Journal of Computational Physics **155,** 410–438 (1999)
Article ID jcph.1999.6345, available online at http://www.idealibrary.com on IDEAL®

A PDE-Based Fast Local Level Set Method[1]

Danping Peng, Barry Merriman, Stanley Osher, Hongkai Zhao,[2] and Myungjoo Kang

Department of Mathematics, University of California at Los Angeles, Los Angeles, California 90095-1555

Received April 16, 1998; revised July 22, 1999

We develop a fast method to localize the level set method of Osher and Sethian (1988, *J. Comput. Phys.* **79**, 12) and address two important issues that are intrinsic to the level set method: (a) how to extend a quantity that is given only on the interface to a neighborhood of the interface; (b) how to reset the level set function to be a signed distance function to the interface efficiently without appreciably moving the interface. This fast local level set method reduces the computational effort by one order of magnitude, works in as much generality as the original one, and is conceptually simple and easy to implement. Our approach differs from previous related works in that we extract all the information needed from the level set function (or functions in multiphase flow) and do not need to find explicitly the location of the interface in the space domain. The complexity of our method to do tasks such as extension and distance reinitialization is $O(N)$, where N is the number of points in space, not $O(N \log N)$ as in works by Sethian (1996, *Proc. Nat. Acad. Sci.* **93**, 1591) and Helmsen and co-workers (1996, *SPIE Microlithography IX,* p. 253). This complexity estimation is also valid for quite general geometrically based front motion for our localized method. © 1999 Academic Press

1. INTRODUCTION

Since its inception in [20], the level set method has been used to capture rather than track interfaces. The advantages of this capturing approach are well known by now. The method is stable, the equations are not unnecessarily stiff, geometric quantities such as curvature become easy to compute, and three-dimensional problems present no difficulties. See [22, 18, 16] for a survey and references. Recent improvements include the computation of multiphase flows in [27, 28] and unstable fronts in [10, 11].

As pointed out in [1], "one drawback of the technique stems from the expense; by embedding the interface as the level set of a higher dimensional function, a one dimensional

[1] Research supported by DARPA/NSF Grant on thin films, NSF Grant DMS 9706827, and ARO Grant DAAG 55-98-1-0323.

[2] Current address: Department of Mathematics, University of California, Irvine, CA 92697. E-mail: zhao@math.uci.edu.

interface problem has been transformed into a two dimensional problem. In three space dimensions, considerable computational labor ($O(n^3)$) is required per time step."

We remark that there are physical problems, e.g., multiphase incompressible fluid dynamics [26, 5], compressible fluid dynamics [17], and melting ice problems [7], in which the additional level set equation adds only a fraction of extra computing time. This is because the underlying field equations must also be solved throughout all space.

In this paper we localize the level set method. Our localization works in as much generality as does the original method and all of its recent variants [27, 28], but requires an order of magnitude less computing effort.

Earlier work on localization was done by Adalsteinsson and Sethian [1]. Our approach differs from theirs in that we use only the values of the level set function (or functions, for multiphase flow) and not the explicit location of points in the domain. Our implementation is easy and straightforward and has been used in [9, 14, 27, 28].

Our approach is partial differential equation (PDE) based, in the sense that our localization, extension, and reinitialization are all based on solving different PDEs. This leads to a simple, accurate, and flexible method. Equations (10) and (11) of Section 2 enable us to update the level set function (or functions in the multiphase case) without any stability problems at the boundary of the tube used to localize the evolution. Such equations are new and do not appear in [1]. In fact, the technique used in [1] has boundary stability problems because Eq. (2) or (3) (the evolution equation of the level set function) is solved right up to this boundary. In contrast, in our method, the speed of evolution degenerates smoothly to 0 at this boundary. This is achieved by modifying the evolution of the level set function near the tube boundary but away from the interface. This modification effectively eliminates the boundary stability issues in [1] and has no impact on the correct evolution of the interface. The reinitialization step will reset the level set function to be a signed distance function to the front. There are no boundary issues in our distance reinitialization or extension of velocity field off the interface. Both of these tasks involve simple hyperbolic equations where characteristics flow out of the tube boundary; thus, upwind schemes remove all boundary difficulties. We make use of the state-of-the-art high order ENO [20, 21] and WENO [13] schemes to do updates, whenever it is appropriate to do so.

The local level set algorithm presented here is almost as easy to program as the global level set method, in-any number of dimensions. Our technique as described in Section 2 requires no complicated data structures to store the information about the tubes around the interface other than the ordinary array; thus, we eliminate the complexity and overhead of updating such data structures.

Another very interesting fast method was recently devised in [23] and [12] in the special case when the normal velocity of the front is a given nonnegative function of position. The key observation there is that the problem can be reduced to first arrival time (see also [19]) and that this time-independent equation can be solved by space marching using a first order accurate upwind scheme, as is commonly done, e.g., in the computation of steady supersonic flows. The complexity of the method developed in [23] and [12] is formally $O(N \log N)$, where N is the number of grid points in space. The complexity of our method is $O(N)$, which is optimal in the sense that it is proportional to the number of points near the front.

Our method works easily in the presence of topological changes, for general speed functions and for a wide class of numerical discretizations, as we demonstrate in Section 5 below. Unlike the method in [12, 23], which is necessarily first order accurate, our method is of an arbitrarily high order of accuracy.

This paper also addresses two important issues in level set computation: extension and reinitialization.

To move the interface, we need the normal velocity u_n in a neighborhood of the interface. In some applications, such as in mean curvature flow, u_n is naturally given globally or at least near the front. However, in some other applications, the normal velocity u_n is only known at the interface. The Stefan problem and Hele–Shaw flow problems fall into this category [7, 9]. We propose in this paper a PDE-based method to extend a quantity that is defined on the interface to a neighborhood of it along the direction normal to the interface. Our approach is conceptually straightforward and easy to implement. It has been successfully used in [7, 9] and elsewhere.

The extension step was first introduced and analyzed in [27]. In a recent interesting paper [2], Adalsteinsson and Sethian obtained a fast global velocity extension method. Their method is closely tied to the fast marching method of [23, 12]. Our extension method has lower formal order of complexity ($O(N)$ instead of $O(N \log N)$), although it remains to be seen which is faster for real local level set calculations. We shall use our method to obtain a very successful level set based computation of an unstable vortex sheet (see Fig. 11b below), considerably improving the results in [11].

In most cases, it is impossible to maintain the level set function as a signed distance function to the moving interface in the advection step. Flat and/or steep regions develop as the interface moves, rendering the computation and contour plotting at those places inaccurate. For numerical reasons, we need to resurrect the level set function to be close to a distance function from time to time. This is the so called distance reinitialization of the level set function. This process can be quite complicated, expensive, and have subtle by-products. A straightforward reinitialization by finding the location of the front and computing the signed distance to it is very expensive and also may bring some unpleasant side effects, such as oscillations in curvature, which is the Laplacian of distance function [15]. This is highly undesirable in situations where such geometric quantities play a crucial role. In [26], the reinitialization is achieved by solving a time-dependent Hamilton–Jacobi type equation to its steady state. This is the desired signed distance function. This approach works well when the level set function is initially not far away from a distance function, but may become too slow when the level set function is flat near the interface, or even worse, it may move the interface across grid points when the interface becomes steep. We propose here a variant to the original Hamilton–Jacobi equation in [26] and a new approximation to the sign function in the equation to ensure that the interface does not move across the grid points, if it moves at all. Our numerical computations verify this.

The format of our paper is as follows. In Section 2, we review the standard level set formulation and present our localization algorithm. In Section 3, we derive the PDE used to extend a quantity on an interface away from the interface. We also discuss the appropriate numerical implementation. In Section 4 we discuss the level set reinitialization in some detail. Numerical examples that are used to demonstrate specific declarations appear near the related text, while the more involved examples that use several of the techniques presented in this paper are presented in Section 5. These new examples include the vortex sheet calculation already mentioned, a double bubble minimizer using three level set functions, the merging of 100 bubbles under mean curvature flow, and the motion of bubbles under curvature dependent acceleration.

2. THE LEVEL SET FORMULATION AND ITS LOCALIZATION

We begin by reviewing the standard level set method, as first developed in [20], and set the conventions that will be followed through out the paper.

Consider a closed moving interface $\Gamma(t)$ in R^n with codimension 1. Let $\Omega(t)$ be the region (possibly multiconnected) that $\Gamma(t)$ encloses. We associate with $\Omega(t)$ an auxiliary function $\phi(x, t)$, called the level set function, which is Lipschitz continuous and satisfies the following conditions

$$\begin{cases} \phi(x,t) < 0 & \text{in } \Omega(t) \\ \phi(x,t) = 0 & \text{on } \Gamma(t) \\ \phi(x,t) > 0 & \text{in } R^n \backslash \bar{\Omega}(t), \end{cases} \tag{1}$$

where $x \in R^n$, $t \in R^+$. Note that our choice of the sign of $\phi(x, t)$ is opposite to the popular sign convention. We have found this choice more convenient. See Fig. 1.

Conversely, if we know ϕ, we may locate the interface by finding the zero level set of ϕ. That is, $\Gamma(t) = \{x : \phi(x, t) = 0\}$. So moving the interface is equivalent to updating ϕ, which can be done by solving a Hamilton–Jacobi type equation. The equation can be derived simply as follows.

Suppose $x(t)$ is a particle trajectory on the interface $\Gamma(t)$ moving with velocity $\vec{u} = \dot{x}(t)$. By definition $\phi(x(t), t) = 0$. Differentiating with respect to t, we get

$$\phi_t + \vec{u} \cdot \nabla\phi = 0. \tag{2}$$

By projecting the velocity $\vec{u}$ onto the direction $\hat{n}$ normal to the interface, Eq. (2) becomes

$$\phi_t + u_n|\nabla\phi| = 0. \tag{3}$$

Typically, the interface $\Gamma(t)$ has a prescribed velocity $\vec{u}$ or normal velocity u_n, which might be a function of space variable x, time t, the normal direction $\hat{n}$, the local mean curvature κ, or some global quantities like the Hausdorff measure $|\Gamma(t)|$ of the interface or the Lesbegue measure $|\Omega(t)|$ of $\Omega(t)$, or some external physics to which the motion of

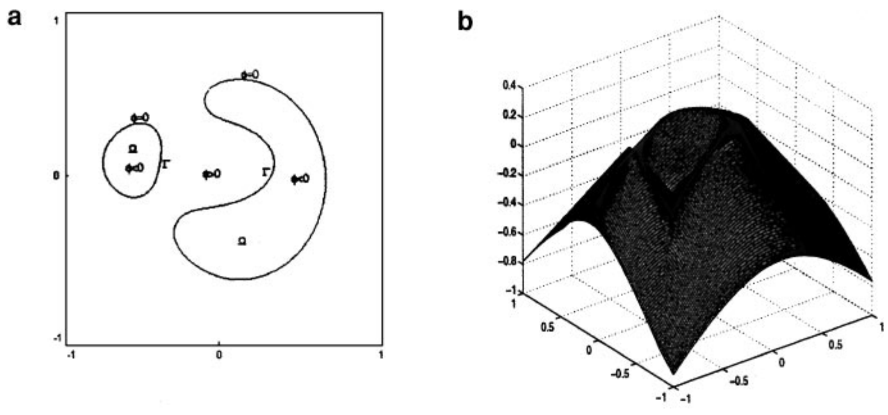

FIG. 1. (a) Zero contour of ϕ representing the front Γ. (b) Surface of $-\phi$.

$\Gamma(t)$ is coupled. One of the nice features of level set formulation is that these geometric quantities have simple representations in terms of ϕ,

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

$$\kappa = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|} \tag{5}$$

$$|\Gamma(t)| = \int \delta(\phi)|\nabla\phi|\, dx \tag{6}$$

$$|\Omega(t)| = \int H(-\phi)\, dx, \tag{7}$$

where $\delta(\phi)$ is the 1D δ-function, $H(\phi)$ is the 1D heaviside function which takes 0 for $\phi < 0$ and 1 otherwise, and κ is chosen such that a sphere has positive mean curvature equal to the reciprocal of its radius. In 2D, $|\Gamma(t)|$ is simply the arclength of $\Gamma(t)$ and $|\Omega(t)|$ the area of $\Omega(t)$, while in 3D, $|\Gamma(t)|$ is the surface area of $\Gamma(t)$ and $|\Omega(t)|$ the volume of $\Omega(t)$.

The level set method is just to extend Eq. (3), or more generally (2), to be valid throughout the space and pick up the zero level set as the front at all later time.

We point out here two issues of practical importance.

First, extending u_n off the interface is not always routine. Localization helps confine the definition of u_n to a small neighborhood of $\Gamma(t) = \{x : \phi(x, t) = 0\}$. In Section 3, we describe a fast method to extend u_n continuously off the front.

Second, for numerical accuracy, the level set function must stay well behaved in the sense that, except for isolated points,

$$0 < c \leq |\nabla\phi| \leq C$$

for some constants c and C. In fact, it is desirable for many problems that $\phi(x, t)$ be a signed distance function, i.e.,

$$|\nabla\phi| = 1.$$

Yet the solution of Eq. (2) or (3) often becomes very flat and/or steep at the front $\Gamma(t)$. So a procedure is needed to resurrect $\phi(x, t)$ so that it behaves well in a neighborhood of the front. Such a procedure is commonly called reinitialization. Again localization makes it only necessary to perform reinitialization within a narrow region around the front. We will address this issue in Section 4.

Now we introduce an algorithm to localize the level set method. For simplicity of presentation, we describe the algorithm in 2D. The idea can be extended to 3D without any change.

Let $0 < \beta < \gamma$ be two constants which are comparable to Δx whose values will be determined below. For a given open region Ω in R^n with boundary Γ^0, we define a level set function $\phi^0(x)$ satisfying (1). If necessary, apply the reinitialization step described in Section 4 to set $\phi^0(x)$ to be $d^0(x)$, the signed distance function to the front Γ^0. Around Γ^0 we define a tube with width γ by

$$T^0 = \{x : |\phi^0(x)| < \gamma\}. \tag{8}$$

Actually we only need $\phi^0(x)$ to coincide with $d^0(x)$ in T^0.

Next, let c be a cut-off function:

$$c(\phi) = \begin{cases} 1 & \text{if } |\phi| \leq \beta \\ (|\phi| - \gamma)^2(2|\phi| + \gamma - 3\beta)/(\gamma - \beta)^3 & \text{if } \beta < |\phi| \leq \gamma \\ 0 & \text{if } |\phi| > \gamma. \end{cases} \tag{9}$$

We update Γ^0 by solving the following equation,

$$\phi_t + c(\phi)\vec{u} \cdot \nabla\phi = 0 \tag{10}$$

or

$$\phi_t + c(\phi)u_n|\nabla\phi| = 0, \tag{11}$$

on T^0 with initial data $\phi^0(x)$ for one time step and get $\tilde{\phi}^1(x)$. Refer to Fig. 2. The time step is chosen such that the front moves less than one grid point. This is equivalent to the CFL condition which requires $\Delta t|u_n| < \Delta x$ uniformly on the front. If the velocity $\vec{u}$ or u_n is only given on Γ^0, an extension step is needed to extend $\vec{u}$ or u_n to T^0. The reason we introduced the cut-off function in (10) and (11) is to prevent numerical oscillations at the tube boundary.

The new location of the front is given by $\Gamma^1 = \{x : \tilde{\phi}^1(x) = 0\}$. Let $d^1(x)$ be the signed distance function to Γ^1. Of course, $d^1(x)$ is not known to us. All we have is $\tilde{\phi}^1(x)$.

To move the front further, we need the shifted tube

$$T^1 = \{x : |d^1(x)| < \gamma\}, \tag{12}$$

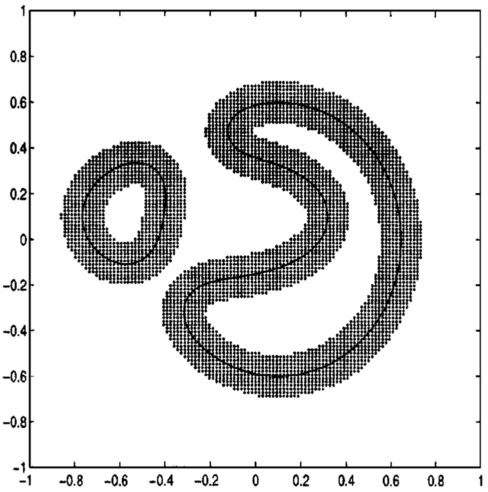

FIG. 2. Computation is only performed on the marked region around Γ.

so we must construct a new level set function $\phi^1(x)$ from $\tilde{\phi}^1(x)$ such that

$$\phi^1(x) = d^1(x) \quad \text{for } |d^1(x)| < \gamma. \tag{13}$$

This can be achieved via a step described below. First we discuss the necessity and some difficulties involved in doing this.

In the updating step above, we solve the correct equation only in a tube of radius β. In the region $\{x : \beta < |\phi^0(x)| < \gamma\}$, the motion is modified by the cut-off function. Outside the tube T^0, we do not update $\tilde{\phi}^1$ at all. A steep gradient develops at the region near the boundary of the tube T^0 in the direction of motion of the interface. Obviously, $\tilde{\phi}^1(x)$ is not a signed distance function to Γ^1. See Fig. 3b. To reset $\tilde{\phi}^1$ to be a signed distance function in a neighborhood of Γ^1 of width γ, the reinitialization must be performed on a region that contains T^1. Since the front moves less than one grid point, we can choose this region to be

$$N^0 = \{x : |\phi^0(x + y)| < \gamma, \text{for certain } |y| < \Delta x\}. \tag{14}$$

Next we perform the reinitialization step on N^0, starting with $\tilde{\phi}^1(x)$, and obtain $d^1(x)$, the signed distance function to Γ^1. Our new level set function is defined by

$$\phi^1(x) = \begin{cases} -\gamma & \text{if } d^1(x) < -\gamma \\ d^1(x) & \text{if } |d^1(x)| \le \gamma \\ \gamma & \text{if } d^1(x) > \gamma. \end{cases} \tag{15}$$

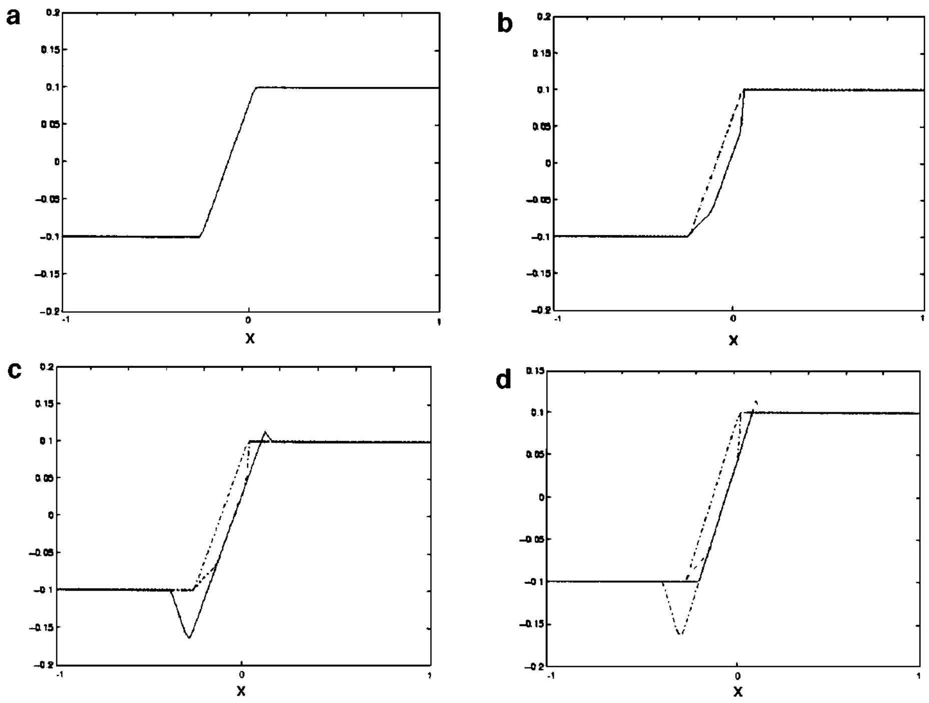

FIG. 3. The current result is plotted with a solid line, while the previous result(s) is shown with a dashed line. (a) The initial local level set profile. (b) After moving one step. Note the steep gradient developing at the right tube boundary. (c) After reinitialization. (d) The final local level set profile after cut-off.

Then we define T^1 and N^1 as follows:

$$T^1 = \{x : |d^1(x)| < \gamma\} \tag{16}$$

$$N^1 = \{x : |\phi^1(x + y)| < \gamma, \text{ for some } |y| < \Delta x\}. \tag{17}$$

Here T^1 is the region where the computation in the next time step will be performed, and N^1 is the region on which the reinitialization will be performed after the next time step.

We are now ready to move Γ^1 one more step. Iterating the above-described steps will move the front further. This process is depicted in Fig. 3.

The actual numerical values of β and γ depend on the width of the stencil of the schemes used to approximate the spatial derivatives. In our computation, we choose $\beta = 2\Delta x$, $\gamma = 4\Delta x$ for a second order ENO scheme [21] or a third order WENO scheme [13], and $\beta = 3\Delta x$, $\gamma = 6\Delta x$ for a third order ENO scheme [21] or a fifth order WENO scheme [13].

We remark here that our extension and reinitialization are done by solving a hyperbolic equation with characteristics flowing out of the tube. No boundary conditions are needed. This is enforced numerically by using an upwind scheme. See Sections 3 and 4 for details.

Having described the local level set method in a semidiscrete way, we now outline the algorithm in the fully discrete form.

MAIN ALGORITHM.

Step 0. Initialize. Given the interface Γ, construct an associated level set function ϕ. If necessary, apply the reinitialization step described in section 4 globally to set ϕ be a signed distance function to Γ.

Step 1. Compute velocity. Calculate u_n or $\vec{u}$ from ϕ or the physical problem coupled with ϕ on or near the zero level set of ϕ.

Step 2. Extend (if necessary). Extend u_n or $\vec{u}$ to tube $T = \{(x_i, y_j) : |\phi_{ij}| \le \gamma\}$ by the method described in Section 3.

Step 3. Advance. Update ϕ in tube T for one time step to get $\tilde{\phi}$ by an ODE time stepping method. See remark 2 below.

Step 4. Reinitialize. Apply the reinitialization step to $\tilde{\phi}$ on the tube $N = \{(x_i, y_j) : \min_{-1 \le \nu,\mu \le 1} |\phi_{i+\nu j+\mu}| \le \gamma\}$. Define the new ϕ by (15). Go back to step 1.

We make several remarks here.

Remark 1. The most straightforward way to implement the above algorithm is to store the values of the level set function(s) and other related quantities at the grid points in 2D arrays. In the updating step, we search through each grid point and test if it falls into the tubes by checking the value of level set function at that point. If it does, then do the computation; otherwise, nothing is done. While easy to code, the drawback of this implementation is that we have to go through all the grid points and do a comparison there. This is not really necessary since computation is performed only at the grid points within the tube T in the main time step and within tube N in the reinitialization step, and such points usually constitute only a small fraction of the total grid points. As the number of grid points or dimension increase, this overhead takes up a considerable amount of CPU time. To overcome this drawback, we introduce an extra 2D array mask of the same size as ϕ that can be used to differentiate the grid points in and out of the tubes and two equal-sized 1D arrays *index*1 and *index*2 that store the index of the grid points in the tube N and whose size is on the order of $O(N)$. These arrays are initialized by a procedure like the following

```
k = 1;
for i = 1 to Nx,
  for j = 1 to Ny
    mask(i,j) = 0;
    if |phi(i,j)| < gamma
      mask(i,j) = 2; index1(k) = i; index2(k) = j; k=k+1;
    else if |phi(p,q)| < gamma and mask(p,q) == 0 for p=i-1,i+1,
    q=j-1,j+1
      mask(p,q) = 1; index1(k) = p; index2(k) = q; k=k+1;
```

Thus, tube T corresponds to grid points where mask$(i, j) = 2$ and tube N corresponds to mask$(i, j) = 1$ or 2. This tube construction step needs $O(N^2)$ operations and is performed just once per main time step. For all the other steps, we only need $O(N)$ operations since the computation is done in a 1D loop. For example, to compute ϕ_x at the grid points within the tube T, we can use the following pseudo-code

```
for k = 1 to K
   if (mask(index1(k),index2(k)) == 2)
      phi_x(index1(k),index2(k)) = (phi(index1(k)+1,index2(k))
                                 - phi(index1(k)-1, index2(k))/
                                   (2*dx);
```

where K is the number of grid points within the tube N found in the construction step above. After the level set function is updated in the tube T, we reinitialize ϕ over the tube N in step 4 by a procedure like

```
for k=1 to K
  phi_new(index1(k),index2(k)) = phi(index1(k),index2(k) + ...
```

We then construct the new, shifted tubes by going back to our tube construction step above. Analogous statements are also true in 3D. This gives us an algorithm of approximately $O(N)$ complexity in 2D and $O(N^2)$ complexity in 3D, as will be shown in Section 5.

A truly $O(N)$ implementation of the main algorithm is possible by replacing the simple tube construction step above with a procedure that require only $O(N)$ operations. We have done this in all the numerical experiments in Section 5 with very little change in our results. We recommend the implementation presented here because of its programming simplicity.

Remark 2. The solutions to Eq. (2) or (3) are often only uniformly continuous with discontinuous derivatives, no matter how smooth the initial data are [20, 21]. Simple central differencing is not appropriate here to approximate the space derivatives. Instead, we use ENO type schemes for Hamilton–Jacobi equations as developed in [20, 21] or WENO schemes developed in [13].

Remark 3. The time stepping methods we used are the TVD Runge–Kutta schemes devised in [24]. Consider the system of ODEs,

$$\begin{cases} \frac{d\phi}{dt} = L(\phi) \\ \phi(0) = \phi_0, \end{cases} \tag{18}$$

where L is some spatial operator. The second order method at the nth step is

$$\begin{cases} \tilde{\phi}^{n+1} = \phi^n + \Delta t L(\phi^n) \\ \phi^{n+1} = \phi^n + \frac{\Delta t}{2}[L(\phi^n) + L(\tilde{\phi}^{n+1})]. \end{cases} \tag{19}$$

The third order method at the nth step is

$$\begin{cases} \tilde{\phi}^{n+1} = \phi^n + \Delta t L(\phi^n) \\ \tilde{\phi}^{n+\frac{1}{2}} = \phi^n + \frac{\Delta t}{4}[L(\phi^n) + L(\tilde{\phi}^{n+1})] \\ \phi^{n+1} = \phi^n + \frac{\Delta t}{6}\left[L(\phi^n) + 4L\left(\tilde{\phi}^{n+\frac{1}{2}}\right) + L(\tilde{\phi}^{n+1})\right]. \end{cases} \tag{20}$$

Remark 4. If the front moves very little in one time step, there is no need to reinitialize every step. We can iterate using only Step 1, 2 and 3 until reinitialization is triggered. Then we go to Step 4. We shall discuss this further in Section 4.

Remark 5. If the motion of the interface involves sensitive quantities such as derivatives of curvature, we need to choose bigger β and γ and compute these quantities in an even narrower tube of width α, where $0 < \alpha < \beta < \gamma$. In this case, we choose $\alpha = 2\Delta x$, $\beta = 4\Delta x$, $\gamma = 6\Delta x$ for a 2nd order ENO scheme [21] or a 3rd order WENO scheme [13], and $\alpha = 3\Delta x$, $\beta = 6\Delta x$, $\gamma = 9\Delta x$ for a 3rd order ENO scheme [21] or a 5th order WENO scheme [13]. Introducing an extra tube also enable us to monitor the behavior of the level set function and the movement of the front and trigger the reinitialization only when necessary. We will discuss this further in Section 4.

3. EXTENDING A QUANTITY OFF AN INTERFACE

In the level set formulation, we need the velocity $\vec{u}$ or normal velocity u_n in a neighborhood of the interface $\Gamma(t)$. In some applications, $\vec{u}$ or u_n is naturally defined globally or at least near the front. Such examples include motion with a constant speed and mean curvature flow [20]. But in some other applications, $\vec{u}$ or u_n is only given on the interface. Away from the interface, it is not defined at all. The Stefan problem [7] and Hele–Shaw flows [9] are such examples. There are also situations where we need to extend other quantities defined on the interface to a neighborhood of the interface. In [7], a PDE-based method was used to do this, and this is the approach that we adopt here. This method is straightforward and easy to implement. We note that the same method was originally proposed and analyzed in the appendix of [27], but the authors did not pursue it there.

Suppose we have a quantity q defined on the interface $\Gamma(t)$. The most natural way to extend q off $\Gamma(t)$ is to let q be a constant along the curve normal to $\Gamma(t)$. This suggests the following hyperbolic PDE,

$$q_t + S(\phi)\frac{\nabla\phi}{|\nabla\phi|} \cdot \nabla q = 0, \tag{21}$$

where $S(\phi)$ is the signature function of ϕ defined as

$$S(\phi) = \begin{cases} -1 & \text{if } \phi < 0 \\ 0 & \text{if } \phi = 0 \\ +1 & \text{if } \phi > 0. \end{cases} \tag{22}$$

The characteristics of Eq. (21) are exactly those that are normal to the level set of ϕ and pointing away from $\Gamma(t)$.

Equation (21) is a particular case of the following general Hamilton–Jacobi equation:

$$q_t + H(\nabla q, x, t) = 0, \quad x \in R^n, t > 0. \tag{23}$$

Accurate and robust numerical schemes exist to compute approximate solutions to Eq. (23). See, for example, [21, 20, 13]. Yet usually in the extension step, numerical accuracy of the method used is not an issue, as long as the method extends the quantity under consideration in a sensible way. In the following, we shall use a first order upwind scheme coupled with a forward Euler time discretization. For simplicity of presentation, we will only write down the formula for the 2D case and drop the explicit (x, y) and t dependence, which is understood to exist in the scheme in a simple fashion, i.e., by fixing (x, y) and t to be their grid values in the numerical Hamiltonian. The extension to higher dimension is straightforward.

We approximate $S(\phi)$ by $S_\delta(\phi) = \phi/\sqrt{\phi^2 + \delta^2}$, where δ is a small smoothing parameter which can be taken as Δx, and denote the nodal value of $S_\delta(\phi)$ as s_{ij}. We compute the fixed quantities $\hat{n} = (n^x, n^y) = (\phi_x/\sqrt{(\phi_x^2 + \phi_y^2)}, \phi_y/\sqrt{(\phi_x^2 + \phi_y^2)})$ by central differencing, and use $\hat{n}_{ij} = (n_{ij}^x, n_{ij}^y)$ to denote their nodal values. The above method reads

$$\begin{aligned} q_{ij}^{n+1} = q_{ij}^n - \Delta t \Bigg\{ & \left(s_{ij} n_{ij}^x\right)^+ \frac{q_{ij} - q_{i-1j}}{\Delta x} + \left(s_{ij} n_{ij}^x\right)^- \frac{q_{i+1j} - q_{ij}}{\Delta x} \\ & + \left(s_{ij} n_{ij}^y\right)^+ \frac{q_{ij} - q_{ij-1}}{\Delta y} + \left(s_{ij} n_{ij}^y\right)^- \frac{q_{ij+1} - q_{ij}}{\Delta y} \Bigg\}, \end{aligned} \tag{24}$$

where $(x)^+ = \max(x, 0)$, $(x)^- = \min(x, 0)$. The nodal values of q on the stencils that the interface cut through can be computed by interpolation [7, 9]. As pointed out in Section 2, no internal boundary condition is needed because the characteristics of the PDE (21) flow out of the interface $\Gamma = \{x : \phi(x, t) = 0\}$. This is numerically enforced naturally by the upwind scheme, since we use only the value of ϕ on the nodes biased to Γ.

We consider the following example. Suppose

$$L = \left\{ (r, \theta) : r(\theta) = 0.1\lambda + 0.7(1 - \lambda), \lambda = \frac{\theta}{6\pi}, 0 \le \theta < 6\pi \right\}$$

is a spiral on the domain $D = [-1, 1] \times [-1, 1]$ that orbits the origin three times. Let Ω be the region consisting of all points with distance to L less than $6/128$, and ϕ be the signed distance function associated with Ω. We use a 128×128 grid and define a quantity q which is equal to θ on points with distance to $\partial\Omega$ less than Δx and 0 otherwise. See Fig. 4 for the result of the extension using the above scheme. Section 5 contains a more elaborate example that uses the extension method described in this section. This is a very successful level set based calculation of an unstable vortex sheet, improving the results of [11] considerably.

4. REINITIALIZATION

It is numerically desirable to keep $\phi(x, t)$ close to a signed distance function. For general $\vec{u}$ or u_n, it is impossible to prevent $\phi(x, t)$ from deviating away from a signed distance

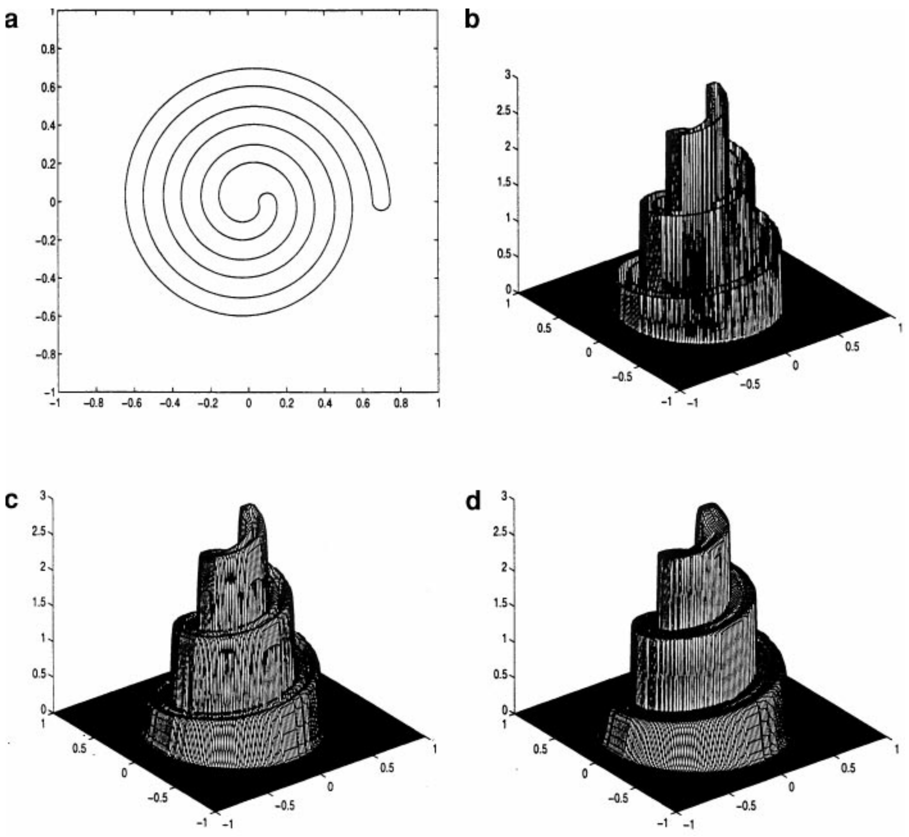

FIG. 4. Extension. (a) Zero contour of ϕ. (b) Initial q. (c) Extend for five steps. (d) Extend for ten steps.

function, except in some very special cases [27]. Typically, flat and/or steep regions will develop at the interface, making further computation and contour plotting highly inaccurate.

As an illustrative example, we consider the simple case of the shrinking of a unit circle in 2D or a sphere in 3D with its mean curvature. In the level set formulation, this is equivalent to solving the following PDE:

$$\begin{cases} \phi_t(x,t) = |\nabla\phi|\nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right) \\ \phi(x,0) = |x| - 1. \end{cases} \tag{25}$$

This equation is rotationally invariant. This suggests that we look for solution of the form $\phi(x,t) = \phi(r,t)$, where $r = |x|$. We have

$$|\nabla\phi| = \frac{\partial\phi}{\partial r}$$

$$\nabla\cdot\frac{\nabla\phi}{|\nabla\phi|} = \frac{1}{r}.$$

Then Eq. (25) becomes

$$\begin{cases} \frac{\partial\phi}{\partial t} = \frac{1}{r}\frac{\partial\phi}{\partial r} \\ \phi(r,0) = r - 1 \end{cases} \tag{26}$$

which has the well-known solution

$$\phi(r,t) = \sqrt{r^2 + 2t} - 1. \tag{27}$$

The interface is located at $r = \sqrt{1-2t}$. On the interface, $|\nabla\phi| = \sqrt{1-2t}$. Clearly, $\phi(x,t)$ becomes more and more flat at the interface and becomes totally flat just before vanishing at $t = \frac{1}{2}$.

Even in this simple case, a direct numerical implementation of Eq. (25) turns out to be problematic at the center, where $|\nabla\phi|$ is always zero. A spike appears at the center after one step. See Fig. 5b. As the interface moves near to the center, the computation becomes highly inaccurate. For simple cases such as this one, we know in advance where the problem will arise, and we can kill the spike by enforcing explicitly an internal boundary condition at the center. For example,

$$\phi_{0,0} = \frac{1}{3}(\phi_{-1,0} + \phi_{1,0} + \phi_{0,-1} + \phi_{0,1}) - \frac{1}{12}(\phi_{-2,0} + \phi_{2,0} + \phi_{0,-2} + \phi_{0,2}). \tag{28}$$

But in general, we do not know in advance where the problem will arise, and such a condition is hard to enforce. A reinitialization step will effectively eliminate such problems without the explicit knowledge of their locations. This is a very important effect of level set reinitialization. In Fig. 5, we plot the errors in computing the radius of the shrinking circle with and without the reinitialization step, and in each case, with and without enforcing the internal boundary condition (28).

Generally, a procedure is needed to reset the level set function $\phi(x,t)$ to be a signed distance function to the front $\Gamma(t)$. One may ask if it is legal to do so. Theoretically, this is justified in [8] and [4], where the authors showed that the interface $\Gamma(t) = \{x : \phi(x,t) = 0\}$ does not depend on the particular choice of the initial data $\phi(x,0)$, as long as its zero level set coincides with $\Gamma(0)$. Reinitialization is simply the process of replacing $\phi(x,t)$ by another function $\tilde{\phi}(x,t)$ that has the same zero contour as $\phi(x,t)$ but behaves better and then taking this new function $\tilde{\phi}(x,t)$ as the initial data to use until the next round of reinitialization.

A straightforward way to reinitialize is to find the location of the front with some interpolation technique and then compute the signed distance function to this front, as is done in [15]. This brute force approach has the advantage that it does not move the interface up to the numerical accuracy of the interpolation scheme. The disadvantage is its high cost and the likelihood of introducing some spurious irregularity into the data, making differentiated quantities such as curvature behave very badly. Some kind of smoothing procedure is usually needed to be coupled with this approach. A more elegant way is presented in [26], where the following Hamilton–Jacobi type equation,

$$\begin{cases} d_\tau + S(d_0)(|\nabla d| - 1) = 0 \\ d(x,0) = d_0(x) = \phi(x,t), \end{cases} \tag{29}$$

is solved to steady state, which is the desired signed distance function. Properly implemented, this method converges quickly in a neighborhood of the front. The reason is quite simple. On the PDE level, d propagates with speed 1 along the characteristics that are normal to the interface and converges in time ϵ to a signed distance function in a neighborhood of $\Gamma(t)$ of width ϵ. We note that the complexity of this kind of reinitialization is proportional to the number of grid points in the ϵ-neighborhood of $\Gamma(t)$, which is of order N when the

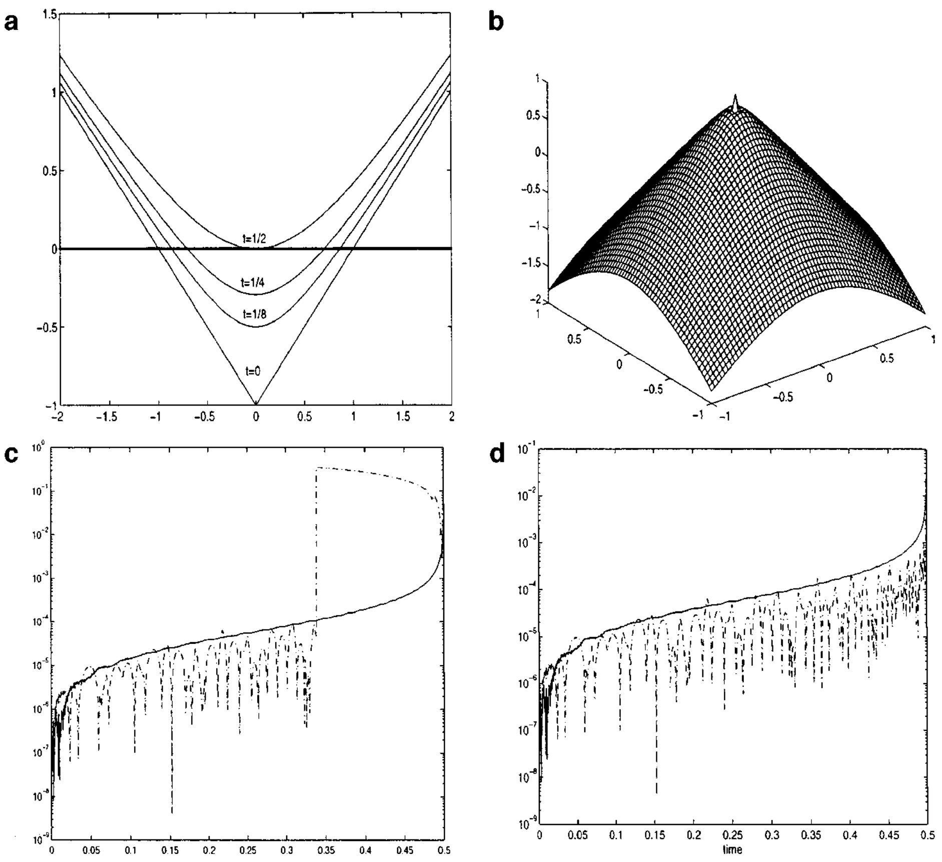

FIG. 5. (a) The mid-slice of analytical level set function of a shrinking circle. (b) A spike appears at the center with a direct implementation. (c) The plot of errors in radius computed with reinitialization (solid line) and without reinitialization and without enforcing the internal boundary condition (28) (dashed line). Note that at about $t = 0.35$, a spurious circle appears at the center. In this case, the radius is taken as the average of the two. (d) The plot of errors in radius computed with reinitialization (solid line) and without reinitialization and with enforcing the internal boundary condition (28) (dashed line).

area of the ϵ-neighborhood occupies a small fraction of the total area. In [26], the authors approximated $S(d)$ by

$$S_\epsilon(d) = \frac{d}{\sqrt{d^2 + \epsilon^2}} \tag{30}$$

with $\epsilon = \Delta x$ and used a second order ENO scheme [21] to approximate the space derivatives. They reported good results.

Their method generally works well when $d_0(x)$ is neither too flat nor too steep near the interface. When d becomes too flat, the quantity $S_\epsilon(d)$ above is small and the propagating speed will be small. More steps will be needed to reset d to be a signed distance function in a neighborhood of $\Gamma(t)$ of fixed width. See Fig. 6a. When d is very steep near the interface, this approach might change the sign of d, thus moving the interface across grid points. See Fig. 7a.

Now we analyze how these issues arise and try to find ways to solve them. To keep things simple and the idea clear, we take the 1D case and use the first order upwind scheme in space

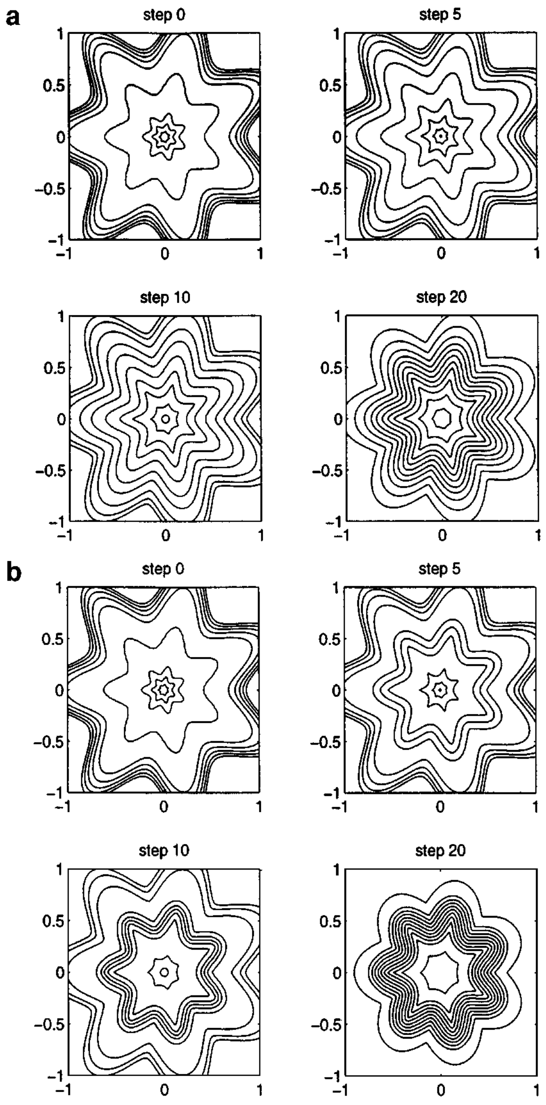

FIG. 6. Convergence rate comparison for a flat ϕ with different approximation to $S(\phi)$. The initial level set in polar coordinates is $\phi(r,\theta) = (r - 0.5 + 0.1r\,\sin(7\theta))^3$. Grid 128×128, $\Delta t = .5\Delta x$. RK3-ENO3. Contour taken over $[-10\Delta x : 2\Delta x : 10\Delta x]$. (a) Approximate $S(\phi)$ by $\phi/\sqrt{\phi^2 + \Delta x^2}$. (b) Approximate $S(\phi)$ by $\phi/\sqrt{\phi^2 + (|\nabla\phi|\Delta x)^2}$.

and the forward Euler method in time discretization. Suppose the interface cuts through the interval $[x_i, x_{i+1}]$ and $d_i < 0 < d_{i+1}$. After one time step, we have

$$d_i^1 = d_i + s_i \Delta\tau \left(1 - \frac{d_{i+1} - d_i}{\Delta x}\right) \tag{31}$$

$$d_{i+1}^1 = d_{i+1} + s_{i+1} \Delta\tau \left(1 - \frac{d_{i+1} - d_i}{\Delta x}\right), \tag{32}$$

where $-1 \le s_i < 0$ and $0 < s_{i+1} \le 1$ are some approximations to $S(d_i)$ and $S(d_{i+1})$,

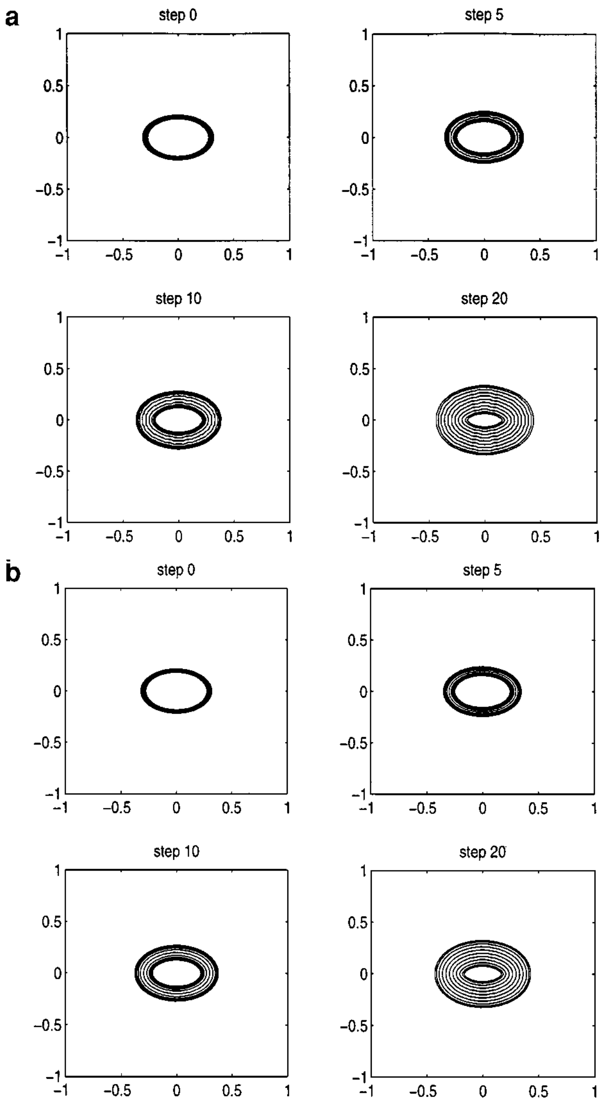

FIG. 7. For a steep ϕ, the interface is moved substantially with the approximation to $S(\phi)$ as (30), while the new one given by (36) alleviates the problem quite a bit. The initial level $\phi(x, y) = x^2/.3^2 + y^2/.2^2 - 1$, and the grid is 128×128, $\Delta t = .5\Delta x$. RK3-ENO3 is used. The contour is taken over $[-10\Delta x : 2\Delta x : 10\Delta x]$. (a) Approximate $S(\phi)$ by $\phi/\sqrt{\phi^2 + \Delta x^2}$. (b) Approximate $S(\phi)$ by $\phi/\sqrt{\phi^2 + (|\nabla\phi|\Delta x)^2}$.

respectively. The above scheme is monotone, which means d_i^1 is nondecreasing in d_i and $d_{i\pm1}$, when the CFL condition $\Delta\tau \le \Delta x$ is satisfied. When $(d_{i+1} - d_i)/\Delta x \le 1$, we see that $d_i^1 < 0 < d_{i+1}^1$, regardless of the choice of $\Delta\tau$, s_i and s_{i+1}. But when $(d_{i+1} - d_i)/\Delta x > 1$, if we approximated $S(d)$ by $S_\epsilon(d)$ with a uniform ϵ as in [26], there is no guarantee that d does not change sign. For example, if we take $\Delta\tau = c\Delta x$, $d_i = -m\Delta x$, and $d_{i+1} = n\Delta x$, we then have

$$d_i^1 = m\Delta x\left[-1 + \frac{c(m+n-1)}{\sqrt{m^2+1}}\right] \tag{33}$$

$$d_{i+1}^1 = n\Delta x\left[1 - \frac{c(m+n-1)}{\sqrt{n^2+1}}\right]. \tag{34}$$

One can easily pick m and n such that d_i^1 or d_{i+1}^1 changes sign, no matter how small c is.

To ensure that d does not change sign, we have to require

$$\frac{\Delta\tau}{\Delta x} \le \frac{|d_k|}{\Delta x|s_k|\left|1 - \frac{d_{i+1}-d_i}{\Delta x}\right|} \quad \text{for } k = i, i+1. \tag{35}$$

In order that the above inequality does not pose a more serious restriction on time step $\Delta\tau$ than the CFL condition does, we choose the approximation to $S(d)$ as

$$s_k = \frac{d_k}{\sqrt{d_k^2 + \left(1 - \left|\frac{d_{i+1}-d_i}{\Delta x}\right|\right)^2 \Delta x^2}}.$$

That is, our approximation to the sign function $S(d)$ depends on the local slope of d.

The above analysis suggests that, in the multidimensional case as well, if we use the forward Euler method in time discretization and approximate $\nabla d(x)$ with some discrete operator Dd (see below), we can choose

$$s = \frac{d}{\sqrt{d^2 + (1 - |Dd|)^2 \Delta x^2}}$$

to ensure that the interface is confined to one cell in the reinitialization step. The higher order TVD Runge–Kutta methods given by (19) and (20) are convex combinations of the first order Euler method. We need only to approximate $S(d)$ with the d in the previous step and $\nabla d(x)$ with updated d in each substep. It is easy to see that the original CFL condition is still valid with this choice of approximation to $S(d)$.

To put in some smoothing effects, we will use the following approximation in our computation:

$$s = \frac{d}{\sqrt{d^2 + |Dd|^2 \Delta x^2}}. \tag{36}$$

This choice of $S(d)$ inherits the above properties. See Figs. 6b and 7b.

The choice of approximation to $S(d)$ by (36) solves the problem of the changing of sign of ϕ (thus moving the interface across the cell boundary) in the reinitialization step when ϕ is steep and speeds up the convergence when ϕ is flat at the interface.

Based on the above analysis, we propose to solve the following Hamilton–Jacobi equation,

$$\begin{cases} d_\tau + S(d)(|\nabla d| - 1) = 0 \\ d(x, 0) = d_0(x) = \phi(x, t), \end{cases} \tag{37}$$

to steady state, with $S(d)$ approximated by (36).

There are two canonical monotone upwind schemes which have been frequently used. In [20] a variant of the Engquist–Osher scheme developed for scalar conservation laws was suggested. In our case it becomes

$$d_{ij}^{n+1} = d_{ij}^n - \frac{\Delta\tau}{\Delta x} s_{ij}^+ \left(\sqrt{(a^+)^2 + (b^-)^2 + (c^+)^2 + (d^-)^2} - 1\right) \\ - \frac{\Delta\tau}{\Delta x} s_{ij}^- \left(\sqrt{(a^-)^2 + (b^+)^2 + (c^-)^2 + (d^+)^2} - 1\right), \tag{38}$$

where s_{ij} is the approximation to $S(d_{ij}^n)$ with (36); a, b, c, d are defined by

$$a = D_x^- d_{ij}^n, \quad b = D_x^+ d_{ij}^n \\ c = D_y^- d_{ij}^n, \quad d = D_y^+ d_{ij}^n.$$

Scheme (38) is easily seen to be monotone; i.e., the right side of (3.3) is a nondecreasing function of all the d_{ij}^n, if

$$\frac{\Delta\tau}{\Delta x} |s_{ij}| \leq \frac{1}{2}. \tag{39}$$

This is also upwind, which means that away from sonic points (for those with either $S(d)d_{x_1} = 0$ or $S(d)d_{x_2} = 0$) the scheme has a domain of dependence which is in the same direction as that of the characteristics.

The second and most canonical example is Godunov's scheme, described abstractly in [21]; see also [3], whose realization for this problem is

$$d_{ij}^{n+1} = d_{ij}^n - \frac{\Delta\tau}{\Delta x} s_{ij}^+ \left(\sqrt{\max[(a^+)^2, (b^-)^2] + \min[(c^+)^2, (d^-)^2]} - 1\right) \\ - \frac{\Delta\tau}{\Delta x} s_{ij}^- \left(\sqrt{\max[(a^-)^2, (b^+)^2] + \min[(c^-)^2, (d^+)^2]} - 1\right), \tag{40}$$

which is also monotone and upwind with the same time step restriction and is slightly less dissipative near sonic points.

If we start each iteration of our main time step with a distance function, the reinitialization typically takes only one or two iterations within the tube.

For simplicity, we described the above two schemes using forward Euler time discretization. In actual computation, we can use a TVD-type Runge–Kutta scheme. See Eqs. (19) and (20) in Section 2. The one-sided differences $D_x^{\pm} d_{ij}$, $D_x^{\pm} d_{ij}$ are computed with ENO [21] or WENO [13] schemes for Hamilton–Jacobi equations.

A very important practical question is when to start our reinitialization step. Shall we do it every time step? There is no simple answer that applies generally. Reinitialization every time step is necessary when the interface undergoes a rapid change and ϕ deviates dramatically away from the signed distance function. Otherwise this is excessive. What we need is some way of monitoring this process to trigger the reinitialization automatically. In our computation, we choose to monitor two factors: whether the average slope of ϕ deviates from 1 substantially and whether the interface moves near the boundary of the tube β (if three tubes are used). (This is another reason to have a middle tube.) If one of these two signals is activated, we start the reinitialization procedure immediately. However, this is not very reliable. If the reinitialization is not triggered by the above monitoring procedure after a fixed number of time steps, it still needs to be executed. Since the monitoring procedure is

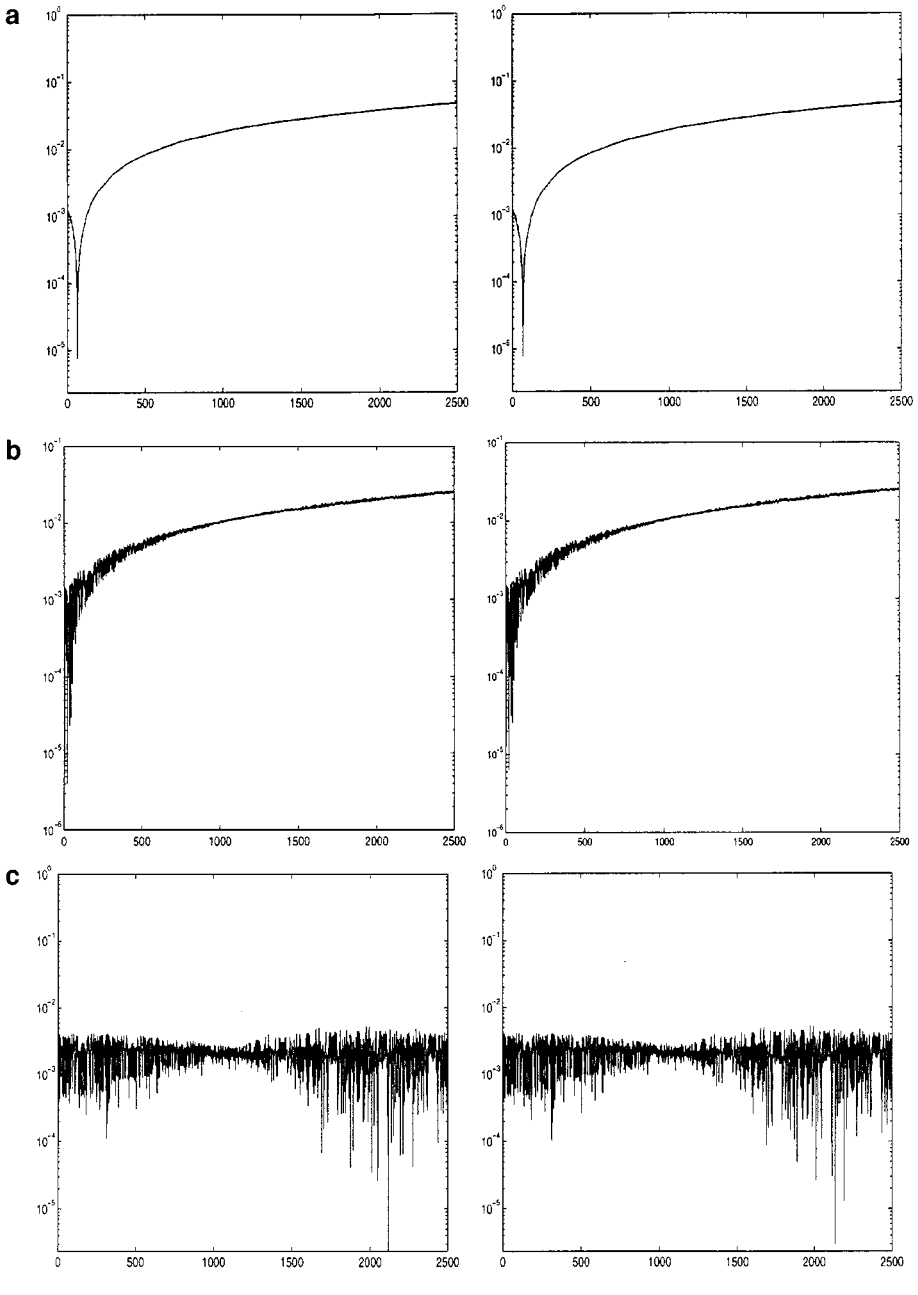

FIG. 8. Data on the left are computed with global method, while those on the right are computed with local method. Grid 128 × 128, RK3-ENO3. (a) Percentage of area loss. (b) Percentage of arclength loss. (c) Deviation from 1 of defect ratio.

only done within a narrow tube with central differencing, the cost of doing this is minimal. We have found this approach works well in our computation. See Section 5 for some numerical examples.

To summarize, our reinitialization step consists of several steps of the PDE solver for (29), with the sign function approximated as discussed above.

5. NUMERICAL RESULTS

In this section, we present some numerical examples that use one or more of the techniques presented in the previous sections and compare the results and performances with that obtained by global level set method. All the computations below are performed on a Sun Sparc-10 workstation.

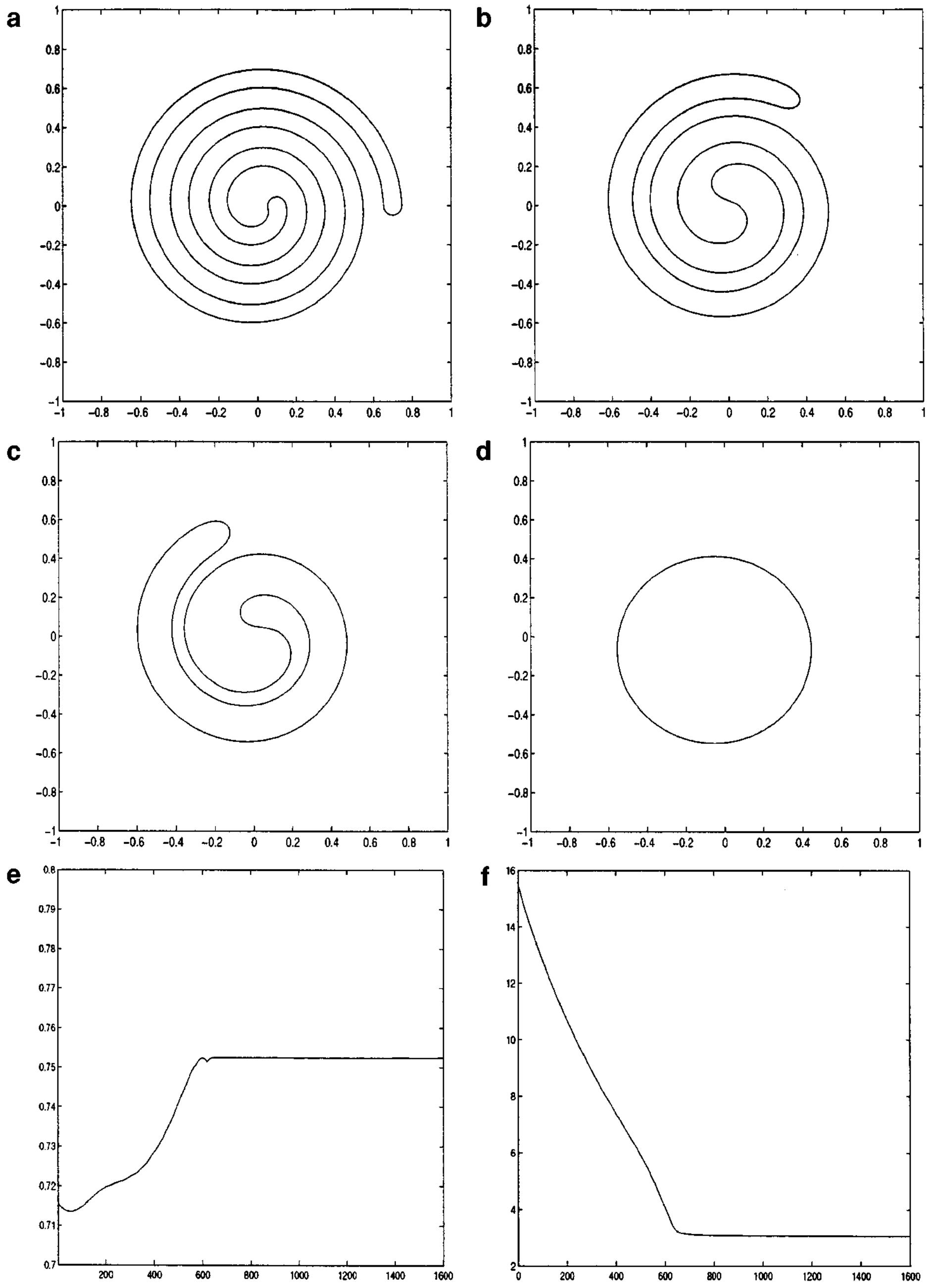

FIG. 9. Volume conserved mean curvature flow. Grid 128 × 128, RK3-WENO5. (a) Initial front. (b) After 200 steps. (c) After 400 steps. (d) After 1600 steps. (e) Area. (f) Arclength.

TABLE 1
Comparison of Global and Local Level Set Methods for Example 1

Mesh size	Global	$O(N^2)$	Local	$O(N)$	Global/Local
32×32	0.03269		0.01986		1.646
64×64	0.1416	4.332	0.04221	2.1225	4.6451
128×128	0.9366	6.614	0.09511	2.253	7.3221
256×256	4.35	4.64	0.2246	2.363	9.2933

EXAMPLE 1. We compare the performances between our local level set method and global level set method on the following problem. We rotate a circle around the origin twice. The circle is initially centered at (0.5, 0), with radius 0.25. The computation is performed on the domain $D = [-1, 1] \times [-1, 1]$, with RK3-ENO3. A reinitialization is performed for three steps in every main time step, also using RK3-ENO3. In the level set formulation, this is equivalent to solve the following equation:

$$\phi_t - y\phi_x + x\phi_y = 0. \tag{41}$$

Table 1 tabulates the CPU time (in seconds) used in each step for the global method and the local method described in Section 2 and their ratios. This table clearly shows the $O(N)$ behavior of our local level set method.

The quality of the numerical solutions is measured by three factors, namely, the loss of area A, the loss of arclength L (in percentages), and the isoperimetric defect ratio $R = L^2/(4\pi A)$. The defect ratio R measures how far the shapes deviated from a circle. For a circle, $R = 1$, and $R > 1$ for any other shape. Because of numerical errors in the algorithms we used to compute the area and arclength, the defect ratios computed are all smaller than 1, but are pretty close to 1. See Table 2. In Fig. 8, we plot the loss of area and arclength (in percentage) and the deviation of defect ratio from 1 in the results computed by local and global method for a grid of size 128×128.

EXAMPLE 2. We test our local level set algorithm on volume-preserving mean curvature flow,

$$\phi_t = (\kappa - \bar{\kappa})|\nabla\phi|, \tag{42}$$

TABLE 2
Area, Arclength Loss, and Defect Ratio for Global Algorithm and Local (Fast) Algorithm in Example 1

	Global			Local		
Mesh size	Area loss	Length loss	Defect ratio	Area loss	Length loss	Defect ratio
32×32	57.3	37.4	0.9174	57.3	37.4	0.9174
64×64	13.6	7.45	0.9908	13.5	7.42	0.9910
128×128	4.88	2.64	0.9964	4.87	2.64	0.9964
256×256	0.48	0.36	0.9977	1.17	0.69	0.9980

where κ is the mean curvature and $\bar{\kappa}$ is the average mean curvature on the front which can be computed by

$$\bar{\kappa} = \frac{\int \kappa \delta(\phi)|\nabla\phi|\,dx}{\int \delta(\phi)|\nabla\phi|\,dx}. \tag{43}$$

To test the performance of our local level set method, two different initial conditions are used in our test. In the first case, the initial front is a spiral as we have used in our extension example in Section 2. In Fig. 9, we plot the fronts at different steps and the area (which should stay constant) and arclength against the computational steps. In the second case, the initial fronts are 100 bubbles spread over $[-1, 1] \times [-1, 1]$, with centers and radii produced by a random number generator. We computed the total area, the total arclength, and the approximate number of bubbles by

$$N = \frac{1}{2\pi}\int \kappa\delta(\phi)|\nabla\phi|\,dx. \tag{44}$$

The results are shown in Fig. 10. With a 128×128 grids, we lose about 4% of the area after 1600 steps. The same computations are also performed with global level set method. No distinguishable differences are noticed between the results of the global and our local methods.

For complexity analysis, we applied our local level set method to this problem with various initial shapes in 2D, namely a circle, an ellipse, a square, a star (see Example 4 below) and the spiral mentioned above. We also simulated the 3D problem of a dumbbell collapsing under its mean curvature for which pinch-off occurs. Table 3 tabulates the CPU time (in seconds) per time step for a different number of grid points and different initial shapes, which clearly demonstrated the $O(N)$ complexity of our local method in 2D and $O(N^2)$ complexity in 3D.

EXAMPLE 3. In this example, we test our extension and reinitialization algorithms discussed in Sections 3 and 4 on the vortex sheet problem. Details of this problem in the level set formulation are contained in [11]. We briefly summarize it here.

The 2D incompressible Euler equation in vorticity-streamline formulation is

$$\omega_t + u\omega_x + v\omega_y = 0, \tag{45}$$

$$\mathrm{curl}(u, v) = \omega, \tag{46}$$

$$\mathrm{div}(u, v) = 0, \tag{47}$$

where ω is vorticity and u and v are the two components of the fluid velocity.

TABLE 3
Complexity Analysis of the Local Level Set Method for Example 2

Mesh	Circle		Ellipse		Square		Star		Spiral		Dumbbell	
32	1.7e-2		2.4e-2		3.6e-2		4.3e-2		5.8e-2		0.4	
64	4.0e-2	2.3	4.9e-2	2.1	7.9e-2	2.2	1.2e-1	2.7	1.5e-1	2.5	2.4	5.9
128	8.3e-2	2.1	1.1e-1	2.3	1.8e-1	2.3	2.8e-1	2.4	4.4e-1	2.9	10.7	4.5
256	2.3e-1	2.8	3.1e-1	2.7	4.8e-1	2.7	7.9e-1	2.8	1.1	2.6		

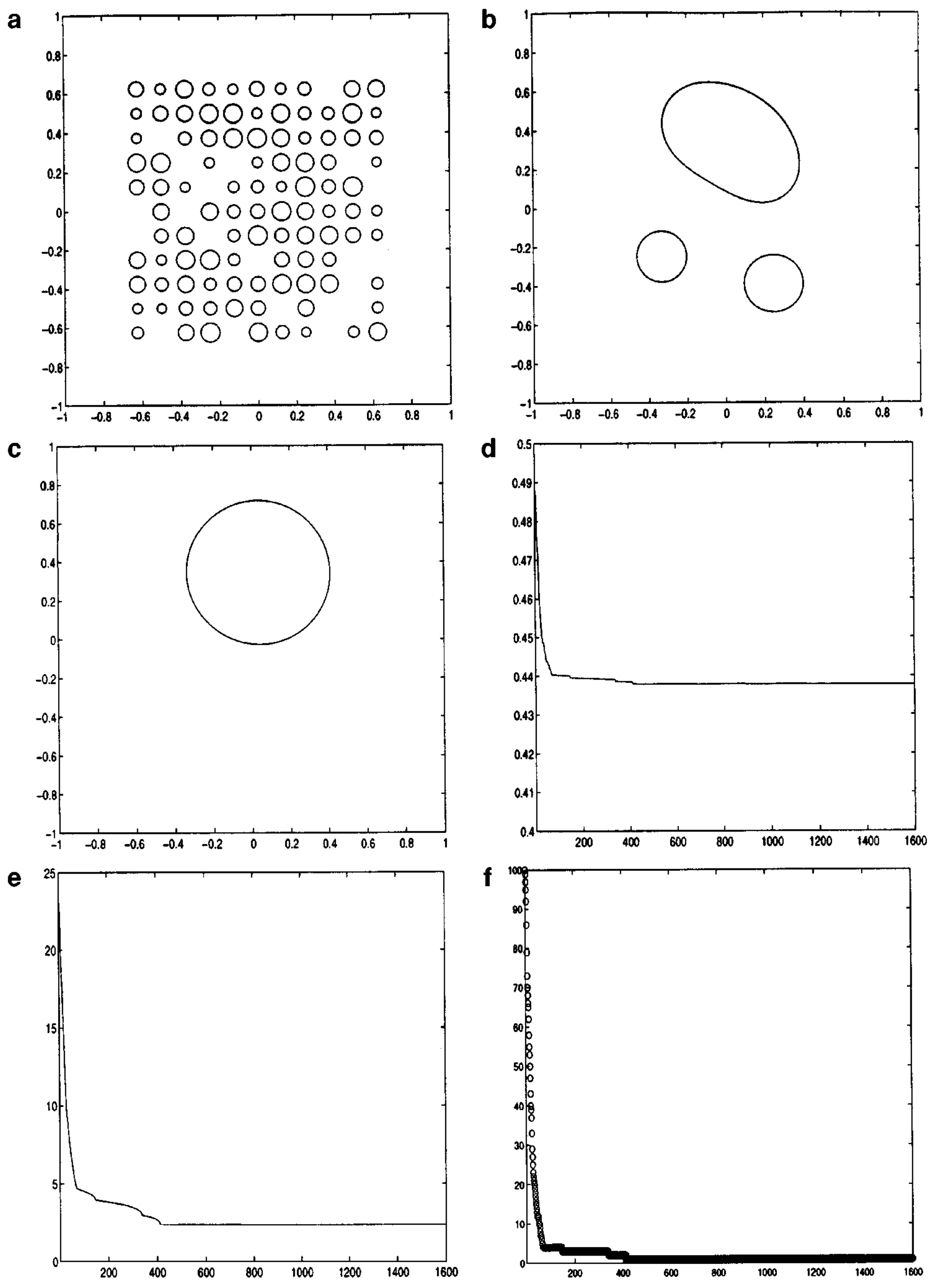

FIG. 10. Volume conserved mean curvature flow. Grid 128 × 128, RK3-WENO5. (a) Initial fronts. (b) After 200 steps. (c) After 1600 steps. (d) Area. The exact area is 0.4613. (e) Arclength. The initial arclength is 23.598. (f) Number of bubbles. The initial number of bubbles is 100.

Suppose ω is only concentrated on a thin curve called a vortex sheet, and let

$$\omega = \delta(\phi)\eta \tag{48}$$

where ϕ is the level set function associated with the vortex sheet and η is a parameter that is related to the strength of vorticity.

Inserting Eq. (48) into (45), and equating the coefficients of $\delta(\phi)$ and $\delta'(\phi)$ terms, we get

$$\phi_t + u\phi_x + v\phi_y = 0, \tag{49}$$

$$\eta_t + u\eta_x + v\eta_y = 0. \tag{50}$$

Introduce the streamline function ψ, such that

$$u = \psi_y, \quad v = -\psi_x. \tag{51}$$

From Eq. (46) we get the following equation for ψ:

$$\psi_{xx} + \psi_{yy} = -\delta(\phi)\eta. \tag{52}$$

Note that η itself does not make physical sense, since $\delta(\phi)$ is not invariant in the sense that ϕ can be replaced by $h(\phi)$ for a function h that satisfies $h' > 0$ and $h(0) = 0$ without changing its value. But $\delta(\phi)|\nabla\phi|$ is invariant. Hence $\eta/|\nabla\phi|$ is the strength of the vorticity. This gives us the following transformation of η if ϕ is changed to $\tilde{\phi}$ that corresponds to the same interface:

$$\tilde{\eta} = \frac{|\nabla\tilde{\phi}|}{|\nabla\phi|}\eta. \tag{53}$$

This relation is used in the reinitialization of ϕ. Another issue that concerns η is that η only makes sense on the vortex sheet, i.e., the zero level set of ϕ. But to update η, we need its value in a neighborhood of the interface. This problem can be solved by the extension procedure we described in Section 3.

Our computation is performed on a rectangular $D = [-1, 1] \times [-1, 1]$. The boundary conditions for ψ, ϕ, and η are:

1. ψ is periodic in x, and $\psi(x, 1) = \psi(x, -1) = 0$.
2. ϕ is periodic in x, and $\phi(x, +1, t) = \phi(x, -1, t) + 2$.
3. η is periodic in both x and y.

The initial conditions for ϕ and η are

$$\phi(x, y, 0) = y + 0.05\sin(\pi x),$$
$$\eta(x, y, 0) = 1.$$

The choice for the approximation to delta function is

$$\delta(\phi) = \begin{cases} \frac{1}{2\epsilon}(1 + \cos\left(\frac{\pi\phi}{\epsilon}\right) & \text{if } \phi < \epsilon \\ 0 & \text{otherwise,} \end{cases} \tag{54}$$

where ϵ is a small parameter that is proportional to mesh size Δx. In our computation, we use a 128×128 grid, with $\epsilon = 12\Delta x$ as in [11] and RK3-WENO5 in main time step and reinitialization. The Laplace equation (52) is solved by the FISHPACK routine *hwscrt.f*. Since the cost of updating ϕ and η is only a small fraction of the cost of solving the Laplace equation, a global level set method is used here. See Figs. 11 and 12 for the remarkable results. In particular, we obtain numerous turns without spurious wiggles. With only the

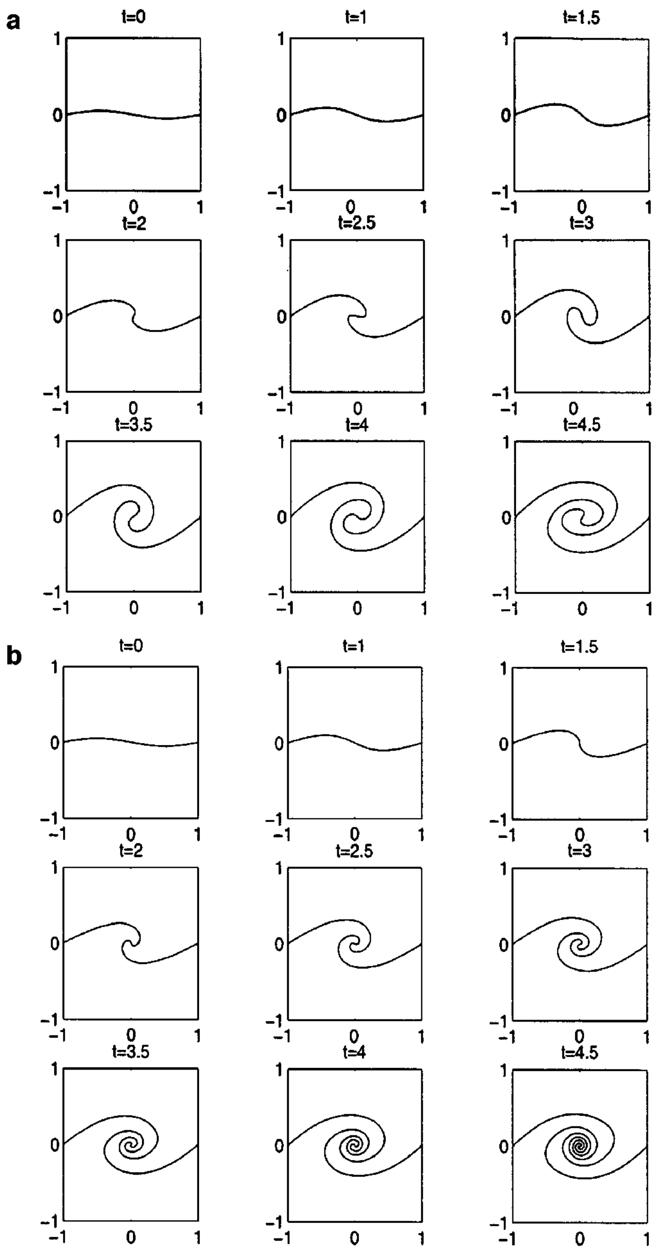

FIG. 11. Vortex sheet problem. $\Delta t = 0.1\Delta x$. (a) Without reinitialization and extension. (b) With reinitialization and extension every 10 time steps.

reinitialization but without the extension step, the computation blows up quickly. The reason for this is that in the rescaling of η using (53), η is not well defined at the grid points where the gradient of ϕ vanishes. The data near the front is polluted by the data over these grid points, especially when the front rolls up, making further computation hard to proceed. An extension step will assign meaningful values to these grid points.

The next two examples showed a comparable savings of one order of magnitude in local and global level set calculations.

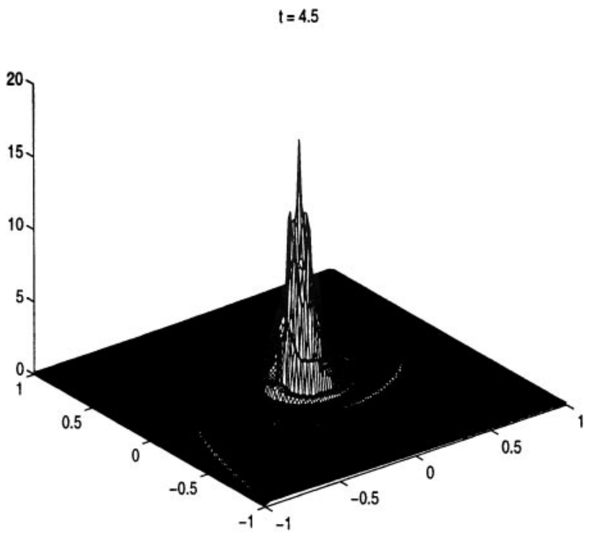

FIG. 12. Vortex sheet problem. The vorticity at time $t = 4.5$.

EXAMPLE 4. We test our local level set algorithm on the motion with curvature dependent acceleration. The general equation governing the motion is

$$\mu \frac{du}{dt} = -[p]n - \sigma\kappa n + f - \mu u \left(\frac{1}{|dA|} \frac{d|dA|}{dt} + \frac{1}{\mu} \frac{d\mu}{dt} \right),$$

where

$$\frac{1}{|dA|} d \frac{|dA|}{dt} = \nabla \cdot u - n \cdot Du \cdot n$$

and

$$[p] = \frac{\int_\Omega \left((\nabla \cdot u)(u \cdot n) - (u \cdot \nabla u) \cdot n - \frac{1}{\mu}(\sigma\kappa - f \cdot n - F \cdot n) \right) \delta(\phi) |\nabla\phi| \, d\Omega}{\int_\Omega \frac{1}{\mu} \delta(\phi) |\nabla\phi| \, d\Omega},$$

where

$$F = -\mu u \left(\frac{1}{|dA|} \frac{d(|dA|)}{dt} + \frac{1}{\mu} \frac{\mu}{dt} \right)$$

and f can be determined depending on the property of the motion. For the details, see [14]. See Fig. 13 for the numerical results.

EXAMPLE 5. In this example, we apply the local level set method to the 3D double bubble minimizer calculation. Initially, a dumbbell-shaped bubble is surrounded by a doughnut-shaped bubble, and the bubbles are in the air. The total surface energy is the integration of the surface tension along the bubble surface. Surface tension can be prescribed to be different at interfaces between bubble and bubble and bubble and air. We are interested in finding the shape which minimizes the total surface energy while keeping the volume of

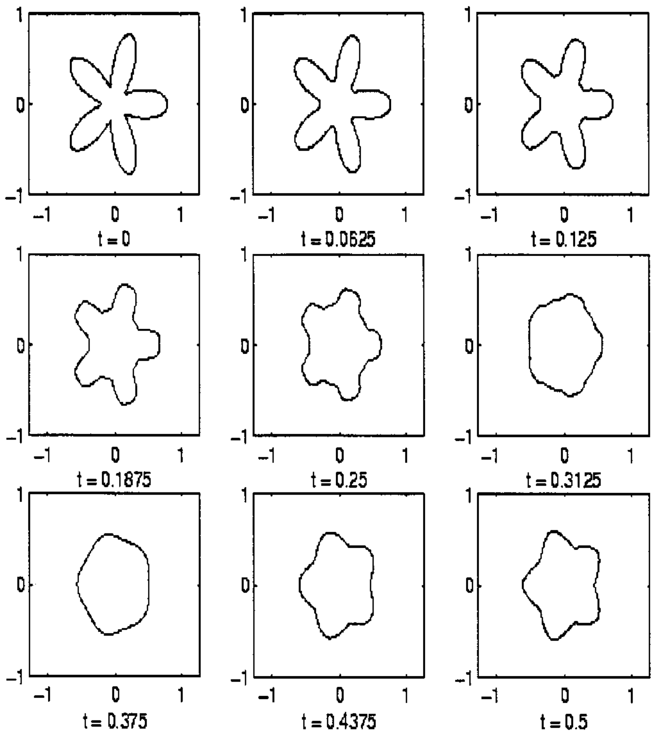

FIG. 13. Oscillating starfish, curvature dependent acceleration (volume preserving). Initial shape is given in polar coordinates by $r = 0.5 + 0.3 \sin 5\theta$.

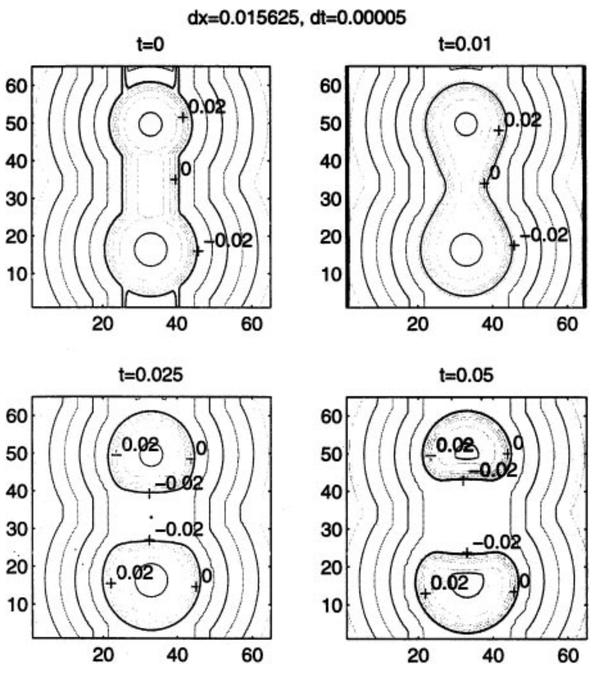

FIG. 14. Middle slice of the 3D double bubble minimizer problem. The surface tension of the dumbbell is smaller than that of the doughnut.

each bubble fixed. Three level sets are used, namely, ϕ_0 for the air, ϕ_1 for the dumbbell, and ϕ_2 for the doughnut. We try to minimize

$$E = \sum_{i=0}^{2} \int \gamma_i \delta(\phi_i(x)) |\nabla \phi_i(x)| \, dx \tag{55}$$

subject to nonoverlap and volume conservation constraints. Here the γ_i's are surface tension. Details can be found in [28].

In our calculation, the surface tension of the doughnut-shaped bubble is taken to be bigger than that of dumbbell-shaped bubble. Figure 14 is the contour plot of the dumbbell-shaped bubble taken on a cross section. Note that we did not cut the level set function to be constant outside the tube around the interface. We see that the dumbbell-shaped bubble is clipped in two. We also see that only in a neighborhood of the interface the level set function is very close to a distance function. Contours that are far away from the interface are not even touched.

REFERENCES

1. D. Adalsteinsson and J. A. Sethian, A fast level set method for propagating interfaces, *J. Comput. Phys.* **118**, 269 (1995).
2. D. Adalsteinsson and J. A. Sethian, The fast construction of extension velocities in level set methods, *J. Comput. Phys.* **148**, 2 (1999).
3. M. Bardi and S. Osher, The nonconvex multi-dimensional Riemann problem for Hamilton–Jacobi equations, *J. Numer. Anal.* **22**, 344 (1991).
4. Y. G. Chen, Y. Giga, and S. Goto, Uniqueness and existence of viscosity solutions of generalized mean curvature flow equations, *J. Diff. Geom.* **33**, 749 (1991).
5. Y. C. Chang, T. Y. Hou, B. Merriman, and S. Osher, A level set formulation of Eulerian interface capturing methods for incompressible fluid flows, *J. Comput. Phys.* **124**, 449 (1996).
6. D. L. Chopp and J. A. Sethian, Flow under curvature: Singularity formulation, minimal surfaces, and geodesics, *J. Exper. Math.* **2**, 235 (1993).
7. S. Chen, B. Merriman, S. Osher, and P. Smereka, A simple level set method for solving Stefan problems, *J. Comput. Phys.* **135**, 8 (1995).
8. L. C. Evans and J. Spruck, Motion of level set via mean curvature I, *J. Diff. Geom.* **33**, 635 (1991).
9. T. Hou, Z. Li, S. Osher, and H. K. Zhao, A hybrid method for moving interface problems with applications to the Hele-Shaw flows, *J. Comput. Phys.* **134**, 1997.
10. E. Harabetian and S. Osher, Regularization of ill-posed problems via the level set approach, *SIAP* **58**, 1689 (1998). [UCLA CAM Report 95-41.]
11. E. Harabetian, S. Osher, and C. W. Shu, An Eulerian approach for vortex motion using a level set regularization procedure, *J. Comput. Phys.* **127**, 15 (1996).
12. J. Helmsen, E. G. Puckett, P. Colella, and M. Dorr, Two new methods for simulating photolithography development in 3D, in *SPIE Microlithography IX, 1996*, p. 253.
13. G. S. Jiang and D. Peng, *WENO Schemes for Hamilton-Jacobi Equations*, UCLA CAM report 97-29, 1997. To appear in *SIAM J. Sci. Comput.*
14. M. Kang, B. Merriman, S. Osher, and P. Smereka, *A Level Set Approach for the Motion of Soap Bubbles with Curvature Dependent Velocity or Acceleration*, UCLA CAM report 96-19.
15. B. Merriman, J. Bence, and S. Osher, Motion of multiple junctions: A level set approach, *J. Comput. Phys.* **112**, 334 (1994).
16. B. Merriman, R. Caflisch, and S. Osher, Level set methods with application to island dynamics, in *Proceedings of Meeting on Free Boundary Problems: Theory and Application, Crete, 1997*, edited by I. Athasopoulos, G. Markrikis, and J. F. Rodrigues (CRC Press, Boca Raton, FL, 1999). [UCLA CAM Report 98-10.]

17. W. Mulder, S. Osher, and J. A. Sethian, Computing interface motion in compressible gas dynamics, *J. Comput. Phys.* **100**, 209 (1992).
18. S. Osher, Subscale capturing in numerical analysis, in *Proceeding of the International Congress of Mathematicians, Zurich, 1994* (Birkhauser, Basel), p. 1449.
19. S. Osher, A level set formulation for the solution of the Dirichlet problem for Hamilton-Jacobi equations, *SIAM J. Numer. Anal.* **24**, 1145 (1993).
20. S. Osher and J. A. Sethian, Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulation, *J. Comput. Phys.* **79**, 12 (1988).
21. S. Osher and C. W. Shu, High-order essentially non-oscillatory schemes for Hamilton-Jacobi equations, *J. Numer. Anal.* **28**, 907 (1991).
22. J. A. Sethian, *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Material Science* (Cambridge Univ. Press, Cambridge, UK, 1996).
23. J. A. Sethian, A fast marching level set method for monotonically advancing fronts, *Proc. Nat. Acad. Sci.* **93**, 1591 (1996).
24. C-W. Shu, Total-variation-diminishing time discretization, *SIAM J. Sci. Stat. Comput.* **9**, 1073 (1988).
25. C. W. Shu and S. Osher, Efficient implementation of essentially nonoscillatory shock capturing schemes, *J. Comput. Phys.* **77**, 439 (1988).
26. M. Sussman, P. Smereka, and S. Osher, A level set method for computing solutions to incompressible two-phase flow, *J. Comput. Phys.* **119**, 146 (1994).
27. H. K. Zhao, T. Chan, B. Merriman, and S. Osher, A variational level set approach to multi-phase motion, *J. Comput. Phys.* **122**, 179 (1996).
28. H. K. Zhao, B. Merriman, S. Osher, and L. Wang, Capturing the behavior of bubbles and drops using the variational level set approach, *J. Comput. Phys.* **143**, 495 (1998).