Interfaces and Free Boundaries **1**, (1999) 107–123

Motion by intrinsic Laplacian of curvature

DAVID L. CHOPP[†]
Engineering Sciences and Applied Mathematics Department, Northwestern University, Evanston, IL 60208, USA
Email: chopp@nwu.edu

AND

J. A. SETHIAN[‡]
Department of Mathematics, University of California, Berkeley, CA 94720, USA
Email: sethian@math.berkeley.edu

[Received 23 December 1998]

In this paper, we discuss numerical schemes to model the motion of curves and surfaces under the intrinsic Laplacian of curvature. This is an intrinsically difficult problem, due to the lack of a maximum principle and the delicate nature of computing an equation of motion which includes a fourth derivative term. We design and analyze a host of algorithms to try and follow motion under this flow, and discuss the virtues and pitfalls of each. Synthesizing the results of these various algorithms, we provide a technique which is stable and handles very delicate motion in two and three dimensions. We apply this algorithm to problems of surface diffusion flow, which is of value for problems in surface diffusion, metal reflow in semiconductor manufacturing, sintering, and elastic membrane simulations. In addition, we provide examples of the extension of this technique to anisotropic diffusivity and surface energy which results in an anisotropic form of the equation of motion.

1. Introduction

In this paper, we discuss numerical schemes to model the motion of curves and surfaces under the intrinsic Laplacian of curvature. By this, in two dimensions we mean motion in a direction normal to a closed, simple curve with a speed function F that depends on the second derivative of the local curvature with respect to arc length; in three dimensions, this translates into the motion of a surface normal to itself with speed that depends on the Laplacian of the mean curvature, where the Laplacian on the surface is constructed from the derivative with respect to arc length in each principal direction. This is an intrinsically difficult problem for three reasons. First, owing to the lack of a maximum principle, an embedded curve need not stay embedded, and this has significant implications in attempting to analyze motion which results in topological change. Second, the equations of motion contain a fourth derivative term, and hence are highly sensitive to errors. Third, this fourth derivative term leads to schemes with very small time steps. In this paper, we design and analyze a host of algorithms to try and follow motion under this flow, and discuss the virtues and pitfalls of each. Synthesizing the results of these various algorithms, we provide a technique which is stable and handles very delicate motion in two and three dimensions.

[†]Supported by the NSF/DARPA VIP program under award #96-15877.
[‡]Supported in part by the Applied Mathematics Subprogram of the Office of Energy Research under contract DE-AC03-76SF00098, and the National Science Foundation.

We apply this algorithm to problems of surface diffusion flow, which is of value for problems in surface diffusion, metal reflow in semiconductor manufacturing, sintering, and elastic membrane simulations. For these flows, a local existence proof for this flow has been given by Elliott & Garcke [13] as well as a proof of the stability of a circle as a limiting shape. Giga & Ito [16] also give an existence and uniqueness proof and in addition prove that embedded plane curves need not remain embedded as they evolve, but can form singularities where they pinch off. An existence and uniqueness proof for certain smooth initial conditions has been found by Escher, Mayer & Simonett [14] as well as a proof of conservation of area/volume and some computational examples. Coleman, Falk & Moakher [11] demonstrate pinch-off in the case of three-dimensional axisymmetric surfaces using a finite element approach.

In addition, we provide examples of the extension of this technique to anisotropic diffusivity and surface energy which results in an anisotropic form of the equation of motion. This anisotropic flow was first proposed by Mullins [19] to model curvature driven diffusion on the surface of a crystal. More mathematical derivations have been examined by both Davi & Gurtin [12], Cahn & Taylor [7], and by Cahn, Elliott & Novick-Cohen [6].

This paper does not present a definitive algorithm for computing fourth derivative surface diffusion interface flows. Instead, our goal is to carefully analyze and discuss some approaches, and to try to extract a workable technique for many problems from the information gleaned. The outline of this paper is as follows. First, we briefly review some background on Level Set Methods and Fast Marching Methods, which lie at the core of our approach. Then, we cast evolution by Laplacian of the curvature in this embedded level set framework. Next, we design several different approaches to solving this embedded level set framework, and discuss the virtues and drawbacks. This leads to a hybrid algorithm which is presented in detail in the following section, accompanied by a collection of numerical results. Finally, we end with a discussion of additional issues that need to be considered in devising an all-purpose algorithm.

2. Background

The straightforward approach to the problem of motion by the second derivative of curvature is through a parameterized view of the equations of motion. Begin by letting γ be a simple, smooth, closed initial curve in R^2, and let $\gamma(t)$ be the one-parameter family of curves generated by moving γ along its normal vector field with speed F. Here, F is the given scalar function. Thus $\mathbf{n} \cdot \mathbf{x}_t = F$, where $\mathbf{x}$ is the position vector of the curve, t is time, and $\mathbf{n}$ is the unit normal to the curve. Let the position vector $\mathbf{x}(s, t)$ parameterize γ at time t, where $0 \leqslant s \leqslant S$, and assume periodic boundary conditions $\mathbf{x}(0, t) = \mathbf{x}(S, t)$. The curve is parameterized so that the interior is on the left in the direction of increasing s. Furthermore, let $\mathbf{n}(s, t)$ be the parameterization of the outward normal and let $\kappa(s, t)$ be the parameterization of the curvature. Then, in the case where the speed function F depends on the second derivative of the curvature, we have

$$F = -\kappa_{ss} = -\frac{1}{\sqrt{x_s^2 + y_s^2}} \left\{ \frac{1}{\sqrt{x_s^2 + y_s^2}} \left[\frac{y_{ss}x_s - x_{ss}y_s}{(x_s^2 + y_s^2)^{3/2}} \right]_s \right\}_s$$

where here we have used the parameterized expression for the curvature, namely

$$\kappa = \frac{y_{ss}x_s - x_{ss}y_s}{(x_s^2 + y_s^2)^{3/2}}.$$

Thus, the equations of motion can then be written in terms of individual components $\mathbf{x} = (x, y)$ as

$$x_t = -\frac{1}{\sqrt{x_s^2 + y_s^2}} \left\{ \frac{1}{\sqrt{x_s^2 + y_s^2}} \left[\frac{y_{ss}x_s - x_{ss}y_s}{(x_s^2 + y_s^2)^{3/2}} \right]_s \right\}_s \left(\frac{y_s}{(x_s^2 + y_s^2)^{1/2}} \right),$$

$$y_t = \frac{1}{\sqrt{x_s^2 + y_s^2}} \left\{ \frac{1}{\sqrt{x_s^2 + y_s^2}} \left[\frac{y_{ss}x_s - x_{ss}y_s}{(x_s^2 + y_s^2)^{3/2}} \right]_s \right\}_s \left(\frac{x_s}{(x_s^2 + y_s^2)^{1/2}} \right).$$

Here, we have used the fact that the normal is given by

$$\mathbf{n} = (y_s, -x_s)/(x_s^2 + y_s^2)^{1/2}.$$

This is a 'Lagrangian' representation because the moving front is represented by the range of $(x(s, t), y(s, t))$.

In this formulation, the speed depends on the fourth derivative of the position vector. An approach using marker particles to follow this motion is very hard to make workable, owing to the high sensitivity of computing fourth derivatives along the front itself (see [22]). The difficulties of such an approach are eloquently described by Van de Vorst [29], who uses marker particle schemes together with elaborate remeshing strategies to keep the calculation alive.

One of the most versatile and effective ways of computing the motion of curves and surfaces under geometry-dependent speed laws is the level set methodology developed by Osher & Sethian [20]. This level set approach grew out of the theory and numerics of curve and surface evolution developed by Sethian in [21–23], which constructed the notion of weak solutions and entropy limits for evolving interfaces, linked upwind numerical methodology for hyperbolic conservation laws to front propagation problems, and developed numerical schemes for curvature-based flows. Geometric calculations using these techniques were presented in [24], in which the breaking singularities of mean curvature flow were studied in detail. The calculations and numerical methodologies introduced in [20, 22, 24] have, in recent years, provided the basis for a large collection of calculations in such areas as combustion and fluid mechanics [30], medical imaging [18], and etching and deposition in semi-conductor manufacturing [2, 3, 5]. For a review and resource on level set methods, see [25].

In the level set approach, both the interface of interest and the interface velocity field are embedded in higher dimensional functions. The standard advantages of a level set approach are that topological changes are handled naturally, that the technique is unchanged in three and higher dimensions, and that finite difference schemes can be used for approximation operators on a fixed Eulerian mesh. It is this last property that will be crucial for studying motion of the second derivative of curvature; calculation of intrinsic geometric properties of the front, including the curvature, comes not solely from information about the interface, but through consideration of the neighboring level sets which carry the embedding. Figuring out how to exploit this embedding is the challenge in devising techniques for surface diffusion driven motion.

3. The level set formulation

3.1 *Equations of motion*

Imagine a closed curve Γ in the plane propagating normal to itself with speed F. We can embed the initial position of the front as the zero level set of a higher dimensional function ϕ, and then identify the evolution of this function ϕ with the propagation of the front itself through a time-dependent initial value problem. At any time, the front is given by the zero level set of the time-dependent level set function ϕ. In order to derive an equation of the motion for this level set function ϕ, we note that the stipulation that the zero level set of the evolving function ϕ always match the propagating hyper-surface means that

$$\phi(x(t), t) = 0. \tag{3.1}$$

By the chain rule,

$$\phi_t + \nabla\phi(x(t), t) \cdot x'(t) = 0. \tag{3.2}$$

Since F supplies the speed in the outward normal direction, then $x'(t) \cdot n = F$ where $n = \nabla\phi/|\nabla\phi|$ and this yields an evolution equation for ϕ, namely,

$$\phi_t + F|\nabla\phi| = 0, \tag{3.3}$$

$$\text{given} \quad \phi(x, t = 0).$$

This is the level set equation introduced by Osher & Sethian [20].

As analyzed by Sethian in [23], the efficient solution of these front propagation problems requires the use of upwind difference schemes and schemes borrowed from the solution of hyperbolic conservation laws. A detailed discussion of such schemes in the context of interface propagation schemes may be found in [25].

3.2 *Adaptivity: The Narrow Band Method*

Note that the embedding of the interface as the zero level set of a higher dimensional function means that calculations are performed over the entire computational domain; which is wasteful. Instead, an efficient modification is to perform work only in a neighborhood of the zero level set; this is known as the *Narrow Band Approach*. This results in an optimal technique which has a much lower operation count. The strategy was introduced in Chopp [9], used in recovering shapes from images in Malladi, Sethian & Vemuri [18], and analyzed extensively by Adalsteinsson & Sethian in [1]. Briefly, the interface propagates until it reaches the edge of its narrow band, at which point a new narrow band is built by re-initializing a new narrow band around the current front's position. The width of the narrow band is a balance between the labor involved in re-initializing and calculations performed on far away points. A narrow band one or two grid cells large requires re-initialization every time step; an infinitely large narrow band requires no re-initialization, but defaults to the regular level set method. For details, see [1].

3.3 *Construction of extension velocities*

In order to apply the level set method, the velocity field F itself must be defined on the entire domain of ϕ, not just the zero level set corresponding to the interface itself. We can be more precise by rewriting the level set equation as

$$\phi_t + F_{\text{ext}}|\nabla\phi| = 0 \tag{3.4}$$

where F_{ext} is some velocity field which, at the zero level set, equals the given speed F. In other words,

$$F_{\text{ext}} = F \text{ on } \phi = 0.$$

This new velocity field F_{ext} is known as the 'extension velocity'.

In [4], a technique was introduced for building this extension velocity field F_{ext} from a velocity field given on the front in a highly efficient and accurate manner. This technique relied on the Fast Marching Method [26], which is the optimal technique for solving the Eikonal equation.

Briefly, the Fast Marching Method solves

$$|\nabla u| = \frac{1}{F(x, y, z)} \tag{3.5}$$

by first replacing the gradient by suitable upwind operators, and then systematically advancing the front by marching outwards from the boundary data in an upwind fashion. The key to the algorithm lies in the observation that an upwind operator implies a causality, and hence grid points with a given value for u cannot be affected by those with a bigger value. Hence, as the solution is advanced, we can maintain a heap sort which keeps track of the smallest element to be updated, and thus always advances the solution 'downwind' of that point. Through the use of this sorting algorithm, each point in the domain is visited *only once*, rather than requiring any iteration; the resulting technique has a total operation count of $O(N \log N)$.

This technique is then used to construct the extension velocity F_{ext} by computing the signed distance function ϕ (obtained by letting $F = 1$ in Eqn 3.5) while simultaneously solving the associated equation

$$\nabla F_{\text{ext}} \cdot \nabla \phi = 0. \tag{3.6}$$

The resulting velocity field F_{ext} equals the velocity on the front, and is reasonably smooth off of the front. We shall use both the Narrow Band Method and a variation of the velocity extension methods based on the Fast Marching Method to construct a stable algorithm for flow under the Laplacian of curvature.

4. Equations of motion for evolution by Laplacian of curvature

4.1 *Flow under curvature*

In the case of flow under curvature, a natural extension velocity is provided by the embedding of the interface as the zero level set of a higher dimensional function. We may let the speed of each level set be the curvature of that particular interface. Thus, we have the equation of motion

$$\phi_t - \kappa|\nabla \phi| = 0. \tag{4.1}$$

To be precise about what is happening, consider a point (x, y). Then the value of $\phi_t(x, y)$ depends on the curvature of the level set passing through the point (x, y). This 'natural' embedding gives us a velocity field by which to move all the level sets in the narrow band, not just the zero level set.

Why does this work? In the case of curvature flow, there is a maximum principle which prevents disjoint parts of the interface from colliding with each other; an embedded curve remains embedded.

In this level set formulation, each level set in the flow moves according to the same speed law. In the case of curvature flow, Evans, Soner & Souganidis [15] proved that the level sets behave nicely in the sense that two different level sets do not cross and in fact remain roughly evenly spaced in time. In terms of the level set function ϕ, this corresponds to the fact that the gradient of ϕ at any given point of a level set, does not change dramatically over time. For the numerical method this translates into numerical stability. The finite difference algorithm to approximate the equations of motion is a straightforward explicit-forward time central-difference space scheme.*

4.2 *Flow under Laplacian of curvature*

Let us try and mimic this technique to produce a level set equation of motion for flow under the Laplacian of curvature.† Let the initial curve γ be described as the zero level set of a function $\phi(x, y)$,

$$\gamma = \{(x, y) : \phi(x, y) = 0\}.$$

From this, we can build an evolution equation for the level set function ϕ given by

$$\phi_t + \kappa_{ss}|\nabla\phi| = 0.$$

Note that κ is the local curvature of the level curve passing through the point (x, y) and can be expressed entirely in terms of the level set function ϕ by

$$\kappa = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}.$$

From this we can express the Laplacian of curvature by

$$\kappa_{ss} = \nabla\left[\nabla\left[\nabla\cdot\frac{\nabla\phi}{|\nabla\phi|}\right]\cdot\frac{(\phi_y, -\phi_x)}{|\nabla\phi|}\right]\cdot\frac{(\phi_y, -\phi_x)}{|\nabla\phi|} \tag{4.2}$$

$$= \frac{\kappa_{xx}\phi_y^2 - 2\kappa_{xy}\phi_x\phi_y + \kappa_{yy}\phi_x^2}{\phi_x^2 + \phi_y^2} - \frac{(\kappa_x\phi_x + \kappa_y\phi_y)\kappa}{\sqrt{\phi_x^2 + \phi_y^2}}. \tag{4.3}$$

One can see from this expression that κ_{ss} is a non-linear term involving up to fourth order derivatives of the function ϕ. For anisotropic surface diffusion, we follow the derivation in [12] where the anisotropic surface diffusion can be written as

$$\begin{aligned}\kappa_{ss} = \nabla\Bigg\{D(\theta)\nabla\Bigg[&(g_0(\theta) + g_0''(\theta))\left(\nabla\cdot\frac{\nabla\phi}{|\nabla\phi|}\right)\\ &+g_0'(\theta)\left(\frac{(\phi_{xx} - \phi_{yy})\phi_x\phi_y + \phi_{xy}(\phi_y^2 - \phi_x^2)}{(\phi_x^2 + \phi_y^2)^{3/2}}\right)\Bigg]\\ &\cdot\frac{(\phi_y, -\phi_x)}{|\nabla\phi|}\Bigg\}\cdot\frac{(\phi_y, -\phi_x)}{|\nabla\phi|}\end{aligned}$$

* For a web page and java script demonstrating this application, see http:/math.berkeley.edu/∼ sethian/level_set.html

† From now on, 'second derivative with respect to curvature (or mean curvature)' shall always mean with respect to the local arc length; we shall just abbreviate and say motion under the Laplacian of curvature.

where θ is the direction of the outward normal, $D(\theta)$ is the diffusivity, and $g_0(\theta)$ is the surface energy. The anisotropic flow reduces to the isotropic flow when $D(\theta) \equiv 1$, $g_0(\theta) \equiv 1$.

It is natural to conjecture that a numerical method similar to the evolution of mean curvature flow, see for example [10] and [8], will result in an equally stable method for speed κ_{ss}. In fact this is not the case. The nice properties of curvature flow do not carry over to the Laplacian of curvature flow. In particular, there is no reason to expect $|\nabla\phi|$ even to remain bounded in time for any particular point traveling with the front. This means the long term stability of such a numerical method is heavily dependent upon the initial data. For initially convex interfaces, we find that long term stability can be maintained. For more complicated interfaces, however, the stability of this method cannot be guaranteed.

The problem is also complicated by the fact that the evolving zero level set under this flow can collide with itself. To see that this is so, we first observe that a circle remains stationary under this motion, since the second derivative of the curvature of a circle is zero and hence there is no motion. Another initial front placed next to this circle can bump into it under its own motion, and hence we cannot guarantee that the level set curves stay separated under this flow; additional examples of this behavior can be found in [14]. Thus, without suitably defining an appropriate model for what happens when fronts collide, the level set function itself need not remain a function for all time. Finally, there are issues related to the sensitivity of κ_{ss} to numerical errors. In the next section, we discuss several approaches to devising an algorithm for this problem.

5. Various techniques for approximating the equations of motion

5.1 *A straightforward embedding*

As a first attempt, we tried to follow the algorithm for curvature flow and use the second derivative of the curvature of each level set to advance that particular level set. Thus, we used the 'natural' embedding. The difficulty with this approach stems from the numerical accuracy of approximating the second derivative of curvature, as well as the behavior of the velocity field near critical points of the level set function.

Consider a grid point where the level set function ϕ has a local maximum. The curvature of the level sets near such points, and hence the approximations of the curvature, become very large. In the case of curvature flow, this does not compromise the calculation because the growth of the curvature is balanced by $|\nabla\phi| \to 0$, and by the fact that the velocity field near such points remains smooth.

The same cannot be said of the derivatives of curvature. The large values in the curvature near such points leads to even larger values for the derivative of curvature, which still is only balanced by $|\nabla\phi|$. The consequence is that the straightforward approximations for κ_{ss} near such singularities yield unstable results. This discussion can also be applied to all other points where $|\nabla\phi| \approx 0$.

5.2 *Using the narrow band approach*

Of course, except for changes in topology, $|\nabla\phi| \neq 0$ on the interface $\phi = 0$ we are evolving. This means that if we take a fine enough mesh, and restrict our calculations to a very narrow band around the level set $\phi = 0$, we can effectively eliminate almost all troublesome calculations near where $|\nabla\phi| = 0$. Depending upon the complexity of the initial interface, this may require an exceptionally small mesh and we must bear in mind that the time step restriction for an explicit method scales as $\Delta t \sim \Delta x^4$. We therefore searched for an alternative which would allow for larger mesh sizes while addressing the problems near points where $|\nabla\phi| = 0$.

5.3 *Variations on velocity extensions*

5.3.1 *A straightforward velocity extension.* As an alternative approach, we tried using the velocity extension methodology of [4]. This method assumes that the velocity is known everywhere on the interface itself, and that velocity is then extended outward in the direction normal to the interface. First, we used fourth order spatial derivatives in computing κ_{ss} from the neighboring level sets, and second, we extended this velocity throughout the narrow band. The numerical computation of κ_{ss} is given by

$$\phi_x \approx \frac{1}{12\Delta x}\left(-\phi_{i+2,j} + 8\phi_{i+1,j} - 8\phi_{i-1,j} + \phi_{i-2,j}\right) \tag{5.1}$$

$$\phi_{xx} \approx \frac{1}{12\Delta x^2}\left(-\phi_{i+2,j} + 16\phi_{i+1,j} - 30\phi_{i,j} + 16\phi_{i-1,j} - \phi_{i-2,j}\right) \tag{5.2}$$

$$\begin{aligned}\phi_{xy} \approx \frac{1}{48\Delta x\Delta y}\big(&-\phi_{i+2,j+2} + 16\phi_{i+1,j+1} + \phi_{i-2,j+2} - 16\phi_{i-1,j+1}\\ &+\phi_{i+2,j-2} - 16\phi_{i+1,j-1} - \phi_{i-2,j-2} + 16\phi_{i-1,j-1}\big)\end{aligned} \tag{5.3}$$

$$\kappa = \frac{\phi_{xx}\phi_y^2 + \phi_{yy}\phi_x^2 - 2\phi_{xy}\phi_x\phi_y}{(\phi_x^2 + \phi_y^2)^{3/2}} \tag{5.4}$$

$$\kappa_{ss} = \frac{\kappa_{xx}\phi_y^2 - 2\kappa_{xy}\phi_x\phi_y + \kappa_{yy}\phi_x^2}{\phi_x^2 + \phi_y^2} - \frac{(\kappa_x\phi_x + \kappa_y\phi_y)\kappa}{(\phi_x^2 + \phi_y^2)^{1/2}} \tag{5.5}$$

where the approximations for ϕ_y, ϕ_{yy}, κ_x, κ_y, κ_{xx}, κ_{xy}, and κ_{yy} are done using the same stencils as for ϕ_x, ϕ_{xx}, and ϕ_{xy}.

While this approach showed marginal success, we found that the use of velocity extensions reduced the numerical accuracy below that necessary to maintain good approximations for κ_{ss}. Using this technique for all points resulted in insufficient accuracy.

5.3.2 *Use of splines.* Next, we tried using splines to approximate κ_{ss} on the interface. The hope here was to provide a more accurate formulation of the second derivative of the curvature to use in the extension velocity. Given a set of points on the interface $\phi = 0$, we constructed a spline $\gamma(s)$ passing through those points. The value of κ_{ss} could then be generated using the parametric formula for κ_{ss} from $\gamma(s)$. In order to generate a smooth velocity field, we required that γ be at least C^4, so for polynomial splines this required quintic polynomials.

As a test, we achieved very good results approximating motion by curvature. However, for Laplacian of curvature, we again encountered the problems associated with velocity extensions which are of lower order accuracy than required for good approximations of κ_{ss}. Furthermore, the quintic polynomials tended to generate very slight oscillations, which led to oscillatory approximations for κ_{ss}, and consequently instability.

5.3.3 *Isolated special points.* These experiments led us to treat points in the narrow band differently depending on whether or not they were either located more than a certain distance from the interface or had gradients below a tolerance threshold. To that end we identify a set of grid points K, described below, which are in the narrow band around $\phi = 0$ and where κ_{ss} can be stably computed. For the points in K, ϕ_t is computed using the original level set method as used for curvature flow, employing the above fourth order finite difference stencil. For all other points, we use a velocity field determined from a velocity extension method.

Thus, the idea is to use the 'natural velocity' value away from troublesome points, and use the extension velocity technique to fill in the gaps. This is the hybrid technique presented below.

6. A hybrid approach

6.1 *Extension of the velocity field*

Based upon our discussions above, our method must consist of directly computing κ_{ss} where $|\nabla\phi| >> 0$, i.e. on some set of grid points K, and using some form of velocity extension for the remaining points in the narrow band.

We designate a grid point to be in K if it is at most five grid points away from the zero level set of ϕ and is not a flat point or immediate neighbor of a flat point. A flat point, our discrete equivalent to a critical point, is a point where $|\nabla\phi| \approx 0$ and can be identified by any one of the following criteria:

1. $\phi_{ij} \geqslant \max(\phi_{i-1,j}, \phi_{i+1,j}, \phi_{i,j-1}, \phi_{i,j+1})$
2. $\phi_{ij} \leqslant \min(\phi_{i-1,j}, \phi_{i+1,j}, \phi_{i,j-1}, \phi_{i,j+1})$
3. $|\nabla\phi| < \epsilon$ for some parameter ϵ.

Before we continue, we make a few comments about the set K. First, the number of grid points in the narrow band that are excluded from the set K is fixed with respect to mesh refinement; as the mesh is refined, the size of the regions where the velocity extension is used shrinks. Second, the width of five grid points is not arbitrary but is based upon the fact that the stencil for computing a fourth order accurate approximation of κ is 5×5. Third, flat points include all local extrema and saddle points.

Again, for the points in the set K, the velocity is computed directly from Eqn (4.2) as approximated in Eqns (5.1–5.5). For the remaining points, we need to patch up the velocity field so that the evolution of ϕ remains stable. The velocity field extension is accomplished by means of a modified version of the Fast Marching Method described above. We tried other versions of velocity extensions and found the results to be unsatisfactory for this particular flow. The goals of the velocity extension for this problem were that the extended velocity should be the same as the nearest directly computed velocity value, the velocities at points in K should not be altered by the velocity extension, and the direction of flow of velocity information may need to be bidirectional in the event that a point adjacent to the zero level set requires an extended velocity value. In fact, we do not extend the interface velocity, κ_{ss}, but the speed of the level set function itself, $\delta\phi = -\kappa_{ss}|\nabla\phi|$ as is done in [4].

The velocity extension method we use here is based upon the principle that where the extension velocity is needed, we use the velocity of the nearest point in the set K. This is a different approach than was used in [4]. For this approach, we simultaneously solve $|\nabla u| = 1$ using the fast marching method, and then solve

$$\nabla u \cdot \nabla(\delta\phi) = 0 \tag{6.1}$$

where the gradients are computed using upwind differences. The difference between this method and that in [4] is that the function u is initialized to be 0 for every point which is in K, and the rest of the points are computed to be the distance to K. We found that this form of extending the velocity gave the best results.

6.2 *Reinitialization*

As we discussed above, we do not have the property that the level sets remain evenly spaced in time. Although this is not a problem for all initial data, e.g. an oval, it is easy to construct initial data for which the level set function will develop sharp gradients. When this occurs, the computation of κ_{ss} can break down, and the method becomes unstable.

A remedy for this is to periodically reconstruct the level set function to force the level sets back into an evenly spaced configuration. This process is called reinitialization. Reinitialization was first introduced in [9] and later improved upon in [1]. An alternative form of reinitialization can be found in [28]. Here we follow the procedure outlined in [1].

We use the Fast Marching Method to solve the equation $|\nabla\phi| = 1$. The resulting function retains the current level set $\phi = 0$ in the same location while repositioning the other level sets to be equally spaced.

The use of reinitialization provides the extra stability required to complete the numerical method, however, it comes at a price. Reinitialization can also be seen as a form of artificial diffusion. The amount of artificial diffusion added to the problem, and the amount of the error introduced in slight movement of the interface varies between reinitialization methods. In any case, we use reinitialization as little as possible without compromising the stability of the method.

Improvements in the method which will reduce the need for reinitialization are the subject of ongoing research.

6.3 *The algorithm and examples*

We now put together the pieces we have detailed above and outline the iterative step. We assume that the initial data is given in the form of a parametric curve $\gamma(s)$.

1. Construct the initial level set function ϕ_0 from $\gamma(s)$ by using the signed distance function.
2. In a band of grid points around the interface, i.e. the set K, compute $\delta\phi = -\kappa_{ss}|\nabla\phi|$.
3. Find all flat points, grid points where ϕ is a local maximum, minimum, or where $|\nabla\phi| < \epsilon$. Remove all such grid points and their neighbors, including diagonally, from the set K.
4. Extend the velocity of the set K to the remainder of the points in the domain using the fast marching method.
5. Advance ϕ in time by an explicit time stepping scheme.
6. Reinitialize if a set number of time steps have elapsed, or if the average of $|\nabla\phi|$ has exceeded some threshold.
7. Go to step 2.

Note that the above computation can be done using either a narrow band method where the number of grid points outside the set K is at least two additional grid points, or the velocity can be extended to the entire domain of ϕ with the same results.

We show some examples of this algorithm. We begin with the example of a star shape in Fig. 1 flowing under isotropic Laplacian of curvature. In this example, we use a 60×60 mesh with a space step size of $\Delta x = 0.0667$. The time step size for this calculation was $\Delta t = 5 \times 10^{-6}$. The shape

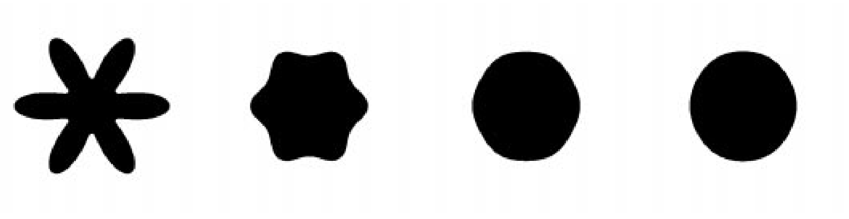

FIG. 1. Evolution of a star shape.

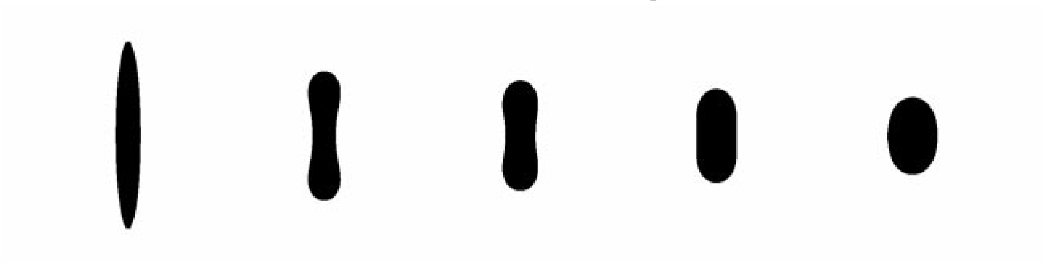

FIG. 2. Evolution of a oval shape.

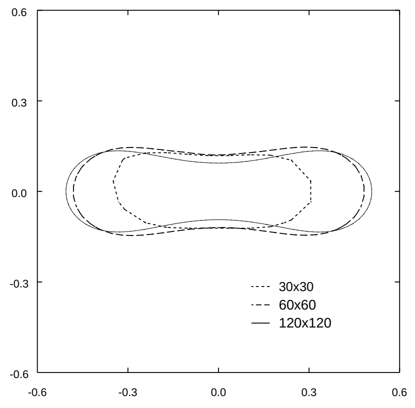

FIG. 3. Computation of initial oval with different spatial resolutions.

is roughly circular within 15 000 iterations, and 43 reinitializations were used. This illustrates how sparingly we apply reinitializations.

Next in Fig. 2, we illustrate how an initially convex oval becomes non-convex before relaxing to a circle. In this example, we use a 60×60 mesh with a space step size of $\Delta x = 0.0667$. The time step size for this calculation was $\Delta t = 5 \times 10^{-6}$. The shape is roughly circular within 20 000 iterations.

We use this same initial oval to verify the convergence and conservation properties of this method. We computed the oval solution to a fixed time using 30×30, 60×60, and 120×120 meshes. The comparison of these different computations can be seen in Fig. 3. Besides the convergence in shape, we can also observe convergence in the area conservation property where these meshes lose approximately 39%, 10%, and 4% of the initial area respectively. It should be noted that the primary

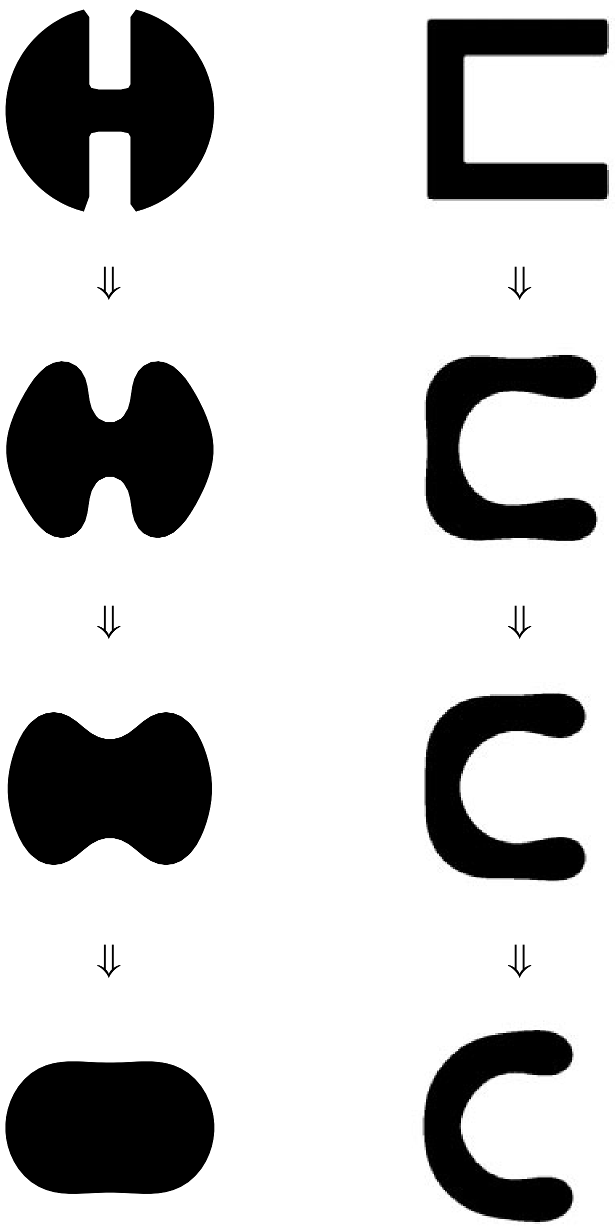
FIG. 4. Motion of non-convex curves under speed $F = \kappa_{ss}$.

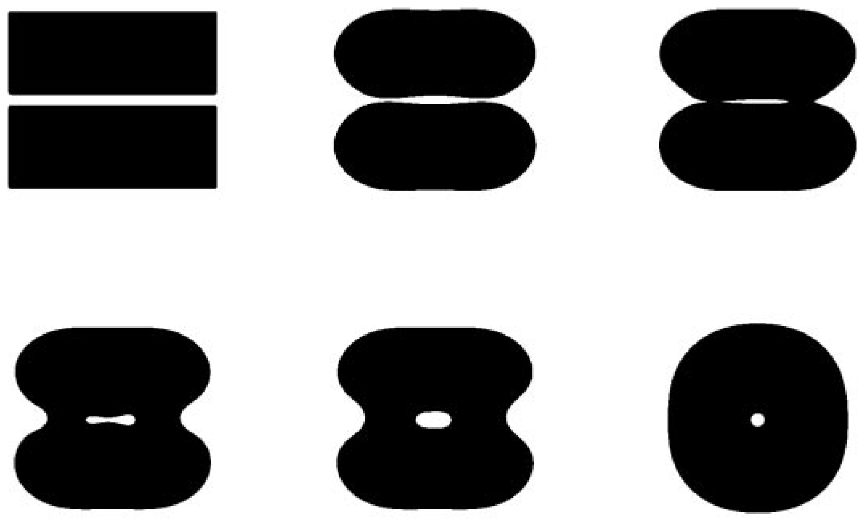

FIG. 5. Example of two shapes merging.

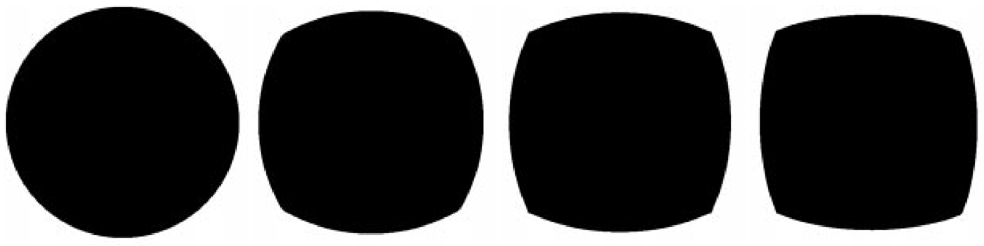

FIG. 6. Anisotropic flow with four-way anisotropy.

cause of loss of mass is due to the occasional reinitialization steps.

We continue in Fig. 4 by showing the motion of two more shapes under this motion.

Our last two-dimensional isotropic example demonstrates merging in a more refined calculation, here we use a 100×100 mesh which seems to be the bare minimum resolution to obtain adequate results. The evolution is depicted in Fig. 5.

In Fig. 6, we illustrate an anisotropic flow. The diffusivity and surface energy functions have been normalized so that they have mean value 1, and the plots for these functions are given in Figs 7 and 8. Note that in Fig. 8, we plot both the energy function $g_0(\theta)$ (solid line) and the sum $g_0(\theta)+g_0''(\theta)$ (dashed line). The surface energy function is based upon data for Aluminum diffusion from Stumpf & Scheffler [27], and the diffusivity function is derived from molecular dynamics modeling provided by Huang, Gilmer & Díaz de la Rubia [17]. The surface starts initially as a

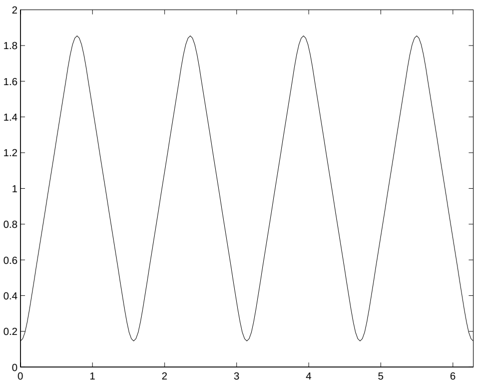

FIG. 7. Plot of diffusivity vs angle of the normal.

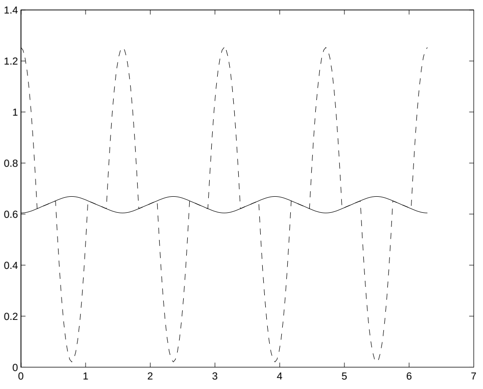

FIG. 8. Plot of energy functions $g_0(\theta)$ and $g_0(\theta) + g_0''(\theta)$.

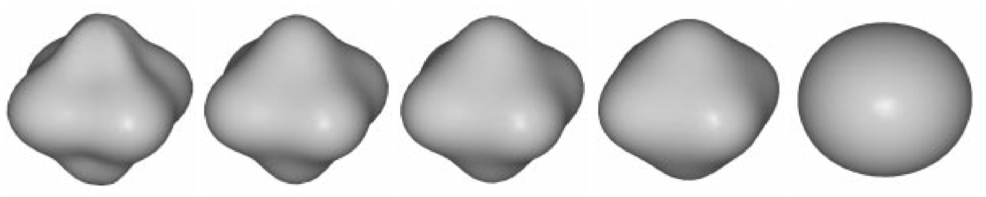

FIG. 9. Relaxation of three-dimensional shape to sphere.

circle, and then evolves towards faceting. Here we have taken a four-fold anisotropy resulting in evolution towards a square.

In our final examples, in Fig. 9 we show isotropic flow in three dimensions. We start with an initially non-convex surface and show how it relaxes to a sphere. For this computation, we used a $30 \times 30 \times 30$ mesh with a space step size of $\Delta x = 0.1$. The time step size for this calculation was $\Delta t = 1 \times 10^{-6}$. The time step restriction is not dramatically worse than for two dimensions, but it

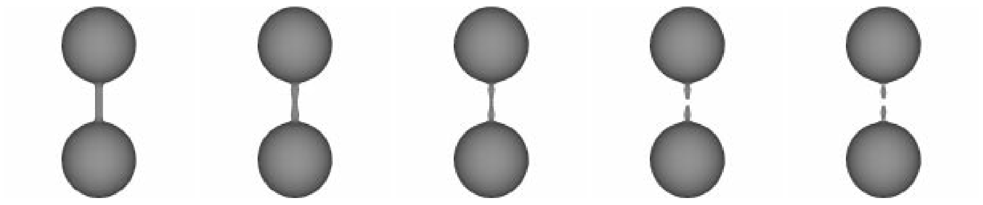

FIG. 10. Pinch-off of a dumbbell shape.

is a little more restrictive as would be expected.

Finally, we start with an initial dumbbell shape consisting of two spheres connected by a thin cylinder as shown in Fig. 10. There is a known instability in the flow which causes perturbations in a cylinder to eventually pinch-off. This example shows the computation passing through pinch-off. The computation used a $60 \times 30 \times 30$ mesh with space step size $\Delta x = 0.2$ and time step size $\Delta t = 5 \times 10^{-6}$.

7. Conclusions and comments

In this paper, we have presented an algorithm for applying the level set method to the flow by Laplacian of curvature. We have shown how to make such a computation stable, with good accuracy for most initial data. At present, the algorithm does not preserve volume perfectly, but improves with refinement of the mesh.

The method extends in a straightforward manner to three-dimensional simulations as demonstrated above. Both the extension velocity methodology and the fourth-order finite difference approximation to the Laplacian of the curvature are unchanged. The more restrictive time step requirement for stability of the method does not appear to change appreciably when going to higher dimensions.

Topological changes in the method can be achieved if the front is more finely resolved. This restricts the time-stepping considerably and we would like to find an alternative approach that will allow topological changes in our more coarse-grained approach. This is the subject of ongoing research.

Simulating this flow using the level set method has proven to be an exceptionally difficult challenge, and the story is not yet complete. We plan to study this and similar flows further in order to improve our method and also to expand the capabilities of the level set method in general.

Acknowledgments

We would like to thank George Gilmer for helpful discussions about anisotropic diffusion flow and suggesting the shape of the diffusivity and surface energy functions derived from experimental data.

REFERENCES

1. ADALSTEINSSON, D. & SETHIAN, J. A. A fast level set method for propagating interfaces. *J. Comput. Phys.* **118**(2), (1995) 269–277.
2. ADALSTEINSSON, D. & SETHIAN, J. A. A unified level set approach to etching, deposition and lithography i: Algorithms and two-dimensional simulations. *J. Comput. Phys.* **120**(1), (1995) 128–144.

3. ADALSTEINSSON, D. & SETHIAN, J. A. A unified level set approach to etching, deposition and lithography ii: Three-dimensional simulations. *J. Comput. Phys.* 122(2):348–366, 1995.
4. ADALSTEINSSON, D. & SETHIAN, J. A. The fast construction of extension velocities in level set methods. *J. Comput. Phys.* **148**, (1999) 2–22.
5. ADALSTEINSSON, D. & SETHIAN, J. A. A unified level set approach to etching, deposition and lithography iii: Complex simulations and multiple effects. *J. Comput. Phys.* **138**(1), (1997) 193–223.
6. CAHN, J. W., ELLIOTT, C. M., & NOVICK-COHEN, A. The Cahn-Hilliard equation with a concentration depedendent mobility: Motion by minus the Laplacian of the mean curvature. *Eur. J. Appl. Math.* **7**, (1996) 287–301.
7. CAHN, J. W. AND TAYLOR, J. E. Surface motion by surface diffusion. *Acta Metallur. Mater.* **42**, (1994) 1045–1063.
8. CHOPP, D. L. Flow under geodesic curvature. Technical Report CAM 92-23, UCLA, May 1992.
9. CHOPP, D. L. Computing minimal surfaces via level set curvature flow. *J. Comput. Phys.* **106**, (1993) 77–91.
10. CHOPP, D. L. & SETHIAN, J. A. Flow under curvature: Singularity formation, minimal surfaces, and geodesics. *J. Exp. Math.* **2**(4), (1993) 235–255.
11. COLEMAN, B. D., FALK, R. S., & MOAKHER, M. Space-time finite element methods for surface diffusion with applications to the theory of the stability of cylinders. *SIAM J. Computing* **17**(6), (1996) 1434–1448.
12. DAVI, F. & GURTIN, M. E. On the motion of a phase interface by surface diffusion. *J. Appl. Math. Phys.* **41**, (1990) 782–811.
13. ELLIOTT, C. M. & GARCKE, H. Existence results for diffusive surface motion laws. *Adv. Math. Sci. Appl.* **7**(1), (1997) 467–490.
14. ESCHER, J., MAYER, U. F., & SIMONETT, G. The surface diffusion flow for immersed hypersurfaces. *SIAM J. Math. Anal.* **29**(6), (1998) 1419–1433.
15. EVANS, L. C., SONER, H. M., & SOUGANIDIS, P. E. Phase transitions and generalized motion by mean curvature. *Commun. Pure Appl. Math.* **45**(9), (1992) 1097–1123.
16. GIGA, Y. & ITO, K. On pinching of curves moved by surface diffusion. *Comm. Appl. Anal.* **2**(3), (1998) 393–405.
17. HUANG, H., GILMER, G. H., & DÍAZ DE LA RUBIA, T. An atomistic simulator for thin film deposition in three dimensions. Preprint, 1997.
18. MALLADI, R., SETHIAN, J. A., & VEMURI, B. C. Evolutionary fronts for topology-independent shape modeling and recovery. In *Proceedings of Third European Conference on Computer Vision, Stockholm, Sweden, Lecture Notes in Computer Science*, Volume 800, pp. 3–13, (1994).
19. MULLINS, W. W. Theory of thermal grooving. *J. Appl. Phy.* **28**, (1957) 333–339.
20. OSHER, S. & SETHIAN, J. A. Fronts propagating with curvature dependent speed: Algorithms based on Hamilton–Jacobi formulation. *J. Comput. Phys.* **79**, (1998) 12–49.
21. SETHIAN, J. A. An Analysis of Flame Propagation. PhD thesis, Dept. of Mathematics, University of California, Berkeley (1982).
22. SETHIAN, J. A. Curvature and the evolution of fronts. *Comm. Math. Phys.* **101**, (1985) 487–499.
23. SETHIAN, J. A. Numerical methods for propagating fronts. In CONCUS, P. & FINN, R. (eds), *Variational Methods for Free Surface Interfaces*. Springer-Verlag, NY (1987).
24. SETHIAN, J. A. Numerical algorithms for propagating interfaces: Hamilton–Jacobi equations and conservation laws. *J. Differen. Geom.* **31**, (1990) 131–161.
25. SETHIAN, J. A. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Material Science*. Cambridge University Press, Cambridge (1996).
26. SETHIAN, J. A. A marching level set method for monotonically advancing fronts. *Proc. Natl. Acad. Sci.*

USA **93**(4), (1996) 1591–1595.

27. Stumpf, R. & Scheffler, M. *Ab Initio* calculations of energies and self-diffusion on flat and stepped surfaces of Al and their implications on crystal growth. *Phys. Rev. B* **53**, (1996).
28. Sussman, M., Smereka, P., & Osher, S. J. A level set method for computing solutions to incompressible two-phase flow. *J. Comput. Phys.* **114**, (1994) 146–159.
29. Van De Vorst, G. A. L. Modelling and Numerical Simulation of Viscous Sintering. PhD thesis, Eindhoven University of Technology (1994).
30. Zhu, J. & Sethian, J. A. Projection methods coupled to level set interface techniques. *J. Comput. Phys.* **102**, (1992) 128–138.