Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 226 (2007) 774–797
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Advecting normal vectors: A new method for calculating interface normals and curvatures when modeling two-phase flows

M. Raessi, J. Mostaghimi *, M. Bussmann

Department of Mechanical and Industrial Engineering, University of Toronto, Canada

Received 13 October 2006; received in revised form 24 April 2007; accepted 28 April 2007
Available online 22 May 2007

Abstract

In simulating two-phase flows, interface normal vectors and curvatures are needed for modeling surface tension. In the traditional approach, these quantities are calculated from the spatial derivatives of a scalar function (e.g. the volume-of-fluid or the level set function) at any instant in time. The orders of accuracy of normals and curvatures calculated from these functions are studied. A new method for calculating these quantities is then presented, where the interface unit normals are advected along with whatever function represents the interface, and curvatures are calculated directly from these advected normals. To illustrate this new approach, the volume-of-fluid method is used to represent the interface and the advected normals are used for interface reconstruction. The accuracy and performance of the new method are demonstrated via test cases with prescribed velocity fields. The results are compared with those of traditional approaches.

Keywords: Curvature; Surface tension; Two-phase flows; Free-surface flows; Spurious currents; Volume-of-fluid; Level set

1. Introduction

In simulating two-phase flows, accurate modeling of surface tension effects is a challenging task. Non-physical velocities, commonly known as spurious or parasitic currents, are easily induced in a flow if the surface tension force is not modelled properly. There are two sources of error when modeling surface tension [1]: the approach to implementing (discretizing) the surface tension and the interface curvature calculation. Various approaches have been presented to reduce or eliminate these errors (e.g. [2–4]). However, Francois et al. [1] recently proposed a method for implementing surface tension into a volume-of-fluid (VOF) [5,6] model that imposes an exact balance between the surface tension and pressure forces. In this method, because the surface tension and pressure forces are treated consistently, no spurious currents are induced in a flow provided that the interface curvature is exact.

* Corresponding author. Tel.: +1 416 978 5604; fax: +1 416 946 8252.
E-mail address: mostag@mie.utoronto.ca (J. Mostaghimi).

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2007.04.023

The focus of this paper then is on the calculation of interface normal vectors and curvatures. There are different methods available, but the most widely used technique is to calculate these quantities from the spatial derivatives of the scalar function which is employed to represent a particular fluid or an interface in a multiphase flow. For instance, the gradient of the VOF function or the level set (LS) [7–9] function yields the normal to an interface represented by either of these. Then, by taking the divergence of the interface unit normal vector, i.e. the second derivative of the VOF or LS function, the interface curvature is obtained. The accuracy of this approach will be assessed later in this paper for both of these functions.

Calculating interface curvature from the VOF or LS function is a straightforward, although not necessarily accurate, approach. There are other methods which are more accurate and sophisticated. For example, the VOF-based height-function method [1,10,11] yields curvatures with second-order accuracy. In this method, fluid "heights" are calculated from the volume fractions in the direction normal (or close to normal) to an interface; the interface curvature is then obtained from the derivatives of the height function. However, this method yields poor results [12] if an interface is not adequately resolved.

Other VOF-based methods for calculating interface curvature utilize curve fits of the volume fraction data; the curvature is then calculated from the equation of the curve. For example, Poo and Ashgriz [13] utilized a second-order polynomial to calculate curvatures in 2D. More recently, in a method known as PROST [14], the data is fitted iteratively with a paraboloid in either 2D or 3D. Although this method yields second-order accurate curvatures, the implementation is not straightforward, especially for complicated interface geometries.

This paper introduces an entirely different approach to calculating interface normals and curvatures: the interface normal vectors are advected along with the interface. The structure of the paper is as follows. First, interface normals and curvatures calculated from the VOF and LS functions are evaluated. Then, the formulation and an approach to implementing the new method are presented. Finally, the accuracy and the performance of the new method are demonstrated via various test cases.

2. Normals and curvatures from VOF and LS functions

2.1. Volume-of-fluid method

In the VOF method, a scalar color function, f, defined as

$$f(\vec{x}) = \begin{cases} 1, & \vec{x} \in \text{fluid1} \\ 0, & \vec{x} \in \text{fluid2} \end{cases} \tag{1}$$

is used to represent fluid 1 in a two-phase system. The VOF function is discontinuous by definition. The discretized form of the VOF function is the fraction of a numerical cell volume V occupied by fluid 1, and is defined as

$$F = \frac{1}{V}\int_V f \,\mathrm{d}v \tag{2}$$

It then follows that $F = 1$ in a cell fully occupied with fluid 1, $F = 0$ in a cell filled with fluid 2, and $0 < F < 1$ in a cell containing a portion of the interface.

The interface is tracked via the following advection equation:

$$\frac{\partial f}{\partial t} + \vec{u}\cdot\nabla f = 0 \tag{3}$$

The interface unit normal vector can be calculated from the gradient of f:

$$\hat{n} = \frac{\nabla f}{|\nabla f|} \tag{4}$$

and the curvature of the interface can then be computed as

$$\kappa = -\nabla\cdot\left(\frac{\nabla f}{|\nabla f|}\right) \tag{5}$$

For a 2D interface between two fluids, depicted in Fig. 1a, the discretized VOF function representing fluid 1 is shown in Fig. 1b. As can be seen, the volume fractions vary sharply from zero to one across the interface. This discontinuous behavior makes it difficult to accurately evaluate the first and second derivatives of f, which leads to inaccurate interface normals and curvatures. Smoothing the f field prior to evaluating ∇f improves the values [12]. We assess the accuracy of $\hat{n}$ and κ calculated from f in Section 2.3. But first, we briefly present the LS method, which is known to yield more accurate normals and curvatures.

2.2. Level set method

In the LS method, the interface is represented by a smooth function ϕ – called the LS function; for a domain Ω, ϕ is defined [15] as a signed distance to the boundary (interface) $\partial\Omega$

$$|\phi(\vec{x})| = \min(|\vec{x} - \vec{x_I}|) \quad \text{for all } \vec{x_I} \in \partial\Omega \tag{6}$$

implying that $\phi(\vec{x}) = 0$ on $\partial\Omega$. Choosing ϕ to be positive inside Ω, we then have

$$\phi(\vec{x}) = \begin{cases} > 0, & \vec{x} \in \Omega \\ 0, & \vec{x} \in \partial\Omega \\ < 0, & \vec{x} \notin \Omega \end{cases} \tag{7}$$

For the 2D interface depicted in Fig. 1a, the discretized LS function, defined at the center of each cell, is shown in Fig. 1c.

The unit normal vector and curvature at any point on the interface are calculated from ϕ by

$$\hat{n} = \frac{\nabla\phi}{|\nabla\phi|} \tag{8}$$

and

$$\kappa = -\nabla \cdot \left(\frac{\nabla\phi}{|\nabla\phi|}\right) \tag{9}$$

Since ϕ is smooth and continuous across the interface (see Fig. 1c), $\nabla\phi$ can be calculated accurately.

In the LS method, the motion of the interface is defined by the following advection equation:

$$\frac{\partial\phi}{\partial t} + \vec{u} \cdot \nabla\phi = 0 \tag{10}$$

When ϕ is advected, the $\phi = 0$ contour moves at the correct interface velocity; however, contours of $\phi \neq 0$ do not necessarily remain distance functions. This can result in an irregular ϕ field that in turn leads to problems

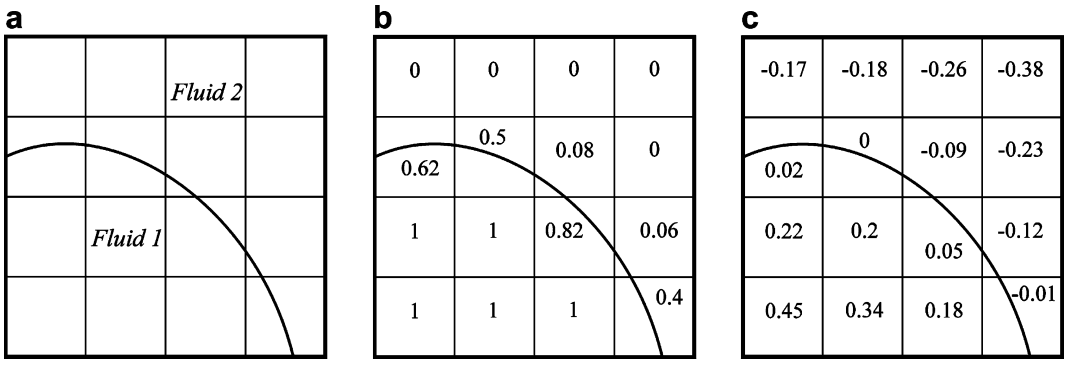

Fig. 1. (a) A 2D interface between fluids 1 and 2, (b) the discretized VOF function representing fluid 1 and (c) the level set function representing distance to the interface.

with mass conservation. To rectify this problem reinitialization methods have been developed, which restore ϕ to a signed distance function without changing the $\phi = 0$ contour.

There are different methods for recalculating a signed distance function [16]. The most popular is an algebraic approach [17]; a steady state solution is obtained to the following PDE at regular intervals:

$$\phi_\tau + S(\phi_0)(|\nabla\phi| - 1) = 0 \tag{11}$$

τ is a pseudo-time, ϕ_0 is the LS function prior to reinitialization, and S is the sign function. Although this approach improves conservation of mass considerably, it fails to achieve it exactly. Furthermore, an important practical question, posed by Peng et al. [18], is that of how often Eq. (11) should be solved, which has no simple answer [19].

To further alleviate the mass conservation problem, Nourgaliev et al. [20] showed that reducing the spatial discretization errors in Eq. (10), by either increasing the accuracy of the spatial discretization schemes and/or locally refining the mesh, improves mass conservation. Gómez et al. [19] too showed that using a locally refined LS approach improves mass conservation. The Fast Marching method [21] is another technique for calculating distance functions, that improves mass conservation properties. And as an example of a "hybrid" method (the LS method coupled with another approach) to address the mass conservation issue, the particle level set method [22] advects marker particles alongside the LS field, and uses the particles to reinitialize the LS function in under-resolved regions.

It must be noted that none of the above techniques exactly conserve mass. To eliminate mass conservation errors entirely, another hybrid scheme, the "coupled level set and VOF" (CLSVOF) method [23–25] has been developed, that advects the VOF function along with the LS function, and reinitializes the LS function from the reconstructed VOF field. The CLSVOF method achieves exact mass conservation if it is based on an exactly conservative VOF approach. For this reason, the CLSVOF method of Son and Hur [25] was implemented for this study. The VOF function is advected by the method of Youngs [26], which is volume conserving.

In the CLSVOF method, the LS and VOF functions ϕ and f are advected from time n to $n+1$; the interface, approximated as piecewise linear, is then reconstructed from f^{n+1} using the interface normal vectors calculated from ϕ^{n+1}. ϕ is then reinitialized by calculating the distance between any cell center (where ϕ is defined) and the VOF interface. In our implementation, the spatial derivatives in Eq. (10) were discretized using a second-order accurate, essentially non-oscillatory (ENO) scheme and the forward Euler scheme was used to discretize the temporal derivative.

2.3. Errors in $\hat{n}$ and κ calculated from VOF and LS functions

As a test problem, consider a circle of radius 0.15 centered at $(0.5, 0.5)$ in a 1×1 domain; the normals and curvatures are calculated from the VOF and LS functions. For any quantity q, the maximum error l_∞ and the average error l_1 are defined as follows:

$$l_\infty = \max_j |(q_{\text{cal.}} - q_{\text{exact}})_j| \tag{12}$$

$$l_1 = \frac{1}{N}\sum_{j=1}^{N} |(q_{\text{cal.}} - q_{\text{exact}})_j| \tag{13}$$

The circle is defined on a uniform mesh, and $\hat{n}$ and κ are calculated at cell vertices and cell centers, respectively. A second-order central differencing scheme is used to calculate ∇f and $\nabla\phi$ to obtain $\hat{n}$ via Eqs. (4) and (8). A second-order operator is also used to calculate $\nabla \cdot \hat{n}$ to compute κ. The l_∞ and l_1 errors are measured only in the N interfacial cells ($0 < F < 1$).

To evaluate $\hat{n}$ and κ from the VOF function, we first initialize the VOF field using recursive local mesh refinement. We refine to 16 levels in interfacial cells, which yields the volume fractions to machine precision. Further subdivisions have no significant effect on the results.

The errors associated with $\hat{n}$ calculated from the VOF function at different mesh resolutions are presented in Table 1. As can be seen, there is a constant error associated with $\hat{n}$, which does not vanish as the mesh is refined, i.e. the normals are zero-order accurate in space, $\hat{n}_{\text{cal.}} = \hat{n}_{\text{exact}} + O(\Delta x^0)$.

Table 1
The errors associated with the unit normal vectors calculated at cell vertices from the VOF function, for a circle of radius 0.15 centered at (0.5,0.5) in a 1×1 domain, at different mesh resolutions

Δx	l_∞	Order	l_1	Order
1/16	0.1520		0.0694	
		0.62		0.94
1/32	0.0992		0.0362	
		−0.14		−0.14
1/64	0.1091		0.0398	
		0.20		0.04
1/128	0.0951		0.0388	
		−0.12		−0.02
1/256	0.1030		0.0392	
		−0.09		0.04
1/512	0.1093		0.0382	
		0.07		−0.02
1/1024	0.1043		0.0386	

The errors in curvatures obtained from the VOF function are presented in Table 2. It can be seen that both l_∞ and l_1 grow linearly when increasing the mesh resolution, $\kappa_{\text{cal.}} = \kappa_{\text{exact}} + \mathrm{O}(1/\Delta x)$. Cummins et al. [12] reported a similar result, although using a different smoothing kernel. This is a serious drawback of the VOF function. Contrary to what one would hope, the accuracy of curvatures deteriorates with increasing mesh resolution. Although converging curvatures calculated from a smoothed VOF function have been reported (see [12] for details), they become computationally expensive as the resolution increases.

To assess $\hat{n}$ and κ from the LS function in a CLSVOF context, we first initialize the VOF field exactly as described above, and then calculate ϕ via the reinitialization procedure, where the normals used in the VOF reconstruction are obtained from the exact ϕ. Note that if we were to use the exact ϕ to calculate $\hat{n}$ and κ, we would only be assessing the accuracy of the operators used for calculating these quantities; this would not reflect the discretization errors associated with the reinitialization of ϕ.

The errors associated with the reinitialized ϕ, and the corresponding order of accuracy, are presented in Table 3. The errors in $\hat{n}$ and κ calculated from ϕ via Eqs. (8) and (9) are presented in Tables 4 and 5, respectively. As the results show, the reinitialized ϕ is second-order accurate, the unit normals are first-order accurate, and the curvatures are zero-order accurate (which implies that a constant error, ranging from 10% to 20%, is always associated with κ regardless of the mesh resolution). These errors are not unexpected because $\hat{n} \sim \nabla\phi$ and $\kappa \sim \nabla \cdot \nabla\phi$.

Table 2
The errors associated with curvatures calculated from the VOF function, for a circle of radius 0.15 centered at (0.5,0.5) in a 1×1 domain, at different mesh resolutions

Δx	l_∞	Order	l_1	Order
1/16	1.09		0.49	
		−0.41		−0.62
1/32	1.44		0.75	
		−1.32		−0.86
1/64	3.59		1.36	
		−1.04		−0.91
1/128	7.38		2.57	
		−1.18		−0.88
1/256	16.75		4.73	
		−1.03		−0.93
1/512	34.14		8.99	
		−1.01		−0.99
1/1024	69.00		17.80	

Table 3
The errors associated with the reinitialized LS function ϕ for a circle of radius 0.15 centered at $(0.5, 0.5)$ in a 1×1 domain, at different mesh resolutions

Δx	l_∞	Order	l_1	Order
1/16	3.03×10^{-3}		1.14×10^{-3}	
		2.27		2.04
1/32	6.30×10^{-4}		2.78×10^{-4}	
		1.71		2.46
1/64	1.93×10^{-4}		5.06×10^{-5}	
		1.86		1.68
1/128	5.33×10^{-5}		1.58×10^{-5}	
		2.03		2.19
1/256	1.31×10^{-5}		3.46×10^{-6}	
		1.88		2.11
1/512	3.54×10^{-6}		8.00×10^{-7}	
		1.82		1.88
1/1024	1.00×10^{-6}		2.18×10^{-7}	

Table 4
The errors associated with the unit normal vectors calculated at cell vertices from the LS function ϕ, for a circle of radius 0.15 centered at $(0.5, 0.5)$ in a 1×1 domain, at different mesh resolutions

Δx	l_∞	Order	l_1	Order
1/16	6.16×10^{-2}		1.58×10^{-2}	
		1.68		2.38
1/32	1.92×10^{-2}		3.03×10^{-3}	
		0.96		0.35
1/64	9.85×10^{-3}		2.38×10^{-3}	
		0.61		0.83
1/128	6.47×10^{-3}		1.34×10^{-3}	
		0.93		1.03
1/256	3.39×10^{-3}		6.56×10^{-4}	
		1.02		1.17
1/512	1.67×10^{-3}		2.92×10^{-4}	
		0.70		0.99
1/1024	1.03×10^{-3}		1.48×10^{-4}	

Table 5
The errors associated with curvatures calculated from the LS function ϕ, for a circle of radius 0.15 centered at $(0.5, 0.5)$ in a 1×1 domain, at different mesh resolutions

Δx	l_∞	Order	l_1	Order
1/16	0.5472		0.2963	
		1.55		1.58
1/32	0.1875		0.0991	
		−1.79		−0.51
1/64	0.6481		0.1407	
		0.61		−0.11
1/128	0.4234		0.1518	
		−0.43		−0.02
1/256	0.5689		0.1537	
		−0.29		0.34
1/512	0.6963		0.1215	
		0.11		−0.01
1/1024	0.6453		0.1227	

Note that the order of accuracy of the CLSVOF method used in this work is consistent [27] with the accuracy of a similar model developed by Sussman and Puckett [24]. Although $\hat{n}$ and κ calculated from the LS function are much more accurate than those calculated from the VOF function, neither approach yields converging curvatures.

Finally, it should be noted that these results only apply to the LS function in a CLSVOF context. Other methods for constructing a distance function (discussed in Section 2.2) may yield converging curvatures, even with second-order accuracy; however, such methods will fail to exactly conserve mass.

The objective, then, of this work was to devise a method to calculate second-order accurate, or at least converging, curvatures. Such a method is presented next, in which interface normals are advected with the flow, and curvatures are calculated directly from the advected normals.

3. Advecting normals: a new method for calculating interface normals and curvatures

3.1. Mathematical fundamentals

As reviewed earlier, the evolution of the LS function is governed by Eq. (10)

$$\frac{\partial \phi}{\partial t} + \vec{u} \cdot \nabla \phi = 0$$

Defining $\vec{N} = \nabla \phi$ as the vector normal to the contours of ϕ, the above equation can be rewritten as

$$\frac{\partial \phi}{\partial t} + \vec{u} \cdot \vec{N} = 0 \tag{14}$$

Taking the gradient of Eq. (14), we obtain

$$\frac{\partial \vec{N}}{\partial t} + \nabla\left(\vec{u} \cdot \vec{N}\right) = 0 \tag{15}$$

Eq. (15) is the advection equation for normals. In 2D Cartesian coordinates, Eq. (15) results in the following equations:

$$\frac{\partial N_x}{\partial t} + \frac{\partial}{\partial x}(uN_x + vN_y) = 0 \tag{16}$$

and

$$\frac{\partial N_y}{\partial t} + \frac{\partial}{\partial y}(uN_x + vN_y) = 0 \tag{17}$$

Next, consider the following lemma [28]:

Let $u_n = \vec{u} \cdot \nabla \phi$ be the normal velocity of each level set, and set $\phi(\vec{x}, 0)$ to be the signed distance function. Then ϕ remains a signed distance function if and only if $\nabla u_n \cdot \nabla \phi = 0$.

The condition $\nabla u_n \cdot \nabla \phi = 0$ can be also expressed as

$$\nabla(\vec{u} \cdot \vec{N}) \cdot \vec{N} = 0 \tag{18}$$

Note that $|\nabla \phi| = |\vec{N}| = 1$. Now, from Eq. (15) we obtain

$$\frac{\partial \vec{N}}{\partial t} \cdot \vec{N} + \nabla(\vec{u} \cdot \vec{N}) \cdot \vec{N} = 0 \tag{19}$$

or

$$\frac{1}{2}\frac{\partial}{\partial t}(|\vec{N}|^2) + \nabla(\vec{u} \cdot \vec{N}) \cdot \vec{N} = 0 \tag{20}$$

If the condition in the above lemma is to be met (i.e. if Eq. (18) is to be satisfied), then we have

$$\frac{\partial}{\partial t}(|\vec{N}|^2) = 0 \tag{21}$$

and since initially $|\nabla\phi| = |\vec{N}| = 1$, then $\vec{N}$ remains a unit vector (this can be also deduced from the above lemma directly). In other words, if $\vec{N}$ is initially a unit vector and we solve Eq. (15) to advect $\vec{N}$ while satisfying Eq. (18), then $\vec{N}$ remains a unit vector. In a sense, satisfying Eq. (18) is similar to reinitialization in the LS method. The method for satisfying Eq. (18) is presented in Section 3.3.

In our proposed method, interface normals are advected via Eq. (15) along with the interface, although the methods for advecting normals and the interface are independent. We use the VOF method to advect the interface but this is simply our choice of implementation, and other methods could be used. Eqs. (3) and (15) are solved side by side; however, for the VOF reconstruction we use the advected normals rather than normals calculated from ∇f. The VOF function and $\vec{N}$ are prevented from decoupling by satisfying Eq. (18) and by the method of defining $\vec{N}$ (Section 3.2).

The interface curvature at any point is then obtained directly from the advected $\vec{N}$ (from this point on we use $\vec{N}$ instead of $\hat{n}$)

$$\kappa = -\nabla \cdot \vec{N} \tag{22}$$

Before presenting the numerical methodology, it must be noted that the idea of advecting interface normals may appear similar to the CIP (cubic interpolated polynomial or constrained interpolation profile) method [29,30], where a color function and its spatial derivatives are advected. However, our proposed method is different for the following reasons:

- In the proposed method, interface unit normal vectors are advected rather than the spatial derivatives of the VOF function. Although unit normals can be obtained from VOF derivatives, they are not the same. Note that normals are initialized as unit vectors and through a special procedure we ensure that they remain unit vectors. In the CIP method, no constraint on the derivatives is applied.
- Rather than advecting the fluid indicator function, in the CIP method a transformed function is advected to minimize numerical diffusion. The transformation is usually via a tangent function [30], and an inverse transform is used to restore the original variable. The tangent transformation performs well only when used with high-order (third or higher) schemes; otherwise, significant dispersion or dissipation errors occur. In our method, we advect the actual indicator function.
- Different than the CIP method, the proposed method involves interface reconstructions that are used for advecting mass (and momentum when implemented in a flow solver). The CIP method does not exactly conserve mass [31] when used for modeling two-phase flows. Although a mass conservative version of the CIP method has been reported [32], to the best of our knowledge it has not been applied to two-phase flows. The proposed method is inherently mass conservative because it utilizes a volume-conserving VOF method.

The numerical methods for solving Eq. (15) are detailed next.

3.2. Numerical methodology

Eq. (15) represents an initial value problem, where the initial value of $\vec{N}$ is determined from the initial geometry of an interface. We directly solve the conservative form of Eq. (15) (in the CIP method the evolution equation for derivatives is solved by splitting the equation into "advection" and "non-advection" parts). The temporal derivative of Eq. (15) is discretized using the third-order total variation diminishing (TVD) Runge–Kutta method [33]. The spatial derivatives in Eq. (15) are discretized using a weighted essentially non-oscillatory (WENO) scheme [15]. However, before evaluating the spatial derivatives, Eq. (18) is satisfied, as detailed in the next section.

In our implementation, normal vectors $\vec{N}$ are located at cell vertices (see Fig. 2); normal vectors at cell centers, which are required for interface reconstruction, are obtained by interpolation. As shown in Fig. 2, we

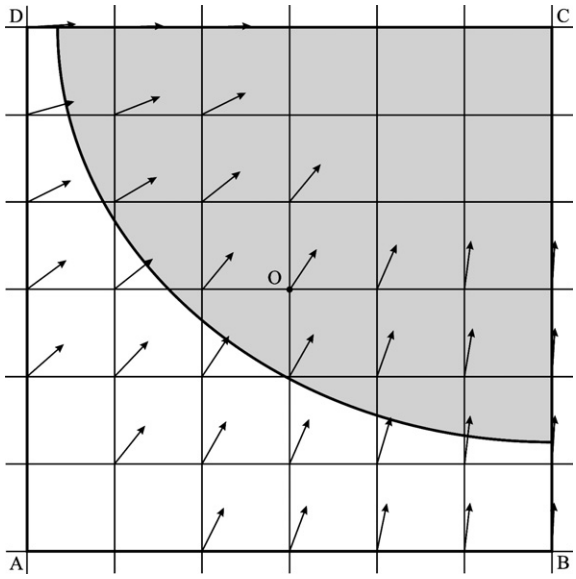

Fig. 2. Normal vectors are defined at cell vertices in a $3\Delta x$ band around the interface. The 6×6 stencil ABCD is used to extend $\vec{u}\cdot\vec{N}$ around point O.

define $\vec{N}$ in a $3\Delta x$ band around the interface defined by the VOF function. Although $\vec{N}$ are only required at the vertices of interfacial cells in order to reconstruct interfaces, when implemented in a flow model, one might require curvatures in neighboring cells as well. Thus, the band spans at least three cells.

3.3. Extension algorithm

To satisfy Eq. (18) we use an "extension" algorithm. This algorithm extends $\vec{u}\cdot\vec{N}$ off the interface in both directions such that the gradient is zero normal to the interface, $\partial(\vec{u}\cdot\vec{N})/\partial\vec{N}=0$. This idea was introduced by Zhao et al. [28] and further developed by Chen et al. [34]. It was used, for example, by Peng et al. [18] to extend interface velocity and by Smereka [35] to extend surface curvature away from the interface. We adopt the same algorithm [18] here but newer extension algorithms [36,37] are also applicable.

In this algorithm, the following hyperbolic PDE is solved to steady state:

$$\frac{\partial q}{\partial \tau}+S(\vec{x})\vec{N}\cdot\nabla q=0, \tag{23}$$

where $q=\vec{u}\cdot\vec{N}$ (q can be any quantity to be extended so that $\partial q/\partial\vec{N}=0$), τ is a pseudo-time, and $S(\vec{x})$ is the sign function defined as

$$S(\vec{x})=\begin{cases}1, & \text{if } \vec{x}\in\Omega\\ 0, & \text{if } \vec{x}\in\partial\Omega\\ -1, & \text{if } \vec{x}\notin\Omega\end{cases} \tag{24}$$

Note that $\vec{N}$ (the normal to the interface $\partial\Omega$) points into Ω. If q on the interface is used as the initial condition for Eq. (23), then the steady solution is the extension of q off the interface, which satisfies $\partial q/\partial\vec{N}=0$.

We use a second-order ENO scheme to discretize the spatial derivative in Eq. (23), and the forward Euler scheme for the temporal derivative. Following [18], the discretized form of Eq. (23) in 2D Cartesian coordinates is

$$q_{ij}^{n+1}=q_{ij}^{n}-\Delta\tau\{(SN_x)_{ij}^{+}D_x^{-}q_{ij}+(SN_x)_{ij}^{-}D_x^{+}q_{ij}+(SN_y)_{ij}^{+}D_y^{-}q_{ij}+(SN_y)_{ij}^{-}D_y^{+}q_{ij}\}^{n} \tag{25}$$

where $(x)^+ = \max(x, 0)$, $(x)^- = \min(x, 0)$, and D_x^+ and D_x^- are ENO operators for calculating the first derivative with respect to x in the forward and backward directions, respectively. No smoothing parameter is used to discretize $S(\vec{x})$.

Eq. (23) is solved locally at each point where $\vec{N}$ is defined. As previously mentioned, we use a WENO scheme to discretize Eq. (15), which requires a 6-cell stencil (three upstream and two downstream of a cell). Therefore, as shown in Fig. 2, we solve Eq. (23) on a 6×6 "extension stencil" centered at any point around which $\vec{u} \cdot \vec{N}$ is to be extended.

Eq. (23) requires normals to indicate the direction of information flow. However, since normals are only defined in a band around the interface, there can be some grid points in an extension stencil that have no normal vector (see Fig. 2). Also, as an interface moves, points which were originally outside the band require an initial value of $\vec{N}$. We assign normal vectors to these points by extending $\vec{N}$, using the same PDE (23) for the components of $\vec{N}$: we first copy the normals in the band to the points outside in order to obtain an initial guess of an $\vec{N}$ field. Then Eq. (23) is solved, but after each iteration we update normals at the points where $\vec{N}$ was not originally defined. Once normals are fully extended in the stencil (steady solution), they are then utilized as the direction for extending $\vec{u} \cdot \vec{N}$.

Note that as an alternative to copying the normals, one could extrapolate them to the points outside the band. This could be done by using the algebraic properties of $\vec{N}$, where in 2D, for example,

$$\frac{\partial^2 \phi}{\partial x \partial y} = \frac{\partial N_x}{\partial y} = \frac{\partial N_y}{\partial x} \tag{26}$$

Finally, the following are a few remarks on the extension algorithm:

- The quality of the solution to Eq. (23) for extending $\vec{u} \cdot \vec{N}$ is directly related to how well we satisfy Eq. (18). As mentioned before, satisfying Eq. (18) is similar to reinitialization in the LS method. If reinitialization is done by solving Eq. (11), errors (residuals) in the steady solution, which are inevitable, result in the distortion of the LS contours and consequently lead to violation of mass conservation. In the proposed method, errors in the steady solution of Eq. (23) have no effect on mass conservation.
- In the hyperbolic PDE (23), $\vec{N}$ plays the role of a characteristic velocity, the magnitude of which is unity, $|\vec{N}| = 1$. When solving Eq. (23), given the mesh size, we choose the pseudo-timestep such that the Courant number is 0.25.
- In instances where the interfaces are under-resolved or merging, there will exist normals in an extension stencil that are associated with different interfaces. In this case, if one uses all of the normal vectors in the stencil as the initial condition without concern for which interface they belong to, then one may not obtain a steady or meaningful solution to Eq. (23). The initial condition must include only those normals which are associated with a single interface.

4. Results

The remainder of this paper presents the results of tests of the new methodology, which we denote as the "$\vec{N}$ method". Using a uniform mesh, we consider only test cases where the velocity fields are prescribed. The results of the $\vec{N}$ method are compared with results obtained from the VOF and LS functions. Note that the only difference lies in the approach to calculating $\hat{n}$.

We also present errors in the volume fraction field:

$$l_\infty = \max_j |(F - F_{\text{exact}})_j| \tag{27}$$

$$l_1 = \frac{1}{N} \sum_{j=1}^{N} |(F - F_{\text{exact}})_j| \tag{28}$$

These errors are calculated over all cells in the computational domain, not just the interface cells. Note that the mass is exactly conserved in the relevant cases.

4.1. Static circle test

First, we return to the problem of Section 2.3. Table 6 presents the errors associated with κ calculated from $\vec{N}$ that are exactly specified. As one would expect, the curvatures are second-order accurate and dramatically better than the curvatures calculated from either the VOF (Table 2) or LS (Table 5) functions.

4.2. Translation test

Consider the translation of a circle of radius 0.15 centered initially at $(0.25, 0.5)$ in a 1×1 domain (Fig. 3a); the velocity field is $(u, v) = (1, 0)$. The circle is advected to $(0.75, 0.5)$ at different mesh resolutions; the Courant number is always 0.125. We study cases where $\hat{n}$ and κ are calculated from the VOF and LS functions, and compare these with results of the $\vec{N}$ method.

Table 7 shows the errors associated with $\hat{n}$ and κ at the end of translation. As Table 7, Panel A shows, the normals calculated from the VOF function do not converge, and curvature errors grow with mesh refinement, similar to what was observed in the static circle test in Section 2.3. When the LS function is used (Table 7, Panel B), $\hat{n}$ converges and is first-order accurate; however, the l_∞ errors of κ increase with mesh refinement while, similar to the static circle test, l_1 errors remain almost constant. The points where curvature error is maximum lie on the leading edge of the circle, approximately along $\pm 45^\circ$ diagonals. Table 7, Panel C presents errors when the $\vec{N}$ method is used. Both $\hat{n}$ and κ converge with almost second-order accuracy.

The final location of the circle and the corresponding volume fractions are exactly known, and are used to calculate the errors associated with the volume fractions at the end of translation (Table 8). For all methods, the volume fraction errors decrease with mesh refinement; however, when the $\vec{N}$ method is used (Table 8, Panel C), the errors are smaller and the convergence rate is slightly faster.

4.3. Rotation test

Consider a circle of radius 0.15 centered at $(0.75, 0.5)$ in a 1×1 domain (Fig. 3b). An angular velocity $\omega = 1$ is specified about $(0.5, 0.5)$. The circle is advected 2π radians at different mesh resolutions, at a maximum Courant number of $\pi/50$. Again, we study cases where $\hat{n}$ and κ are calculated from the $\vec{N}$ method and from the VOF and LS functions.

Table 9 shows the errors associated with $\hat{n}$ and κ at the end of one rotation. Again, when the VOF function is used (Table 9, Panel A), the normals do not converge, and the curvature errors grow with mesh refinement. The LS function yields normals which are first-order accurate (Table 9, Panel B); the errors associated with κ

Table 6
The errors associated with curvatures calculated by the $\vec{N}$ method, for a circle of radius 0.15 centered at $(0.5, 0.5)$ in a 1×1 domain, at different mesh resolutions

Δx	l_∞	Order	l_1	Order
1/16	3.87×10^{-1}		2.11×10^{-2}	
		1.72		1.73
1/32	1.18×10^{-1}		6.33×10^{-2}	
		2.11		2.20
1/64	2.73×10^{-2}		1.38×10^{-2}	
		2.07		1.94
1/128	6.51×10^{-3}		3.60×10^{-3}	
		1.95		2.03
1/256	1.68×10^{-3}		8.83×10^{-4}	
		1.97		2.00
1/512	4.29×10^{-4}		2.21×10^{-4}	
		2.02		2.01
1/1024	1.06×10^{-4}		5.50×10^{-5}	

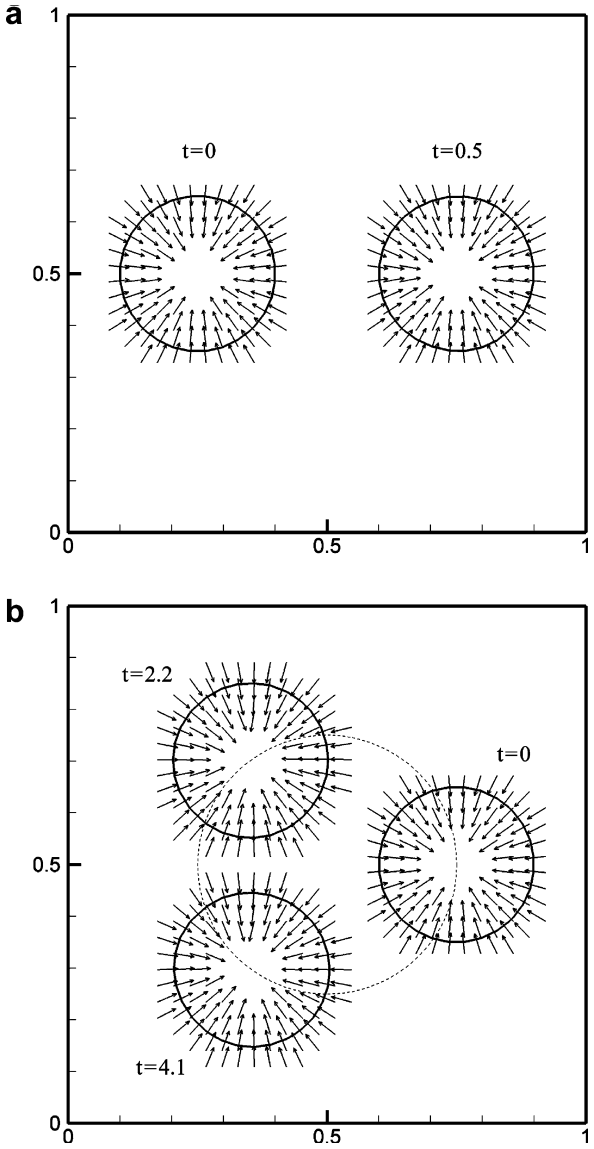

Fig. 3. Interface geometry and normal vectors $\vec{N}$ in (a) translation and (b) rotation of a circle when the $\vec{N}$ method is used. $\Delta x = 1/32$.

remain almost constant with mesh refinement. As Table 9, Panel C shows, when the $\vec{N}$ method is used, the normals are second-order accurate, and curvatures converge with a good order of accuracy, although not quite at second-order.

The errors associated with the volume fractions at the end of one rotation are presented in Table 10. All errors decrease with refinement; however, the magnitudes are smaller when the $\vec{N}$ method is used.

4.4. Shrinking/expanding circle test

Next, we consider shrinking and expanding a well resolved circle, and expanding a poorly resolved circle. In each case, we use the $\vec{N}$ method and the VOF and LS functions to calculate $\hat{n}$, and we study the errors associated with $\hat{n}$, κ and the volume fractions.

First, consider a circle of radius 0.25 centered at $(x_c, y_c) = (0.5, 0.5)$ in a 1×1 domain. The following velocity field is specified in the domain:

Table 7
The errors associated with $\hat{n}$ and κ calculated from (Panel A) the VOF function, (Panel B) the LS function and (Panel C) the $\vec{N}$ method, for a circle of radius 0.15 after being translated at an x-velocity of 1 for 0.5 at a Courant number of 0.125

Δx	$\hat{n}$				κ			
	l_∞	Order	l_1	Order	l_∞	Order	l_1	Order
Panel A								
1/32	1.397×10^{-1}		4.65×10^{-2}		2.275		0.996	
		0.24		0.24		−0.70		−0.55
1/64	1.184×10^{-1}		3.95×10^{-2}		3.730		1.460	
		−0.02		0.10		−1.12		−0.80
1/128	1.203×10^{-1}		3.69×10^{-2}		8.060		2.540	
Panel B								
1/32	2.746×10^{-1}		5.30×10^{-2}		5.753		1.561	
		0.75		1.26		−0.76		0.19
1/64	1.638×10^{-1}		2.22×10^{-2}		9.775		1.370	
		0.90		1.18		−0.41		0.04
1/128	8.80×10^{-2}		9.82×10^{-3}		12.986		1.332	
Panel C								
1/32	1.359×10^{-1}		2.96×10^{-2}		3.340		0.527	
		1.63		1.72		1.83		1.44
1/64	4.44×10^{-2}		9.00×10^{-3}		0.941		0.194	
		1.87		1.63		1.87		1.35
1/128	1.20×10^{-2}		2.91×10^{-3}		0.258		0.076	

The circle is initially centered at $(0.25, 0.5)$ in a 1×1 domain.

Table 8
Volume fraction errors in the translation test; normals $\hat{n}$ are calculated from (Panel A) the VOF function, (Panel B) the LS function and (Panel C) the $\vec{N}$ method

Δx	l_∞	Order	l_1	Order
Panel A				
1/32	8.67×10^{-2}		6.92×10^{-4}	
		0.18		1.83
1/64	7.66×10^{-2}		1.94×10^{-4}	
		0.91		2.05
1/128	4.07×10^{-2}		4.70×10^{-5}	
Panel B				
1/32	1.68×10^{-1}		1.04×10^{-3}	
		0.86		1.95
1/64	9.23×10^{-2}		2.69×10^{-4}	
		0.46		2.43
1/128	6.73×10^{-2}		5.0×10^{-5}	
Panel C				
1/32	5.08×10^{-2}		4.10×10^{-4}	
		0.60		1.90
1/64	3.36×10^{-2}		1.10×10^{-4}	
		1.23		2.46
1/128	1.43×10^{-2}		2.0×10^{-5}	

$$u(x,y) = -\frac{x - x_c}{\sqrt{(x - x_c)^2 + (y - y_c)^2}} \tag{29}$$

$$v(x,y) = -\frac{y - y_c}{\sqrt{(x - x_c)^2 + (y - y_c)^2}} \tag{30}$$

Table 9
The errors associated with $\hat{n}$ and κ calculated from (Panel A) the VOF function, (Panel B) the LS function and (Panel C) the $\vec{N}$ method, for a circle of radius 0.15 after being rotated for 2π radians at an angular velocity of 1 and at a maximum Courant number of $\pi/50$

Δx	$\hat{n}$				κ			
	l_∞	Order	l_1	Order	l_∞	Order	l_1	Order
Panel A								
1/32	1.464×10^{-1}		5.16×10^{-2}		2.155		0.837	
		0.26		0.31		−0.94		−0.75
1/64	1.226×10^{-1}		4.17×10^{-2}		4.127		1.408	
		−0.02		0.09		−1.01		−0.91
1/128	1.243×10^{-1}		3.93×10^{-2}		8.330		2.653	
Panel B								
1/32	1.662×10^{-1}		3.35×10^{-2}		3.256		0.571	
		1.19		1.38		−0.43		−0.27
1/64	7.28×10^{-2}		1.29×10^{-2}		4.388		0.690	
		1.09		1.59		0.41		0.51
1/128	3.41×10^{-2}		4.28×10^{-3}		3.296		0.485	
Panel C								
1/32	1.506×10^{-1}		4.36×10^{-2}		1.867		0.408	
		3.09		2.81		2.87		2.54
1/64	1.77×10^{-2}		6.20×10^{-3}		0.256		0.070	
		2.11		2.41		0.85		1.22
1/128	4.11×10^{-3}		1.17×10^{-3}		0.142		0.030	

The circle, initially centered at $(0.75, 0.5)$ in a 1×1 domain, rotates around the centre of the domain.

Table 10
Volume fraction errors in the rotation test; normals $\hat{n}$ are calculated from (Panel A) the VOF function, (Panel B) the LS function and (Panel C) the $\vec{N}$ method

Δx	l_∞	Order	l_1	Order
Panel A				
1/32	1.783×10^{-1}		2.85×10^{-3}	
		0.53		1.91
1/64	1.231×10^{-1}		7.59×10^{-4}	
		0.24		1.64
1/128	1.039×10^{-1}		2.44×10^{-4}	
Panel B				
1/32	2.376×10^{-1}		2.90×10^{-3}	
		1.36		1.93
1/64	9.24×10^{-2}		7.59×10^{-4}	
		0.59		2.07
1/128	6.15×10^{-2}		1.81×10^{-4}	
Panel C				
1/32	1.125×10^{-1}		1.81×10^{-3}	
		0.76		1.72
1/64	6.65×10^{-2}		5.49×10^{-4}	
		0.82		1.86
1/128	3.77×10^{-2}		1.51×10^{-4}	

This is a radial velocity with a magnitude of one ($u_r = -1$) which shrinks the circle; see Fig. 4a. Since $\nabla \cdot \vec{u} \neq 0$ at (x_c, y_c), we modified the volume advection algorithm in the vicinity of (x_c, y_c). The interface is advected until $t = 0.125$, and the Courant number is 0.25.

Table 11 presents errors associated with $\hat{n}$ and κ for different methods. Again, we see that the VOF function yields non-converging normals, and the errors in curvatures grow with refinement (Table 11, Panel A). As

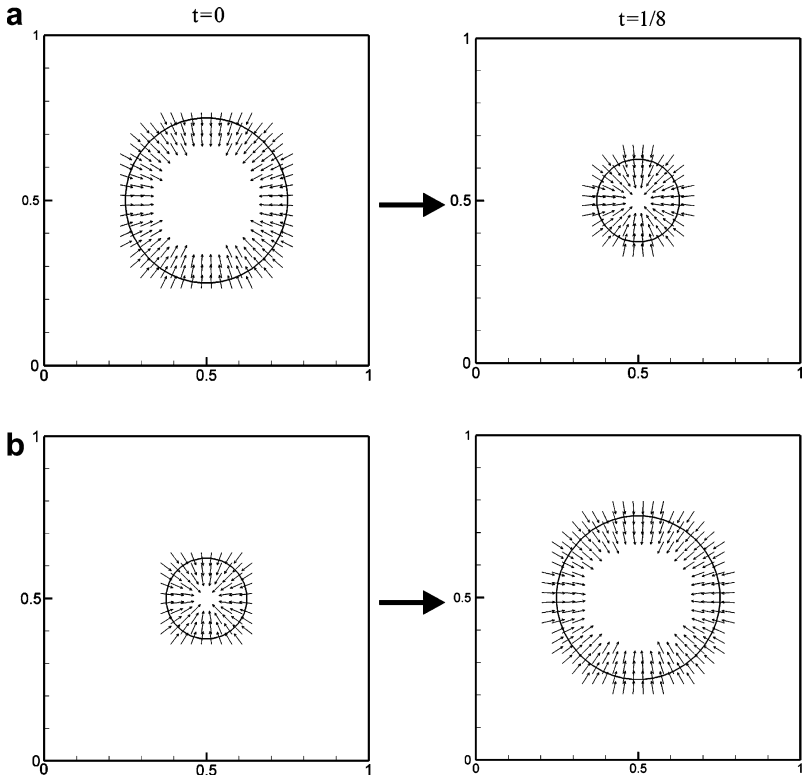

Fig. 4. Interface geometry and normal vectors $\vec{N}$ for (a) a shrinking and (b) an expanding circle, when the $\vec{N}$ method is used. $\Delta x = 1/32$, $\Delta t = 1/128$.

Table 11
The errors associated with $\hat{n}$ and κ for a shrinking circle; normals $\hat{n}$ are calculated from (Panel A) the VOF function, (Panel B) the LS function and (Panel C) the $\vec{N}$ method

Δx	$\hat{n}$				κ			
	l_∞	Order	l_1	Order	l_∞	Order	l_1	Order
Panel A								
1/32	8.28×10^{-2}		3.32×10^{-2}		1.335		0.689	
		−0.41		−0.42		−0.95		−1.12
1/64	1.098×10^{-1}		4.45×10^{-2}		2.579		1.499	
		0.11		0.31		−1.44		−0.79
1/128	1.019×10^{-1}		3.58×10^{-2}		6.982		2.596	
		−0.03		−0.05		−1.29		−0.93
1/256	1.039×10^{-1}		3.70×10^{-2}		17.031		4.947	
Panel B								
1/32	5.32×10^{-2}		2.11×10^{-2}		1.338		0.507	
		1.18		1.93		0.18		0.80
1/64	2.35×10^{-2}		5.54×10^{-3}		1.182		0.291	
		1.49		1.48		1.06		0.56
1/128	8.36×10^{-3}		1.99×10^{-3}		0.567		0.197	
		0.87		1.12		0.23		−0.11
1/256	4.56×10^{-3}		9.15×10^{-4}		0.485		0.213	
Panel C								
1/32	2.13×10^{-2}		6.12×10^{-3}		0.656		0.333	
		1.96		1.48		2.13		1.84
1/64	5.48×10^{-3}		2.19×10^{-3}		0.150		0.093	
		1.58		1.57		1.56		1.63
1/128	1.83×10^{-3}		7.39×10^{-4}		0.051		0.030	
		1.95		1.84		1.28		1.91
1/256	4.73×10^{-4}		2.06×10^{-4}		0.021		0.008	

expected, the normals calculated from the LS function are first-order accurate (Table 11, Panel B), but the curvature errors remain constant with refinement. As Table 11, Panel C shows, the $\vec{N}$ method yields second-order normals, and curvatures that converge with a good order of accuracy.

The volume fraction errors of the circle at $t = 0.125$ are presented in Table 12. When $\hat{n}$ is calculated from the VOF function, the l_∞ errors vary little at different mesh resolutions, although the l_1 errors converge. These errors decrease with mesh refinement when $\hat{n}$ is calculated from the LS function or the $\vec{N}$ method. However, when the $\vec{N}$ method is used, the volume fractions are most accurate.

Now, consider an expanding circle ($u_r = 1$) of initial radius 0.125 centered at $(0.5, 0.5)$; see Fig. 4b. Again, the interface is advected until $t = 0.125$, and the Courant number is 0.25. Table 13 shows the errors associated with $\hat{n}$ and κ at $t = 0.125$. The results from the VOF and LS functions are as before: non-converging normals and diverging curvatures from the VOF function, and first-order accurate normals and non-converging curvatures from the LS function. However, we see from Table 13, Panel C that the convergence rate of normals calculated by the $\vec{N}$ method is now reduced to first-order. Furthermore, curvatures do not converge as well as before, and there is an anomaly in the l_∞ error at $\Delta x = 1/128$; the error arises at interfacial cells located along $\pm 25^\circ$ and $\pm 155^\circ$ diagonals.

The reason for this change in accuracy is that, unlike the normals from the VOF or LS functions, there is a time history associated with normals when the $\vec{N}$ method is employed. The accuracy of normals at a given timestep depends on the accuracy at the previous timestep(s), whereas when the VOF or LS functions are used, new normals are calculated at each timestep. Thus, if $\vec{N}$ is initially inaccurate, subsequent values of $\vec{N}$ will reflect this inaccuracy. In the case of the expanding circle, the normals are advected from a small, not-well-resolved circle to a large one; thus, the accuracy of normals (and curvatures) on the expanded circle are only as good as the ones defined on the initial circle.

In the extreme case, we study the expansion of a very poorly resolved circle for $t = 0.125$. The radius of the circle is initially 0.02, resolved by $\Delta x = 1/32$; thus, the circle is defined within a 2×2 stencil about a

Table 12
The errors associated with volume fractions for a shrinking circle; normals $\hat{n}$ are calculated from (Panel A) the VOF function, (Panel B) the LS function and (Panel C) the $\vec{N}$ method

Δx	l_∞	Order	l_1	Order
Panel A				
1/32	2.84×10^{-2}		4.14×10^{-4}	
		0.36		1.41
1/64	2.21×10^{-2}		1.66×10^{-4}	
		−0.35		1.20
1/128	2.82×10^{-2}		7.24×10^{-5}	
		0.15		1.11
1/256	2.54×10^{-2}		3.36×10^{-5}	
Panel B				
1/32	2.98×10^{-2}		4.10×10^{-4}	
		1.41		2.08
1/64	1.12×10^{-2}		9.67×10^{-5}	
		0.71		2.04
1/128	6.87×10^{-3}		2.35×10^{-5}	
		0.79		2.06
1/256	3.97×10^{-3}		5.62×10^{-6}	
Panel C				
1/32	1.83×10^{-2}		2.84×10^{-4}	
		1.05		2.39
1/64	8.86×10^{-3}		5.43×10^{-5}	
		0.87		1.85
1/128	4.85×10^{-3}		1.51×10^{-5}	
		0.92		1.94
1/256	2.57×10^{-3}		3.93×10^{-6}	

Table 13
The errors associated with $\hat{n}$ and κ for an expanding circle; normals $\hat{n}$ are calculated from (Panel A) the VOF function, (Panel B) the LS function and (Panel C) the $\vec{N}$ method

Δx	$\hat{n}$				κ			
	l_∞	Order	l_1	Order	l_∞	Order	l_1	Order
Panel A								
1/32	1.271×10^{-1}		4.16×10^{-2}		1.173		0.679	
		0.32		0.22		−1.59		−0.99
1/64	1.015×10^{-1}		3.57×10^{-2}		3.527		1.347	
		−0.02		−0.07		−1.21		−0.88
1/128	1.028×10^{-1}		3.74×10^{-2}		8.180		2.479	
		−0.05		0.05		−1.06		−0.87
1/256	1.061×10^{-1}		3.62×10^{-2}		17.070		4.521	
Panel B								
1/32	1.59×10^{-2}		6.29×10^{-3}		0.386		0.127	
		0.44		0.93		0.11		−0.48
1/64	1.17×10^{-2}		3.29×10^{-3}		0.358		0.177	
		1.27		1.24		−0.20		0.31
1/128	4.86×10^{-3}		1.39×10^{-3}		0.410		0.143	
		0.87		1.25		−0.07		0.25
1/256	2.65×10^{-3}		5.83×10^{-4}		0.429		0.120	
Panel C								
1/32	4.97×10^{-3}		1.83×10^{-3}		0.075		0.041	
		0.93		1.09		0.84		0.83
1/64	2.60×10^{-3}		8.61×10^{-4}		0.042		0.023	
		1.13		1.23		−0.13		1.06
1/128	1.19×10^{-3}		3.67×10^{-4}		0.046		0.011	
		1.41		1.00		1.62		1.14
1/256	4.47×10^{-4}		1.83×10^{-4}		0.015		0.005	

cell vertex at $(0.5, 0.5)$. The results are displayed in Fig. 5. When the VOF or LS functions are used (Fig. 5a and b), the expanded circles are relatively round; however, with the $\vec{N}$ method, the final geometry is deformed and asymmetric; see Fig. 5c. The deformed circle is horizontally symmetric, but the vertical asymmetry is due to the algorithm for extending normals when solving for the evolution of $\vec{N}$: $\vec{N}_x$ is extended first, and then $\vec{N}_y$ (the asymmetry is developed even if the order is alternated). The errors associated with the volume fractions of the expanded circle are presented in Table 14. As expected from Fig. 5, the volume fractions are the least accurate when the $\vec{N}$ method is used; the best result is obtained from the LS function.

At this resolution, the poorly resolved circle is initially reconstructed as a diamond (Fig. 5). This test then reveals a weakness of the $\vec{N}$ method in dealing with corners. A normal to a corner is mathematically undefined. However, numerically, at a point equidistant from both sides of a sharp corner, $\vec{N}$ can point to one side or the other, or can lie along the bisector. The latter, which gives a smooth variation of $\vec{N}$, is ideal, and can be initialized but not necessarily maintained when solving for the evolution of $\vec{N}$. Depending on which side of a corner $\vec{N}$ points to, the evolution of $\vec{N}$, and thus the shape of the interface, will be different. A remedy for this issue might be to use normals from the VOF function at early stages of interface evolution. Once the circle expanded to some minimum resolution (mesh per radius), one could then switch to the $\vec{N}$ method and at the transition, use normals calculated from the VOF function as the initial condition for the evolution equation of $\vec{N}$.

4.5. Vortex test

Finally, we consider a problem that includes shear. The following velocity field is specified in a 1×1 domain:

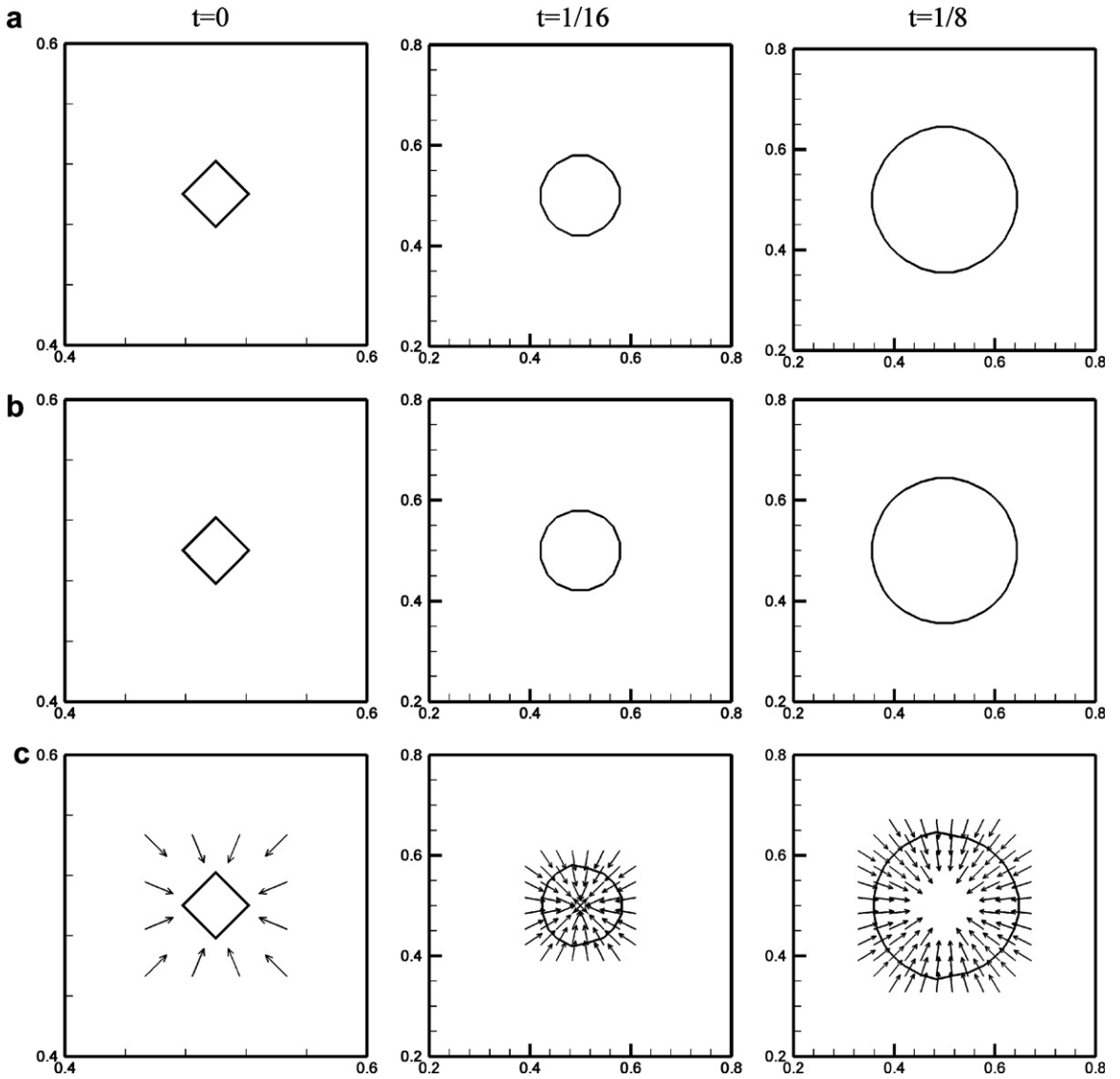

Fig. 5. Expansion of a poorly resolved circle when (a) VOF function, (b) LS function and (c) $\vec{N}$ is used for calculating $\hat{n}$. $R_{\text{initial}} = 0.02$, $\Delta x = 1/32$.

Table 14
The errors associated with volume fractions when a poorly resolved circle is expanded

	l_∞	l_1
VOF	5.15×10^{-2}	8.90×10^{-4}
LS	1.85×10^{-2}	2.94×10^{-4}
$\vec{N}$	1.37×10^{-1}	1.78×10^{-3}

$$u = \sin^2(\pi x)\sin(2\pi y) \tag{31}$$

$$v = -\sin^2(\pi y)\sin(2\pi x) \tag{32}$$

A circle of radius 0.15 is initially placed at $(0.5, 0.75)$; see Fig. 6. The circle is advected to $t = 1$; then the sign of the velocity field is reversed and advection continues to $t = 2$, at which point the circle should return to its initial configuration. Using the $\vec{N}$ method, the VOF and LS functions to calculate $\hat{n}$, tests were performed at two mesh resolutions: 15 cells per radius (cpr) and 30 cpr, at a maximum Courant number of 0.1. The results are presented in Figs. 6–8. The errors associated with volume fractions at $t = 2$ are also presented in Table 15.

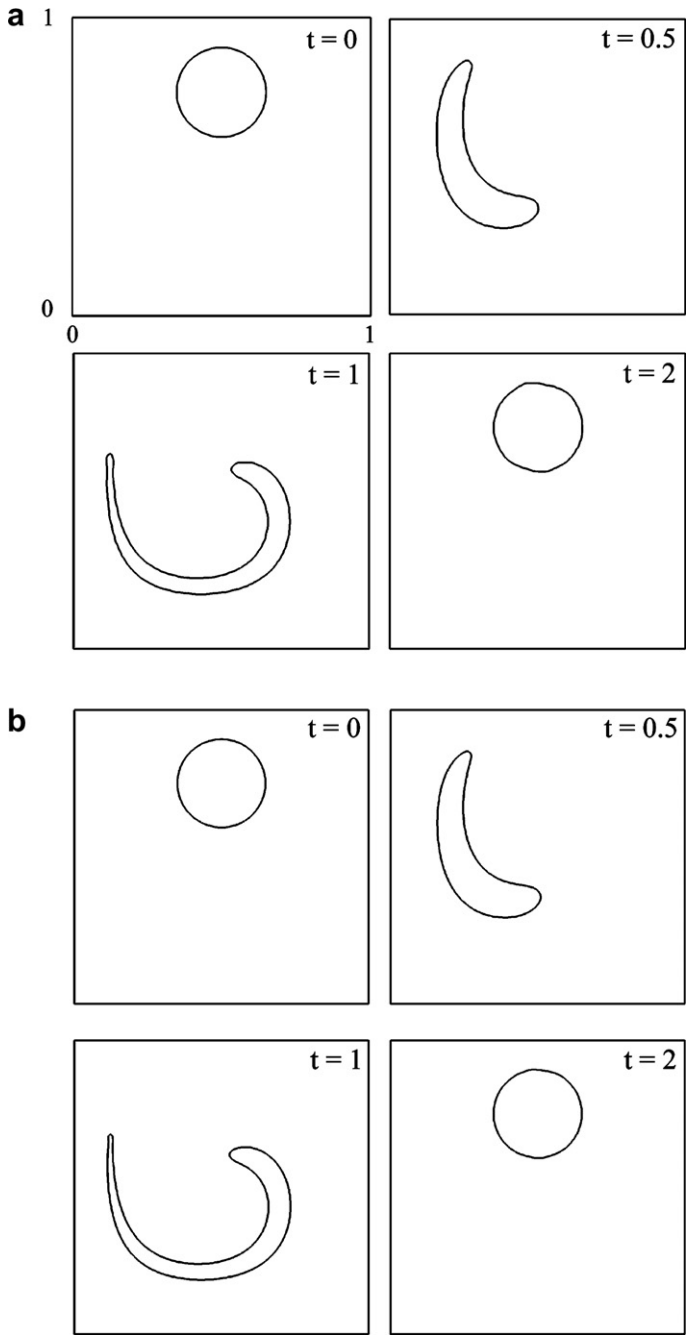

Fig. 6. The evolution of a circle in a vortex field at (a) 15 and (b) 30 cells per radius when the VOF function is used to calculate $\hat{n}$.

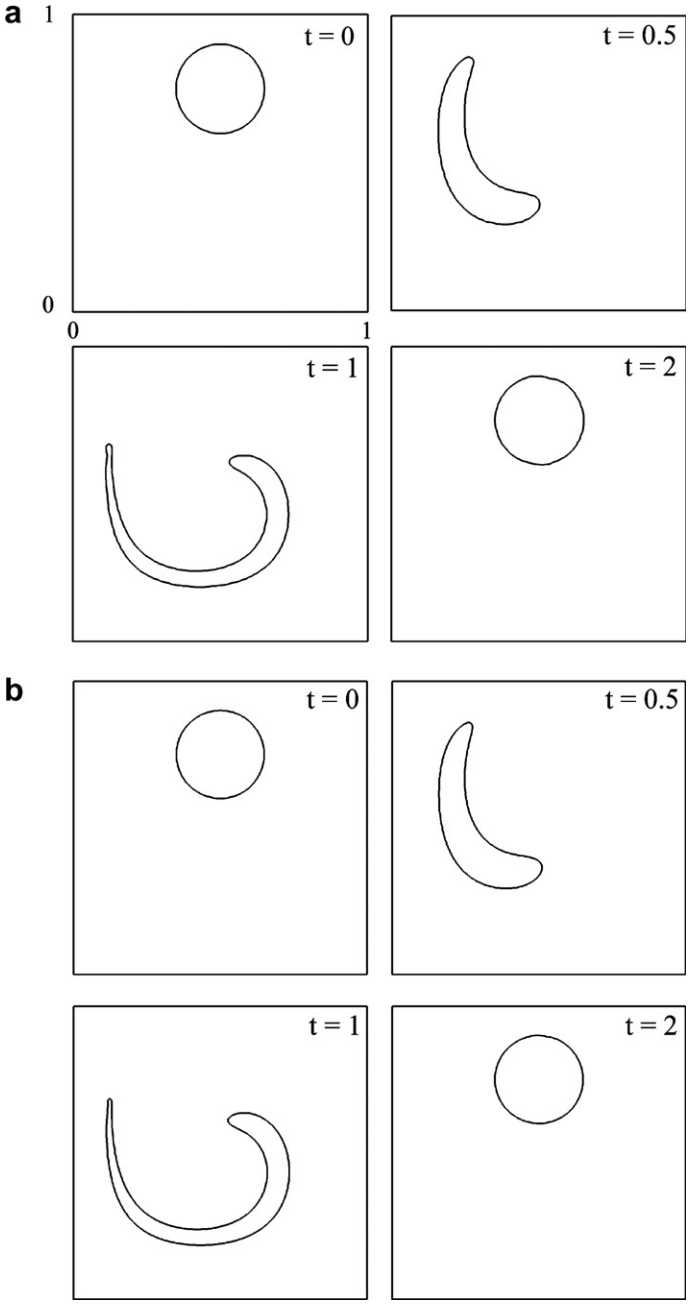

Fig. 7. The evolution of a circle in a vortex field at (a) 15 and (b) 30 cells per radius when the LS function is used is used to calculate $\hat{n}$.

As the figures and Table 15 show, the best result is obtained when using the LS function. The results from the $\vec{N}$ method (Fig. 8) are promising, although the circles at $t = 2$ are not as round as we might wish. The reason is related to what happens at the tail, shown at $t = 1$ in Fig. 9 in a close-up view: a corner forms, which has no clear normal vector. Furthermore, the interface near the tail is relatively poorly resolved, and as the interfaces on either side approach each other, the normal vectors vary sharply across one cell. When the velocity field reverses, the tail expands, and the inaccurate normals near the tail "pollute" subsequent normals and eventually lead to the deformed circle at $t = 2$, similar to the expansion of the poorly resolved circle.

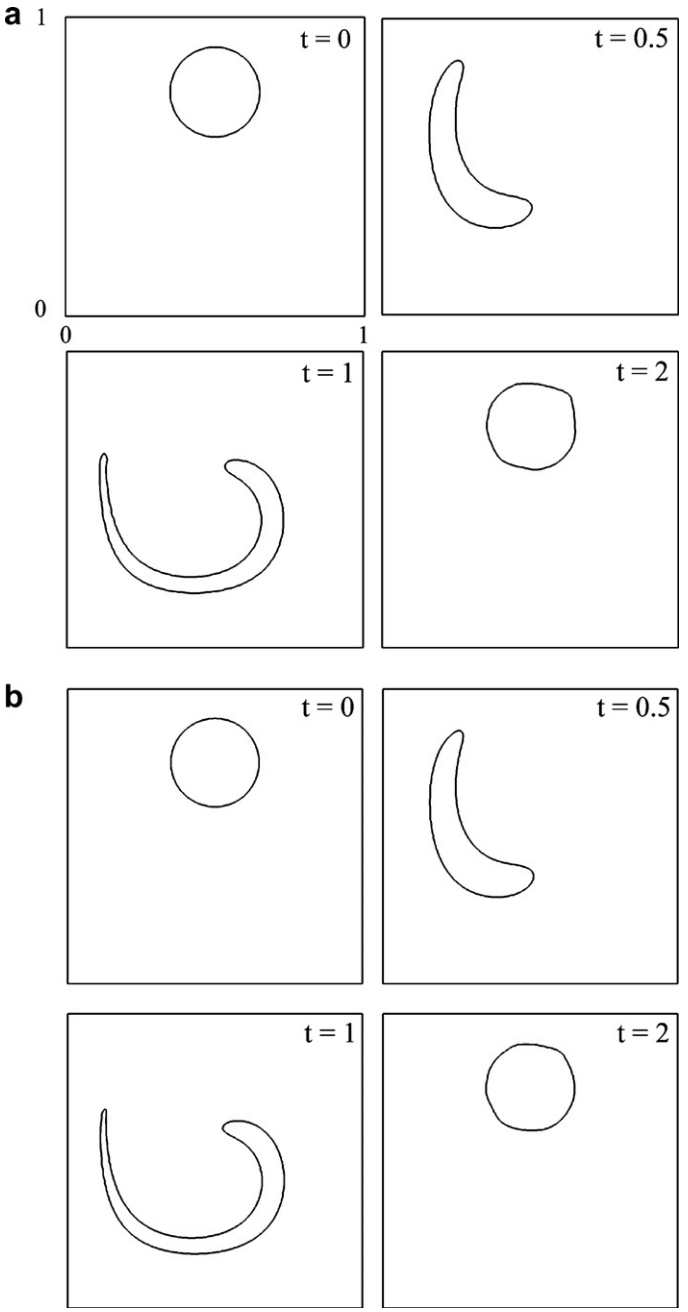

Fig. 8. The evolution of a circle in a vortex field at (a) 15 and (b) 30 cells per radius when the $\vec{N}$ method is used.

Again, a solution to this problem would be to locally use the VOF function to calculate $\hat{n}$ in poorly resolved areas where $\vec{N}$ varies sharply. Once a minimum resolution required for a smooth variation of $\vec{N}$ exists, one could then switch to the $\vec{N}$ method. We tried this for the vortex test at 15 cpr: normals $\hat{n}$ were calculated from the $\vec{N}$ method except during $0.7 \leqslant t \leqslant 1.3$, when the VOF function was used. At $t = 1.3$, normals calculated from the VOF function were used as the initial condition for the $\vec{N}$ method. The results improved significantly, as the l_∞ and l_1 errors were reduced to 0.665 and 2.38×10^{-3}, respectively. This suggests that for satisfactory performance the $\vec{N}$ method requires a minimum resolution in areas where normals vary sharply.

Table 15
The errors associated with volume fractions in the vortex test at $t = 2$ when normals $\hat{n}$ are calculated from (Panel A) the VOF function, (Panel B) the LS function and (Panel C) the $\vec{N}$ method

Δx	l_∞	l_1
Panel A		
1/100	0.433	1.70×10^{-3}
1/200	0.416	3.18×10^{-4}
Panel B		
1/100	0.302	1.31×10^{-3}
1/200	0.259	1.00×10^{-4}
Panel C		
1/100	1	4.0×10^{-3}
1/200	0.971	1.57×10^{-3}

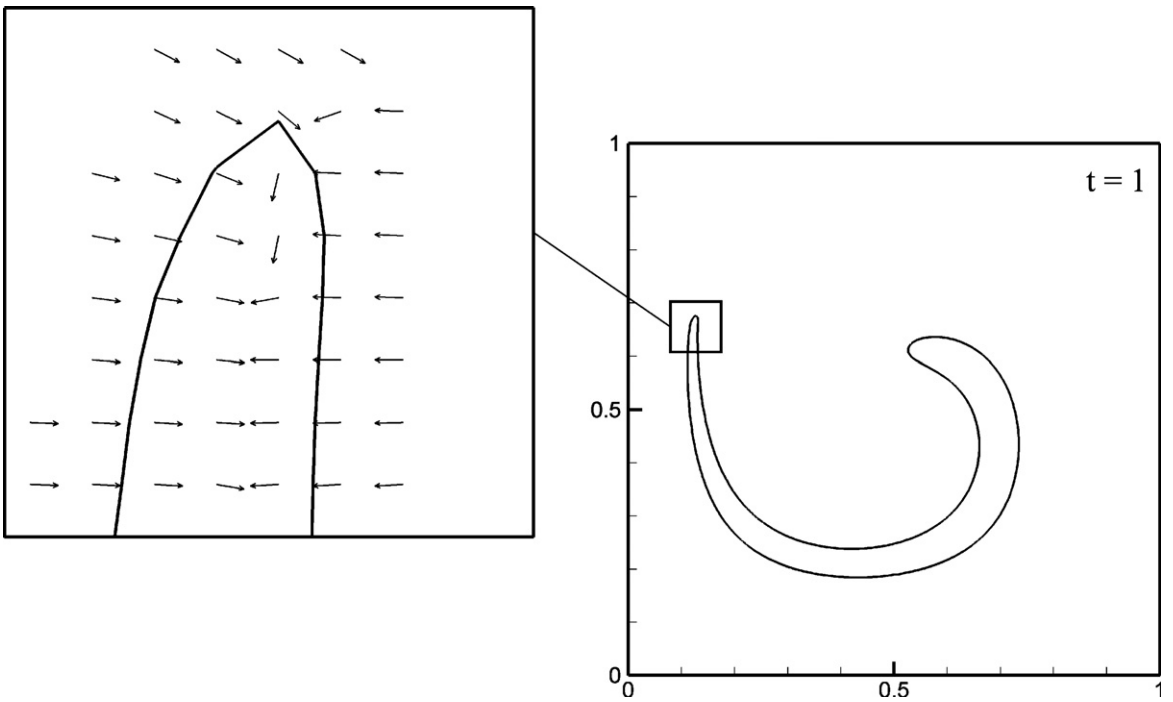

Fig. 9. A close-up view of the tail of a deformed circle in a vortex field, when the $\vec{N}$ method is used. Time $t = 1$, resolution is 30 cells per radius.

5. Summary and conclusions

It has been shown that the accuracy of curvatures calculated from the VOF function deteriorates with mesh refinement, and that there is a constant error associated with curvatures calculated from the LS function in a coupled LS and VOF method, irrespective of mesh refinement. A new methodology for calculating interface normal vectors and curvatures is proposed, in which the normal vectors are advected along with the interface, and the curvatures are calculated directly from the advected normals.

The new methodology has been incorporated into a volume-conserving VOF method, where the advected normals are employed for interface reconstruction. The normals are initialized as unit vectors and via an extension algorithm applied at each timestep they remain unit vectors. This is similar to reinitializing the distance function in the LS method. However, unlike the LS method, volume conservation does not depend on

the extent to which this condition is satisfied. Via a series of test cases, it has been shown that the method is second-order accurate when calculating interface normals, and that it yields curvatures that converge with a good order of accuracy. Although the evolution of normals and volume fractions are solved via separate equations, the test results show that there is no decoupling between normals and volume fractions. Furthermore, compared to using the VOF or LS functions to calculate normals, it has been shown that the new methodology yields volume fractions more accurately. This must be due to the more accurate normals provided by the new method.

The new methodology has some drawbacks: when an interface develops a corner, or when it is under-resolved or near another interface, the normal vectors are poorly defined and/or vary sharply. Since the new method advects normals in time, inaccurate normals associated with poorly resolved interfaces will affect the accuracy of normals later in the calculation. A potential remedy is to locally use the VOF function to calculate normals in poorly resolved areas, and then switch to the new methodology once a minimum resolution exists. At transition, the normals calculated from the VOF function are used as the initial condition for the advection equation of $\vec{N}$. However, such an approach requires further study. The other drawback lies with the extension algorithm used to satisfy Eq. (18). This algorithm is computationally expensive. For example, when the VOF function was used to calculate normals, the translation test (Section 4.2) took 2 min at $\Delta x = 1/64$; the new methodology required 13 min. Other algorithms [36,37] will be implemented to try to reduce the computation time.

Finally, the new methodology has been successfully implemented and used to simulate actual flow problems; these results will be presented in another article. And the extension of the new method to 3D is straightforward.

References

[1] M.M. Francois, S.J. Cummins, E.D. Dendy, D.B. Kothe, J.M. Sicilian, M.W. Williams, A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework, J. Comput. Phys. 213 (1) (2006) 141–173.
[2] D. Jamet, D. Torres, J.U. Brackbill, On the theory and computation of surface tension: the elimination of parasitic currents through energy conservation in the second-gradient method, J. Comput. Phys. 182 (1) (2002) 262–276.
[3] S. Shin, S.I. Abdel-Khalik, V. Daru, D. Juric, Accurate representation of surface tension using the level contour reconstruction method, J. Comput. Phys. 203 (2) (2005) 493–516.
[4] E. Shirani, N. Ashgriz, J. Mostaghimi, Interface pressure calculation based on conservation of momentum for front capturing methods, J. Comput. Phys. 203 (1) (2005) 154–175.
[5] C.W. Hirt, B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, J. Comput. Phys. 39 (1981) 201–225.
[6] R. Scardovelli, S. Zaleski, Direct numerical simulation of free-surface and interfacial flow, Ann. Rev. Fluid Mech. 31 (1999) 567–603.
[7] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1988) 12–49.
[8] S. Osher, R. Fedkiw, Level set methods: an overview and some recent results, J. Comput. Phys. 169 (2001) 463–502.
[9] J.A. Sethian, Evolution, implementation, and application of level set and fast marching methods for advancing fronts, J. Comput. Phys. 169 (2) (2001) 503–555.
[10] J. Helmsen, P. Colella, E.G. Puckett, Non-convex profile evolution in two dimensions using volume of fluids, Technical Report LBNL-40693, Lawrence Berkeley National Laboratory, 1997.
[11] 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.
[12] S.J. Cummins, M.M. Francois, D.B. Kothe, Estimating curvature from volume fractions, Comput. Struct. 83 (2005) 425–434.
[13] J.Y. Poo, N. Ashgriz, A computational method for determining curvatures, J. Comput. Phys. 84 (2) (1989) 483–491.
[14] Y. Renardy, M. Renardy, PROST: a parabolic reconstruction of surface tension for the volume-of-fluid method, J. Comput. Phys. 183 (2) (2002) 400–421.
[15] S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer-Verlag, New York, 2003.
[16] F. Losasso, R. Fedkiw, S. Osher, Spatially adaptive techniques for level set methods and incompressible flow, Comput. Fluids 35 (10) (2006) 995–1010.
[17] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, J. Comput. Phys. 114 (1) (1994) 146–159.
[18] D. Peng, B. Merriman, S. Osher, H. Zhao, M. Kang, A PDE-based fast local level set method, J. Comput. Phys. 155 (2) (1999) 410–438.
[19] P. Gómez, J. Hernández, J. López, On the reinitialization procedure in a narrow-band locally refined level set method for interfacial flows, Int. J. Numer. Meth. Fluids 63 (2005) 1478–1512.

[20] R.R. Nourgaliev, S. Wiri, N.T. Dinh, T.G. Theofanous, On improving mass conservation of level set by reducing spatial discretization errors, Int. J. Multiphase Flow 31 (12) (2005) 1329–1336.
[21] J.A. Sethian, Fast marching methods, SIAM Rev. 41 (2) (1999) 199–235.
[22] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface capturing, J. Comput. Phys. 183 (1) (2002) 83–116.
[23] A. Bourlioux, A coupled level set volume-of-fluid algorithm for tracking material interfaces, in: 6th Int. Symp. on Computational Fluid Dynamics, Lake Tahoe, CA, 1995, pp. 15–22.
[24] M. Sussman, E.G. Puckett, A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows, J. Comput. Phys. 162 (2000) 301–337.
[25] G. Son, N. Hur, A coupled level set and volume-of-fluid method for the buoyancy-driven motion of fluid particles, Numer. Heat Transfer, Part B 42 (2002) 523–542.
[26] D.L. Youngs, Time-dependent multi-material flow with large fluid distortion, in: K.W. Morton, M.J. Baines (Eds.), Numerical Methods for Fluid Dynamics, Academic, New York, 1982, pp. 273–285.
[27] M. Sussman, Personal communication, 2005.
[28] H.-K. Zhao, T. Chan, B. Merriman, S. Osher, A variational level set approach to multiphase motion, J. Comput. Phys. 127 (1) (1996) 179–195.
[29] H. Takewaki, A. Nishiguchi, T. Yabe, Cubic interpolated pseudo-particle method (CIP) for solving hyperbolic-type equations, J. Comput. Phys. 61 (1985) 261–268.
[30] T. Yabe, F. Xiao, T. Utsumi, The constrained interpolation profile method for multiphase analysis, J. Comput. Phys. 169 (2) (2001) 556–593.
[31] T. Yabe, F. Xiao, Description of complex and sharp interface with fixed grids in incompressible and compressible fluid, Comput. Math. Appl. 29 (1) (1995) 15–25.
[32] R. Tanaka, T. Nakamura, T. Yabe, Constructing exactly conservative scheme in a non-conservative form, Comput. Phys. Commun. 126 (2000) 232–243.
[33] S. Gottlieb, C.-W. Shu, Total variation diminishing Runge–Kutta schemes, Math. Comput. 67 (221) (1998) 73–85.
[34] S. Chen, B. Merriman, S. Osher, P. Smereka, A simple level set method for solving Stefan problems, J. Comput. Phys. 135 (1) (1997) 8–29.
[35] P. Smereka, Semi-implicit level set methods for curvature and surface diffusion motion, J. Sci. Comput. 19 (1–3) (2003) 439–456.
[36] P. Macklin, J. Lowengrub, Evolving interfaces via gradients of geometry-dependent interior Poisson problems: application to tumor growth, J. Comput. Phys. 203 (1) (2005) 191–220.
[37] C. Wu, J. Deng, F. Chen, Fast data extrapolating, J. Comput. Appl. Math., in press.