cell
input $\boldsymbol{n}$ ▷ Normal vector field defined in each computational cell with interface (0 otherwise)
▷ Compute normal vector using neighboring values (deals with cells without an interface)
$\hat{\boldsymbol{n}} \leftarrow \sum_{i'=i-1}^{i+1} \sum_{j'=j-1}^{j+1} \sum_{k'=k-1}^{k+1} \boldsymbol{n}_{i',j',k'}$ ▷ Average normal
$\hat{\boldsymbol{n}} \leftarrow \hat{\boldsymbol{n}} / ||\hat{\boldsymbol{n}}||$ ▷ Normalize
▷ Select $\hat{\boldsymbol{t}}_2$ that is not parallel with $\hat{\boldsymbol{n}}$
$\hat{\boldsymbol{t}}_2 \leftarrow [0, 0, -1]$ ▷ Set guess for $\hat{\boldsymbol{t}}_2$
if $\left|\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{t}}_2\right| > 0.6$ **then** ▷ Check if $\hat{\boldsymbol{n}}$ and $\hat{\boldsymbol{t}}_2$ are close to parallel
  $\hat{\boldsymbol{t}}_2 \leftarrow [0, -1, 0]$ ▷ Set another guess for $\hat{\boldsymbol{t}}_2$
  if $\left|\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{t}}_2\right| > 0.6$ **then** ▷ Check if $\hat{\boldsymbol{n}}$ and $\hat{\boldsymbol{t}}_2$ are close to parallel
    $\hat{\boldsymbol{t}}_2 \leftarrow [-1, 0, 0]$ ▷ Set final guess for $\hat{\boldsymbol{t}}_2$
  end if
end if
$\hat{\boldsymbol{t}}_1 \leftarrow \hat{\boldsymbol{n}} \times \hat{\boldsymbol{t}}_2 / \left|\hat{\boldsymbol{n}} \times \hat{\boldsymbol{t}}_2\right|$ ▷ Compute $\hat{\boldsymbol{t}}_1$ that is orthogonal to $\hat{\boldsymbol{n}}$ and $\hat{\boldsymbol{t}}_2$
$\hat{\boldsymbol{t}}_2 \leftarrow \hat{\boldsymbol{n}} \times \hat{\boldsymbol{t}}_1 / \left|\hat{\boldsymbol{n}} \times \hat{\boldsymbol{t}}_1\right|$ ▷ Compute $\hat{\boldsymbol{t}}_2$ that is orthogonal to $\hat{\boldsymbol{n}}$ and $\hat{\boldsymbol{t}}_1$
return $\left[\hat{\boldsymbol{n}}, \hat{\boldsymbol{t}}_1, \hat{\boldsymbol{t}}_2\right]$ ▷ Local orthonormal coordinate system for cell i, j, k

Appendix B. Fourth-order height function method

Following the work of Sussman and Ohta [21], we expands the interface height, described by a continuous function, $h(x, z)$ (for a mostly horizontal interface), using the multivariate Taylor series expansion about (x_i, z_k) leading to

$$h(x,z) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \frac{(x-x_i)^m (z-z_k)^n}{m!\,n!} \frac{\partial^{m+n} h(x_i, z_k)}{\partial x^m\, \partial z^n}. \tag{B.1}$$

Integrating the Taylor series within columns provides a Taylor series expansion of the heights used in the height function method, i.e.,

$$H_{i',k'} = \frac{1}{\Delta x\, \Delta z} \int_{x_{i'}-\Delta x/2}^{x_{i'}+\Delta x/2} \int_{z_{k'}-\Delta z/2}^{z_{k'}+\Delta z/2} h(x,z)\, \mathrm{d}x\, \mathrm{d}z, \tag{B.2}$$

for $i' = i-2, \ldots, i+2$ and $k' = k-2, \ldots, k+2$.

A fourth-order curvature calculation requires the derivative h_x, h_z, h_{xx}, h_{zz}, and h_{xz} are computed with fourth-order accuracy using the heights $H_{i'k'}$. The derivatives are assumed to be a linear combination of the heights, i.e.,

$$\frac{\partial^{(D_x+D_z)} h(x_i, z_k)}{\partial x^{(D_x)}\, \partial z^{(D_z)}} \approx \sum_{i'=-2}^{2} \sum_{k'=-2}^{2} A_{i',k'}^{D_x,D_z} H_{i',k'}, \tag{B.3}$$

where D_x and D_z specify the order of the derivative.

The coefficients $A_{i',k'}^{D_x,D_z}$ are found using a standard technique to compute finite-difference operators, but the input information is the Taylor series expansions of the heights, instead of Taylor series expansions of discrete data points [42]. We plug Eqs. (B.1) and (B.2) into Eq. (B.3) and set as many of the low-order error terms to zero as possible. This procedure can be organized using Table 2 where we have denoted the m, n term in the Taylor series expansion of the height $H_{i',k'}$ as

Table 2
Taylor series expansions organized into a table to compute, for example, $\frac{\partial h(x_i,z_k)}{\partial x}$ based on the notation of Moin [42]. Other derivatives are found by changing the second row in the table.

	$h(x_i,z_k)$	$\frac{\partial h(x_i,z_k)}{\partial x}$	$\frac{\partial h(x_i,z_k)}{\partial y}$	$\cdots$	$\frac{\partial^8 h(x_i,z_k)}{\partial x^4 \partial y^4}$	$\cdots$
$-\frac{\partial h(x_i,z_k)}{\partial x}$	0	-1	0	$\cdots$	0	$\cdots$
$A_{-2,-2}H_{-2,-2}$	$A_{-2,-2}\hat{H}^{0,0}_{-2,-2}$	$A_{-2,-2}\hat{H}^{1,0}_{-2,-2}$	$A_{-2,-2}\hat{H}^{0,1}_{-2,-2}$	$\cdots$	$A_{-2,-2}\hat{H}^{4,4}_{-2,-2}$	$\cdots$
$A_{-2,-1}H_{-2,-1}$	$A_{-2,-1}\hat{H}^{0,0}_{-2,-1}$	$A_{-2,-1}\hat{H}^{1,0}_{-2,-1}$	$A_{-2,-1}\hat{H}^{0,1}_{-2,-1}$	$\cdots$	$A_{-2,-1}\hat{H}^{4,4}_{-2,-1}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$A_{2,2}H_{2,2}$	$A_{2,2}\hat{H}^{0,0}_{2,2}$	$A_{2,2}\hat{H}^{1,0}_{2,2}$	$A_{2,2}\hat{H}^{0,1}_{2,2}$	$\cdots$	$A_{2,2}\hat{H}^{4,4}_{2,2}$	$\cdots$

Algorithm 2 MATLAB code used to compute weights for fourth-order height function method.

```
clear; clc
syms x z xik zik dx dz
% Order of expansion
M=4; N=4;
% Taylor series expansion
h=sym(zeros(M+1,N+1));
for n=0:N
    for m=0:M
        h(m+1,n+1)=(x-xik)^m*(z-zik)^n/(factorial(m)*factorial(n));
    end
end
% Allocate Matrices
Z=sym(zeros((M+1)*(N+1),(M+1)*(N+1)));
B=sym(eye  ((M+1)*(N+1),(M+1)*(N+1)));
% Loop over cells in stencil
c=0;
for kp=-2:2
    for ip=-2:2
        c=c+1; % Column number
        fprintf('%i/25 \n',c)
        % Integrate h over this cell to form height
        H=int(int(h,z,zik+(kp-1/2)*dz,zik+(kp+1/2)*dz) ...
            ,x,xik+(ip-1/2)*dx,xik+(ip+1/2)*dx)/(dx*dz);
        % Get coefficents from H
        Z(:,c)=H(:);
    end
end
% Solve for A's (in a vector)
Av=Z\B;
% Weights for needed derivatives
c=0;
for n=0:N
    for m=0:M
        c=c+1;
        if     m==1 && n==0;  hx =reshape(Av(:,c),5,5); % h_x =dh/dx
        elseif m==0 && n==1;  hz =reshape(Av(:,c),5,5); % h_z =dh/dz
        elseif m==2 && n==0;  hxx=reshape(Av(:,c),5,5); % h_xx=d^2h/dx^2
        elseif m==0 && n==2;  hzz=reshape(Av(:,c),5,5); % h_zz=d^2h/dz^2
        elseif m==1 && n==1;  hxz=reshape(Av(:,c),5,5); % h_xz=d^2h/(dx dz)
        end
    end
end
```

$$\hat{H}^{m,n}_{i',k'} = \frac{1}{\Delta x\,\Delta z}\int_{x_{i'}-\Delta x/2}^{x_{i'}+\Delta x/2}\int_{z_{k'}-\Delta z/2}^{z_{k'}+\Delta z/2}\frac{(x-x_i)^m(z-z_k)^n}{m!\,n!}\frac{\partial^{m+n}h(x_i,z_k)}{\partial x^m\,\partial z^n}\,\mathrm{d}x\,\mathrm{d}z. \tag{B.4}$$

With this notation $H_{i',k'} = \sum_{m=0}^{4}\sum_{n=0}^{4}\hat{H}^{m,n}_{i',k'}$. Minimizing the error requires setting the sum of as many columns in Table 2 equal to zero as possible. Note that the sum of the first column is Eq. (B.3) and the other columns are the terms in the Taylor series expansions of the heights. This procedure results in a set of 25 equations for the 25 unknowns $A^{D_x,D_z}_{i',k'}$. We solved this system of equations using MATLAB and provide our code in Algorithm 2. These weights are combined with Eq. (B.3) to compute each of the derivatives in Eq. (12).

References

[1] G. Tryggvason, R. Scardovelli, S. Zaleski, Direct Numerical Simulations of Gas–Liquid Multiphase Flows, Cambridge University Press, 2011.
[2] W. Dettmer, P.H. Saksono, D. Peric, On a finite element formulation for incompressible Newtonian fluid flows on moving domains in the presence of surface tension, Commun. Numer. Methods Eng. 19 (9) (2003) 659–668.
[3] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1) (1988) 12–49.
[4] C. Hirt, B. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, J. Comput. Phys. 39 (1) (1981) 201–225.
[5] J.U. Brackbill, D.B. Kothe, C. Zemach, A continuum method for modeling surface tension, J. Comput. Phys. 100 (2) (1992) 335–354.
[6] S.J. Cummins, M.M. François, D.B. Kothe, Estimating curvature from volume fractions, Comput. Struct. 83 (6–7) (2005) 425–434.
[7] M.D. Torrey, L.D. Cloutman, R.C. Mjolsness, C.W. Hirt, NASA-VOF2d: A Computer Program for Incompressible Flows with Free Surfaces, NASA STI/Recon Technical Report N 86 1985, 30116.
[8] B.D. Nichols, C.W. Hirt, R.S. Hotchkiss, SOLA-VOF: A Solution Algorithm for Transient Fluid Flow with Multiple Free Boundaries, NASA STI/Recon Technical Report N 81 1980, 14281, 00415.
[9] J. Helmsen, P. Colella, E.G. Puckett, Non-Convex Profile Evolution in Two Dimensions Using Volume of Fluids, Tech. Rep. LBNL–40693, Lawrence Berkeley Lab., CA (United States), 1997.
[10] 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, J. Comput. Phys. 230 (4) (2011) 851–862.
[11] S. Afkhami, M. Bussmann, Height functions for applying contact angles to 2d VOF simulations, Int. J. Numer. Methods Fluids 57 (4) (2008) 453–472.
[12] M. Sussman, K.M. Smith, M.Y. Hussaini, M. Ohta, R. Zhi-Wei, A sharp interface method for incompressible two-phase flows, J. Comput. Phys. 221 (2) (2007) 469–505.
[13] M. Sussman, A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles, J. Comput. Phys. 187 (1) (2003) 110–136.
[14] S. Popinet, An accurate adaptive solver for surface-tension-driven interfacial flows, J. Comput. Phys. 228 (16) (2009) 5838–5866.
[15] M. Owkes, O. Desjardins, A mesh-decoupled height function method for computing interface curvature, J. Comput. Phys. 281 (2015) 285–300.
[16] M. Owkes, O. Desjardins, A discontinuous Galerkin conservative level set scheme for interface capturing in multiphase flows, J. Comput. Phys. 249 (15) (2013) 275–302.
[17] A.Z. Zinchenko, M.A. Rother, R.H. Davis, A novel boundary-integral algorithm for viscous interaction of deformable drops, Phys. Fluids 9 (6) (1997) 1493–1511.
[18] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, Y.J. Jan, A front-tracking method for the computations of multiphase flow, J. Comput. Phys. 169 (2) (2001) 708–759.
[19] J.P. Gois, A. Nakano, L.G. Nonato, G.C. Buscaglia, Front tracking with moving-least-squares surfaces, J. Comput. Phys. 227 (22) (2008) 9643–9669.
[20] M.W. Williams, D.B. Kothe, E.G. Puckett, Accuracy and convergence of continuum surface tension models, in: Fluid Dynamics at Interfaces, Univ. Press, 1998, pp. 294–305.
[21] M. Sussman, M. Ohta, High-order techniques for calculating surface tension forces, in: Free Boundary Problems, Birkhäuser, Basel, 2006, pp. 425–434.
[22] M.M. Francois, B.K. Swartz, Interface curvature via volume fractions, heights, and mean values on nonuniform rectangular grids, J. Comput. Phys. 229 (3) (2010) 527–540.
[23] Q. Zhang, A. Fogelson MARS, An analytic framework of interface tracking via mapping and adjusting regular semialgebraic sets, SIAM J. Numer. Anal. 54 (2) (2016) 530–560.
[24] R. DeBar, Fundamentals of the KRAKEN Code, Tech. Rep. UCIR-760, LLNL, 1974.
[25] W.J. Rider, D.B. Kothe, Reconstructing volume tracking, J. Comput. Phys. 141 (2) (1998) 112–152.
[26] D. Youngs, Time-dependent multi-material flow with large fluid distortion, Numer. Methods Fluid Dyn. (1982) 273–285.
[27] M. Owkes, O. Desjardins, A mass and momentum conserving unsplit semi-Lagrangian framework for simulating multiphase flows, J. Comput. Phys. 332 (2017) 21–46.
[28] J.E. Pilliod, E.G. Puckett, Second-order accurate volume-of-fluid algorithms for tracking material interfaces, J. Comput. Phys. 199 (2) (2004) 465–502.
[29] E. Puckett, A volume-of-fluid interface reconstruction algorithm that is second-order accurate in the max norm, Commun. Appl. Math. Comput. Sci. 5 (2) (2010) 199–220.
[30] M. Owkes, O. Desjardins, A computational framework for conservative, three-dimensional, unsplit, geometric transport with application to the volume-of-fluid (VOF) method, J. Comput. Phys. 270 (1) (2014) 587–612.
[31] R.N. Goldman, IV.1 — Area of planar polygons and volume of polyhedra, in: James Arvo (Ed.), Graphics Gems II, Morgan Kaufmann, San Diego, 1991, pp. 170–171.
[32] M. Coquerelle, S. Glockner, A fourth-order accurate curvature computation in a level set framework for two-phase flows subjected to surface tension forces, J. Comput. Phys. 305 (Suppl. C) (2016) 838–876.
[33] C.B. Ivey, P. Moin, Accurate interface normal and curvature estimates on three-dimensional unstructured non-convex polyhedral meshes, J. Comput. Phys. 300 (2015) 365–386.
[34] J. Dendy, Black box multigrid, J. Comput. Phys. 48 (3) (1982) 366–386.
[35] O. Desjardins, G. Blanquart, G. Balarac, H. Pitsch, High order conservative finite difference scheme for variable density low Mach number turbulent flows, J. Comput. Phys. 227 (15) (2008) 7125–7159.
[36] H. Choi, P. Moin, Effects of the computational time step on numerical solutions of turbulent flow, J. Comput. Phys. 113 (1) (1994) 1–4.
[37] O. Desjardins, V. Moureau, H. Pitsch, An accurate conservative level set/ghost fluid method for simulating turbulent atomization, J. Comput. Phys. 227 (18) (2008) 8395–8416.
[38] A. Prosperetti, Motion of two superposed viscous fluids, Phys. Fluids 24 (7) (1981) 1217–1223.
[39] M. Herrmann, A balanced force refined level set grid method for two-phase flows on unstructured flow solver grids, J. Comput. Phys. 227 (4) (2008) 2674–2706.
[40] L. Rayleigh, On the capillary phenomena of jets, Proc. R. Soc. Lond. 29 (1879) 71–97.
[41] D.E. Fyfe, E.S. Oran, M.J. Fritts, Surface tension and viscosity with Lagrangian hydrodynamics on a triangular mesh, J. Comput. Phys. 76 (2) (1988) 349–384.
[42] P. Moin, Fundamentals of Engineering Numerical Analysis, Cambridge University Press, 2010.