Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 227 (2008) 4063–4088
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A new interface tracking method: The polygonal area mapping method

Qinghai Zhang [a,*], Philip L.-F. Liu [a,b]

[a] School of Civil and Environmental Engineering, Cornell University, Ithaca, NY 14853, USA
[b] Institute of Hydrological and Oceanic Science, National Central University, Jhongli, Taiwan

Received 16 May 2007; received in revised form 8 October 2007; accepted 14 December 2007
Available online 31 December 2007

Abstract

We present a new method, the polygonal area mapping (PAM) method, for tracking a non-diffusive, immiscible material interface between two materials in two-dimensional incompressible flows. This method represents material areas explicitly as piecewise polygons, traces characteristic points on polygon boundaries along pathlines and calculates new material areas inside interface cells via polygon-clippings in a discrete manner. The new method has very little spatial numerical diffusion and tracks the interface singularities naturally and accurately. In addition to high accuracy, the PAM method can be directly used on either a structured rectangular mesh or an unstructured mesh without any modifications. The mass conservation is enforced by heuristic algorithms adjusting the volume of material polygons. The results from a set of widely used benchmark tests show that the PAM method is superior to existing volume-of-fluid (VOF) methods.

Keywords: Interface tracking; Area mapping; Polygon-clipping; Compact point set; Boolean set operations; Volume-of-fluid methods

1. Introduction

Over the past 40 years, three major formulations have been developed and used for interface tracking in multi-phase simulations; namely, front-tracking methods, level set methods and VOF methods.

The front-tracking method treats the interface as a jump of density and *explicitly* marks the interface by a fixed number of points. Each marker is tracked by solving an ordinary differential equation (ODE). This method works well if the line segments connecting the markers have roughly equal lengths and do not intersect each other. Unfortunately, even the most trivial velocity fields can distort the interface and the accuracy of this method can deteriorate dramatically. To remedy this problem, complicated surgical operations, such as attaching and detaching boundary elements, have been suggested so as to maintain some degrees of mesh regularity as the interface deforms. Interested readers can find an discussion of these surgical operations in [36] and an state-of-art review of the front-tracking method in [31].

* Corresponding author.
E-mail addresses: qz27@cornell.edu (Q. Zhang), pll3@cornell.edu (P.L.-F. Liu).

0021-9991/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2007.12.014

Totally different from front-tracking methods, level set methods [21,19,20] define an function ϕ to *implicitly* represent the interface, a typical choice for ϕ being the signed distance from the interface. A partial differential equation (PDE) of ϕ is used to evolve the interface. Level set methods have become increasingly popular in a diverse group of applications including optimal design, computer-aided design (CAD), optimal control, and computer graphics. One difficulty of level set methods, particularly concerning fluid flow interfaces, is the lack of mass conservation: existing methods conserve mass only in a global sense to a fraction of one percent.

For a certain material W, VOF methods define the *color function* as

$$f(\boldsymbol{x}) = \begin{cases} 1 & \text{if there is material } W \text{ at } \boldsymbol{x}, \\ 0 & \text{otherwise}, \end{cases} \tag{1}$$

and the *volume fraction* of material W in cell (i,j) as

$$f_{i,j} = \frac{\int\int_{\mathcal{C}_{i,j}} f(x,y)\,\mathrm{d}x\,\mathrm{d}y}{\Delta x \Delta y}, \tag{2}$$

where $\mathcal{C}_{i,j} = [x_{i-1/2}, x_{i+1/2}] \times [y_{j-1/2}, y_{j+1/2}]$ denotes the cell area. For simplicity we assume $\Delta x = \Delta y = h$, thus cell centers are located at $[x_i, y_j]^T$, vertical edges at $x_{i\pm1/2} = x_i \pm h/2$ and horizontal edges at $y_{j\pm1/2} = y_j \pm h/2$. The interface in cell (i,j) is represented both *implicitly* by the volume fraction $f_{i,j}$ and *explicitly* by the interface locus, often a linear function or a parabola. In VOF literature, either the *advective* form of

$$f_t + \boldsymbol{u} \cdot \nabla f = 0 \tag{3}$$

or the equivalent *conservative* form

$$f_t + \nabla \bullet (f\boldsymbol{u}) = 0 \tag{4}$$

in a divergence-free flow is often referred to as the governing equation of VOF methods. However, the use of (3) or (4) is dubious since the color function defined in (1) is discrete and thus non-differentiable in time and in space.

During each time step, VOF methods track the interface by two sub-steps. In the first advection step, a new value of volume fraction, $f_{i,j}^{n+1}$, is calculated based on $f_{i,j}^{n}$ and the current interface locus; in the second reconstruction step, the new interface locus is determined solely by $f_{i,j}^{n+1}$ in a neighborhood of cell (i,j).

The advection schemes of VOF methods fall into two categories: operator splitting methods and unsplit methods. Because of its simplicity, the operator splitting approach is dominant in the early development of VOF methods, such as SLIC [17], Chorin [4], SOLA-VOF [10], FCT–VOF [26] and Youngs' method [34]. A relatively recent splitting scheme, EI-LE [27], features an Eulerian implicit step followed by an Lagrangian explicit one. This method conserves mass exactly with no flotsam (small fragments of materials surrounded by a different material) and no under/overshoots. In recent years unsplit methods [23,8,9] have become more popular. These methods are often based on explicit tracing of pathlines. One can find the root of these methods in the theory of the multi-dimensional hyperbolic conservation law [13]. More recently, Lopez [14] proposed the edge-matched flux polygon advection (EMFPA) method. This method constructs edge-matched flux polygons at cell faces to calculate the flux. A later improvement [15] allows the interface to be represented by two non-contiguous line segments in a cell for tracking thin fluid structures. Lagrangian markers are placed at the centers of these interface segments to locate and orient the interface.

Early VOF methods reconstruct the interface with a piecewise constant function aligned to one of the coordinates. Since the constructed interface can only be either horizontal or vertical, these methods are crude in accuracy and they generate spurious flotsam even in simple flows. Nowadays the dominant reconstruction schemes are those so-called piecewise linear interface calculation (PLIC). Youngs' method [34] calculates the interface normal by taking gradient of the volume fraction values; this method does not reconstruct all linear interfaces exactly. In contrast, the least-squares VOF interface reconstruction algorithm (LVIRA) [24] can reconstruct a linear interface exactly. Under the constraint of the reconstructed volume having the same volume fraction in cell (i,j), this method places the linear function in a way to minimize the reconstruction error in the 3×3 stencil centered at this cell. To reduce the computation time of the least-square minimization, the Efficient LVIRA (ELVIRA) [22] method only calculates nine candidates for the interface normal

via the combination of backward, central and forward differentiations of $f_{i,j}$. The reconstruction residual is then calculated for each candidate and the one that has the smallest residual is chosen as the solution. Both LVIRA and ELVIRA reconstruct an linear interface within a cell exactly.

Apart from the three major methods, researchers have been trying to combine the best features of the three major formulations, resulting in a few hybrid methods, e.g. the coupled level set with VOF method (CLSVOF) [30], the hybrid level-contour front-tracking method [28], and the hybrid particle level set method [5]. In another hybrid marker-VOF method [1,2], the interface is described by a continuous chain of line segments connecting Lagrangian markers advected along pathlines. Grid intersection markers locate the interface on the cell edges and ensure continuity while mass conservation markers are redistributed uniformly along the interface for local area-preserving.

Despite the successes of existing interface tracking methods, further improvements are still in need. This is particularly true when the interface contains large curvature or singular points. In this case, spurious numerical diffusion always rounds sharp corners and prevent convergence to the exact solution. In solid–fluid interaction applications, this is often unacceptable.

Attempting to overcome the above shortcomings, we have developed a new method, called the polygonal area mapping (PAM) method. In this method, material areas are explicitly represented by polygons. The velocity field is considered as a mapping of these polygons from the beginning of a time step to the end of it. The material area to be advected into a certain cell is calculated by discrete Boolean set operations defined on polygons in a discrete manner such that no calculus is involved. As a result of restricting spatial operation to these polygon-clipping operations, the PAM method is almost free of numerical diffusion. Another original feature of the PAM method is its independence of the mesh topology. Although numerical tests in this paper are carried out on structured rectangular grids, the PAM method can be applied to unstructured meshes without any modification. Also, the PAM method can be extended to three-dimensional space via Boolean set operations on polyhedra [7]. In this paper, we focus on the PAM method in two-dimensional space and we will report the three-dimensional extension in the future.

The remainder of this paper is organized as follows. In Section 2 we introduce the formulation of the PAM method. In Section 3 we summarize the main steps of the PAM method and details the heuristic algorithms for mass conservation. In Section 4 a number of benchmark interface tracking problems are performed to test the stability, accuracy, and efficiency of the new method; the results are compared to those of existing VOF methods. Finally we draw our conclusions in Section 5.

2. Formulation

2.1. Interface representation

A computational cell is called an interface cell if it contains more than one material, otherwise it is called *a pure cell*. An *interface candidate cell* is either an interface cell or a pure cell adjacent to an interface cell. For example, in Fig. 1 cells $[i-2,j-2],[i-2,j-1],[i-1,j-1],[i-1,j],[i,j],[i+1,j],[i+1,j+1]$ are interface cells and all other cells are pure cells; all cells except $[i+1,j-2]$ are interface candidate cells. Note that the sets of the three types of cells changes in time since the locus of the interface moves.

Let $p_0,p_1,\ldots,p_{n-1}$ be n points, we establish the indexing convention of i as $i \bmod n$, implying a cyclic ordering of the points, thus $p_n = p_0, p_{n+1} = p_1$ and so on. The line segments $\overline{p_0p_1},\ldots,\overline{p_ip_{i+1}},\ldots,\overline{p_{n-1}p_n}$ bounds a *simple polygon* [18] if and only if:

$$\begin{cases} \overline{p_ip_{i+1}} \cap^* \overline{p_{i+1}p_{i+2}} = \{p_{i+1}\} & i = 0,\ldots,n-1, \\ \overline{p_ip_{i+1}} \cap^* \overline{p_jp_{j+1}} = \emptyset & 1 < \mathrm{mod}(|i-j|,n) < n-1, \end{cases} \tag{5}$$

where $\cap^*$ denote the *ordinary* set-intersection operation. The points are the *vertexes* of the polygon and the line segments are the *edges* of the polygon. A vertex and its two adjacent edges defines two angles, the one that intersects the interior of the polygon is called the *internal angle*. A vertex is *concave* if its internal angle is strictly bigger than π, otherwise it is *convex*. A *convex polygon* $\mathcal{P}$ contains the line segment $\overline{pq}$ if both endpoints p and q belong to $\mathcal{P}$:

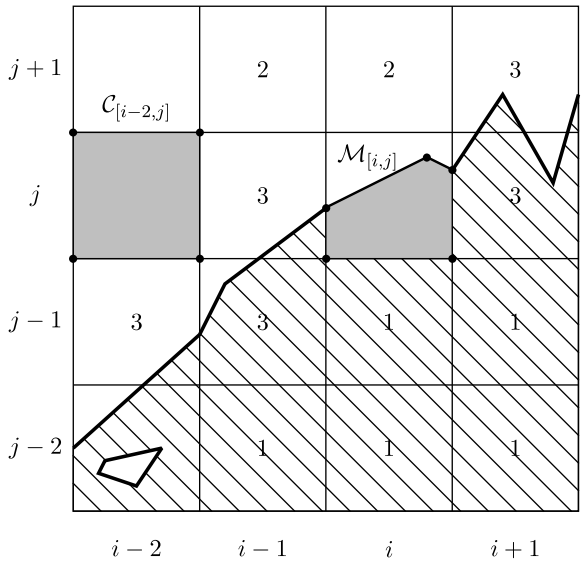


Fig. 1. Material representation on a structured rectangular mesh. The thick polygonal line represents the interface that separate water (hatched area) from air. Shaded polygon $\mathcal{M}_{[i,j]}$ is the area occupied by water (W) in cell $[i,j]$. Shaded polygon $\mathcal{C}_{[i-2,j]}$ is the area of cell $[i-2,j]$, also the area occupied by air in this cell. Labels 1, 2, and 3 indicate a pure water cell, a pure air cell, and an interface cell, respectively. The quadrilateral in cell $[i-2,j-2]$ represent an air bubble trapped in water.

$$\mathcal{P} \text{ is convex} \iff \alpha p + (1-\alpha)q \in \mathcal{P} \quad \forall p, q \in \mathcal{P}, \ \ \alpha \in [0,1]. \tag{6}$$

The *convex hull* of a set of points is the smallest convex polygon that contains the point set [18].

A simple polygon represents a *simply-connected* area, in which every closed curve in it can be continuously deformed to a point. We denote such an area by an uppercase calligraphic letter, e.g. the area occupied by material W in cell $[i,j]$ is denoted by $\mathcal{M}_{[i,j]}$ and the area of cell $[i-2,j]$ denoted by $\mathcal{C}_{[i-2,j]}$, as shown by the shaded polygons in Fig. 1.

In addition to (5), we make the following assumptions on the polygon we are using.

$$p_i \text{ is a convex vertex of } \mathcal{P} \Rightarrow q \in \mathcal{P} \quad \forall q \in \Delta(p_i, p_{i-1}, p_{i+1}), \tag{7a}$$

$$p_i \text{ is a concave vertex of } \mathcal{P} \Rightarrow q \notin \text{ interior}(\mathcal{P}) \quad \forall q \in \Delta(p_i, p_{i-1}, p_{i+1}), \tag{7b}$$

where $\Delta(p_i, p_{i-1}, p_{i+1})$ denote the triangle formed by vertices p_i, p_{i-1}, p_{i+1}. (7) limits our attention to a family of 'well-shaped' polygons. Two examples violating (7) are shown in Fig. 2.

In general, one simple polygon is not enough to describe the material area in a fixed cell, as in the cases of cell $[i-2,j-2]$ and cell $[i+1,j+1]$ in Fig. 1. Let $\mathcal{P}_i$ be a simple polygon with counter-clockwise orientation,

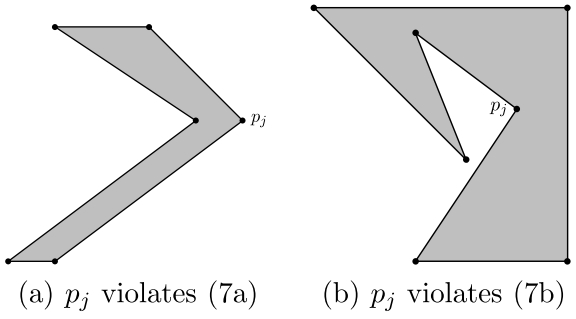


Fig. 2. Examples of polygons violating (7).

representing a 'positive' area; let $\mathcal{H}_j$ be a simple polygon with clockwise orientation, representing a 'negative' area (a hole). We define an abstract data type, the *set of simple polygons* (SSP),

$$\mathcal{M} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{s_p}, \mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_{s_h}\} \tag{8}$$

to represent the two-dimensional area of the tracked material in an interface cell

$$\mathcal{M} = (\cup_{i=1}^{s_p} \mathcal{P}_i) \setminus \left(\cup_{j=1}^{s_h} \mathcal{H}_j\right), \tag{9}$$

where s_p, s_h are non-negative integers and the *regularized* Boolean set operations [11] are defined as

$$\mathcal{P} \cup^* \mathcal{Q} = \{p : p \in \mathcal{P}, \text{or}, p \in \mathcal{Q}\}, \tag{10a}$$

$$\mathcal{P} \cup \mathcal{Q} = \text{closure}(\text{interior}(\mathcal{P} \cup^* \mathcal{Q})), \tag{10b}$$

$$\mathcal{P} \setminus^* \mathcal{Q} = \{p : p \in \mathcal{P} \text{ and } p \notin \mathcal{Q}\}, \tag{10c}$$

$$\mathcal{P} \setminus \mathcal{Q} = \text{closure}(\text{interior}(\mathcal{P} \setminus^* \mathcal{Q})). \tag{10d}$$

In other words, a regularized Boolean set operation can be obtained by taking closure of the interior of the point set resulting from the corresponding *ordinary* Boolean set operation (marked by *). The set-intersection has similar definition. Regularized Boolean set operations eliminate lower dimensional features, such as isolated vertices and antennas and guarantee the representation to be physically meaningful. As shown in Fig. 3, the Boolean set operations are also *discrete* in that there are no differentiation associated. A generic polygon-clipping (GPC) algorithm [32] and a modified GPC library [16] are used to perform these operations.

The number of simple polygons in $\mathcal{M}$ and the number of vertices in a simple polygon should be minimal so that the representation is unique.

$$i \neq j \Rightarrow \mathcal{P}_i \cap \mathcal{P}_j = \emptyset, \tag{11a}$$

$$i \neq j \Rightarrow \mathcal{H}_i \cap \mathcal{H}_j = \emptyset, \tag{11b}$$

$$\mathcal{H}_i \in \mathcal{M} \Rightarrow \exists \mathcal{P}_j \in \mathcal{M}, \quad \text{s.t. } \mathcal{H}_i \subset \mathcal{P}_j, \ \ \mathcal{P}_j \setminus \mathcal{H}_i \text{ is not a simple polygon}, \tag{11c}$$

$$\forall \mathcal{P}_i, \mathcal{H}_j \in \mathcal{M}, \ \text{no three consecutive vertices are collinear}. \tag{11d}$$

(11a) and (11b) ensure that polygons of the same sign are pairwise disjoint. (11c) requires a 'hole' polygon to be properly contained in a 'positive' polygon. (11d) gets rid of redundant vertices. Consequently, the total volume of $\mathcal{M}$ can be easily calculated by

$$\|\mathcal{M}\| = \sum_{i=1}^{s_p} \|\mathcal{P}_i\| - \sum_{j=1}^{s_h} \|\mathcal{H}_j\|, \tag{12}$$

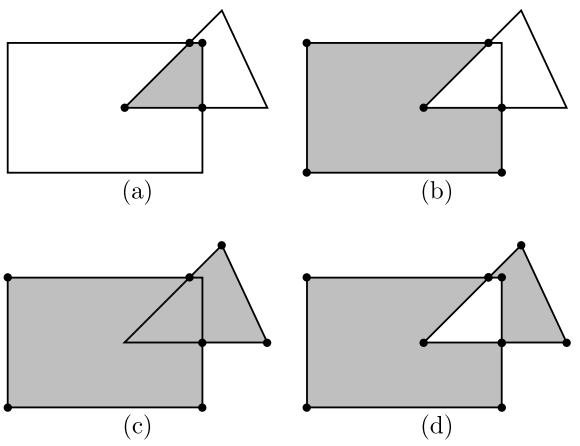


Fig. 3. Boolean set operations on polygons. $\mathcal{P}$ is the rectangle and $\mathcal{Q}$ is the triangle, the shaded region represents the result of an operator. (a) $\mathcal{P} \cap \mathcal{Q}$, (b) $\mathcal{P} \setminus \mathcal{Q}$, (c) $\mathcal{P} \cup \mathcal{Q}$, and (d) $\mathcal{P} \wedge \mathcal{Q}$.

and the volume of a simple polygon, $\mathcal{P}(p_0, p_1, \ldots, p_{n_\mathcal{P}-1})$, is given by

$$\|\mathcal{P}\| = \frac{1}{2}\left|\sum_{i=0}^{n_\mathcal{P}-3}(p_{i+1} - p_i) \times (p_{i+2} - p_{i+1})\right|, \tag{13}$$

where $n_\mathcal{P}$ denotes the number of vertices of $\mathcal{P}$ and '$\times$' the cross product of two vectors.

We collect (5), (7), and (11) as the *representation invariant* of the SSP and enforce them on the concrete data structure all the time.

Compared to simple polygons, SSPs has two advantages: (i) SSPs are closed under the Boolean set operations; (ii) any two-dimensional compact point sets can be approximated by a SSP arbitrarily well. For an interface cell in a two-phase interface tracking problem, we thus only store a SSP for one material, from which the SSP of the other material and the interface can be deduced. An integer label is also attached to each cell in the domain to indicate the types of materials contained in that cell, as shown in Fig. 1.

2.2. Area mapping by the velocity field

A particle $p = p(t_n) = [x_p(t_n), y_p(t_n)]^T$ labeled with its position at time t_n changes its location in subsequent time according to the velocity field,

$$\frac{\mathrm{d}\boldsymbol{x}_p}{\mathrm{d}t} = \boldsymbol{u}(\boldsymbol{x}_p, t). \tag{14}$$

To find out where $p(t_n)$ came from, we define another ODE system to trace back p in time,

$$\frac{\mathrm{d}\boldsymbol{x}_p}{\mathrm{d}t} = \boldsymbol{u}_B(\boldsymbol{x}_p, t), \tag{15}$$

where

$$\boldsymbol{u}_B(\boldsymbol{x}_p, t) = -\boldsymbol{u}(\boldsymbol{x}_p, t). \tag{16}$$

The *image* and *preimage* of $p(t_n)$ with respect to a time interval δt are defined by forward tracing ($\overrightarrow{\cdot}$) and backward tracing ($\overleftarrow{\cdot}$), respectively,

$$\overrightarrow{p(t_n, \delta t)} = p(t_n + \delta t) = p(t_n) + \int_{t_n}^{t_n+\delta t} \boldsymbol{u}(p(t), t)\,\mathrm{d}t, \tag{17}$$

$$\overleftarrow{p(t_n, \delta t)} = p(t_n - \delta t) = p(t_n) + \int_{t_n}^{t_n+\delta t} \boldsymbol{u}_B(p(t), t)\,\mathrm{d}t, \tag{18}$$

where the time increment δt is always positive. A simple polygon also has its image and preimage defined from those of its vertices. Similarly, a SSP $\mathcal{M}(t_n)$ has the corresponding image $\overrightarrow{\mathcal{M}(t_n, \delta t)}$ and preimage $\overleftarrow{\mathcal{M}(t_n, \delta t)}$, defined from those of its element polygons.

The underlying velocity field can be regarded as a mapping of two-dimensional compact point sets represented by polygonal areas. For incompressible flow, the image and preimage of a polygonal area is *homomorphic*. In other words, a simple polygon remains topologically a *disk* through a divergence-free mapping. From the above observations we derive the name of our new method.

Let $\mathcal{M}(t)$ denote the area of the interested material in the computational domain at time t, the two-phase immiscible interface tracking problem can be formulated as *the determination of* $\mathcal{M}(T)$ *from the given velocity field* $\boldsymbol{u}(\boldsymbol{x}, t)$ $(t \in [t_0, T])$ *and the initial area* $\mathcal{M}(t_0)$.

3. Algorithm

In the following, we shall present our new method on a structured mesh ($\Delta x = \Delta y = h$) with uniform time steps $t_n = nk$ where k is the time step size. These assumptions are not necessary in the formulation of the PAM method, but are used here only for the convenience of exposition.

At the initial time, all cells in the domain are labeled according to the initial condition $\mathcal{M}(t_0)$. The material area in an interface cell is also calculated by

$$\mathcal{M}^0_{[c]} = \mathcal{M}(t_0) \cap \mathcal{C}_{[c]}, \tag{19}$$

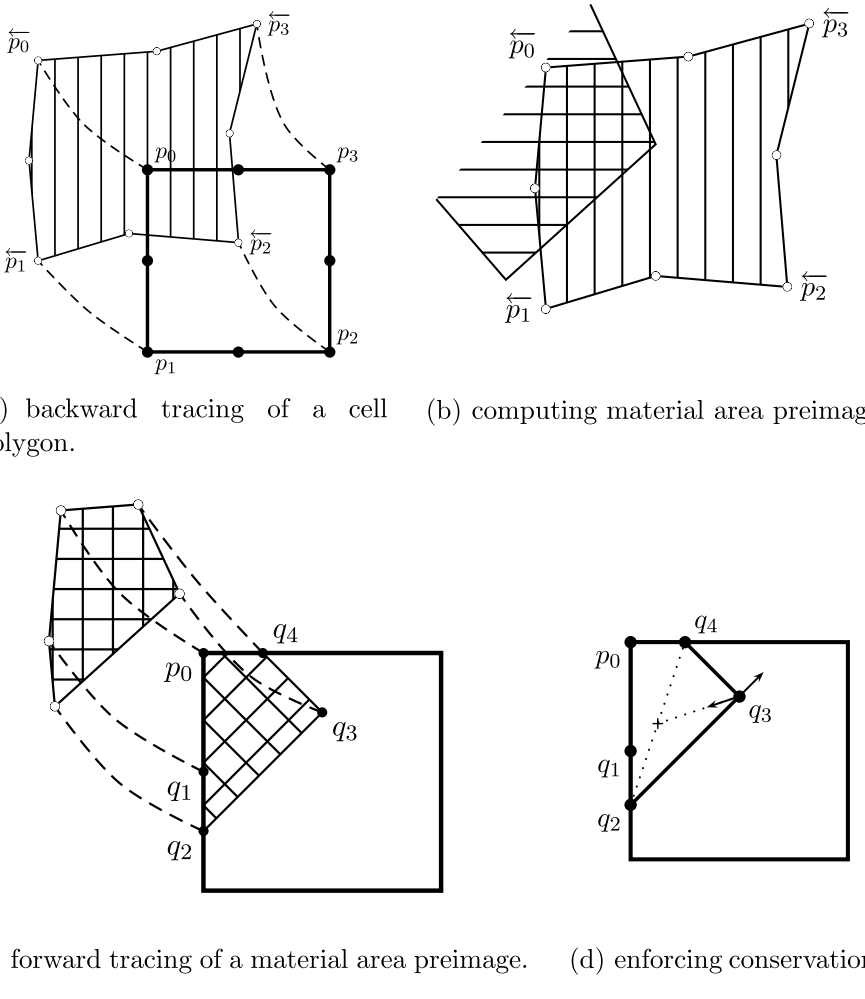
where the subscript $[c]$ is the index of that cell and $\mathcal{C}_{[c]}$ is the shell polygon of cell $[c]$.

3.1. Major steps

We describe the steps of our algorithm for one time step. The PAM method tracks the interface by applying these steps to all interface candidate cells in every time step.

Step 1 – trace backward to obtain cell preimages

For each vertex of $\mathcal{C}_{[c]}$, we use (18) to obtain its preimage. To improve accuracy, we also compute the midpoint preimages of the edges of $\mathcal{C}_{[c]}$. These preimages of vertices and midpoints define the preimage of the cell polygon. This step is illustrated in Fig. 4(a), where the cell preimage, $\overleftarrow{\mathcal{C}^{n+1}_{[c]}}$ (a shorthand for $\overleftarrow{\mathcal{C}_{[c]}(t_{n+1}, k)}$) is represented by vertical hatches.



(a) backward tracing of a cell polygon. (b) computing material area preimage.

(c) forward tracing of a material area preimage. (d) enforcing conservation.

Fig. 4. The polygonal area mapping algorithm. (a)–(d) correspond to the four sub-steps of the PAM method. Solid dots (●) and hollow dots (○) represent image points and preimage points, respectively. Dashed lines denote pathlines.

In our implementation, pairs of images and preimages are associated in a map data structure. Two maps are maintained: the backward tracing map has image points as the keys and their preimages as the values; the forward tracing map has the same pairs of images and preimages, but has the preimages as the keys and images as the values. In forming the preimage of a cell polygon, we first check the backward tracing map for any vertex preimages that have already been calculated. If a preimage is not found, the standard fourth-order Runge–Kutta (RK4) formula is used for numerical integration of (18) and the result is stored in the two maps for future use. In this way, the preimage of any vertex is only calculated once. Since grid intersection vertices of a cell are shared by many neighboring cells and the Runge–Kutta integration is the most expensive part of the algorithm, the use of the maps reduces the computational time considerably.

Step 2 – calculate material area preimages

Let $\{[b]\}$ denote the index set of the cells inside the 3-by-3 stencil centered at cell $[c]$, i.e. $\{[b]\} = \{[i-1,j-1],\ [i-1,j],\ [i-1,j+1],\ [i,j-1],\ [i,j],\ [i,j+1],\ [i+1,j-1],\ [i+1,j],\ [i+1,j+1]\}$. The union of material areas in the neighborhood of cell $[c]$ is given by

$$\mathcal{N}^n_{[c]} = \bigcup_{\{[b]\}} \mathcal{M}^n_{[b]}, \tag{20}$$

where $\mathcal{M}^n_{[b]}$ denotes the material SSP inside cell $[b]$ at time t_n.

$\overleftarrow{\mathcal{C}^{n+1}_{[c]}}$ contains all the particles to be advected into cell $[c]$ at the end of the time step, these particles may or may not be of the material being tracked. $\mathcal{N}^n_{[c]}$ contains all the particles of the tracked material at the beginning of the time step. The intersection of these two polygons yields the set of points $\{p\}$ that satisfies:

- p is of the tracked material;
- $p(t_n,k) \in \mathcal{C}_{[c]}$.

The material area preimage is thus obtained by

$$\overleftarrow{\mathcal{M}^{n+1}_{[c]}} = \overleftarrow{\mathcal{C}^{n+1}_{[c]}} \cap \mathcal{N}^n_{[c]}. \tag{21}$$

This step is illustrated in Fig. 4(b) where $\mathcal{N}^n_{[c]}$ is represented by horizontal hatches and $\overleftarrow{\mathcal{M}^{n+1}_{[c]}}$ by cross hatches. For proper results of (21), we require that the Courant number be no greater than one so that the neighborhood material area $\mathcal{N}^n_{[c]}$ covers all possible particles for cell $[c]$ in this time step.

Step 3 – trace forward material area preimages

Because of (16)–(18), the image of the preimage of a point is itself. In other words, tracing backward followed by tracing forward with identical δt yields the same point. The image of $\overleftarrow{\mathcal{M}^{n+1}_{[c]}}$ is thus $\mathcal{M}^{n+1}_{[c]}$, which can be obtained by applying (17) to each vertex of $\overleftarrow{\mathcal{M}^{n+1}_{[c]}}$.

In Fig. 4(c), $\overleftarrow{\mathcal{M}^{n+1}_{[c]}}$ is represented by the 0° cross hatches while $\mathcal{M}^{n+1}_{[c]}$ by the 45° cross hatches. Some vertices images of $\overleftarrow{\mathcal{M}^{n+1}_{[c]}}$, e.g. p_0 and q_1, can be obtained by looking up the forward tracing map since their preimages, $\overleftarrow{p_0}$ and $\overleftarrow{q_1}$, are already calculated in step 1. Other vertex images, e.g. q_2, q_3 and q_4, are calculated by the RK4 formula. Note that the line segments $\overline{q_2q_3}$ and $\overline{q_3q_4}$ in Fig. 4(c) define the interface locus in that cell.

Step 4 – simplify representation and conserve mass

Although the RK4 formula generates very small errors for point locations, in general the mass difference between the image SSP and preimage SSP

$$\delta_{[c]} = \left\|\mathcal{M}^{n+1}_{[c]}\right\| - \left\|\overleftarrow{\mathcal{M}^{n+1}_{[c]}}\right\| \tag{22}$$

is not zero. To conserve mass for incompressible flow, the following must be enforced:

$$\left\|\mathcal{M}^{n+1}_{[c]}\right\| = \left\|\overleftarrow{\mathcal{M}^{n+1}_{[c]}}\right\|. \tag{23}$$

A related stability issue is that the number of simple polygons in a SSP, s_p, s_h, as well as the number of points in one polygon, might increase quickly in a complicated velocity field. To maintain stability and efficiency, we simplify the representation and conserve mass by the following steps:

(1) delete all the hole polygons from $\mathcal{M}_{[c]}^{n+1}$;
(2) if $s_p > 1$ or (7) is violated, we apply a two-dimensional *convex hull* algorithm [6] on the finite point set $\cup_{i=1}^{s_p} \mathsf{V}_{\mathcal{P}_i}$ where the set of all vertices of polygon $\mathcal{P}$ is denoted by
$$\mathsf{V}_{\mathcal{P}} = \mathsf{V}(\mathcal{P}) = \{p_0, \ldots, p_{n_{\mathcal{P}}-1}\}; \tag{24}$$
(3) move interface points of $\mathcal{P}_{\mathrm{CH}}$ so that $\|\mathcal{P}_{\mathrm{CH}}\| = \|\overleftarrow{\mathcal{M}_{[c]}^{n+1}}\|$, where $\mathcal{P}_{\mathrm{CH}}$ is the simple polygon resulting from the convex hull algorithm. In Section 3.2, we explain in full details Algorithms 2 and 3 for generic geometric configurations;
(4) introduce a global constant $n_{\max}$ as the maximal number of points allowed in a simple polygon. If $n_{\mathcal{P}_{\mathrm{CH}}} > n_{\max}$, keep deleting the least significant points from $\mathcal{P}_{\mathrm{CH}}$ until $n_{\mathcal{P}_{\mathrm{CH}}} = n_{\max}$, with the significance of a vertex p_i being measured by $\|\Delta(p_i, p_{i+1}, p_{i-1})\|$;
(5) set $\mathcal{M}_{[c]}^{n+1} = \{\mathcal{P}_{\mathrm{CH}}\}$ and call Algorithm 2 or Algorithm 3 again to satisfy (23).

An example illustrating the above steps is shown in Fig. 5. Note that we do allow more than one polygons in the material SSP in major Step 2 and Step 3. A premature simplification before major Step 4 generates large errors for merging. This is why we do not use one simple polygon as the representation of the material area in an interface cell.

More often than not, only one material polygon needs to be adjusted for mass conservation after tracing forward material preimages. An example is shown in Fig. 4(d).

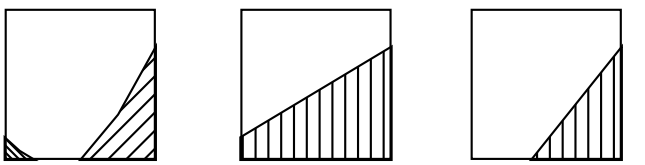
3.2. Adjustment of a simple polygon under the constraint of another

Let $\mathcal{P}$ and $\mathcal{P}'$ be the polygons before and after the adjustment, respectively, $\mathcal{C}$ the cell polygon. The following crucial observations guide the design of the heuristic algorithms.

- The change of $\mathcal{P}$ should be correct so that $\mathcal{P}' \subseteq \mathcal{C}$.
- The change of $\mathcal{P}$ should be minimized so that $\mathcal{P} \subset \mathcal{P}'$ for expanding and $\mathcal{P}' \subset \mathcal{P}$ for shrinking.
- The change of $\mathcal{P}$ should reduce $n_{\mathcal{P}}$ as much as possible so that the number of points that has to be deleted is minimal.
- Only points on the interface can be moved, among which those inside the cell are moved before those on the cell boundaries to maintain maximal continuity of the interface.
- To maintain 'shape' stability and ensure (7), we prefer $\mathcal{P}'$ to be a convex polygon rather than a concave polygon.

We capture the constraint of $\mathcal{C}$ on the expansion of $\mathcal{P}$ in the following definitions.

Definition 1 (*Expansion area*). Given two convex polygons $\mathcal{P} \subset \mathcal{C}$, the jth expansion area, $\mathcal{E}_j$, associated with the jth edge of $\mathcal{P}$ is



(a) end of step 3 (b) Convex hull (c) Adjustment

Fig. 5. Step 4 of the PAM method for two material polygons. After taking the convex hull of disjoint polygons, mass conservation is enforced by Algorithm 2 or Algorithm 3 in Section 3.2. See Fig. 9(b) and (c) for a different scenario.

$$\mathcal{E}_j(\mathcal{P},\mathcal{C}) = \mathcal{C} \cap \mathcal{R}_j(\mathcal{P},\mathcal{C}), \tag{25}$$

where $\mathsf{R}(q_0,q_1)$ denotes the ray starting from q_0 and shooting in the direction of $q_0 - q_1$:

$$\mathsf{R}(q_0,q_1) = \{p | p = q_0 + \alpha(q_0 - q_1), \alpha \geqslant 0\}, \tag{26}$$

and $\mathcal{R}_j(\mathcal{P},\mathcal{C})$ is the area bounded by $\mathsf{R}(p_j,p_{j-1}), \mathsf{R}(p_{j+1},p_{j+2})$ and $\overline{p_jp_{j+1}}$.

The convexity of $\mathcal{P}$ implies the angle between $\mathsf{R}(p_j,p_{j-1})$ and $\mathsf{R}(p_{j+1},p_{j+2})$ to be less than π, thus permits no ambiguity of $\mathcal{R}_j(\mathcal{P},\mathcal{C})$, although it might be unbounded. Clearly $\mathcal{E}_j = \emptyset$ if and only if $\overline{p_jp_{j+1}} \subset \partial\mathcal{C}$.

Definition 2 (*Expansion point*). If $\overline{p_jp_{j+1}} \subset \partial\mathcal{C}$, we say the jth expansion point, $G_j(\mathcal{P},\mathcal{C})$, does not exist; otherwise $G_j(\mathcal{P},\mathcal{C})$ is the vertex of $\mathcal{E}_j$ satisfying

$$\|\Delta(G_j,p_j,p_{j+1})\| = \inf_{q\in\mathsf{V}(\mathcal{E}_j),\ q\notin\overline{p_jp_{j+1}}} \|\Delta(q,p_j,p_{j+1})\|. \tag{27}$$

we also call $\mathcal{T}_j = \Delta(G_j,p_j,p_{j+1})$ the jth expansion triangle.

Four typical cases of the expansion point and expansion area are shown in Fig. 6.

From the above definitions, it is clear that $G_j(\mathcal{P},\mathcal{C}) \in \mathcal{C}$. By (6) the convexity of $\mathcal{C}$ implies that the expanded polygon is within $\mathcal{C}$:

$$\widetilde{\mathcal{P}} = \mathcal{P} \cup \mathcal{T}_j \subseteq \mathcal{C}. \tag{28}$$

Claim 3. $\widetilde{\mathcal{P}}$ is convex.

Proof. An edge of a convex polygon $\mathcal{P}$ is part of a supporting line, of which all interior points of $\mathcal{P}$ lie to the same side [33]. Take the half-space of line p_jp_{j-1} with $\mathcal{P}$ inside and that of line $p_{j+1}p_{j+2}$ also with $\mathcal{P}$ inside, denote their intersection by $\mathcal{K}$. $\mathcal{K}$ is clearly convex since the intersection of convex sets is convex [33]. By Definitions 1 and 2, $\overline{p_jp_{j+1}}$ divides $\mathcal{K}$ into two parts $\mathcal{K}_1$ and $\mathcal{K}_2$ such that $\mathcal{P} \subseteq \mathcal{K}_1$ and $\mathcal{T}_j \subseteq \mathcal{K}_2$. If two points q_0, q_1 are both inside $\mathcal{P}$ or $\mathcal{T}_j$, $\overline{q_0q_1} \subset \widetilde{\mathcal{P}}$ because of the convexity of $\mathcal{P}$ and $\mathcal{T}_j$. Consider $q_0 \in \mathcal{P}$ and $q_1 \in \mathcal{T}_j$, $\overline{q_0q_1} \subset \mathcal{K}$ because of the convexity of $\mathcal{K}$. The line segment from q_0 in $\mathcal{K}_1$ to q_1 in $\mathcal{K}_2$ thus must intersect the common boundary $\overline{p_jp_{j+1}}$, say, at q_2. The convexity of $\widetilde{\mathcal{P}}$ is thus proved by $\overline{q_0q_2} \subset \mathcal{P} \subset \widetilde{\mathcal{P}}$, $\overline{q_1q_2} \subset \mathcal{T}_j \subset \widetilde{\mathcal{P}}$, and (6). □

Definition 4 (*Classification of expansion points*). A stable expansion point (SEP) is an expansion point that satisfies

$$n_{\widetilde{\mathcal{P}}} \leqslant n_{\mathcal{P}}, \tag{29}$$

otherwise it is called an unstable expansion point (UEP).

Claim 5. If $G_j(\mathcal{P},\mathcal{C})$ is a UEP,

$$\begin{cases} G_j(\mathcal{P},\mathcal{C}) \notin \mathsf{R}(p_j,p_{j-1}), \\ G_j(\mathcal{P},\mathcal{C}) \notin \mathsf{R}(p_{j+1},p_{j+2}). \end{cases} \tag{30}$$

Proof. Since $\overline{p_jp_{j+1}}$ is a common edge of $\mathcal{P}$ and $\mathcal{T}_j$, the only possible new vertex is G_j. Suppose $G_j(\mathcal{P},\mathcal{C}) \in \mathsf{R}(p_j,p_{j-1})$, then p_j, p_{j-1}, and G_j are collinear. However, after the insertion of G_j, p_j will be removed after fulfilling the representation invariant (11d). Thus $n_{\widetilde{\mathcal{P}}} = n_{\mathcal{P}}$, which is a contradiction to Definition 4. □

Claim 6. If $G_j(\mathcal{P},\mathcal{C})$ is a UEP, $G_j(\mathcal{P},\mathcal{C})$ is a vertex of $\mathcal{C}$.

Proof. By Definitions 1 and 2, there are three types of vertices in $\mathcal{E}_j$. (i) p_j and p_{j+1}, (ii) vertices of $\mathcal{C}$, (iii) intersection of $\partial\mathcal{C}$ to either $\mathsf{R}(p_j,p_{j-1})$ or $\mathsf{R}(p_{j+1},p_{j+2})$. (i) is not possible by Definition 2. (iii) is not possible by Claim 5. □

An example of UEP is shown in Fig. 6(d), all other expansion points in Fig. 6 are SEPs.

Definition 7 (*Least expansion triangle*). A least expansion triangle (LET) is the smallest triangle among existing expansion triangles:

$$\|\mathcal{T}_{\min}(\mathcal{P},\mathcal{C})\| = \inf \left\|\mathcal{T}_j(\mathcal{P},\mathcal{C})\right\| \quad (1 \leqslant j \leqslant n \text{ and } G_j(\mathcal{P},\mathcal{C}) \text{ exists}). \tag{31}$$

In the following, we will omit the parameters of a function form if no ambiguity is generated by doing so, e.g. $\mathcal{T}_j = \mathcal{T}_j(\mathcal{P},\mathcal{C})$.

Procedure `AdjustTriangle` $(p_0, p_1, p_2, q, \delta)$

Data:The maximal volume change $\|\Delta(q,p_1,p_2)\| - \|\Delta(p_0,p_1,p_2)\|$ and the expected volume change δ are both positive or negative.

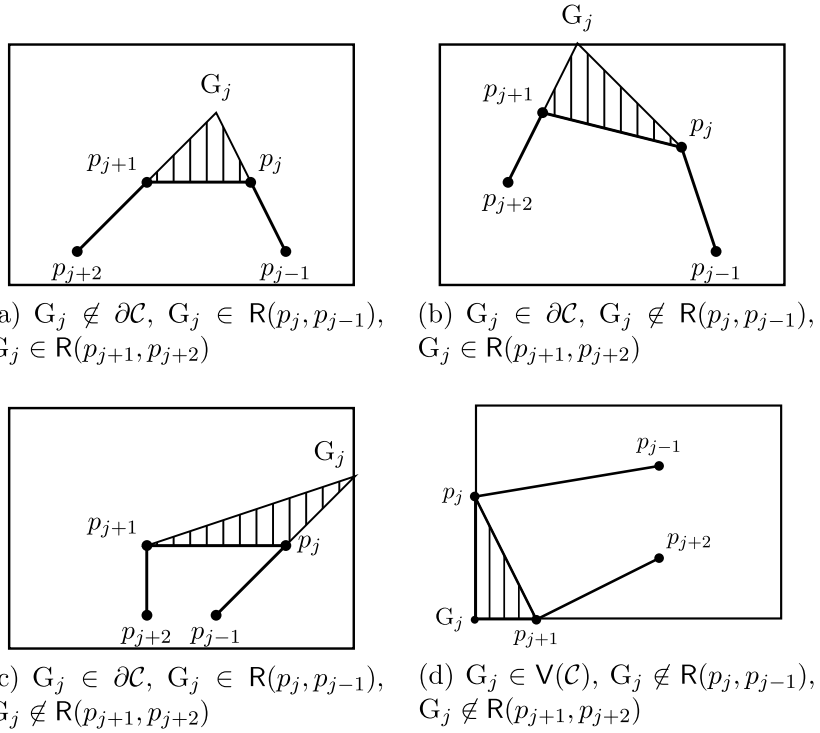
Result: p_0 is changed; the actual volume change is returned.

1 $a \leftarrow \|\Delta(q,p_1,p_2)\| - \|\Delta(p_0,p_1,p_2)\|$
2 $r \leftarrow \delta/a$
3 if $r > 1$ **then** $r = 1$
4 $p_0 \leftarrow rq + (1-r)p_0$
5 return ra

A simple procedure, `AdjustTriangle`, tries to change a triangle by moving its first point p_0 to a fixed point q until p_0 reaches q or the expected volume change δ is fulfilled. The actual volume change is returned. The algorithms of expanding and shrinking a simple polygon use `AdjustTriangle` as the fundamental building block.

Algorithm 2 formalizes the steps of expanding a simple polygon. This algorithm seeks to reduce $n_{\mathcal{P}}$ at each step by adjusting concave triangles or the LETs.

Claim 8. Given a convex polygon $\mathcal{C}$, another simple polygon $\mathcal{P} \subset \mathcal{C}$, and a volume increment $0 < \delta \leqslant \|\mathcal{C}\| - \|\mathcal{P}\|$, Algorithm 2 changes $\mathcal{P}$ to $\mathcal{P}'$ such that $\|\mathcal{P}'\| = \|\mathcal{P}\| + \delta$ and $\mathcal{P} \subset \mathcal{P}' \subseteq \mathcal{C}$.



(a) $G_j \notin \partial\mathcal{C}$, $G_j \in \mathsf{R}(p_j, p_{j-1})$, $G_j \in \mathsf{R}(p_{j+1}, p_{j+2})$

(b) $G_j \in \partial\mathcal{C}$, $G_j \notin \mathsf{R}(p_j, p_{j-1})$, $G_j \in \mathsf{R}(p_{j+1}, p_{j+2})$

(c) $G_j \in \partial\mathcal{C}$, $G_j \in \mathsf{R}(p_j, p_{j-1})$, $G_j \notin \mathsf{R}(p_{j+1}, p_{j+2})$

(d) $G_j \in \mathsf{V}(\mathcal{C})$, $G_j \notin \mathsf{R}(p_j, p_{j-1})$, $G_j \notin \mathsf{R}(p_{j+1}, p_{j+2})$

Fig. 6. Typical cases of expansion points and expansion areas. The rectangle represents the cell polygon. Hatched area is the jth expansion triangle.

Proof. Volume of a polygon is a continuous function with respect to vertex locations. $\|\mathcal{P}'\| = \|\mathcal{P}\| + \delta$ can thus be enforced by `AdjustTriangle` and the fact that we do not change more than one triangle at the same time. $\mathcal{P} \subset \mathcal{P}'$ holds because each invoking of `AdjustTriangle` only appends extra area to $\mathcal{P}$. In lines 2–4, (7b) implies $\Delta(p_i, p_{i+1}, p_{i-1}) \cap \mathcal{P} = \emptyset$ and the convexity of $\mathcal{C}$ implies $\Delta(p_i, p_{i+1}, p_{i-1}) \subset \mathcal{C}$, which guarantees $\mathcal{P}' \subseteq \mathcal{C}$; In lines 6–19, $\mathcal{P}' \subseteq \mathcal{C}$ holds because of (28). Since expansion triangles are non-degenerating, the value of δ is always reduced, thus the while loops eventually terminates. By Claim 3, once the execution passes line 5. $\mathcal{P}$ can never be concave, this realizes line 7. The conditions in line 8–18 are based on Claim 5. By Definition 2, the non-existence of any expansion triangles implies $\overline{p_j p_{j+1}} \subset \partial\mathcal{C}$ for $1 \leqslant j \leqslant n_{\mathcal{P}}$, which in turn implies $\mathcal{P} = \mathcal{C}$. Thus the while loops terminates properly. □

Algorithm 2. Expand a simple polygon

Data: convex polygon $\mathcal{C}$; simple polygon $\mathcal{P} \subset \mathcal{C}$; a volume increment $0 < \delta \leqslant \|\mathcal{C}\| - \|\mathcal{P}\|$
Result:change $\mathcal{P}$ to $\mathcal{P}'$ s.t. $\|\mathcal{P}'\| = \|\mathcal{P}\| + \delta$, $\mathcal{P} \subset \mathcal{P}' \subseteq \mathcal{C}$

```
1  while δ > 0 and at least one concave vertex exists. do
2      i ← the index of the least significant concave vertex
3      δ ← δ + AdjustTriangle(p_i, p_{i-1}, p_{i+1}, (p_{i-1} + p_{i+1})/2, −δ)
4      if ‖P(p_i, p_{i+1}, p_{i-1})‖ = 0 then delete p_i from P.
5  end
6  while δ > 0 do
7      i ← the index of LET
8      if G_i ∉ R(p_i, p_{i-1}) and G_i ∈ R(p_{i+1}, p_{i+2}) then
9          δ ← δ − AdjustTriangle(p_{i+1}, p_i, p_{i+1}, G_i, δ)
10     else if G_i ∈ R(p_i, p_{i-1}) and G_i ∉ R(p_{i+1}, p_{i+2}) then
11         δ ← δ − AdjustTriangle(p_i, p_{i+1}, p_i, G_i, δ)
12     else if G_i ∈ R(p_i, p_{i-1}) and G_i ∈ R(p_{i+1}, p_{i+2}) then
13         δ ← δ − AdjustTriangle(p_i, p_{i+1}, p_i, G_i, δ)
14         if ‖P(p_i, p_{i+1}, G_i)‖ = 0 then delete p_{i+1} from P.
15     else
16         insert q = p_i into P between p_i and p_{i+1}
17         δ ← δ − AdjustTriangle(q, p_i, p_{i+1}, G_i, δ)
18     end
19 end
```

We show a polygon expanded by different values of δ in Fig. 7. Algorithm 2 almost always reduces the number of vertices in a simple polygon except in the case of a UEP, which corresponds to lines 16–17 of Algorithm 2. However, because of Claim 6, the number increase is bounded by the number of vertices in $\mathcal{C}$. We emphasis that this increase is over *all* time steps. In fact, the case shown in Fig. 6(d) is very rare, our numerical tests also confirmed this.

The shrink of $\mathcal{P}$ is easier than the expansion of $\mathcal{P}$ in that the constraint of C is much weaker.

Definition 9 (*Convex shrink triangle*). Let $\mathcal{P}$ be a convex polygon, the jth convex shrink triangle (VST) associated with the jth vertex p_j is

$$\mathcal{S}_j = \Delta(p_j, p_{j+1}, p_{j-1}). \tag{32}$$

Definition 10 (*Concave shrink triangle*). Let $\mathcal{P}$ be a concave polygon, p_j a concave vertex, and p_k an adjacent convex vertex. The concave shrink triangle (CST) associated with an edge $\overline{p_j p_k}$ $(k = j \pm 1)$ is defined as

$$\mathcal{S}_{j,j+1} = \begin{cases} \Delta(p_j, p_{j+1}, p_{j+2}) & \text{if } \mathsf{R}(p_j, p_{j-1}) \cap^* \overline{p_{j+1} p_{j+2}} = \emptyset, \\ \Delta(p_j, p_{j+1}, q) & \text{if } \mathsf{R}(p_j, p_{j-1}) \cap^* \overline{p_{j+1} p_{j+2}} = \{q\}. \end{cases} \tag{33}$$

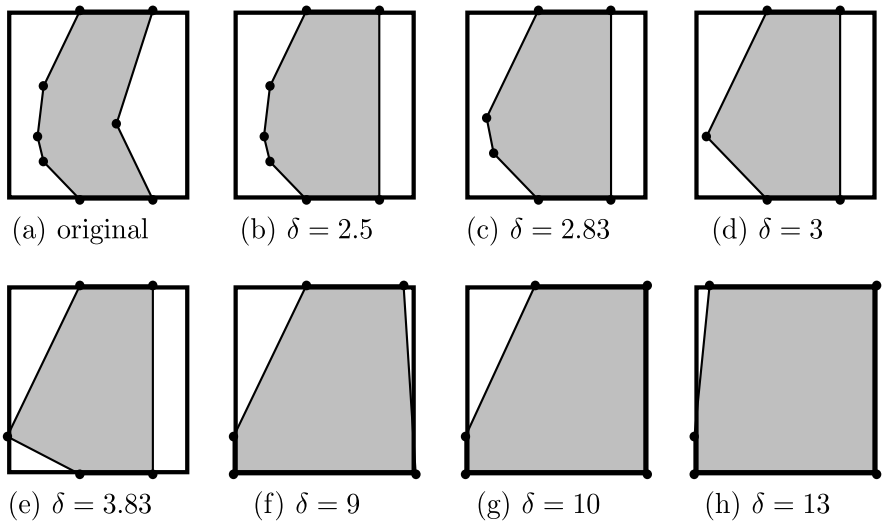


Fig. 7. Expanding a simple polygon. The rectangle is a cell with $\|\mathcal{C}\| = 25$, the shaded area is $\mathcal{P}$. The correspondence to Algorithm 2 is: (a) → (b): lines 3–4; (b) → (c) → (d): lines 12–14; (d) → (e): lines 10–11; (e) → (f): lines 15–17 and lines 12–13; (f) → (g): lines 15–17; (g) → (e): lines 12–13.

$\mathcal{S}_{j,j-1}$ is similarly defined by changing each '+' to '−' and '−' to '+' in all the subscripts in (33).

Definition 11 (*Least shrink triangle*). For a concave polygon $\mathcal{P}$, the least shrink triangle (LST), $\mathcal{S}_{\min}(s_0, s_1, s_2)$, is the CST with the smallest volume. For a convex polygon, the LST is the VST with the smallest volume.

Note that s_0, s_1, s_2 are place holders corresponding to the points in (32) and (33) *in the same order*. In practice, we only consider vertices $p_j \notin \partial\mathcal{C}$ until no such vertices exists in $\mathcal{P}$. This reflects our preference of interior points in adjusting the material polygon. Also, p_j has to be a vertex on the interface, otherwise spurious interface will be created by adjusting the associated triangle.

Algorithm 3 formalizes the steps of shrinking a simple polygon. The proof of correctness is similar to that of Claim 8 and is a much easier one. The key points are (7a) and the convexity of $\mathcal{P}$. In contrast to Algorithm 2, the algorithm *always* reduces the number of vertices of the polygon, since there is no insertion of new vertices.

Algorithm 3. Shrink a simple polygon

Data: convex polygon $\mathcal{C}$; simple polygon $\mathcal{P} \subset \mathcal{C}$; a volume decrease $\|\mathcal{P}\| \leqslant \delta < 0$
Result: change $\mathcal{P}$ to $\mathcal{P}'$ s.t. $\|\mathcal{P}'\| = \|\mathcal{P}\| + \delta$, $\mathcal{P}' \subset \mathcal{P} \subset \mathcal{C}$

```
1  while δ < 0 do
2      S_min(s0, s1, s2) ← the LST of P
3      if S_min is a CST then
4          δ ← δ − AdjustTriangle(s1, s0, s2, s2, δ)
5          if ‖Δ(s1, s0, s2)‖ = 0 then delete s1 from P
6      else
7          q ← (s1 + s2)/2
8          if s0s1 ⊂ ∂C then q ← s1
9          else if s0s2 ⊂ ∂C then q ← s2
10             δ ← δ − AdjustTriangle(s0, s1, s2, q, δ)
11             if ‖Δ(s0, s1, s2)‖ = 0 then delete s1 from P
12     end
13 end
```

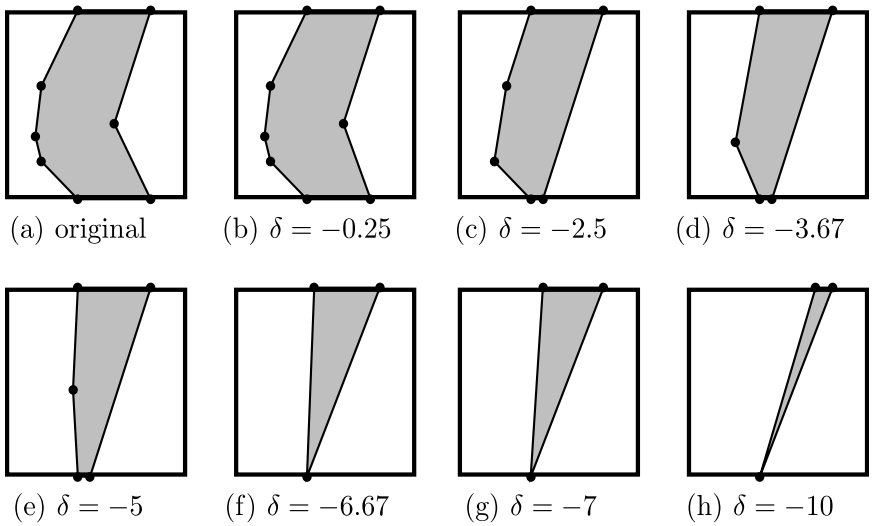


Fig. 8. Shrinking a simple polygon. The rectangle is a cell with $\|\mathcal{C}\| = 25$. The correspondence to Algorithm 3 is: (a) → (b): lines 4; (b) → (c): lines 4–5; (c) → (d) → (e) → (f): lines 7,10–11; (f) → (g) → (h): lines 8–11.

A polygon shrunken by different values of δ is shown in Fig. 8. It is clear that each step in the above algorithm trims a triangle off the simple polygon, thus the new polygon is always a subset of the original one.

Although Algorithms 2 and 3 has worst-case complexity of $O(n_{\mathcal{P}}^2)$, $\delta_{[c]}$ in (22) is typically much smaller than the total volume of the material area, thanks to the high accuracy of the RK4 formula. In addition, $n_{\max}$, the maximum number of points in a material polygon, is set to a number much smaller than $1/h$. Therefore the overhead caused by these algorithms is insignificant.

In essence, Algorithms 2 and 3 are simple greedy algorithms with the local optimal goals to make material polygons more stable and efficient in the sense that

- each time a LET or a LST is found, δ is consumed as little as possible to maximize the possibility of reducing number of vertices. This changes the polygon towards a more stable one.
- the least significant point is always deleted first. Physically this corresponds to omitting the smallest detail the polygon can resolve. This changes the polygon towards a more efficient one.

Including mass conservation, the advantages of the volume adjusting algorithms are thus threefold.

3.3. Merging of two material polygons

We exemplify the PAM method by showing how the algorithms work for the merging of two material polygons, as shown in Fig. 9(a).

If the Courant number is large enough, the two image polygons will have some overlapping area, as in Fig. 9(b). After fulfilling the representation invariant (11), the two polygons will be united into one, which is adjusted further for mass conservation, as in Fig. 9(c). We call the above scenario a *successful merging*. In this case, polygon union and volume adjusting serve as the merging 'mechanisms'.

However, if the Courant number is not big enough, as in Fig. 9(d), the two image polygons will stay disjoint after the tracing forward, as in Fig. 9(e). According to the simplification steps, a convex hull is calculated to approximate the material SSP, as in Fig. 9(f), before the mass conservation is enforced in Fig. 9(g). We call the above scenario an *unsuccessful merging*. In this case, the convex hull algorithm instead of polygon union is used.

It is clear from Fig. 9 that an unsuccessful merging generates much more error than a successful merging. Even worse, an unsuccessful merging can happen again and again, given the Courant number is small enough.

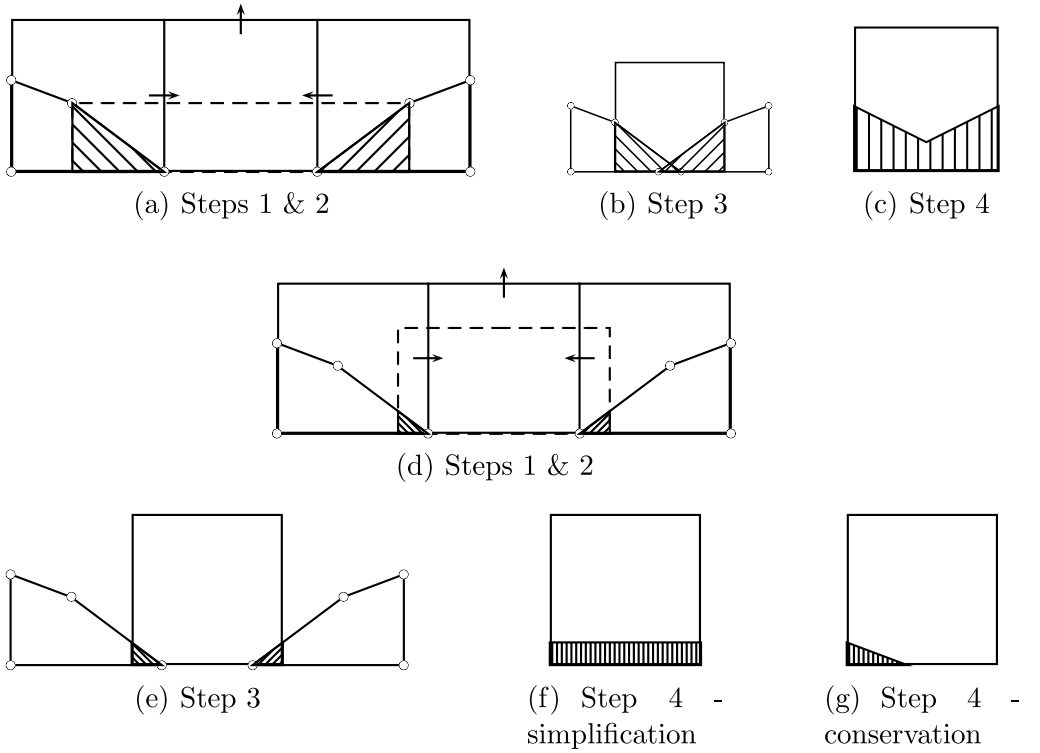


Fig. 9. Merging of two material polygons. (a)–(c) represent a successful merging with $C_r = 0.6$. (d)–(g) represent a unsuccessful merging with $C_r = 0.2$. Steps corresponds to the major steps in Section 3.1. The cell preimage is enclosed in the dashed lines. Hatched areas represent material preimages in (a) and (d), and material images in other sub-figures.

Also, finer grids tend to have more unsuccessful merging than coarser grids for a fixed Courant number because of a larger number of interface cells. These are confirmed in the deformation tests in Section 4.4.

4. Tests

In this section, we test the accuracy and efficiency of the new PAM method on structured rectangular grids for various different scenarios: translation, rotation, shear, breaking up, and merging, all of which have been widely accepted as benchmark tests for interface tracking methods ([26,25,8,9,1,3,2,27,23,14,15]). We also compare the PAM method to other state-of-art VOF methods using two different error measurement criteria. One is the *relative* distortion of the material area [26]:

$$E_r = \frac{\sum_{i,j}\left|f_{i,j}(T) - f_{i,j}^{T/k}\right|}{\sum_{i,j} f_{i,j}^0}, \tag{34}$$

where $f_{i,j}(T)$ is the exact volume fraction for cell $[i,j]$ at the end of the test $t = T$ and $f_{i,j}^n$ is the computational result at time step n; for the PAM method, $f_{i,j}^n$ is calculated by $f_{i,j}^n = \|\mathcal{M}_{i,j}^n\|/\|\mathcal{C}_{i,j}^n\|$. The other measures the *absolute* difference between the exact and the computed results [25]:

$$E_g = h^2 \sum_{i,j} \left\| f_{i,j}(T) - f_{i,j}^{T/k} \right|, \tag{35}$$

which is also called the geometrical error in [1,2].

Apart from the interface tracking errors, the *relative* mass conservation error is defined as

$$E_m = \frac{\left|\sum_{i,j} f_{i,j}(T) - \sum_{i,j} f_{i,j}^{T/k}\right|}{\sum_{i,j} f_{i,j}^0}. \tag{36}$$

Parameter	Value
domain	$[0,4]\times[0,4]$
$n_{\max}$	10
grids	200×200
shape	$D=0.8,\ d=0.4$
θ	26.57°
h	0.02
C_r	0.25
n_{step}	500
velocity	$\boldsymbol{u}_1=[1,0]^T, \boldsymbol{u}_2=[2,1]^T$
time step	$k_1=0.005, k_2=0.0025$

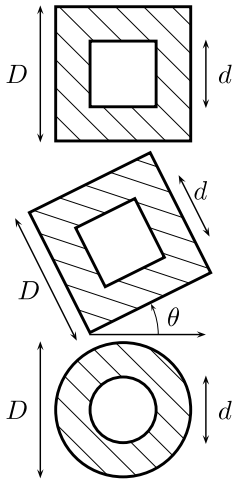


Fig. 10. Initial setup of translation tests.

It follows from (34) and (36) that $E_m \leqslant E_r$. Thus, we do not have to consider E_m so long as the interface tracking error is small enough. However, a large value of E_m/E_r points to the right direction how a tracking method can be improved: one should improve the mass conservation before improving the tracking method itself.

4.1. Translation test

Translation tests measure the most basic feature of an interface tracking method. In these tests [26], a hollow square, a tilted hollow square and a hollow circle are advected in two constant velocity fields, resulting six different cases. The initial setup and other test parameters are shown in Fig. 10. The material area and the shape should remain unchanged through the translation process.

The errors of the interface tracking are measured by (34) and shown in Table 1.

Table 1
Results of the translation tests

Methods		Square (0°)	Square (26.57°)	Circle
Velocity field $\mathrm{u}_1=[1,0]^T$				
VOF	SLIC	8.42e−8	1.46e−2	1.30e−2
	Hirt–Nichols	1.03e−8	6.91e−2	4.55e−2
	FCT–VOF	3.89e−8	2.32e−2	1.28e−2
	Youngs	1.08e−3	5.35e−3	3.08e−3
	Stream/Puckett [8]	1.61e−3	4.57e−3	1.42e−3
	DDR/ELVRIA	1.65e−3	4.80e−3	2.98e−3
PAM		3.50e−15	1.33e−4	3.04e−13
Velocity field $\mathrm{u}_2=[2,1]^T$				
VOF	SLIC	1.32e−1	1.08e−1	9.18e−2
	Hirt–Nichols	6.86e−3	1.60e−1	1.90e−1
	FCT–VOF	1.63e−8	8.15e−2	3.99e−2
	Youngs	2.58e−2	3.16e−2	2.98e−2
	Stream/Puckett[8]	3.33e−2	3.15e−2	6.96e−3
	DDR/ELVIRA	1.50e−1	1.45e−1	1.54e−1
PAM		7.77e−15	4.43e−4	4.30e−11

Results of the first four methods are taken from [26]; those of DDR/ELVIRA are from the authors' implementation.

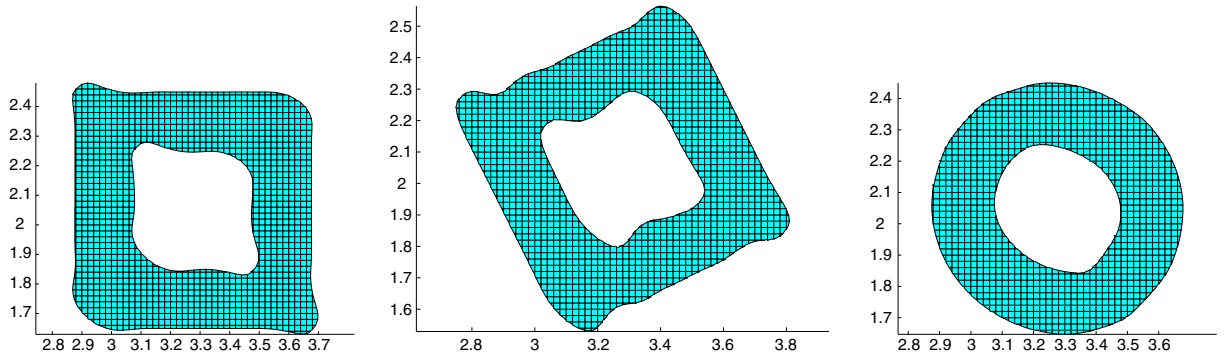


Fig. 11. Results of the author's implementation of DDR/ELVIRA for the translation tests under velocity $\mathbf{u} = [2, 1]^T$. Material area polygons instead of contour lines of $f_{i,j}$ are shown in the subplots and all other figures hereafter.

Most of the advection schemes of VOF methods predict accurately the values of $f_{i,j}$ for this trivial velocity field, but the reconstruction schemes do not have the same accuracy because of two reasons: the limitation of the representation of material interface within a cell as a piecewise linear function and the numerical diffusion resulting from utilizing the information of neighboring cells in reconstructing the material interface. Consequently, singularities of the material interface, e.g. the corners of a square, are rounded and lost. In Fig. 11, we show the results of the DDR method [9] coupled with ELVIRA [22]. The corners are rounded by ELVIRA and the material area is "squashed" in the corner direction because the DDR method neglects the fluxes in the cell-diagonal directions.

In contrast, the PAM method preserves the corners of the square as shown in Fig. 12 and yields more accurate results. In fact, the ODE solver calculates point locations to the machine precision for this constant velocity field; furthermore, the interface does not deform during the translation. Therefore, the mass conservation is satisfied naturally and there is no need to adjust the material polygons. In the cases that only one polygon is sufficient to represent the material area in a cell, the PAM method should yield 'exact' solution close to machine round-off error. This is confirmed by the results of the hollow square case and the hollow circle case, as shown in Table 1. However, the tilted hollow square has larger errors than the other two cases. The reason is that cells near the concave corners of the inner square sometimes have two separate material bodies (see Fig. 5(a) for an example), which get caught by the simplification steps. In this tests, the convex hull algorithm is the sole source of numerical errors.

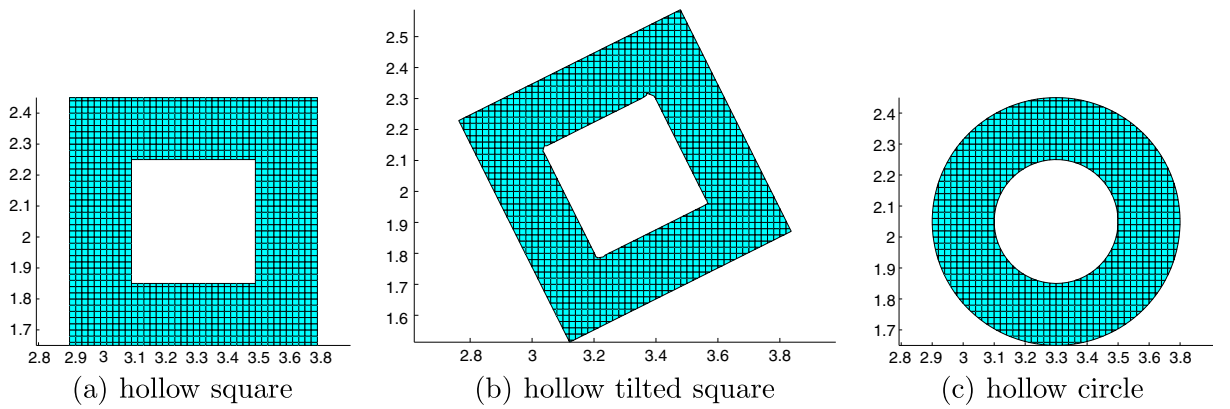


Fig. 12. Results of the PAM method for the translation tests under velocity $\mathbf{u} = [2, 1]^T$.

4.2. Zalesak disk rotation test

First introduced by Zalesak [35] and later used by Rudman [26] and many other researchers, this test places a slotted circle in a pure rotation velocity field with the following stream function:

$$\psi(x,y) = -\frac{\omega}{2}[(x-x_O)^2 + (y-y_O)^2], \tag{37}$$

where $O = [x_O, y_O]^T$ is the center of the rotation and ω is a constant angular velocity. The setup of this test and other parameters are shown in Fig. 13. After a full revolution of 2π rotation, the slotted circle returns to its initial location.

The interface tracking errors are measured by (34) and shown in Table 2. The results of the PAM method are better than those of all the listed VOF methods. In Fig. 14, the material interface of the PAM method at four time instants are compared to those of the DDR/ELVIRA method. As shown in the subplots (b), (d), (f), and (h), numerical diffusion in the DDR/ELVIRA method rounds the corners of the slot. The errors generated at the corners of the slot propagate to other interface cells and grow larger for longer simulation time. This is a common feature of all VOF methods since the reconstruction step utilizes the information of the nearby cells.

Parameter	Value
domain	$[0,4] \times [0,4]$
$n_{\max}$	10
grids	200×200
slot	$r = 0.4,\ s = 0.06$
circle	$R = 0.5,\ C = [2.0, 2.75]^T$
O	$[2.0, 2.0]^T$
ω	0.5
C_r	0.25
h	0.02
k	0.00498
n_{step}	2524

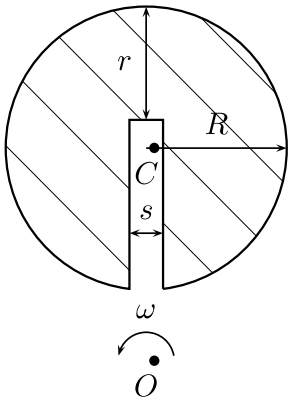


Fig. 13. Initial setup of Zalesak's rotation disk test.

Table 2
Results of the Zalesak rotation test

Methods		E_r
VOF	SLIC	8.38e−2
	Hirt–Nichols	9.62e−2
	FCT–VOF	3.29e−2
	Youngs	1.09e−2
	Stream/Puckett [8]	1.00e−2
	DDR/ELVIRA	5.25e−2
	Π/ELVIRA [3]	1.00e−2
	Lagrangian/Quadratic fit + continuity [27]	4.16e−3
PAM		3.97e−4

The first four lines are taken from [26]. DDR/ELVIRA is implemented by the authors.

In contrast, there is much less numerical diffusion using the PAM method, as shown in the subplots (a), (c), (e), and (g) in Fig. 14. There are no observable change at the two convex corners; as for the two concave corners, they are rounded to a small degree. In addition, the number of nearby cells influenced by the numerical diffusion is much less than that of DDR/ELIVIRA methods. For this nonlinear velocity field, the forward tracing and the backward tracing are not exact, even though the material interface does not deform under the pure rotation. The numerical diffusion of the PAM method in this test comes not only from the convex hull approximation, but also from the tendency in Algorithms 2 and 3 to get rid of a concave vertex from the material polygon. This is different from the translation tests.

4.3. Rider–Kothe reversed single vortex test

In this test [25], a non-uniform vorticity field with the following stream function:

$$\psi(x,y) = \frac{1}{\pi}\sin^2(\pi x)\sin^2(\pi y)\cos\left(\frac{\pi t}{T}\right) \tag{38}$$

is imposed on a circular material. The initial setup and other test parameters are shown in Fig. 15. The velocity field stretches the material interface into a filament that spirals toward the vortex center, potentially tearing the material apart if the grids resolution is not fine enough or the interface tracking methods are not good enough. At time $t = T/2$, the velocity field is reversed by the cosinusoidal temporal factor such that the exact interface at $t = T$ is the same as the initial setup [12].

The errors are measured by (35) and compared to those of previous work in Table 3. Relative tracking error, E_r, can be obtained by scaling E_g with the reciprocal of the total volume of the circle, $E_r = 14.15E_v$. The order of accuracy $\mathcal{O}_h$ is calculated by

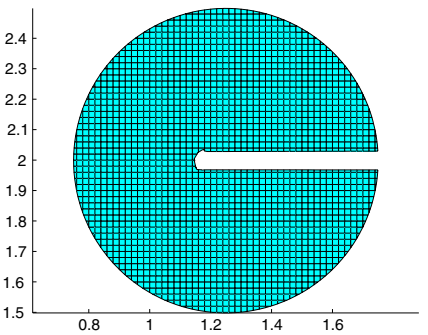
$$\mathcal{O}_h = \log_2\left(\frac{E_g\left(\frac{1}{2h}\right)}{E_g\left(\frac{1}{h}\right)}\right), \tag{39}$$

where $E_g\left(\frac{1}{h}\right)$ is the total volume difference on grids with $\Delta x = \Delta y = h$.

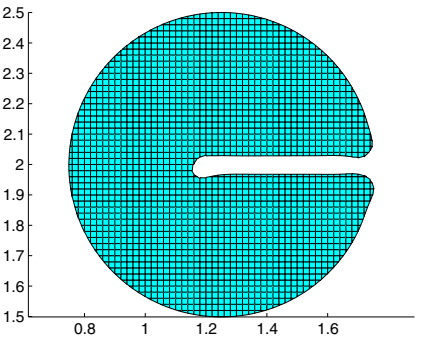
Three periods, $T = 0.5, 2.0$ and 8.0, are tested using different Courant numbers. In the cases of $T = 0.5$ and $T = 2$, different Courant numbers yields roughly the same results for the PAM method. However, in the case of $T = 8$, the errors of the $h = 1/32$ grids grow noticeably as the Courant number is reduced. A closer investigation shows that there are many cells with similar topology as that shown in Fig. 5(a) around the concave side of the filament (see Fig. 16). Smaller Courant number thus implies more frequent invoking of the convex hull approximation procedure, and consequently increasing errors. However, this problem is reduced when the grids are refined, as in the column of $E_g(128)$.

In Fig. 16, we compare the results of the PAM method and the DDR/ELVIRA method for $T = 8$. The material area is subject to stretching when $t \in [0, \frac{T}{2}]$. Even on the coarsest grids of 32×32, the PAM method does not break up the material filament. In contrast, VOF methods tend to produce fragmentation of the material filament when its thickness becomes close to the cell size. The reason is that calculation of the interface normal on 3×3 stencils yields spurious results when there are more than one interface segment inside the same stencil. Consequently, the advection algorithm in the subsequent time steps also generates large errors, as shown in Table 3. Although the improved EMFPA-SIR methods [15] was specifically designed for tracking thin filaments, its accuracy is still not as good as that of the PAM method.

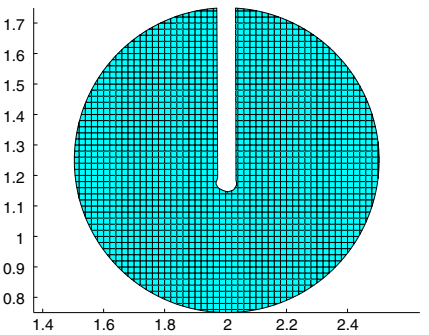
In VOF methods, information on the interface orientation is lost at the end of the advection step since the only results retained are the values of volume fractions. Thus the task of reconstructing the interface locus *solely* from the values of volume fractions is not well-posed. Choosing any fixed form of the interface, either a line, or a parabola, or several line segments, prevents convergence because the real form of the interface might not be any of the pre-determined ones. Consequently, the reconstruction step is the source of numerical diffusion and is responsible for the smearing of sharp corners. The reconstruction error will propagate along the interface in subsequent time steps and eventually the tracking accuracy of all interface cells deteriorates. In contrast, the PAM method does not specify a fixed form of the interface, but determines its locus from its pre-image via the mapping of the velocity field. This allows a natural and accurate tracking of the interface,



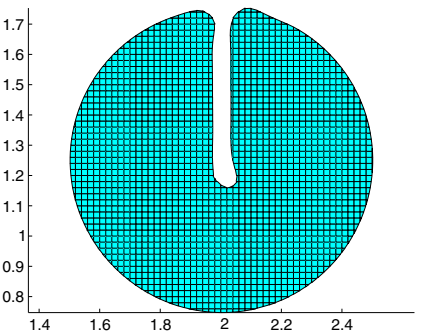
(a) $t = \frac{1}{4}T$, PAM



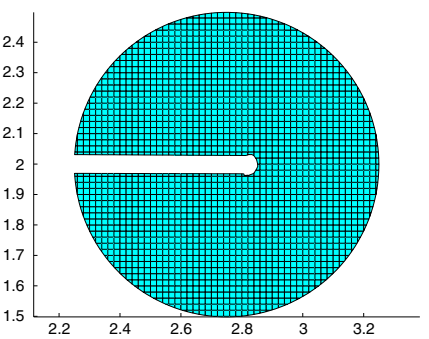
(b) $t = \frac{1}{4}T$, DDR/ELVIRA



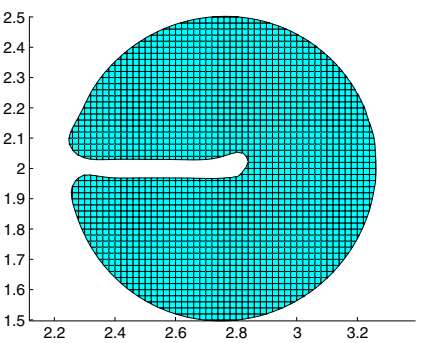
(c) $t = \frac{1}{2}T$, PAM



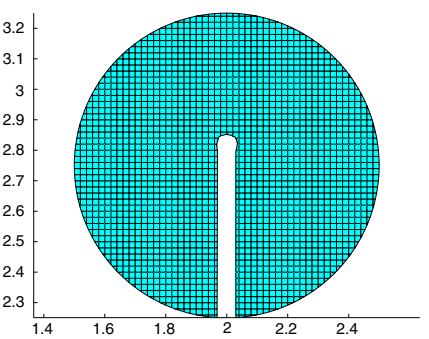
(d) $t = \frac{1}{2}T$, DDR/ELVIRA



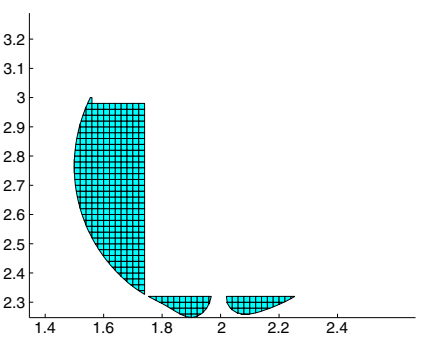
(e) $t = \frac{3}{4}T$, PAM



(f) $t = \frac{3}{4}T$, DDR/ELVIRA



(g) $t = T$, PAM



Parameter	Value
domain	$[0,1]\times[0,1]$
circle	$C=[0.5,0.75]^T$, $R=0.15$
$n_{\max}$	10
C_r	1, 0.1, 0.05
period	$T_1=0.5$, $T_2=2$, $T_3=8$
grid size	$h_1=1/32$, $h_2=1/64$, $h_3=1/128$

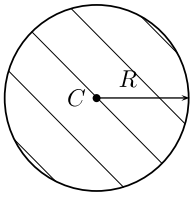


Fig. 15. Initial setup of the Rider and Kothe reversed single vortex test.

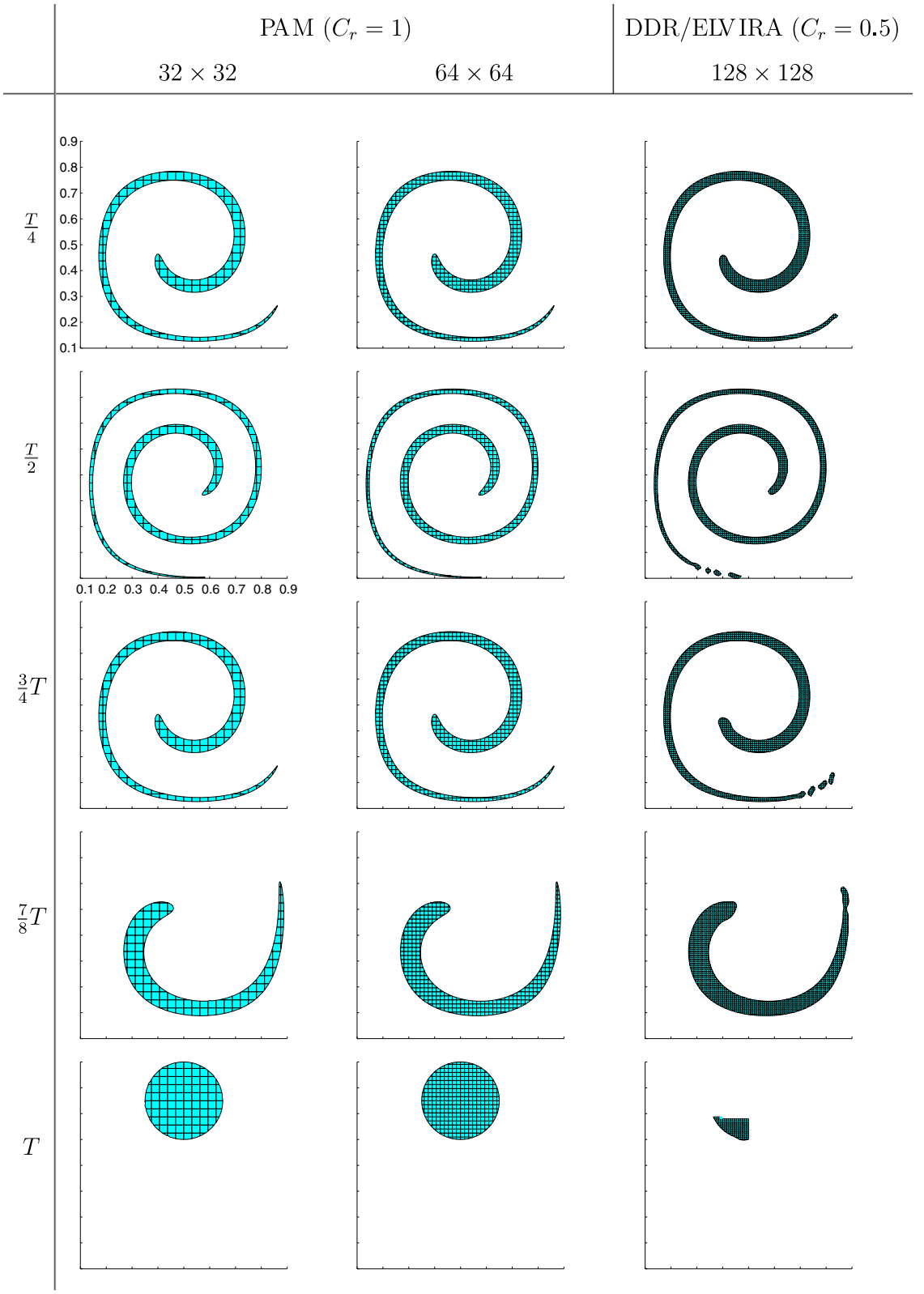
Table 3
Results of the Rider and Kothe reversed single vortex test

Methods		$E_g(32)$	$E_g(64)$	$\mathcal{O}_h$	$E_g(128)$	$\mathcal{O}_h$
$T=0.5$						
VOF	Rider and Kothe/Puckett [25]	7.29e−4	1.42e−4	2.36	3.90e−5	1.86
	Stream/Youngs [8]	3.42e−4	1.72e−4	0.99	9.60e−5	0.84
	EMFPA-SIR [14]	2.07e−4	5.36e−5	1.95	1.24e−5	2.11
PAM	$C_r=1.0$	1.41e−5	2.44e−6	2.53	5.15e−7	2.24
	$C_r=0.1$	9.62e−6	2.30e−6	2.32	5.74e−7	2.00
	$C_r=0.05$	1.15e−5	2.16e−6	2.41	5.22e−7	2.05
$T=2$						
VOF	Rider and Kothe/Puckett [25]	2.36e−3	5.85e−4	2.01	1.31e−4	2.16
	Stream/Youngs [8]	2.49e−3	7.06e−4	1.82	2.23e−4	1.66
	EI-LE/quadratic fit [27]	1.88e−3	4.42e−4	2.08	9.36e−5	2.24
	EMFPA-SIR [14]	8.62e−4	2.37e−4	1.86	5.62e−5	2.08
Hybrid markers-VOF [1] ($C_r=1$)		1.00e−3	2.69e−4	1.89	5.47e−5	2.30
PAM	$C_r=1.0$	4.24e−5	7.24e−6	2.55	1.35e−6	2.42
	$C_r=0.1$	2.43e−5	7.03e−6	1.79	1.59e−6	2.14
	$C_r=0.05$	2.45e−5	6.81e−6	1.85	1.36e−6	2.32
$T=8$						
VOF	Rider and Kothe/Puckett [25]	4.78e−2	6.96e−3	2.78	1.44e−3	2.27
	Stream/Youngs [8]	3.61e−2	1.00e−2	1.85	2.16e−3	2.22
	EMFPA-SIR [14]	4.64e−2	5.94e−3	2.97	5.39e−4	3.46
	Improved EMFPA–SIR [15]	5.78e−3	1.77e−3	1.71	3.30e−4	2.42
Hybrid markers–VOF [1] ($C_r=1$)		2.53e−2	2.78e−3	3.19	4.78e−4	2.54
Hybrid markers-conservation [2]	$C_r=1$	2.52e−3	3.23e−4	2.96	4.06e−5	2.99
	$C_r=0.1$	3.09e−4	3.99e−5	2.95	5.19e−6	2.94
PAM	$C_r=1.0$	2.07e−4	4.76e−5	2.12	6.87e−6	2.79
	$C_r=0.1$	1.02e−3	7.23e−5	3.82	5.37e−6	3.75
	$C_r=0.05$	1.32e−3	1.04e−4	3.67	8.59e−6	3.60

$E_r=14.15E_g$.

whether it contains singular points or not. Since volume adjusting algorithms only generate errors within the accuracy of the RK4 formula, the convex hull approximation is the only reason in which numerical diffusion can enter the computation of the PAM method. However, this convex hull algorithm is not frequently triggered in most applications. This is why the PAM method is more accurate than the VOF methods.

We compare the CPU time consumed by the PAM method to that of the DDR/ELVIRA method and find that the PAM method runs faster than the DDR/ELVIRA method for non-deforming interfaces in the translation and rotation tests, in which a material polygon only have four to six vertices. In the single vortex tests, the PAM method is slower about 20%. The reason is that the number of vertices of a material polygon increases up to 10 per interface cell due to a more complicated velocity field. Consequently, the most


PAM ($C_r = 1$)
DDR/ELVIRA ($C_r = 0.5$)
32×32
64×64
128×128
$\frac{T}{4}$
$\frac{T}{2}$
$\frac{3}{4}T$
$\frac{7}{8}T$
T
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9

time-consuming component, the RK4 formula, is repeated more often for tracing the vertices of the material polygons. Generally speaking, the CPU time of the PAM method increases as the velocity field gets more complicated. However, in a real application the velocity field rarely gets as complicated as that of the single vortex test. Even so, the CPU time consumed by the PAM method is of the same order as that of the DDR/ELVIRA method. In fact, interface tracking algorithm is often used with a velocity solver, e.g. a Navier–Stokes (N–S) solver. In a structured $1/h \times 1/h$ mesh, the computational cost of the N–S solver is $O(1/h^2)$ while that of the interface tracking algorithm is only $O(1/h)$, due to the fact that an interface is one-dimensional lower than the computational domain. Thus, the CPU time difference of various interface tracking methods are not noticeable in a large-scale multi-physics application.

Considering the accuracy and the fact that $n_{\max}$ is only 10, we conclude that the efficiency of the PAM method is satisfactory.

4.4. Deformation field test

Introduced in [29] and also used in [25,5,1,2], this test has been considered the most stringent among popular benchmarking problems. The velocity field is given by the stream function as

$$\psi(x,y) = \frac{1}{n\pi}\sin(n\pi(x+0.5))\cos(n\pi(y+0.5))\cos\left(\frac{\pi t}{T}\right), \tag{40}$$

where n is the number of vortexes in the computational domain and the temporal factor reverses the flow at $t = T/2$.

The initial setup and other test parameters are shown in Fig. 17 and the results of the PAM method are shown in Table 4. The exact solution returns to the initial condition at $t = T$. We show eight frames of the simulation results in Fig. 18. During the first half period, the material body is continuously stretched and very

Parameter	Value
domain	$[0,1] \times [0,1]$
circle	$C = [0.5, 0.5]^T$, $R = 0.15$
$n_{\max}$	20
C_r	1.0, 0.1
period	$T = 2$
grid size	$h_1 = 1/64$, $h_2 = 1/128$, $h_3 = 1/256$
number of vortexes	$n = 4$

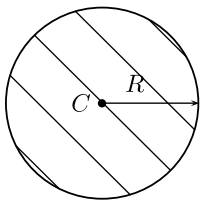
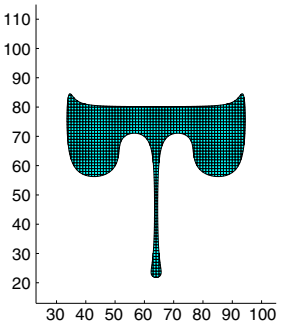


Fig. 17. Initial setup of the deformation field test.

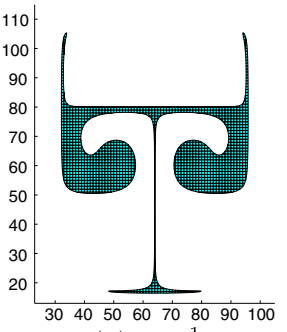
Table 4
Results of the PAM method for the deformation test

Grids	C_r	E_m	E_g	$\mathcal{O}_h$	E_m/E_r
64×64	1.0	7.11e−5	7.73e−5	–	6.50e−2
	0.1	7.16e−6	8.89e−5	–	5.66e−3
128×128	1.0	7.44e−6	7.19e−6	3.43	7.31e−2
	0.1	7.42e−7	1.41e−5	2.67	3.71e−3
256×256	1.0	7.45e−7	3.85e−7	4.22	1.37e−1
	0.1	9.02e−8	1.86e−6	2.92	3.43e−3

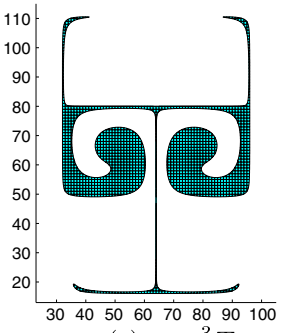
($E_r = 14.15E_g$).



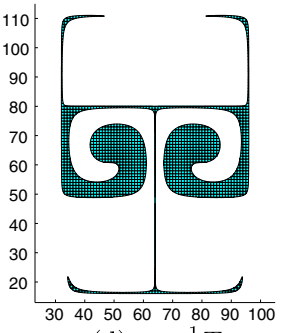
(a) $t = \frac{1}{8}T$



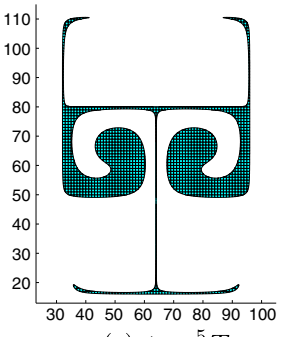
(b) $t = \frac{1}{4}T$



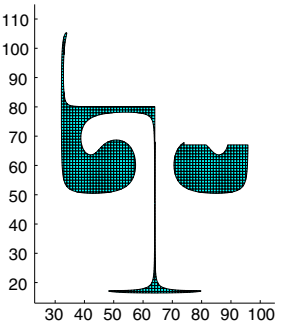
(c) $t = \frac{3}{8}T$



(d) $t = \frac{1}{2}T$



(e) $t = \frac{5}{8}T$



thin antennas are formed at $t = T/2$; while in the second half period the thin filaments and spiral areas merge back into each other.

As shown in Table 4, relative mass conservation errors become smaller when either the grids are refined or the Courant number is reduced. The small ratio of E_m/E_r implies that the mass conservation error is not the main error in the PAM method. However, as the Courant number become smaller, tracking errors grow. This is particularly true on fine grids. The reason of this has been explained in Section 3.3. An upper bound of the error generated by an unsuccessful merging is $O(h^2)$ for one interface cell. For grids with the same resolution, the growth of errors with respect to the Courant number is thus bounded by C_{r1}/C_{r2} where C_{r1} is the large Courant number and C_{r2} is the smaller one. In a practical simulation, though, large Courant numbers tend to be used to save CPU time and a Courant number smaller than 0.1 is seldom used. More importantly, the results of same Courant number on different grids shows a high convergence rate and the overall accuracy is closed to those in [2,5] and compares favorably to those in [25].

5. Conclusions

A new interface tracking method is proposed for better accuracy and flexibility. This method represents 2-D areas by polygons and tracks an interface by discrete Boolean set operations on polygons. A convex hull algorithm and two heuristic polygon volume adjusting algorithms are used to obtain stability. These novel features endow the new method with the following advantages:

(1) The representation of material areas and computational cells by polygons makes the PAM method independent of mesh topology.
(2) Inside an interface cell, VOF methods always assume the interface to be piecewise linear, quadratic, or cubic for the reconstruction step. This assumption generates numerical diffusion that smears the interface. In contrast, the PAM method makes no such assumption and determines an interface from its preimage naturally. It thus has very little numerical diffusion and yields much better results, especially for interfaces with singularities and high curvatures.
(3) Although both front-tracking methods and the PAM method solve ODEs to track Lagrangian markers, the PAM method is in essence an *area tracking* method while the front-tracking methods tracks fronts that are one-dimensional lower than the computational domain. Because operations upon polygons are much more robust than attaching and detaching line segments in complicated interface topologies, the PAM method appears to be more stable than front-tracking methods in the sense that the surgical operations for maintaining the interface topology are not needed.

Numerical results of benchmark tests confirms the high accuracy and the advantages of the PAM method. Furthermore, computational cost of the PAM method is roughly the same as those of the VOF methods.

Acknowledgements

We would like to acknowledge the supports from National Science Foundations through research grants to Cornell University. We also thank our referees for their helpful comments.

References

[1] E. Aulisa, S. Manservisi, R. Scardovelli, A mixed markers and volume-of-fluid method for the reconstruction and advection of interfaces in two-phase and free-boundary flows, J. Comput. Phys. 188 (2003) 611–639.
[2] E. Aulisa, S. Manservisi, R. Scardovelli, A surface marker algorithm coupled to an area-preserving marker redistribution method for three-dimensional interface tracking, J. Comput. Phys. 197 (2004) 555–584.
[3] E. Aulisa, S. Manservisi, R. Scardovelli, S. Zaleski, A geometrical area-preserving volume-of-fluid advection method, J. Comput. Phys. 192 (2003) 355–364.
[4] A.J. Chorin, Flame advection and propagation algorithm, J. Comput. Phys. 35 (1980) 1–11.
[5] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface tracking, J. Comput. Phys. 183 (2002) 83–116.

[6] R.L. Graham, An efficient algorithm for determining the convex hull of a finite planar set, Inform. Process. Lett. 1 (1972) 132–133.
[7] M. Granados, P. Hachenberger, S. Hert, L. Ketter, K. Mehlhorn, M. Seel, Boolean operations on 3d selective nef complexes: data structure, algorithms, and implementation, in: G.D. Battista, U. Zwick (Eds.), Algorithms – ESA 2003: 11th Annual European Symposium, Lecture Notes in Computer Science, vol. 2832, Springer, 2003, pp. 174–186.
[8] D.J.E. Harvie, D.F. Fletcher, A new volume of fluid advection algorithm: the stream scheme, J. Comput. Phys. 162 (2000) 1–32.
[9] D.J.E. Harvie, D.F. Fletcher, A new volume of fluid advection algorithm: the defined donating region scheme, Int. J. Numer. Meth. Fluids 35 (2001) 151–172.
[10] C.W. Hirt, B.D. Nichols, Volume of fluid (vof) method for the dynamics of free boundaries, J. Comput. Phys. 39 (1981) 201–225.
[11] C.M. Hoffmann, Solid modeling, in: J.E. Goodman, J. O'Rourke (Eds.), Handbook of Discrete and Computational Geometry, second ed., Chapman & Hall, 2004, pp. 1257–1278, chap. 56.
[12] R.J. LeVeque, High-resolution conservative algorithms for advection in incompressible flow, SIAM J. Numer. Anal. 33 (2) (1996) 627–665.
[13] R.J. LeVeque, Finite-Volume Methods for Hyperbolic Problems, Cambridge University Press, Cambridge, 2002, pp. 64–68, Chapter 4.
[14] J. López, J. Hernández, P. Gómez, F. Faura, A volume of fluid method based on multidimensional advection and spline interface reconstruction, J. Comput. Phys. 195 (2004) 718–742.
[15] J. López, J. Hernández, P. Gómez, F. Faura, An improved plic-vof method for tracking thin fluid structures in incompressible two-phase flows, J. Comput. Phys. 208 (2005) 51–74.
[16] A. Murta, A generic polygon clipping library: C sources and documentation. <http://www.cs.man.ac.uk/~toby/alan/software>, 2004.
[17] W.F. Noh, P. Woodward, Slic (simple line interface calculation), in: A.I. van Dooren, P.J. Zandbergen (Eds.), Lecture Notes in Physics, vol. 59, Springer-Verlag, Berlin, 1976, pp. 330–340.
[18] J. O'Rourke, Computational Geometry in C, second ed., Cambridge University Press, New York, 1998.
[19] S. Osher, R.P. Fedkiw, Level set methods: an overview and some recent results, J. Comput. Phys. 169 (2001) 463–502.
[20] S. Osher, R.P. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer-Verlag, New York, 2003.
[21] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1988) 12–49.
[22] J.E. Pilliod, An Analysis of Piecewise Linear Interface Reconstruction Algorithms for Volume-of-fluid Methods, Master's Thesis, University of California, Davis, December 1992.
[23] J.E. Pilliod Jr., E.G. Puckett, Second-order accurate volume-of-fluid algorithms for tracking material interfaces, J. Comput. Phys. 199 (2004) 465–502.
[24] E.G. Puckett, A volume-of-fluid interface tracking algorithm with applications to computing shock wave refraction, in: H. Dwyer, (Ed.), Proceedings of the Fourth International Symposium on Computational Fluid Dynamics, 1991, pp. 933–938.
[25] W.J. Rider, D.B. Kothe, Reconstructing volume tracking, J. Comput. Phys. 141 (1998) 112–152.
[26] M. Rudman, Volume-tracking methods for interfacial flow calculations, Int. J. Numer. Methods Fluids 24 (1997) 671–691.
[27] R. Scardovelli, S. Zaleski, Interface reconstruction with least-square fit and split Eulerian–Lagrangian advection, Int. J. Numer. Meth. Fluids 41 (2003) 251–274.
[28] S. Shin, D. Juric, Modeling three-dimensional multiphase flow using a level contour reconstruction method for front tracking without connectivity, J. Comput. Phys. 180 (2002) 427–470.
[29] P.K. Smolarkiewicz, The multi-dimensional crowley advection scheme, Month. Weather Rev. 110 (1982) 1968–1983.
[30] M. Sussman, E. Puckett, A coupled level set and vof method for computing 3d and axisymmetric incompressible two-phase flows, J. Comput. Phys. 162 (2000) 301–337.
[31] G. Tryggvason, B. Bunner, D. Juric, W. Tauber, S. Nas, J. Han, N. Al-Rawahi, Y.-J. Jan, A front-tracking method for the computations of multiphase flow, J. Comput. Phys. 169 (2001) 708–759.
[32] B.R. Vatti, A generic solution to polygon clipping, Commun. ACM 35 (7) (1992) 56–63.
[33] I.M. Yaglom, V.G. Boltyanskii, Convex Figures, Holt, Rinehart and Winston, Inc., 1961.
[34] 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 Press, New York, 1982, pp. 273–285.
[35] S.T. Zalesak, Fully multi-dimensional flux corrected transport algorithms for fluid flow, J. Comput. Phys. 31 (1979) 335–362.
[36] Y.L. Zhang, K.S. Yeo, B.C. Khoo, C. Wang, 3D jet impact and toroidal bubbles, J. Comput. Phys. 166 (2001) 336–360.