Journal of Computational Physics 229 (2010) 1017–1042

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Diffusion generated motion using signed distance functions

Selim Esedoglu [a,*], Steven Ruuth [b], Richard Tsai [c]

[a] Department of Mathematics, University of Michigan, Ann Arbor, MI 48109, USA
[b] Department of Mathematics, Simon Fraser University, Burnaby, B.C., Canada V5A 1S6
[c] Department of Mathematics, University of Texas, Austin, TX 78712, USA

ARTICLE INFO

Article history:
Received 20 March 2009
Received in revised form 15 September 2009
Accepted 2 October 2009
Available online 24 October 2009

Keywords:
Diffusion generated motion
Grain growth
Multiphase flow
Level set
Mean curvature motion

ABSTRACT

We describe a new class of algorithms for generating a variety of geometric interfacial motions by alternating two steps: Construction of the signed distance function (i.e. redistancing) to the interface, and convolution with a suitable kernel. These algorithms can be seen as variants of Merriman, Bence, and Osher's threshold dynamics [25]. The new algorithms proposed here preserve the computational efficiency of the original threshold dynamics algorithm. However, unlike threshold dynamics, the new algorithms also allow attaining high accuracy on uniform grids, without adaptive refinement.

1. Introduction

In [25], Merriman, Bence, and Osher (MBO) proposed an intriguing algorithm for approximating the motion by mean curvature of an interface by alternating two computationally efficient steps: Convolution, and simple thresholding. To be precise, let $\Sigma \subset \mathbb{R}^N$ be a domain whose boundary $\partial\Sigma$ is to be evolved via motion by mean curvature. Given a time step size $\delta t > 0$, the MBO algorithm generates a time discrete approximation $\{\partial\Sigma_n\}$ to motion by mean curvature (where $\partial\Sigma_n$ is the approximation at time $t = n\delta t$) according to the following prescription for obtaining Σ_{n+1} from Σ_n:

1. *Convolution step:* Form $u : \mathbb{R}^N \to \mathbb{R}$ as

$$u(\mathbf{x}) = (G_t * \mathbf{1}_{\Sigma_n})(\mathbf{x}) \tag{1}$$

where $G_t(\mathbf{x})$ is the N-dimensional Gaussian kernel

$$G_t(\mathbf{x}) = \frac{1}{(4\pi t)^{N/2}} e^{-\frac{|\mathbf{x}|^2}{4t}}. \tag{2}$$

2. *Thresholding step:* Return to the realm of sets:

$$\Sigma_{n+1} = \left\{ \mathbf{x} \ : \ u(\mathbf{x}) \geqslant \frac{1}{2} \right\}. \tag{3}$$

This algorithm has been rigorously verified to converge to motion by mean curvature in the limit $\delta t \to 0^+$; see e.g. [17,4]. One of its major advantages is unconditional stability: The choice of time step size δt is constrained only by accuracy

* Corresponding author.
E-mail addresses: esedoglu@umich.edu (S. Esedoglu), sruuth@math.sfu.ca (S. Ruuth), ytsai@math.utexas.edu (R. Tsai).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.10.002

considerations; the scheme remains stable (in fact, *monotone*) for all choices, independently of spatial resolution. In addition, for any choice of δt, the computational complexity of each time step is low: The bottleneck is the convolution step, which can be accomplished using e.g. the fast Fourier transform (FFT) at $O(n \log n)$ cost when a uniform grid of n points is used for spatial discretization. This is a major benefit over standard level set based approaches [27] which invariably involve the solution of a degenerate, very nonlinear PDE; however, see e.g. [36] for some semi-implicit level set schemes. The thresholding based algorithm of MBO has been generalized in subsequent papers (e.g. [24,30,32]) to other geometric motions, and more recently to some fourth order flows [20,14,15].

Although the MBO algorithm is thus very attractive from a computational complexity point of view, it has well known drawbacks. Chief among them is its inaccuracy on uniform grids. Indeed, unless grid size is refined concurrently with the time step size, the approximate motion generated by the algorithm gets "stuck" [25]. Less severely, even at moderately large time step sizes, there can be very large errors in the computed dynamics. Hence, in practice, it is necessary to discretize the MBO scheme using a method which can provide subgrid resolution of the interface position. This is accomplished in [31] while maintaining the efficiency of the algorithm through the use of unequally spaced FFTs. Unequally spaced FFTs also enable the use of adaptive grids to concentrate the computational effort near the interface. Such a spatially adaptive strategy proves especially indispensable in simulating high order motions.

This paper explores a different class of diffusion generated motion algorithms, where the thresholding step is replaced by another fast procedure: Construction of the signed distance function to the interface. The motivation is very easy to explain: Unlike characteristic functions, signed distance functions can be accurately represented on uniform grids at subgrid accuracies due to their Lipschitz continuity. This alleviates the inaccuracies involved in the original (finite difference) MBO algorithm. The second step of MBO type algorithms, namely the convolution step, remains the same in character (though details might need to be different; see Section 5). Since there are a variety of existing algorithms for fast computation of signed distance functions (e.g. fast marching, fast sweeping, etc. [40,35,39,29,11]), the modification to the original MBO algorithm proposed here does not sacrifice efficiency (up to a constant factor, of course) for the resulting improved accuracy. Moreover, they lead to highly accurate computations on uniform grids – in our opinion, one of the greatest benefits of the proposed method.

2. Main idea and outline of results

In this section we discuss the main ideas of the paper, provide general motivation for the proposed algorithms, and give an outline of the results presented.

The inaccuracy of the original MBO threshold dynamics algorithm on uniform grids stems from representing characteristic functions of sets (i.e. binary functions) on such grids. Indeed, the thresholding step of the algorithm necessitates this at every time step. However, using a binary function on a uniform grid, the boundary of the set cannot be located with better accuracy than δx, the grid size. In particular, there is no way to *interpolate* and thus locate the interface with subgrid accuracy: The interface is essentially forced to follow grid lines.

Our observation is quite simple: In order to derive MBO type algorithms, the essential point is to

Represent the interface by a level set function $\psi(x)$ whose 1D profile $\phi : \mathbb{R} \to \mathbb{R}$ along every normal to the interface is identical, and satisfies

$$\phi'(0) \neq 0 \text{ and } \phi''(0) = 0. \tag{4}$$

In particular, this 1D profile need not be the Heaviside function as it is in the original algorithm, the discontinuous nature of which is the cause of poor accuracy on uniform grids. For example, the 1D profile ϕ can be chosen to be any other smooth, odd, monotone function of one variable that takes the value 0 at 0. Indeed, in case the level set function $\psi(\mathbf{x})$ representing the interface has identical profiles along every normal, it can then be written as

$$\psi(\mathbf{x}) = \phi(d(\mathbf{x}))$$

in a neighborhood of the interface, where d is the signed distance function to the interface. Then, for any such representation we have

$$\Delta\psi(\mathbf{x}) = \phi'(d(\mathbf{x}))\Delta d(\mathbf{x}) + \phi''(d)|\nabla d|^2 = \phi'(0)\kappa(\mathbf{x})$$

when evaluated at a point $\mathbf{x}$ *on* the interface (so that $d(\mathbf{x}) = 0$), under the assumption (4) on ϕ; here κ denotes mean curvature.

Indeed, as long as the level set function that represents the interface has the same profile along every normal to the interface, it is easy to see that alternating the construction of such a representation for the interface and convolution with positive, symmetric, unit mass kernels would always generate motion by mean curvature as the leading order motion, just as the original MBO scheme. In addition, a smooth profile with a uniform bound on its derivative would allow interpolation to locate the interface with accuracy considerably greater than δx. See Fig. 1 for an illustration of this basic point. The simplest smooth, odd, 1D profile is the identity function $\phi(\xi) = \xi$; this leads us to represent the interface with the *signed distance function*, and is the basis of the algorithms derived in the present paper. In particular, in this paper we ask: *What kind of interesting geometric flows can we generate by alternating convolution and the construction of the signed distance function?*

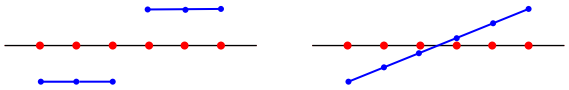

Fig. 1. The discontinuous, Heaviside function based profile of the characteristic function used to represent interfaces in the original MBO threshold dynamics does not allow to locate the interface at subgrid accuracy via interpolation (left). A function with a smooth normal profile, such as the signed distance function, does (right).

To explore this idea, in Section 4 we obtain Taylor expansions for the distance function of an interface in the plane in terms of its geometric quantities (such as curvature and derivatives of curvature). In particular, we concentrate on two situations: Points where the interface is smooth, and points where two smooth curves meet in a corner with given angle. The first expansion is relevant for deriving algorithms in two-phase flow, and the second is used for algorithms for multiphase motion, such as the motion of triple junctions. Section 5 is devoted to utilizing the expansions in Section 4 to derive algorithms for various types of motion. Section 5.1 focuses on the rather familiar case of motion by mean curvature, with the slight modification of an additional spatially varying normal speed. Although we suspect that the first algorithm we write down for this motion may not be completely unexpected, it is a good place to start and we do follow it up with more interesting high order in time variants. Section 5.2 describes a *monotone* algorithm for generating the motion $f(\kappa)$ where $f:\mathbb{R}\to\mathbb{R}$ is an odd, increasing, Lipschitz continuous function, and κ is the curvature of the interface. Section 5.3 explores an algorithm for motion of triple junctions under curvature flow with prescribed angle conditions at the junctions, and takes significant steps towards its justification by estimating the local truncation error at the junctions. Section 5.4 describes a tentative algorithm for motion by surface diffusion – a *fourth order* flow – using the signed distance function based approach of this paper. Finally in Section 6 we present numerical results and convergence studies with the algorithms proposed in Section 5. Although most of our derivations and algorithms are stated in two dimensions, some of them have immediate and straightforward extensions to higher dimensions; we briefly indicate these wherever appropriate.

3. Previous work

First and foremost, as already mention in Section 1, the approach to interfacial motion advocated in this paper is motivated by Merriman, Bence, and Osher's threshold dynamics [25]. The accuracy issue concerning this algorithm when implemented on uniform grids is well known and constitutes one of the main thrusts behind not only the present paper, but also that of several previous works. In [31], an adaptive refinement strategy for the MBO scheme is proposed and efficiently implemented using a spectral method in order to address the original scheme's accuracy shortcomings; that method represents an alternative strategy to the path taken here. Additionally, some of our discussions in Section 5 on how to generate a variety of interfacial motions using the signed distance function follows the analogous developments that use characteristic functions in [31,30]. In particular, our discussion of high order in time schemes for curvature flow in Section 5.1, as well as our treatment of multiphase flow of networks (junctions) in Section 5.3 have precursors in [31,30]. The signed distance function representation utilized in the present paper is less explicit a representation of an interface than a characteristic function representation. This makes the algorithms and especially the analysis in this paper quite different from these previous works.

It is interesting to make the connection, even though it is indirect, between one of the algorithms presented in this paper, namely the most basic one (64) and (65) of the several mean curvature motion algorithms from Section 5.1, and a recent algorithm for the same motion proposed by Chambolle in [8,9]. In these works, the author proposes an algorithm for implementing Almgren et al. discrete in time variational approximation [1] to motion by mean curvature that entails the construction of the distance function to an interface at every time step; this step of his algorithm is identical to step (65) of the one presented in Section 5.1 of the present paper. However, the second step of the algorithms presented in [8,9] involves the solution of a computationally very non-trivial total variation based optimization problem as in [28] per time step – this aspect is drastically different from the algorithms proposed in the present paper.

A distance function based level set-like algorithm for the special case of motion by mean curvature plus a constant is proposed in [22]. Although their algorithm also constructs the signed distance function to the interface at every time step and thus may be likened to one of the proposed algorithms, namely (64) and (65) in this paper, it is actually quite different. Indeed, the algorithm of [22] utilizes the signed distance function only in evaluating the right hand side of the explicit in time version of standard level set equation for mean curvature flow. Therefore, unlike the algorithms proposed in this paper, it lacks unconditional stability.

Finally, convergence to the viscosity solution [17] of the discrete in time solutions generated by the most basic one (64) and (65) of the several mean curvature motion algorithms presented in Section 5 has been established in [10].

4. Expansions for the distance function

In this section, we first write down a Taylor expansion of the signed distance function $d_\Sigma(\mathbf{x})$ in the neighborhood of a point $p\in\partial\Sigma$ on the smooth boundary $\partial\Sigma$ of a set Σ. For simplicity, we work mostly in $\mathbb{R}^2$ where we write $\mathbf{x}=(x,y)$. This expansion then allows us to obtain a Taylor expansion for the convolution of $d_\Sigma(x,y)$ with a Gaussian kernel $G_t(x,y)$. Our goal is to express the expansion coefficients in terms of the geometry (curvature and derivatives of curvature) of $\partial\Sigma$.

4.1. Expansion for a smooth interface

We will eventually work in the plane for convenience; but first, let us recall a few well known properties of the signed distance function that hold more generally in $\mathbb{R}^N$; see e.g. [18,12].

For $\mathbf{x} \in \partial\Sigma$, let $n(\mathbf{x})$ denote the unit outer normal to $\partial\Sigma$ at $\mathbf{x}$. The first familiar property we note is based on the fact that the normals to a smooth interface do not focus right away, so that the signed distance function is smooth in a tubular neighborhood of $\partial\Sigma$, and is linear with slope one along the normals:

Proposition 1. *Let $\partial\Sigma$ be $C^{k,\ell}$ (i.e. kth derivative Holder continuous with exponent ℓ) where $k \geqslant 2$ and $\ell \geqslant 0$ in a neighborhood of $p \in \partial\Sigma$. Then, there exists a neighborhood $T \subset \mathbb{R}^N$ of p such that $d_\Sigma(\mathbf{x})$ is $C^{k,\ell}$ in T. The closest point projection map $P : \mathbb{R}^N \to \partial\Sigma$ is well-defined on T. Furthermore, d_Σ and P satisfy*

$$d_\Sigma(\mathbf{x}) = (\mathbf{x} - P(\mathbf{x})) \cdot n(P(\mathbf{x})) \tag{5}$$

in T. In addition, $d(\mathbf{x})$ satisfies

$$|\nabla d_\Sigma| = 1 \text{ for all } \mathbf{x} \in T, \text{ with the boundary condition } d_\Sigma|_{\mathbf{x}\in\partial\Sigma} = 0. \tag{6}$$

The second important fact we recall is that Laplacian of the signed distance function d_Σ at a point x gives us essentially the mean curvature of the isosurface of d passing through x:

$$\Delta d_\Sigma(\mathbf{x}) = (N-1)H(\mathbf{x}) \tag{7}$$

where $H(\mathbf{x})$ denotes the mean curvature of the level set $\{\xi : f(\xi) = f(\mathbf{x})\}$ at $\mathbf{x}$.

Specializing to the planar (2D) setting, let $\gamma : (-\varepsilon, \varepsilon) \to \mathbb{R}^2$ be a unit speed parametrization of the curve $\partial\Sigma$ around $p \in \partial\Sigma$, with $\gamma(0) = p$ and positive orientation. Let $\kappa(\mathbf{x})$ denote the *curvature* of the curve at $\mathbf{x}$:

$$\gamma_{ss}(0) \cdot n(p) = \kappa(p). \tag{8}$$

Note that curvature κ of the boundary of convex sets is negative according to convention (8).

We may rotate and translate the set Σ so that $p = 0 \in \mathbb{R}^2$ and the outer unit normal $n(0)$ at $p = 0$ is given by the vector $n(0) = (0, -1)$; see Fig. 2 for the setup.

Let $f(x)$ be the smooth function whose graph $(x, f(x))$ describes the interface $\partial\Sigma$ in a neighborhood of the origin. Let us write simply $\kappa(x)$ to denote the curvature $\kappa(x, f(x))$ of $\partial\Sigma$ at $(x, f(x))$. We then have the following relations implied:

$$f(0) = 0, \; f'(0) = 0, \text{ and } f''(0) = -\kappa(0). \tag{9}$$

For the signed distance function $d_\Sigma(x, y)$ to Σ, we drop the Σ in its notation and adopt the convention that $d(x, y) < 0$ if $y < f(x)$ (and hence $d(x, y) > 0$ if $y > f(x)$). In 2D, Eq. (7) reads

$$d_{xx}(x, f(x)) + d_{yy}(x, f(x)) = \kappa(x) \tag{10}$$

on the interface.

The following useful formulas follow immediately from (5) in Proposition 1:

Lemma 1. *For sufficiently small y, we have*

$$d(0, y) = y \tag{11}$$

so that

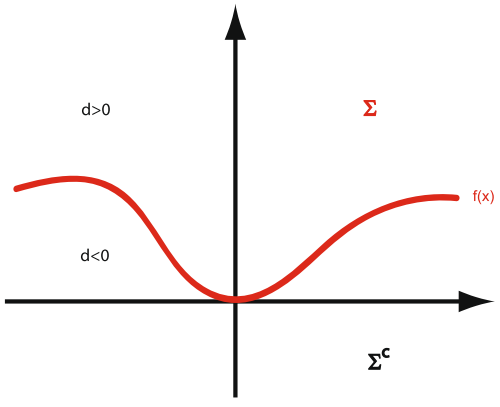

Fig. 2. The setup.

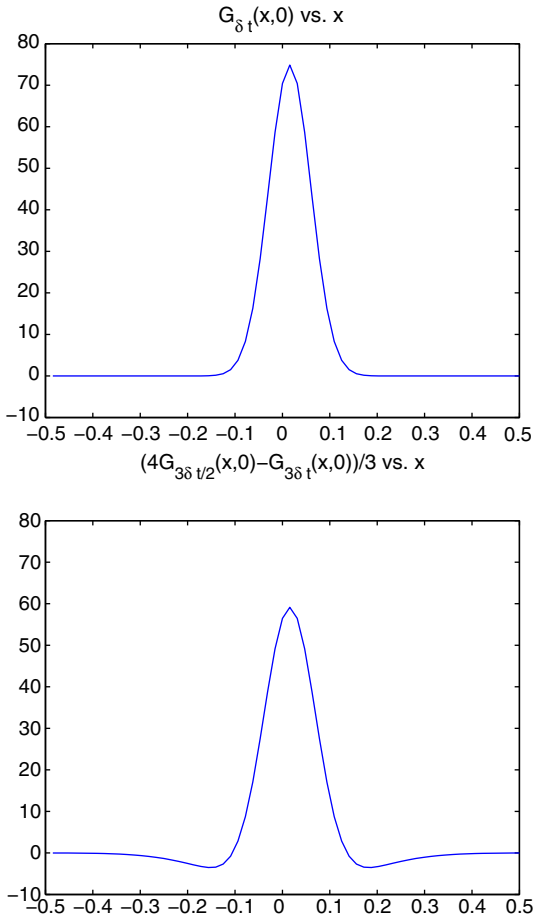

Fig. 3. The convolution kernel involved in the more accurate algorithm for mean curvature motion is not positive; the resulting algorithm therefore may not be monotone.

$$d_y(0,y) = 1, \tag{12}$$

$$\frac{\partial^k}{\partial y^k} d(0,y) = 0 \text{ for } k = 2,3,4,\ldots \tag{13}$$

and

$$d_x(0,y) = 0. \tag{14}$$

Proof. That $d(0,y) = y$ follows from (5), and the y partial derivatives follow from this expression. Then, (14) follows from these and the Eikonal Eq. (6). □

Lemma 2. *The following hold*

$$\frac{\partial^k}{\partial y^k} d_x(0,y) = 0 \text{ for } k = 1,2,3,\ldots \tag{15}$$

for all sufficiently small y.

Proof. Set $A(x,y) := d_x^2(x,y) + d_y^2(x,y)$. Then, by (6) we have

$$A(x,y) \equiv 1 \text{ for all small enough } (x,y). \tag{16}$$

Differentiating (16) w.r.t. x and y, we have

$$\begin{aligned} &\frac{1}{2}\frac{\partial}{\partial x} A(x,y) = d_x(x,y) d_{xx}(x,y) + d_y(x,y) d_{xy}(x,y) \equiv 0, \text{ and} \\ &\frac{1}{2}\frac{\partial}{\partial y} A(x,y) = d_x(x,y) d_{xy}(x,y) + d_y(x,y) d_{yy}(x,y) \equiv 0. \end{aligned} \tag{17}$$

Evaluating the first equality in (17) at $x = 0$ and using (13) and (14) we get

$$d_{xy}(0,y) \equiv 0 \text{ for all small enough } y. \tag{18}$$

Further differentiating (18) with respect to y, we get (15). □

Lemma 3. *The following hold:*

$$d_{xx}(0,0) = \kappa(0), \tag{19a}$$
$$d_{xxy}(0,0) = -\kappa^2(0), \tag{19b}$$
$$d_{xxx}(0,0) = \kappa_x(0). \tag{19c}$$

Proof. Eq. (19a) follows from evaluating (10) at $x = 0$ and (13).

To obtain (19b), we first differentiate (17) with respect to x once again:

$$\frac{1}{2}A_{xx}(x,y) = d_{xx}^2 + d_x d_{xxx} + d_{xy}^2 + d_y d_{xxy} \equiv 0. \tag{20}$$

Evaluating (20) at $(x,y) = (0,0)$ and using 19a, 14, 13 and 15 we get (19b).

To obtain (19c), we differentiate (10) with respect to x:

$$d_{xxx}(x,f(x)) + d_{xxy}(x,f(x))f'(x) + d_{yyx}(x,f(x)) + d_{yyy}(x,f(x))f'(x) \equiv \kappa_x(x). \tag{21}$$

Evaluating (21) at $x = 0$ and using (9) and (15), we get (19c). □

Lemma 4. *The following hold:*

$$d_{xxxy}(0,0) = -3\kappa(0)\kappa_x(0), \tag{22a}$$
$$d_{xxyy}(0,0) = 2\kappa^3(0), \tag{22b}$$
$$d_{xxxx}(0,0) = \kappa_{xx}(0) - 3\kappa^3(0). \tag{22c}$$

Proof. Differentiating (20) with respect to x once again, we obtain

$$\frac{1}{2}A_{xxx}(x,y) = 3d_{xx}d_{xxx} + d_x d_{xxxx} + 3d_{xy}d_{xxy} + d_y d_{xxxy} \equiv 0. \tag{23}$$

Evaluating at $x = 0$ and using 19a, 19c, 14, 15 and 13 we get (22a).

Differentiating (20) this time with respect to y, we get

$$\frac{1}{2}A_{xxy}(x,y) = 2d_{xx}d_{xxy} + d_{xy}d_{xxx} + d_x d_{xxxy} + 2d_{xy}d_{xyy} + d_{yy}d_{xxy} + d_y d_{xxyy} \equiv 0. \tag{24}$$

Evaluating at $x = 0$ and using 19a, 19b, 15, 14 and 13 yields (22b).

Differentiating (21) once more with respect to x, we find

$$d_{xxxx}(x,f) + 2d_{xxxy}(x,f)f' + d_{xxyy}(x,f)(f')^2 + d_{xxy}(x,f)f'' + d_{xxyy}(x,f) + 2d_{xyyy}(x,y)f' + d_{xyyyy}(x,f)(f')^2 + d_{yyy}(x,f)f'' = \kappa_{xx}. \tag{25}$$

Evaluating at $x = 0$ and using 9, 19b, 22b and 13 we get (22c). □

Collecting terms from Lemmas 1 to 4, we arrive at the desired Taylor expansion:

Proposition 2. *The signed distance function $d(x,f(x))$ has the following Taylor expansion at $x = 0$:*

$$d(x,y) = y + \frac{1}{2}\kappa(0)x^2 + \frac{1}{6}\kappa_x(0)x^3 - \frac{1}{2}\kappa^2(0)x^2y + \frac{1}{24}(\kappa_{xx}(0) - 3\kappa^3(0))x^4 - \frac{1}{2}\kappa(0)\kappa_x(0)x^3y + \frac{1}{2}\kappa^3(0)x^2y^2 + O(|\mathbf{x}|^5). \tag{26}$$

We can now substitute the expansion (26) into the convolution integral

$$\int_{\mathbb{R}^2} G_t(\xi,\eta)d(x-\xi, y-\eta)\, d\xi d\eta \tag{27}$$

to get a Taylor expansion for the convolution $(G_t * d)(x,y)$ at $(x,y) = (0,0)$. The terms we need are:

$$(x^2 * G_t)(0,y) = 2t \quad (x^2y * G_t)(0,y) = 2ty \tag{28}$$
$$(x^4 * G_t)(0,y) = 12t^2 \quad (x^2y^2G_t)(0,y) = 2ty^2 + 4t^2 \tag{29}$$

Using these, we arrive at the following expansion:

Proposition 3. *Convolution of the signed distance function d with the Gaussian kernel G_t has the following expansion*

$$(d * G_t)(0,y) = y + \kappa(0)t - \kappa^2(0)yt + \frac{1}{2}(\kappa_{xx}(0) + \kappa^3(0))t^2 + O(t^3) \tag{30}$$

provided that $y = O(t)$.

Remark 1. Because $f'(0) = 0$, we in fact have

$$\kappa_{xx}(0) = \kappa_{ss}(0)$$

where κ_{ss} denotes second derivative of curvature with respect to arc-length. Thus coefficients in the expansion of Proposition 3 can be easily expressed in completely geometric quantities, if desired.

4.2. Expansion at a junction

For convenience, let us introduce the following notation for the 1D Gaussian:

$$g_t(x) = \frac{1}{\sqrt{4\pi t}} e^{-\frac{x^2}{4t}} \text{ so that } G_t(x,y) = g_t(x)g_t(y). \tag{31}$$

Let us record the formulas

$$\int_0^\infty \xi g_t(\xi)\, d\xi = \frac{\sqrt{t}}{\sqrt{\pi}}, \int_0^\infty \xi^2 g_t(\xi)\, d\xi = t, \text{ and } \int_{-\infty}^x \xi g_t(\xi)\, d\xi = -\frac{\sqrt{t}}{\sqrt{\pi}} e^{-\frac{x^2}{4t}}. \tag{32}$$

We now consider the set up where three C^2 curves meet in a triple point located at the origin, such that their tangents have the angles $2\theta_1, 2\theta_2, 2\theta_3 \in (0,\pi)$ between them; see Fig. 4 for an illustration. If we zoom in to the origin, the set up would look like three sectors, as indicated in the right hand side plot of Fig. 4. Hence, we start by writing down explicit formulas for the distance function to a sector.

Let a sector S of *opening angle* 2θ be given as follows:

$$S = \left\{(x,y) \;:\; y < \tan\left(\frac{3\pi}{2} - \theta\right)x \text{ and } y < \tan\left(\frac{3\pi}{2} + \theta\right)x\right\}. \tag{33}$$

See Fig. 5 for an illustration. We will say that the *ridge* of S is the set on the complement of which the signed distance function to S is smooth; it consists of the following three lines:

$$\begin{aligned} \ell_1 &:= \{(x,y) \;:\; x \geqslant 0 \text{ and } y = (\tan\theta)x\}, \\ \ell_2 &:= \{(x,y) \;:\; x \leqslant 0 \text{ and } y = -(\tan\theta)x\}, \\ \ell_3 &:= \{(x,y) \;:\; x = 0 \text{ and } y \leqslant 0\}. \end{aligned} \tag{34}$$

Then, we distinguish the three regions defined by the ridge (as shown in Fig. 5), in each of which the signed distance function is smooth, as follows:

- *Region 1:* $R_1 := \{(x,y) \;:\; y \leqslant (\tan\theta)x \text{ and } x \geqslant 0\}$,
- *Region 2:* $R_2 := \{(x,y) \;:\; y \leqslant -(\tan\theta)x \text{ and } x \leqslant 0\}$, and

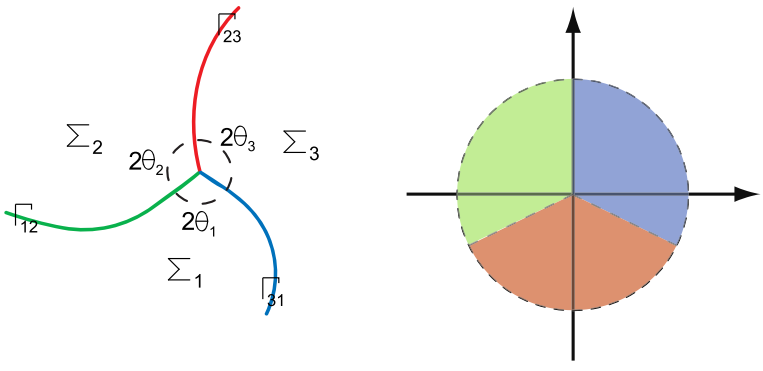

Fig. 4. Let: A triple junction where three C^2 curves meet. Right: When we zoom in on the junction, the three sets (phases) meeting at the triple point can be approximated by an arrangement of sectors. The discussion in Sections 4.2 and 5.3 perturbs from this configuration.

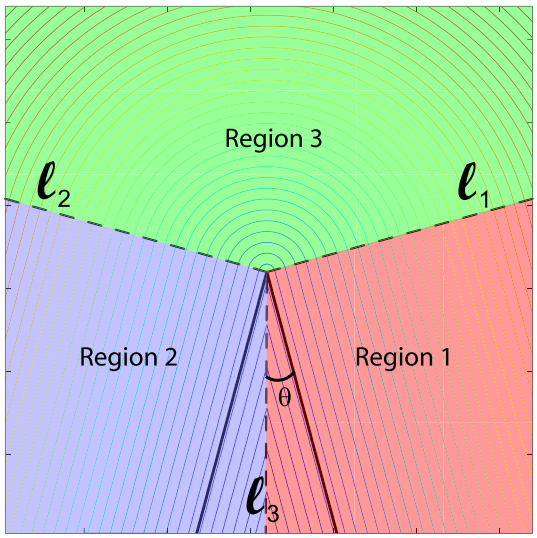

Fig. 5. Distance function to a sector of opening angle 2θ. The boundary of the sector is shown in solid black; the other contours are isocontours of the signed distance function. The *ridge* is the union of the three dashed black lines (denoted ℓ_1, ℓ_2, and ℓ_3) that separate the plane into three regions in each of which the signed distance function is smooth.

- *Region 3:* $R_3 := \{(x,y) \ : \ y \geqslant (\tan\theta)|x|\}$.

It is then easy to see that the signed distance function is given as follows in these regions:

- *Region 1:* $d(x,y) = -x\cos\theta - y\sin\theta$ for $(x,y) \in R_1$,
- *Region 2:* $d(x,y) = x\cos\theta - y\sin\theta$ for $(x,y) \in R_2$, and
- *Region 3:* $d(x,y) = -\sqrt{x^2+y^2}$ for $(x,y) \in R_3$.

See Fig. 5 for illustration of the regions and level curves of the signed distance function. We will compute the Taylor expansion at the origin of the convolution of the signed distance function d to the sector S with the Gaussian kernel.

We start with the constant term in the expansion. To that end, we first note:

$$\int_{R_1} xG_t(x,y)\ dx\ dy = \int_{-\infty}^{0}\int_{0}^{\infty} xG_t(x,y)\ dx\ dy + \int_{0}^{\infty}\int_{\frac{y}{\tan\theta}}^{\infty} xG_t(x,y)\ dx\ dy = \frac{\sqrt{t}}{2\sqrt{\pi}} + \frac{\sqrt{t}}{2\sqrt{\pi}}\frac{\tan\theta}{\sqrt{1+\tan^2\theta}} \tag{35}$$

$$\int_{R_1} yG_t(x,y)\ dx\ dy = \int_{0}^{\infty}\int_{-\infty}^{(\tan\theta)x} yG_t(x,y)\ dy\ dx = -\frac{\sqrt{t}}{2\sqrt{\pi}}\frac{1}{\sqrt{1+\tan^2\theta}} \tag{36}$$

Observing that

$$\begin{aligned}\int_{R_1\cup R_2} d(x',y')G_t(x-x',y-y')\ dx'\ dy'\Big|_{x=0,\ y=0} &= 2\int_{R_1} d(x',y')G_t(x',y')\ dx'\ dy' \\ &= -2\cos\theta\int_{R_1} x'G_t(x',y')\ dx'\ dy' - 2\sin\theta\int_{R_1} y'G_t(x',y')\ dx'\ dy'\end{aligned} \tag{37}$$

and using (35) and (36), we get:

$$\int_{R_1\cup R_2} d(x',y')G_t(x-x',y-y')\ dx'\ dy'\Big|_{x=0,\ y=0} = -\frac{\sqrt{t}}{\sqrt{\pi}}\cos\theta. \tag{38}$$

Next, we calculate

$$\int_{R_3} d(x',y')G_t(x-x',y-y')\ dx'\ dy'\Big|_{x=0,\ y=0} = -\int_{0}^{\pi-2\theta}\int_{0}^{\infty} r^2\frac{1}{4\pi t}e^{-\frac{r^2}{4t}}\ dr\ d\theta = (2\theta-\pi)\frac{\sqrt{t}}{2\sqrt{\pi}} \tag{39}$$

We can now put together (38) and (39) to get the 0th order term in the desired expansion:

$$\int_{\mathbb{R}^2} d(x',y')G_t(x-x',y-y')\ dx'\ dy'\Big|_{x=0,\ y=0} = \frac{\sqrt{t}}{\sqrt{\pi}}\left(\theta - \frac{\pi}{2} - \cos\theta\right). \tag{40}$$

Moving on to the calculation of higher order terms, we first note:

$$\int_{R_1} xyG_t(x,y)\ dx\ dy = \int_0^\infty \int_{-\infty}^{(\tan\theta)x} xyG_t(x,y)\ dx\ dy = \int_0^\infty xg_t(x) \int_{-\infty}^{(\tan\theta)x} yg_t(y)\ dy\ dx = -\frac{t}{\pi(1+\tan^2\theta)}. \tag{41}$$

Also useful is:

$$\int_{R_1} y^2 G_t(x,y)\ dx\ dy = \int_0^\infty g_t(x) \int_{-\infty}^{(\tan\theta)x} y^2 g_t(y)\ dy\ dx = -\frac{t}{\pi}\frac{(\tan\theta)}{(1+\tan^2\theta)} + \frac{(\pi+2\theta)}{2\pi}t. \tag{42}$$

Using (41) and (42) we get

$$\begin{aligned}\int_{R_1\cup R_2} d(x',y')\frac{\partial}{\partial y}G_t(x-x',y-y')\ dx'\ dy'\Big|_{x=0,\ y=0} &= \frac{1}{t}\int_{R_1} d(x,y)yG_t(x,y)\ dx\ dy\\ &= -\frac{1}{t}\int_{R_1}(x\cos\theta + y\sin\theta)\ y\ G_t(x,y)\ dx\ dy\\ &= -\frac{\cos\theta}{t}\int_{R_1} xyG_t(x,y)\ dx\ dy - \frac{\sin\theta}{t}\int_{R_1} y^2 G_t(x,y)\ dx\ dy\\ &= -\frac{1}{\pi}\left(\left(\frac{\pi}{2}+\theta\right)\sin\theta - \cos\theta\right).\end{aligned} \tag{43}$$

We now also calculate

$$\begin{aligned}\int_{R_3} d(x',y')\frac{\partial}{\partial y}G_t(x-x',y-y')\ dx'\ dy'\Big|_{x=0,\ y=0} &= \frac{1}{2t}\int_{R_3} d(x,y)\ y\ G_t(x,y)\ dx\ dy = -\frac{1}{2t}\int_\theta^{\pi-\theta}\int_0^\infty r^3\sin\theta\frac{e^{-\frac{r^2}{4t}}}{4\pi t}\ dr\ d\theta\\ &= -\frac{2\cos\theta}{\pi}.\end{aligned} \tag{44}$$

Putting together (43) and (44) we find

$$\int_{\mathbb{R}^2} d(x',y')\frac{\partial}{\partial y}G_t(x-x',y-y')\ dx'\ dy'\Big|_{x=0,\ y=0} = -\frac{1}{\pi}\left(\left(\frac{\pi}{2}+\theta\right)\sin\theta + \cos\theta\right). \tag{45}$$

which is the coefficient of y in the expansion we seek. Noting that the coefficient of x must be zero on symmetry grounds, we next consider quadratic terms. To that end, first compute:

$$\begin{aligned}\int_{R_1\cup R_2} d(x',y')\frac{\partial^2}{\partial x^2}G_t(x-x',y-y')\ dx'\ dy'\Big|_{x=0,y=0} &= 2\int_{R_1} d(x',y')\frac{\partial^2}{\partial x^2}G_t(x',y')\ dx'\ dy'\\ &= -2\int_0^\infty\int_{-\infty}^{x\tan\theta}(x'\cos\theta + y'\sin\theta)\frac{\partial^2}{\partial x^2}G_t(x',y')\ dy'\ dx'\\ &= -\frac{1}{2\sqrt{\pi t}}(1+2\sin\theta)\cos\theta.\end{aligned} \tag{46}$$

Then,

$$\begin{aligned}\int_{R_3} d(x',y')\frac{\partial^2}{\partial x^2}G_t(x-x',y-y')\ dx'\ dy'\Big|_{x=0,y=0} &= -\int_{R_3}\sqrt{(x')^2+(y')^2}\frac{\partial^2}{\partial x^2}G_t(x',y')\ dx\ dy\\ &= \int_0^\infty\int_\theta^{\pi-\theta}\frac{r^2}{16\pi t^3}(2t - r^2\cos^2\theta')e^{-\frac{r^2}{4t}}\ d\theta'\ dr\\ &= \frac{1}{8\sqrt{\pi t}}(3\sin(2\theta) + 2\theta - \pi).\end{aligned} \tag{47}$$

Putting (46) and (47) together, we find

$$\int_{\mathbb{R}^2} d(x',y')\frac{\partial^2}{\partial x^2}G_t(x-x',y-y')\ dx'\ dy'\Big|_{x=0,y=0} = \frac{1}{8\sqrt{\pi t}}(2\theta - 4\cos\theta - \sin(2\theta) - \pi). \tag{48}$$

which would be the coefficient of x^2 in the expansion. Noting once again on symmetry grounds that the coefficient of xy must be zero, it remains only to compute the coefficient of y^2, as follows:

$$\int_{R_1\cup R_2} d(x',y')\frac{\partial^2}{\partial y^2}G_t(x-x',y-y')\ dx'\ dy'\Big|_{x=0,y=0} = -2\int_{R_1}(x'\cos\theta + y'\sin\theta)\frac{\partial^2}{\partial y^2}G_t(x',y')\ dx'\ dy' = \frac{1}{2\sqrt{\pi t}}\sin(2\theta) \tag{49}$$

and

$$\int_{R_3} d(x',y')\frac{\partial^2}{\partial y^2}G_t(x-x',y-y')\,dx'\,dy'\Big|_{x=0,y=0} = \frac{1}{8\sqrt{\pi t}}(2\theta - 3\sin(2\theta) - \pi). \tag{50}$$

Putting together (49) and (50) we obtain

$$\int_{\mathbb{R}^2} d(x',y')\frac{\partial^2}{\partial y^2}G_t(x-x',y-y')\,dx'\,dy'\Big|_{x=0,y=0} = \frac{1}{8\sqrt{\pi t}}(\sin(2\theta) + 2\theta - \pi) \tag{51}$$

which is the coefficient of y^2. Additionally, note that since the signed distance function d to any set is Lipschitz, we have

$$\left|\frac{\partial^k}{\partial x_{j_1}\cdots\partial x_{j_k}}(G_t * d)(x)\right| \leqslant C_k t^{\frac{1-k}{2}} \tag{52}$$

for $k = 1, 2, 3, \ldots$, where the constants C_k are universal.

Finally, putting the formulas 40, 45, 48, 51 and the bound (52) together with the following Taylor expansion at the origin

$$(G_t * d)(\mathbf{x}) = c_{00}(t) + c_{10}(t)\,x + c_{01}(t)\,y + c_{20}(t)\,x^2 + c_{11}(t)\,xy + c_{02}(t)y^2 + \cdots \tag{53}$$

we arrive at the following.

Proposition 4. *Convolution of the signed distance function d for the sector (33) with a Gaussian kernel satisfies the following Taylor expansion:*

$$\begin{aligned}\int_{\mathbb{R}^2} d(x',y')G_t(x-x',y-y')\,dx'\,dy' = {}& \frac{\sqrt{t}}{\sqrt{\pi}}\left(\theta - \frac{\pi}{2} - \cos\theta\right) - \frac{1}{\pi}\left(\left(\frac{\pi}{2}+\theta\right)\sin\theta + \cos\theta\right)y \\ & + \frac{1}{\sqrt{t}}\frac{1}{16\sqrt{\pi}}(2\theta - 4\cos\theta - \sin 2\theta - \pi)x^2 + \frac{1}{\sqrt{t}}\frac{1}{16\sqrt{\pi}}(\sin 2\theta + 2\theta - \pi)y^2 + O\left(\frac{|\mathbf{x}|^3}{t}\right).\end{aligned} \tag{54}$$

We now turn to obtaining the analogous expansion at a corner point of an open set Σ whose boundary at the corner point consists of the meeting of two C^2 arcs, namely Γ_1 and Γ_2. Let 2θ denote the (interior) angle formed by these curves at the junction. Assume that $\theta < \frac{\pi}{2}$. Also, assume that Σ has been rotated and translated if necessary so that the corner point is at the origin, and $\Sigma \cap B_r(0)$ is contained in the lower half plane for small enough $r > 0$, and its boundary curves Γ_1 and Γ_2 make angles of θ with the axis $\{y \leqslant 0\}$ as shown in Fig. 6.

Consider the *approximating sector S* to the set Σ at the origin. More precisely, S is the sector the boundary curves of which are tangent to those of Σ at the origin, so that in particular we have

$$\limsup_{r\to 0^+}\frac{H(\partial S \cap B_r(0),\ \partial\Sigma \cap B_r(0))}{r^2} < \infty \tag{55}$$

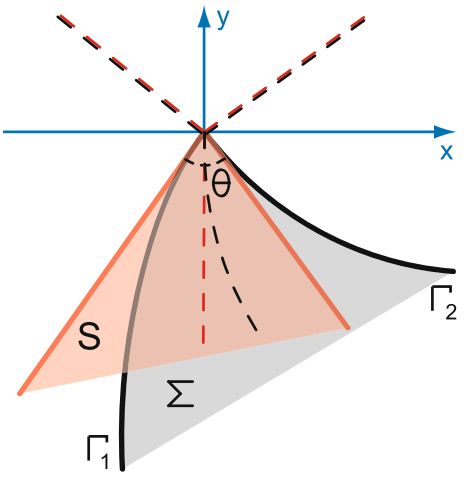

Fig. 6. A set Σ with a corner on its boundary, and the approximating sector S at the corner. The *ridge* for the sector is shown in red by the dashed lines; that of the set Σ is shown in black dashed curves that are tangent to the red dashed lines at the corner.

where $H(\cdot,\cdot)$ denotes the Hausdorff distance. See Fig. 6 for an illustration. A consequence of (55) is

$$\limsup_{r\to 0^+} \frac{|B_r(0)\cap(\Sigma\cap S)|}{r^3} < \infty. \tag{56}$$

Let d denote the signed distance function to Σ, and let $\tilde{d}$ denote the signed distance function to the sector S. By (55), we have

$$|d(\mathbf{x}) - \tilde{d}(\mathbf{x})| = O(|\mathbf{x}|^2) \text{ as } \mathbf{x}\to\mathbf{0}. \tag{57}$$

Therefore,

$$|d * G_t(\mathbf{0}) - \tilde{d} * G_t(\mathbf{0})| \leqslant \int_{\mathbb{R}^2} \left|(d-\tilde{d})(-\mathbf{x})|G_t(\mathbf{x})\, d\mathbf{x} \leqslant \int_{\mathbb{R}^2} O(|\mathbf{x}|^2)G_t(\mathbf{x})\, d\mathbf{x} = O(t) \text{ as } t\to 0. \tag{58}$$

Moreover,

$$\left|\frac{\partial}{\partial x_j}(d*G_t)(\mathbf{0}) - \frac{\partial}{\partial x_j}(\tilde{d}*G_t)(\mathbf{0})\right| = \left|\left(d*\partial_{x_j}G_t\right)(\mathbf{0}) - \left(\tilde{d}*\partial_{x_j}G_t\right)(\mathbf{0})\right| \leqslant \int_{\mathbb{R}^2} O(|\mathbf{x}|^2)\left|\partial_{x_j}G_t\right| d\mathbf{x} = O(\sqrt{t}) \text{ as } t\to 0. \tag{59}$$

We also need to estimate second derivatives of $d(\mathbf{x})$. To that end, we first note the following easy lemma:

Lemma 5. *The ridge of the signed distance function to Σ in a neighborhood of the origin consists of three simple C^2 arcs γ_1, γ_2 and γ_3 that touch each other only at the origin. The arcs γ_1 and γ_2 coincide with the lines ℓ_1, ℓ_2 that constitute part of the ridge of the approximating sector (see (34)). The third arc, γ_3, is tangent to ℓ_3 at the origin so that in particular*

$$H(\gamma_3\cap B_r(\mathbf{0}),\ \ell_3\cap B_r(\mathbf{0})) = o(r) \text{ as } r\to 0 \tag{60}$$

where $H(\cdot,\cdot)$ denotes the Hausdorff distance. See Fig. 6.

Proof. The lemma is easy to establish by use of Proposition 1 and the implicit function theorem. □

Lemma 5 shows that near the origin, ∇d and $\nabla\tilde{d}$ disagree at $O(1)$ level on only a thin set. Based on this observation, it is easy to establish the following estimate:

$$\left|\frac{\partial^2}{\partial x_i\partial x_j}(d*G_t)(0,0) - \frac{\partial^2}{\partial x_i\partial x_j}(\tilde{d}*G_t)(0,0)\right| \leqslant \int_{\mathbb{R}^2}\left|\frac{\partial}{\partial x_i}(d-\tilde{d})(x',y')\frac{\partial}{\partial x_j}G_t(x',y')\right| dx'\, dy' = O(1) \tag{61}$$

that holds for any $i,j\in\{1,2\}$.

Since $G_t * d$ is a C^∞ function for any $t>0$, it has a Taylor expansion of the form (53). Indeed, expansion (54) that applies to $\tilde{d}$, together with the bounds 52, 56 and 61 give the following expansion for d:

Proposition 5. *Convolution with a Gaussian kernel of the signed distance function d of the domain Σ at its corner located at the origin formed by the meeting of two C^2 arcs Γ_1 and Γ_2 as in Fig. 6 satisfies the following Taylor expansion:*

$$\begin{aligned}\int_{\mathbb{R}^2} d(x',y')G_t(x-x',y-y')\, dx'\, dy' = {}& \frac{\sqrt{t}}{\sqrt{\pi}}\left(\theta-\frac{\pi}{2}-\cos\theta+C_1(t)\right)+C_2(t)\, x-\frac{1}{\pi}\left(\left(\frac{\pi}{2}+\theta\right)\sin\theta+\cos\theta+C_3(t)\right)y\\ &+\frac{1}{\sqrt{t}}\frac{1}{16\sqrt{\pi}}(2\theta-4\cos\theta-\sin 2\theta-\pi+C_3(t))x^2\\ &+\frac{1}{\sqrt{t}}\frac{1}{16\sqrt{\pi}}(\sin 2\theta+2\theta-\pi+C_4(t))y^2+\frac{1}{\sqrt{t}}C_5(t)xy+\text{H.O.T.}\end{aligned} \tag{62}$$

The coefficients $C_j(t)$ satisfy $C_j(t)=O(\sqrt{t})$ as $t\to 0^+$.

5. Algorithms

In this section, we utilize the expansions obtained in the previous sections to describe a number of new algorithms for interfacial motion.

5.1. Warm up: curvature motion

In this section, we describe several algorithms for simulating the motion of an interface with normal speeds of the form

$$v_n = \kappa + S(\mathbf{x}) \tag{63}$$

where $S:\mathbb{R}^2\to\mathbb{R}$ is a given function.

We start with the following algorithm for the slightly generalized curvature motion (63), which is very easily obtained from expansion (30).

Algorithm 1. Given the initial set Σ_0 through its signed distance function $d_0(x)$ and a time step size $\delta t > 0$, generate the sets Σ_j via their signed distance functions $d_j(x)$ at subsequent discrete times $t = j(\delta t)$ by alternating the following steps:

1. Form the function
$$A(\mathbf{x}) := G_{\delta t} * d_j + (\delta t) S(\mathbf{x}). \tag{64}$$
2. Construct the signed distance function d_{j+1} by
$$d_{j+1}(\mathbf{x}) = \mathbf{Redist}(A). \tag{65}$$

It is easy to see that the algorithm is *monotone*, since each of its steps preserves order. Consistency of one step of the algorithm with the desired motion (63) is immediate from (30), which shows that the zero level set of the convolution $d * G_{\delta t}$ crosses the y-axis at

$$y = -\kappa(\delta t) - S(0,0)(\delta t) + \text{H.O.T.} \tag{66}$$

leading to the advertised motion. Furthermore, we can read off the form of the leading order truncation error at every time step by substituting (66) into expansion (30) to find:

$$\text{Error} = \left(\frac{1}{2}\kappa_{ss} + \frac{3}{2}\kappa^3 + \kappa^2 S\right)(\delta t)^2 + O((\delta t)^3). \tag{67}$$

We now turn our attention to designing more accurate versions of algorithms such as the one above. For instance, we can utilize once again the expansion in (30) to get more accurate evaluation of curvature κ at every time step using a Richardson extrapolation-like procedure. This incurs no additional computational cost whatsoever, since the only modification to algorithm (64) and (65) is to replace the convolution kernel in step (64) by a linear combination of two Gaussians.

Algorithm 2. Given the initial set Σ_0 through its signed distance function $d_0(\mathbf{x})$ and a time step size $\delta t > 0$, generate the sets Σ_j via their signed distance functions $d_j(\mathbf{x})$ at subsequent discrete times $t = j(\delta t)$ by alternating the following steps:

1. Form the function
$$A(\mathbf{x}) := K_{\delta t} * d_j + (\delta t) S(\mathbf{x}) \tag{68}$$
where K_t is the kernel:
$$K_t = \frac{1}{3}\left(4G_{\frac{3}{2}t} - G_{3t}\right). \tag{69}$$
2. Construct the signed distance function d_{j+1} by
$$d_{j+1} = \mathbf{Redist}(A). \tag{70}$$

Indeed, from expansion (30) we see that

$$K_{\delta t} * d(0,y) = y + (\kappa + S)(\delta t) - \kappa^2 y(\delta t) + O((\delta t)^3) \tag{71}$$

leading to

$$\text{Error} = (\kappa^3 + \kappa^2 S)(\delta t)^2 + O((\delta t)^3) \tag{72}$$

which suggests better controlled accuracy in (implicit) evaluation of curvature at every time step by eliminating the dependence of the leading order error term on derivatives of curvature (and improving the constant of the remaining term). Numerical experiments with algorithm (68)–(70) indeed lead to more accurate results in practice than algorithm (64) and (65); evidence to this effect is presented in Section 6.1. However, we should note that although the original algorithm (64) and (65) is obviously monotone due to the positivity of its convolution kernel, we cannot say the same for algorithm (68)–(70) since *its* convolution kernel K_t is no longer positive; see Fig. 3. Monotonicity cannot be guaranteed for the high order in time schemes discussed below, either.

We now turn to designing *higher order in time* versions of (64) and (65), or (68)–(70), at the expense of increasing slightly the number of convolution or redistancing operations involved at each time step (which are, although fast, still the most computationally intensive tasks of our algorithms). For example, the following algorithm requires three convolution and two redistancing operations per time step, but formally has quadratic convergence rate in t:

Algorithm 3. Given the initial set Σ_0 through its signed distance function $d_0(\mathbf{x})$ and a time step size $\delta t > 0$, generate the sets Σ_j via their signed distance functions $d_j(\mathbf{x})$ at subsequent discrete times $t = j(\delta t)$ by alternating the following steps:

1. Form the functions

$$\begin{aligned} A_1(\mathbf{x}) &:= K_{\frac{\delta t}{2}} * d_j + \frac{(\delta t)}{2} S(\mathbf{x}) \\ d_{j+\frac{1}{2}}(\mathbf{x}) &:= \mathbf{Redist}(A_1) \\ A_2(\mathbf{x}) &:= K_{\frac{\delta t}{2}} * d_{j+\frac{1}{2}} + \frac{(\delta t)}{2} S(\mathbf{x}) \\ A_3(\mathbf{x}) &:= K_{\delta t} * d_j + (\delta t) S(\mathbf{x}). \end{aligned} \tag{73}$$

where K_t is one of the two kernels:

$$K_t = G_t \text{ or } K_t = \frac{1}{3}\left(4G_{\frac{3}{2}t} - G_{3t}\right). \tag{74}$$

2. Construct the signed distance function d_{j+1} by

$$d_{j+1} = \mathbf{Redist}(2A_2 - A_3). \tag{75}$$

By using a *multistep* strategy, we can reduce the per time step cost of the high order in time algorithm (73)–(75) above to just two convolution and one redistancing operations. Indeed, the variant below still has formally quadratic convergence rate in time:

Algorithm 4. Multistep version of (73)–(75):

1. Form the functions

$$\begin{aligned} A_1(\mathbf{x}) &:= K_{2\delta t} * d_{j-1} + 2(\delta t) S(\mathbf{x}) \\ A_2(\mathbf{x}) &:= K_{\delta t} * d_j + (\delta t) S(\mathbf{x}) \end{aligned} \tag{76}$$

2. Construct the signed distance function d_{j+1} by

$$d_{j+1} = \mathbf{Redist}\frac{1}{3}(4A_2 - A_1). \tag{77}$$

The algorithms in this section have been discussed in $\mathbb{R}^2$ for curves, but they generalize verbatim to hypersurfaces in $\mathbb{R}^N$. In this case, they approximate the flow of hypersurfaces under normal speeds of the form

$$v_n = (N-1)H + S(\mathbf{x}) \tag{78}$$

where H denotes the mean curvature of the interface, and $S : \mathbb{R}^N \to \mathbb{R}$ is a given function.

5.2. Motion by $f(\kappa)$

In this section we present *unconditionally monotone* schemes for propagating interfaces with normal speeds given by

$$v_n = f(\kappa) \tag{79}$$

where $f : \mathbb{R} \to \mathbb{R}$ is an odd, increasing, Lipschitz function with constant L_f.

For any constant $M > 0$, we consider the following algorithm:

Algorithm 5. Given the initial set Σ_0 through its signed distance function $d_0(\mathbf{x})$ and a time step size $\delta t > 0$, generate the sets Σ_j via their signed distance functions $d_j(\mathbf{x})$ at subsequent discrete times $t = j(\delta t)$ by alternating the following steps:

1. Form the function

$$A(\mathbf{x}) := d_j + (\delta t) f\left(\frac{1}{M(\delta t)}\left\{G_{M(\delta t)} * d_j - d_j\right\}\right) \tag{80}$$

2. Construct distance function d_{j+1} by

$$d_{j+1}(\mathbf{x}) = \mathbf{Redist}(A). \tag{81}$$

At the jth step of the algorithm, the set Σ_j can be recovered if desired through the relation

$$\Sigma_j = \{\mathbf{x} \ : \ d_j(\mathbf{x}) > 0\}.$$

Consistency of this algorithm is easy to verify on a C^2 curve using the expansion (30) in Proposition 3. Indeed, using (30) together with (11), we have

$$\frac{1}{M(\delta t)}\{G_{M(\delta t)} * d - d\} = \kappa + O(\delta t). \tag{82}$$

Since f is Lipschitz, we therefore also have

$$f(\kappa) = f\left(\frac{1}{M(\delta t)}\{G_{M(\delta t)} * d - d\}\right) + O(\delta t). \tag{83}$$

Once again using (11) in conjunction with (80), we see that the 0-level set of

$$d + (\delta t) f\left(\frac{1}{M(\delta t)}\{G_{M(\delta t)} * d - d\}\right) \tag{84}$$

has moved with the desired speed in the normal direction; we also see that the scheme is first order accurate in time (though higher order in time versions may be possible as in Section 5.1), and the constant in the error term depends on M as well as the Lipschitz constant of f. In addition to this consistency result, we have the following monotonicity property:

Proposition 6. *If $M \geqslant L_f$, then algorithm (80) and (81) is monotone for any choice of time step size $\delta t > 0$.*

Proof. Let Σ_1 and Σ_2 be two sets satisfying $\Sigma_1 \subset \Sigma_2$. Let $d_1(x)$ and $d_2(x)$ be the signed distance functions to Σ_1 and Σ_2, respectively. Then, first of all,

$$d_1(x) \leqslant d_2(x) \text{ for all } x. \tag{85}$$

Using the same notation as in the description of the algorithm, let

$$\begin{aligned} A_1(x) &= d_1(x) + (\delta t) f\left(\frac{1}{M(\delta t)}\{G_{M(\delta t)} * d_1 - d_1\}\right) \\ A_2(x) &= d_2(x) + (\delta t) f\left(\frac{1}{M(\delta t)}\{G_{M(\delta t)} * d_2 - d_2\}\right). \end{aligned} \tag{86}$$

Then, just calculate:

$$\begin{aligned} A_2 - A_1 &= (d_2 - d_1) + (\delta t)\left\{f\left(\frac{1}{M(\delta t)}\{G_{M(\delta t)} * d_2 - d_2\}\right) - f\left(\frac{1}{M(\delta t)}\{G_{M(\delta t)} * d_1 - d_1\}\right)\right\} \\ &\geqslant (d_2 - d_1) + (\delta t)\left\{f\left(\frac{1}{M(\delta t)}\{G_{M(\delta t)} * d_2 - d_2\}\right) - f\left(\frac{1}{M(\delta t)}\{G_{M(\delta t)} * d_2 - d_1\}\right)\right\} \end{aligned}$$

(where we used $d_2 \geqslant d_1$ and that f is increasing)

$$\geqslant (d_2 - d_1) - (\delta t)\frac{1}{M(\delta t)} L_f (d_2 - d_1)$$

(where we used the hypothesis that f is Lipschitz)

$$\geqslant 0. \tag{87}$$

(where we used the hypothesis on M).

This verifies monotonicity of the first step of the algorithm; that of the second is obvious as already noted. □

As in Section 5.1, the algorithm of this section generalizes almost verbatim to the flow of hypersurfaces in $\mathbb{R}^N$, where it can generate motions with normal speed

$$v_n = f(H) \tag{88}$$

where H denotes the mean curvature of the interface. We note that other generalizations, for example explicit spatial and time dependence in f, i.e. $f = f(\cdot, \mathbf{x}, t)$, are also easily possible.

5.3. Multiple junctions

In this section, we describe an algorithm based on the signed distance function for simulating the motion of multiple junctions under curvature motion and subject to the symmetric (i.e. 120°) Herring conditions [21]. This is an important application that comes up in materials science, e.g. in simulating the grain boundary motion in polycrystals [26]. We will

use the expansions in Section 4.2 to justify the algorithm and estimate the local truncation error. Our algorithm has the following form:

Algorithm 6. Given the initial sets $\Sigma_0^1, \ldots, \Sigma_0^m$ through their signed distance functions $d_0^1(\mathbf{x}), \ldots, d_0^m(\mathbf{x})$ as well as a time step size $\delta t > 0$, generate the sets $\Sigma_j^1, \ldots, \Sigma_j^m$ via their signed distance functions $d_j^1(\mathbf{x}), \ldots, d_j^m(\mathbf{x})$ at subsequent discrete times $t = j(\delta t)$ by alternating the following steps:

1. Form the convolutions

$$L_j^k := K_{\delta t} * d_j^k \tag{89}$$

for $k = 1, \ldots, m$ where K_t is one of the kernels:

$$K_t = G_t \text{ or } K_t = \frac{1}{3}\left(4G_{\frac{3}{2}t} - G_{3t}\right).$$

2. Construct the signed distance functions d_{j+1}^k for $k = 1, \ldots, m$ according to

$$d_{j+1}^k = \textbf{Redist}\left(L_j^k - \max\{L_j^\ell \ : \ \ell \neq k\}\right). \tag{90}$$

The *reassignment* step (90) of the algorithm stems from the $t \to \infty$ limit of gradient descent on the multiwell potential that constitutes the nonlinear, pointwise term in vectorial phase-field energies such as the ones in [3,6]. It is identical to the reassignment step in [25].

This algorithm differs from the one proposed in [25] in important ways. First of all, the authors in [25] utilize redistancing (construction of the signed distance function) optionally, only as a means to prevent the level set function from becoming too steep or too flat – this is the standard role of redistancing in level set computations, as used even in two-phase flows, and is typically employed only occasionally during the flow (once per a large number of time steps). Indeed, the authors state explicitly that as long as the level set does not become too steep or too flat, any level set representation can be used. However, in order to get the desired Herring angle (i.e. boundary) conditions at junctions, it is absolutely essential to make sure that the profile of level sets representing the various phases be the same near the junctions. In other words, using arbitrary (even if not too flat, not too steep) level sets to represent the phases as is suggested in [25] will lead to $O(1)$ errors in the angles at the junctions; this simple fact can be easily understood by considering what happens to an arrangement of three phases where one of the phases is represented by a level set function half as steep as the other two; Fig. 7 shows a numerical experiment verifying this effect. (To be fair, authors of [25] eventually propose an "Algorithm B" that redistances at every time step, but claim this is only to keep the level set from becoming too steep or too flat. In reality, level sets do not degenerate during the multiphase motion as rapidly as is suggested by the authors – this is not the correct reason for redistancing).

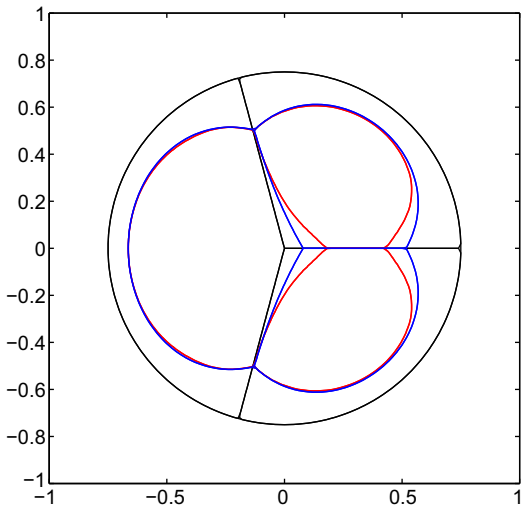

Fig. 7. The black curve shows the initial phase configuration. The red curve is the inconsistent approximation resulting from unequal level set representation of the three phases around the junction, as in "Algorithm A" of [25], after a few iterations; the inconsistency arises even though the level sets have not become too steep or too flat during the short evolution. The blue curve is the algorithm proposed in this paper, which stably preserves the correct angle condition at the junction.

Furthermore, the algorithms for multiple junctions proposed in [25] and utilized in the more recent application paper [42], or in the related work [37], all evolve the level set functions representing the phases via the standard level set formulation of curvature motion. The algorithm proposed here uses convolution (with a Gaussian or more accurate kernel) at every step to evolve the different phases, and therefore maintains the unconditional stability of the two-phase case. It should be pointed out that various other numerical algorithms have been proposed for computing the motion of networks of junctions, including front tracking, level set, and phase field based methods; see e.g. [5,7,6,3,19,44,38,23,16] and their references. We repeat that, in this context of networks as in others, front tracking based methods do not allow painless treatment of topological changes and are therefore hard to extend to three dimensions; phase field based methods require the spatial resolution of a rapid transition layer and involve the small parameter describing the thickness of this layer as a stiffness parameter; and standard level set based methods require the solution of highly nonlinear, degenerate PDE.

We now provide some justification for algorithm (89) and (90), focusing on the case $K_t = G_t$ for simplicity. First, note that the algorithm moves interfaces with motion by curvature *away* from the triple junction; this is immediate based on expansion (30) and the definition of the algorithm. The more interesting point is its behavior at a triple junction, which we now focus on. Our goal is to establish that the scheme not only preserves but in fact *imposes* symmetric (120°) Herring angle conditions at the junction. An analogous calculation for the thresholding based precursor of our algorithm has been carried out in [30]. The argument presented below is a bit more abstract and less explicit, as the representation of the interface we deal with, namely the signed distance function, is not as explicitly given in terms of the interface as a characteristic function is.

Consider a triple junction at the origin formed by the meeting of three C^2 curves Γ_{12}, Γ_{23}, and Γ_{31}. Let the three curves Γ_j constitute the boundaries of three phases Σ_1, Σ_2, and Σ_3 as shown in Fig. 4. Let the angle subtended at the triple junction by phase Σ_j be $2\theta_j$, with $j = 1, 2, 3$. We will assume that θ_j are in a small enough neighborhood of $\frac{\pi}{3}$. Let us also assume that the configuration has been rotated if necessary so that Σ_1 is contained in the lower half plane, and its boundary curves make angles of θ_1 with the axis $\{y \leqslant 0\}$ as shown in Fig. 4. Let d^1, d^2, and d^3 denote the signed distance functions of Σ_1, Σ_2, and Σ_3, respectively. For convenience, let us define the following functions:

$$\begin{aligned} A(\theta) &= \frac{1}{\sqrt{\pi}}\left(\theta - \frac{\pi}{2} - \cos\theta\right) \\ B(\theta) &= -\frac{1}{\pi}\left(\left(\frac{\pi}{2} + \theta\right)\sin\theta + \cos\theta\right) \\ Q_1(\theta) &= \frac{1}{16\sqrt{\pi}}(2\theta - 4\cos\theta - \sin 2\theta - \pi) \\ Q_2(\theta) &= \frac{1}{16\sqrt{\pi}}(\sin 2\theta + 2\theta - \pi). \end{aligned} \tag{91}$$

Using (62) in Proposition 5, we then have the following Taylor expansions for the functions $d^j * G_t$ at the origin:

$$\begin{aligned} d^1 * G_t &= A(\theta_1)\sqrt{t} + B(\theta_1)y + \frac{1}{\sqrt{t}}Q_1(\theta_1)x^2 + \frac{1}{\sqrt{t}}Q_2(\theta_1)y^2 + C_1^1(t)\sqrt{t} + C_2^1(t)x + C_3^1(t)y + \text{H.O.T.} \\ d^2 * G_t &= A(\theta_2)\sqrt{t} + B(\theta_2)(\sin(\theta_1+\theta_2)x + \cos(\theta_1+\theta_2)y) + \frac{1}{\sqrt{t}}Q_1(\theta_2)(\cos(\theta_1+\theta_2)x - \sin(\theta_1+\theta_2)y)^2 \\ &\quad + \frac{1}{\sqrt{t}}Q_2(\theta_2)(\sin(\theta_1+\theta_2)x + \cos(\theta_1+\theta_2)y)^2 + C_1^2(t)\sqrt{t} + C_2^2(t)x + C_3^2(t)y + \text{H.O.T.} \\ d^3 * G_t &= A(\theta_3)\sqrt{t} + B(\theta_3)(-\sin(\theta_1+\theta_3)x + \cos(\theta_1+\theta_3)y) + \frac{1}{\sqrt{t}}Q_1(\theta_3)(\cos(\theta_1+\theta_3)x + \sin(\theta_1+\theta_3)y)^2 \\ &\quad + \frac{1}{\sqrt{t}}Q_2(\theta_3)(-\sin(\theta_1+\theta_3)x + \cos(\theta_1+\theta_3)y)^2 + C_1^3(t)\sqrt{t} + C_2^3(t)x + C_3^3(t)y + \text{H.O.T.} \end{aligned} \tag{92}$$

where $C_j^i(t) = O(\sqrt{t})$ as $t \to 0^+$. Expansions for $d^2 * G_t$ and $d^3 * G_t$ were obtained from that of $d^1 * G_t$ simply by the appropriate rotations. From (92) we can immediately read off the following:

1. If $\theta_1 = \theta_2 = \theta_3 = \frac{\pi}{3}$, then the triple junction moves with speed at most $O(1)$ in a time step of size δt.
2. If θ_j differs from $\frac{\pi}{3}$ by $O(1)$ for any $j \in \{1, 2, 3\}$, then the triple junction moves with speed at least $O(\delta t^{-\frac{1}{2}})$ in a time step of size δt.

These properties suggest that the motion of the junction has the expected behavior under algorithm (89) and (90). Indeed, the second point suggests that if the Herring angle conditions are not satisfied initially, then the numerical solution will "adjust" the location of the triple junction at a fast time scale (infinitely fast as $\delta t \to 0^+$). We will now see if this fast adjustment takes the angles towards or away from the Herring condition. To that end, assume that the angles θ_1 and θ_2 at the beginning of a time step with the algorithm are not too far from $\frac{\pi}{3}$. We can solve for the new coordinates (x_*, y_*) of the triple junction *after* the time step using (92). Indeed, the three surfaces in (92) intersect in three curves, which in return intersect in a point in the (x, y)-plane, whose coordinates can be found by solving

$$(d^1 * G_t)(x_*, y_*) = (d^2 * G_t)(x_*, y_*) = (d^3 * G_t)(x_*, y_*) \tag{93}$$

for x_* and y_*.

From (92) we see that (x_*, y_*) are given by

$$\begin{aligned} x_* &= -\frac{\sqrt{3}}{3}\frac{A'\left(\frac{\pi}{3}\right)}{B\left(\frac{\pi}{3}\right)}\sqrt{t}(2\theta_2 + \theta_1 - \pi) + \text{H.O.T.} = \frac{2\sqrt{3\pi}(2+\sqrt{3})}{6+5\pi\sqrt{3}}\sqrt{t}(2\theta_2 + \theta_1 - \pi) + \text{H.O.T.} \\ y_* &= -\frac{A'\left(\frac{\pi}{3}\right)}{B\left(\frac{\pi}{3}\right)}\sqrt{t}\left(\theta_1 - \frac{\pi}{3}\right) + \text{H.O.T.} = \frac{6\sqrt{\pi}(2+\sqrt{3})}{6+5\pi\sqrt{3}}\sqrt{t}\left(\theta_1 - \frac{\pi}{3}\right) + \text{H.O.T.} \end{aligned} \tag{94}$$

Differentiating (92) and evaluating the result at the new junction location (x_*, y_*), we see that the *normals* to the three curves at the junction are given by

$$\begin{aligned} N_{12} &:= \nabla(G_t * (d_1 - d_2))(x_*, y_*), \\ N_{23} &:= \nabla(G_t * (d_2 - d_3))(x_*, y_*), \\ N_{31} &:= \nabla(G_t * (d_3 - d_1))(x_*, y_*). \end{aligned} \tag{95}$$

up to high order terms; see Fig. 8. The two angles (θ_1, θ_2) between the curves at the beginning of the time step get sent to a new pair of angles $(\tilde{\theta}_1, \tilde{\theta}_2)$ at the end of the time step. These new angles can be expressed using (95) as

$$\begin{aligned} \tilde{\theta}_1 &= \frac{1}{2}\cos^{-1}\left(\frac{N_{31}\cdot N_{12}}{\|N_{31}\|\ \|N_{12}\|}\right) + \text{H.O.T.}, \text{ and} \\ \tilde{\theta}_2 &= \frac{1}{2}\cos^{-1}\left(\frac{N_{12}\cdot N_{23}}{\|N_{12}\|\ \|N_{23}\|}\right) + \text{H.O.T.} \end{aligned} \tag{96}$$

Noting again that θ_3 is determined in terms of θ_1 and θ_2, the task at hand is to study the map

$$(\theta_1, \theta_2) \to (\tilde{\theta}_1, \tilde{\theta}_2). \tag{97}$$

in terms of its fixed points and their stability. To that end, first define the map $\phi : \mathbb{R}^2 \to \mathbb{R}^2$ as

$$\phi(\theta_1, \theta_2) := \frac{1}{2}\left(\cos^{-1}\left(\frac{N_{31}\cdot N_{12}}{\|N_{31}\|\ \|N_{12}\|}\right),\ \cos^{-1}\left(\frac{N_{12}\cdot N_{23}}{\|N_{12}\|\ \|N_{23}\|}\right)\right). \tag{98}$$

Then, consider the related map $\psi : \mathbb{R}^2 \to \mathbb{R}^2$ given by

$$\psi(\theta_1, \theta_2) := \left(\frac{N_{31}\cdot N_{12}}{\|N_{31}\|\ \|N_{12}\|},\ \frac{N_{12}\cdot N_{23}}{\|N_{12}\|\ \|N_{23}\|}\right). \tag{99}$$

Note that

$$\psi\left(\frac{\pi}{3}, \frac{\pi}{3}\right) = -\left(\frac{1}{2}, \frac{1}{2}\right). \tag{100}$$

We also compute, using MAPLE, that

$$(D\psi)\left(\frac{\pi}{3}, \frac{\pi}{3}\right) = \begin{pmatrix} \gamma & 0 \\ 0 & \gamma \end{pmatrix} \tag{101}$$

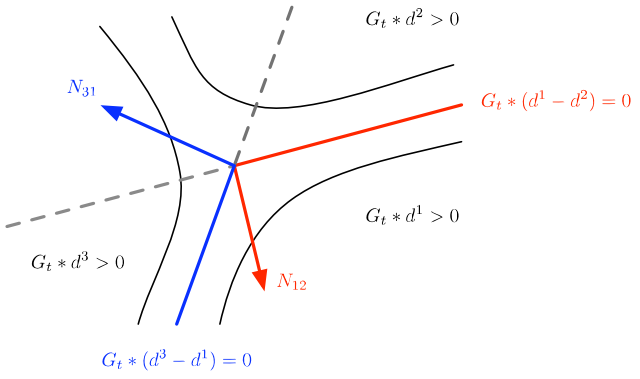

Fig. 8. The three convolved distance functions, $G_t * d_j$ with $j = 1, 2, 3$, intersect in three curves whose projections onto the xy-plane intersect in a point: the new location of the triple junction at the end of the time step.

with

$$\gamma = -\frac{3}{4}\frac{12\sqrt{3}+27\pi+50\pi^2\sqrt{3}-36\pi\sqrt{3}}{(6+5\pi\sqrt{3})^2} \approx -0.52. \tag{102}$$

Write the components of ψ as $\psi = (\psi_1, \psi_2)$ so that we have

$$\phi(\theta_1,\theta_2) = \frac{1}{2}\left(\cos^{-1}(\psi_1(\theta_1,\theta_2)),\ \cos^{-1}(\psi_2(\theta_1,\theta_2))\right). \tag{103}$$

Now, expand ϕ in a Taylor series near $\left(\frac{\pi}{3},\frac{\pi}{3}\right)$. We have

$$\phi\left(\frac{\pi}{3},\frac{\pi}{3}\right) = \left(\frac{\pi}{3},\frac{\pi}{3}\right) + O(\sqrt{t}) \tag{104}$$

and

$$(D\phi)\left(\frac{\pi}{3},\frac{\pi}{3}\right) = \frac{1}{2}\frac{d}{d\xi}\cos^{-1}(\xi)\Big|_{\xi=-\frac{1}{2}} (D\psi)|_{(\theta_1,\theta_2)=\left(\frac{\pi}{3},\frac{\pi}{3}\right)} + \text{H.O.T.} = -\frac{\sqrt{3}}{3}\begin{pmatrix}\gamma & 0\\ 0 & \gamma\end{pmatrix} + \text{H.O.T.} \tag{105}$$

Putting (105) together with (101) and (102), we see that

$$\begin{pmatrix}\tilde{\theta}_1 - \frac{\pi}{3}\\ \tilde{\theta}_2 - \frac{\pi}{3}\end{pmatrix} = M\begin{pmatrix}\theta_1 - \frac{\pi}{3}\\ \theta_2 - \frac{\pi}{3}\end{pmatrix} + O(\sqrt{t}) + \text{H.O.T.} \tag{106}$$

where M is a 2×2 constant matrix whose largest singular value σ satisfies

$$\sigma \approx 0.3 \tag{107}$$

for all small enough $t > 0$ according to (105) along with continuous dependence of the eigenvalues of a matrix on its entries. We thus see that algorithm (89) and (90) stably imposes the Herring angle conditions with an error of the form $O(\sqrt{t})$.

5.4. High order motions

In this *very experimental* and rather *speculative* section, we briefly note how distance function based algorithms for high order motions, such as Willmore flow or surface diffusion flow, might be designed via the expansions of Section 4. This is in analogy with the threshold dynamics based algorithms for Willmore and surface diffusion flows in [20,15].

Here, we focus just on surface diffusion. For this flow, the normal speed of the interface is given by

$$v_n = -\kappa_{ss} = -\kappa_{xx} = -f^{(iv)}(0) + 3(f''(0))^3.$$

There are several alternatives for achieving this speed. For example, as in [20,15], one can first take convolution of the signed distance function to the interface using two different kernels, then take the correct linear combination of the two convolutions so that the lower order, curvature related terms in expansion (30) drop out, leaving behind derivatives of curvature exposed in the dominant terms. Interestingly, the exact form of the algorithm (in particular, the weights used) turn out to be different than in the threshold dynamics case:

Algorithm 7. Given the initial set Σ_0 through its signed distance function $d_0(x)$ and a time step size $\delta t > 0$, generate the sets Σ_j via their signed distance functions $d_j(x)$ at subsequent discrete times $t = j(\delta t)$ by alternating the following steps:

1. Form the functions

$$\begin{aligned} A &:= \left(2G_{\sqrt{\delta t}} - G_{2\sqrt{\delta t}}\right) * d_j,\\ B &:= \left(G_{\sqrt{\delta t}} * d_j - d_j\right)^3. \end{aligned} \tag{108}$$

2. Construct the signed distance function d_{j+1} by

$$d_{j+1}(\mathbf{x}) = \textbf{Redist}\left(\sqrt{\delta t}A + B\right). \tag{109}$$

6. Numerical results

In this section we present several numerical results and convergence studies with the algorithms proposed in the previous sections. Since construction of the signed distance function to a set constitutes a major common step of all the algorithms, it is worth a brief discussion. Fast and accurate solution of Hamilton–Jacobi equations such as the Eikonal

equation is an extensive field in its own right. Here, we are only interested in constructing standard *Euclidean* distance functions, which makes the corresponding Eikonal equation particularly simple and a great variety of existing algorithms applicable. In the computations presented below, a very simple procedure for second order accurate computation of the Euclidean distance function in a tubular neighborhood of the interface was utilized. Specifically, it is based on starting with a first order reconstruction (for which there are indeed many fast algorithms) and then improving it to second (or higher) order by a few steps of a line search strategy at every grid point. Of course, high order versions of more sophisticated algorithms such as [40,35,29,39,43,11] can also be used, perhaps with better results.

Some comments on the computational complexity of the proposed algorithms are also in order. For the sake of simplicity, let us leave possible gains from local versions of the algorithms (such as redistancing only in a tubular neighborhood of the interfaces) out of this brief discussion; although these enhancements are entirely feasible (e.g. in practice one needs to construct the signed distance function only in a tubular neighborhood of thickness $\approx \sqrt{\delta t}$ for second order flows), they are in any case not always as worthwhile as might be suspected – indeed, in applications such as large scale grain boundary motion simulations [13], the evolving network of curves or surfaces is, at least initially, so dense that even a relatively thin tubular neighborhood of them covers almost the entire grid. When the proposed algorithms are thus implemented globally on an $N \times N$, uniform computational grid discretizing e.g. the unit square $[0,1]^2$, the convolution operations can be completed at $O(N^2 \log N)$ complexity using the fast Fourier transform. As mentioned above, the redistancing steps of the proposed algorithms can be accomplished using e.g. fast marching, whose complexity is also $O(N^2 \log N)$ on the whole computational grid provided that first order accurate in space solutions are acceptable. If high order accurate distance functions are required, the first order accurate solutions from e.g. fast marching can be improved to higher order using the strategy mentioned above (and used in numerical results of this paper) at $O(N^2)$ cost. Hence, overall, the complexity of each time step of the proposed algorithms is essentially linear in the number of grid points.

6.1. Curvature motion

We first consider the convergence of algorithm (64) and (65) computed over the time interval $\left[0, \frac{3}{256}\right]$. The initial condition is a circle of radius $\frac{1}{4}$.

Resolution	# time steps	Relative error (%)	Order
32×32	10	0.98	–
64×64	20	0.41	1.25
128×128	40	0.19	1.11
256×256	80	0.093	1.03
512×512	160	0.045	1.05

The errors cited are the errors in the radius of the shrinking circle, the exact value $R(t)$ of which is given by

$$\frac{dR}{dt} = -\frac{1}{R} \;\Rightarrow\; R(t) = \sqrt{\frac{1}{16} - 2t} \tag{110}$$

which gives $R = \sqrt{\frac{5}{128}}$ at $t = \frac{3}{256}$.

Convergence of algorithm (68)–(70) on the same test with an initial circle:

Resolution	# of time steps	Relative error (%)	Order
32×32	10	0.39	–
64×64	20	0.16	1.29
128×128	40	0.088	0.86
256×256	80	0.044	1.00
512×512	160	0.022	1.00

The order of convergence of the time integration is of course still linear, but the results are more accurate by a factor of two with no difference in computational cost: Only a different kernel is used in the convolution step.

We now present results of the higher order method (73)–(75) on the same shrinking circle example:

Resolution	# of time steps	Relative error (%)	Order
32×32	10	0.17	–
64×64	20	0.047	1.85
128×128	40	0.013	1.85
256×256	80	0.0033	1.98
512×512	160	0.00086	1.94

Here, the standard Gaussian kernel G_t was used in the convolutions.

On the same problem, the multistep version (76) and (77) of the high order in t algorithm gives the following results:

Resolution	# of time steps	Relative error (%)	Order
32×32	10	0.092	–
64×64	20	0.027	1.77
128×128	40	0.0078	1.79
256×256	80	0.0020	1.96
512×512	160	0.00052	1.94

The next set of results concern the motion by curvature of the more interesting curve shown in Fig. 9. Since an explicit solution is not available in this case, we monitor the error in the *von Neumann* law of area loss [41]:

$$\frac{d}{dt}A(t) = -2\pi \tag{111}$$

in this (i.e. two-phase) case. The table below shows the relative error in this rate of area loss of the shape as it evolves, using the 2nd order in time scheme (73)–(75).

Resolution	# of time steps	Relative error (%)	Order
32×32	10	1.12	–
64×64	20	0.81	0.47
128×128	40	0.21	1.95
256×256	80	0.036	2.54
512×512	160	0.0028	3.68

6.2. Motion by $f(\kappa)$

From an applications point of view, one of the most important geometric motions with normal speed of the form

$$v_n = f(\kappa)$$

is the *affine invariant motion by curvature* [2,33,34], which has the precise form

$$v_n = \kappa^{\frac{1}{3}}. \tag{112}$$

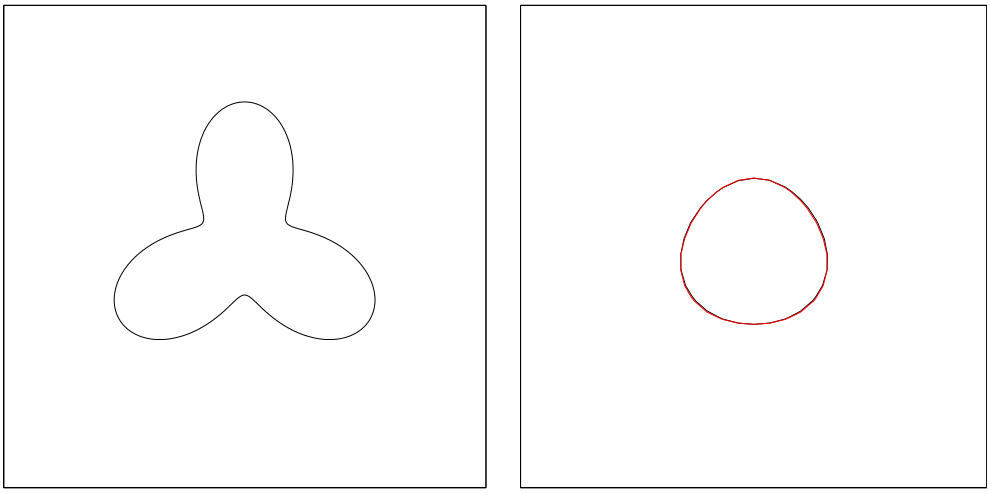

Fig. 9. A more interesting curve under curvature motion. The initial curve is shown on the left. The image on the right shows superimposed on each other the solution at time $t = \frac{3}{256}$ computed at two different resolutions: The black curve at 32×32 spatial resolution using 10 time steps, and the red curve at 1024×1024 resolution using 320 time steps. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

It arises in computer vision applications where algorithms for such fundamental operations on images as denoising and segmentation are expected to be invariant under small changes of the viewpoint. Of course, in this case, $f(\xi) = \xi^{\frac{1}{3}}$ is not Lipschitz, so that our proposed algorithm in Section 5.2 for this type of motion is not monotone in this case. However, we can of course regularize f, for example as

$$f_\varepsilon(\xi) = \operatorname{sign}(\xi)\left\{(\xi^2+\varepsilon)^{\frac{1}{6}} - \varepsilon^{\frac{1}{6}}\right\}. \tag{113}$$

With $\varepsilon > 0$, f_ε is Lipschitz with Lipschitz constant

$$L_\varepsilon = \frac{160^{\frac{1}{6}}\sqrt{6}}{30\varepsilon^{\frac{1}{3}}}. \tag{114}$$

We can then choose the constant M in the description of algorithm (80) and (81) large enough (and dependent on ε) to ensure monotonicity via Proposition 6. On the other hand, in numerical experiments we find that just fixing M e.g. at $M = 2$ and taking $\varepsilon = 0$ (i.e. no regularization) does not seem to lead to any instabilities. The results presented in this section were therefore obtained with no regularization and the said value of M.

Fig. 10 shows the important example of an ellipse, which should remain an ellipse of fixed eccentricity as it evolves due to the affine invariance of the flow, unlike under standard motion by mean curvature that takes the curve asymptotically to a circle. Large time steps were taken to demonstrate stability.

As important as the example of an ellipse is, it is not a particularly challenging test case for affine invariant motion since its curvature remains bounded away from 0 and hence the algorithm never has to deal with the (regularized) singularity at inflection ($\kappa = 0$) points. Fig. 11 shows the result of the algorithm on the more interesting example of an initially flower shaped curve.

6.3. Junctions

This section presents a couple of simple examples of computing the motion of a triple junction under curvature motion in the plane using the algorithm (89) and (90) in Section 5.3. Algorithm (89) and (90) can in fact be generalized to allow accurate and efficient computation of very large scale grain networks in both two and three space dimensions, which is of high interest in materials science applications. Extensive demonstration of a generalized version of the algorithm in this capacity has been carried and will be reported separately in an upcoming paper [13] by one of the authors. Here, we confine ourselves to the simple cases of only three or four phases.

Fig. 12 shows a computation with three phases. The initial data consists of two partially overlapping disks with a straight interface in between, as shown in Fig. 12. The two partial disks constitute two of the phases; the background (complement of the disks) constitutes the third. Hence, all three phases have $n = 2$ triple junctions on their boundary throughout the evolution. If we let $A(t)$ denote area of one of the partial disks, the von Neumann area loss law [41] this time implies

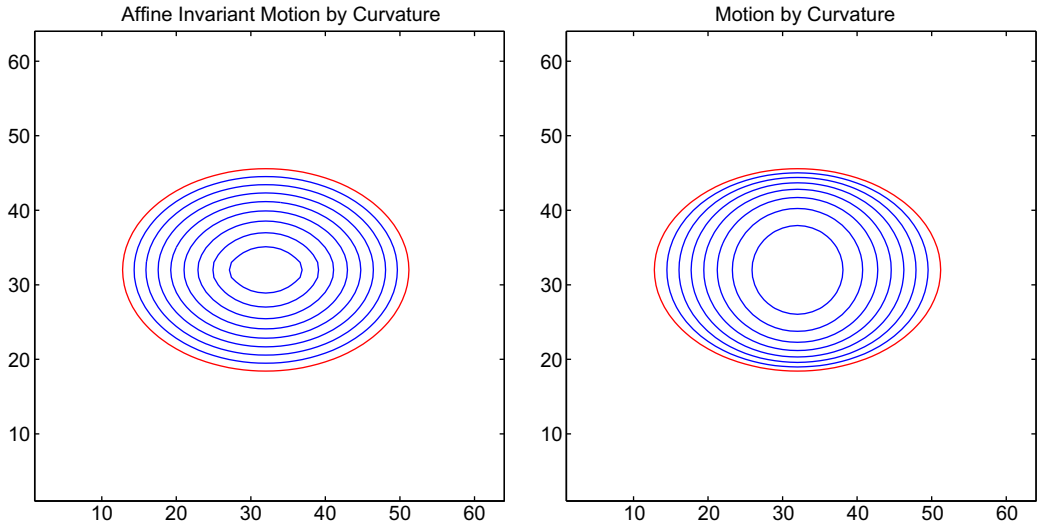

Fig. 10. Comparison between regular (left) and affine invariant curvature motion (right). The initial curve, an ellipse, is shown in red. Under affine invariant curvature motion, in remains an ellipse of fixed eccentricity. Computation was carried out on a 64×64 domain with coarse time steps; blue curves show the evolving curve at consecutive time steps. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

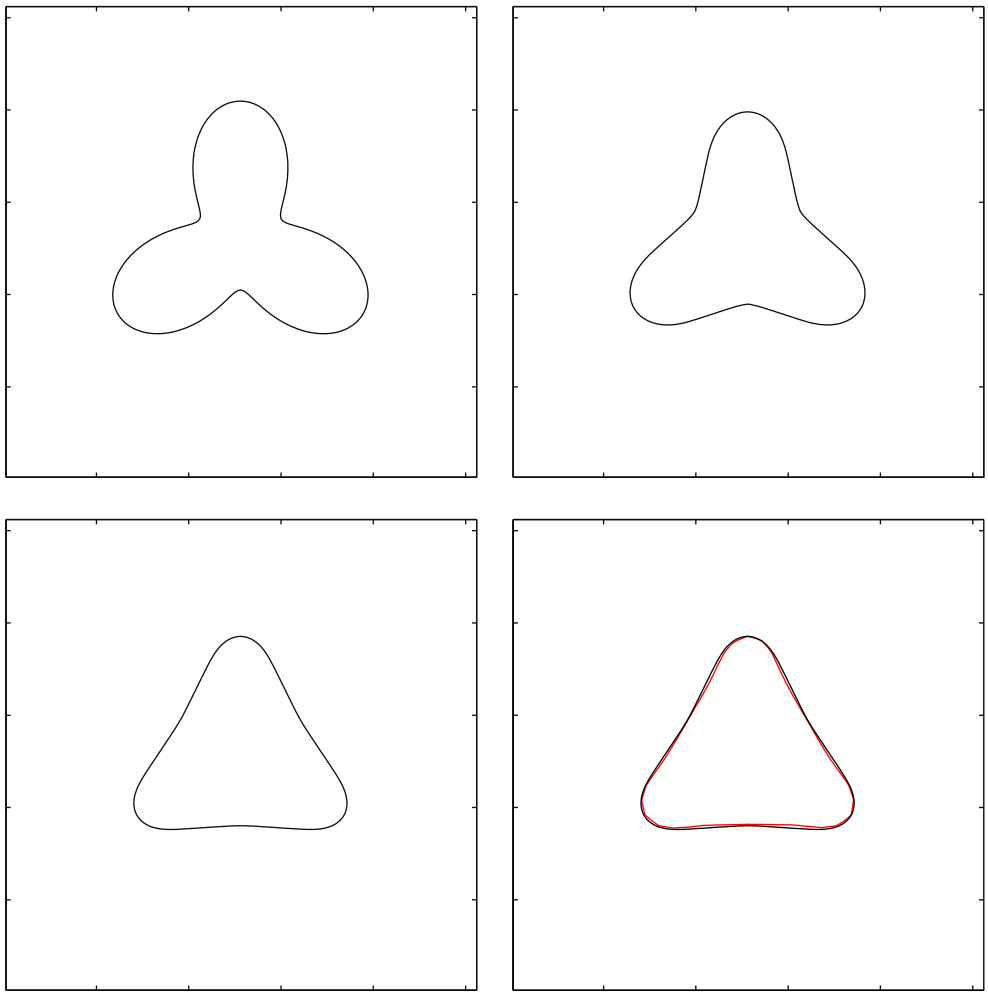

Fig. 11. Motion of a flower shaped curve under affine invariant curvature motion computed using algorithm (80) and (81). The upper left plot shows the initial curve. The upper right and lower left plots show the evolution of the curve at subsequent times. These computations were carried out on a 256×256 grid using 80 and 160 time steps, respectively. The lower right plot shows the result from the 256×256 computation at 160 time steps (black curve) superimposed with the same solution computed on a 32×32 grid using only 20 (eight times larger) time steps (red curve). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$$\frac{d}{dt}A(t) = \frac{\pi}{3}(n-6) = -\frac{4\pi}{3} \tag{115}$$

To assess the accuracy of the simulation, we measure the *rate* of area loss in one of the partial disks. The table below shows the percentage relative error in this quantity over the time interval $\left[0, \frac{1}{64}\right]$. Gaussian kernel was used in the convolution step (89).

Resolution	# of time steps	Relative error (%)	Order
32×32	30	3.91	–
64×64	60	2.07	0.918
128×128	120	1.28	0.693
256×256	240	0.84	0.608
512×512	480	0.53	0.664

The errors reported in the table were obtained as an average over 10 runs with the initial condition rotated and translated randomly to preempt possible interference from grid effects. The truncation error analysis carried out in Section 4.2 implies $O(\sqrt{t})$ error at junctions, some evidence of which can be seen in the table. Fig. 12 shows the computed solution at 32×32 and 512×512 resolution superimposed.

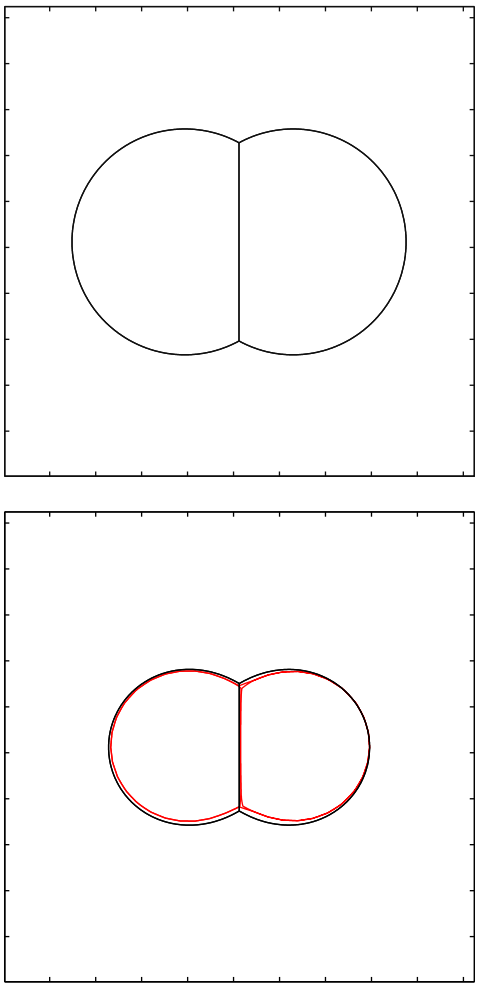

Fig. 12. The first plot shows the initial condition for algorithm (89) and (90). The second plot shows the solution at $t = \frac{1}{64}$ computed using 30 time steps at 32×32 spatial resolution (red curve), superimposed with the computed result using 480 time steps at 512×512 resolution (black curve). The error in the latter is about one thirteenth of the former. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 13 shows a computation with four phases: Three partial disks and the background. This time, von Neumann law gives:

$$\frac{d}{dt}A(t) = \frac{\pi}{3}(n-6) = -\pi. \tag{116}$$

The relative error in (116) is tabulated in the table below at various resolutions:

Resolution	# of time steps	Relative error (%)	Order
32×32	30	3.82	–
64×64	60	2.10	0.863
128×128	120	1.26	0.737
256×256	240	0.71	0.828
512×512	480	0.44	0.690

The plot on the right in Fig. 13 compares as in the previous example the solution obtained at high and low (spatial and temporal) resolutions.

Fig. 14 shows the 512×512 computation at later times. At some point, one of the phases that started out as a partial disk disappears. The algorithm handles that transition seamlessly, and carries on as a three phase flow from that point onwards – one of the well-known advantages of implicit interface representations is thus maintained in our algorithms, as expected.

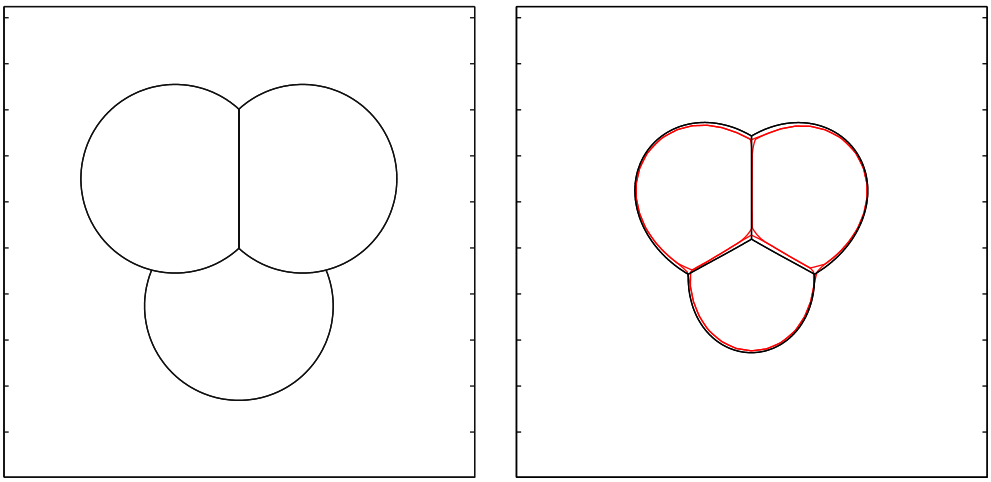

Fig. 13. Sample computation with four phases. Plot on the left is the initial condition. The one on the right is at time $t = \frac{1}{64}$. The black contour is the solution computed at 512×512 spatial resolution using 480 time steps; the red contour is the solution computed at 32×32 spatial resolution using 30 time steps. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

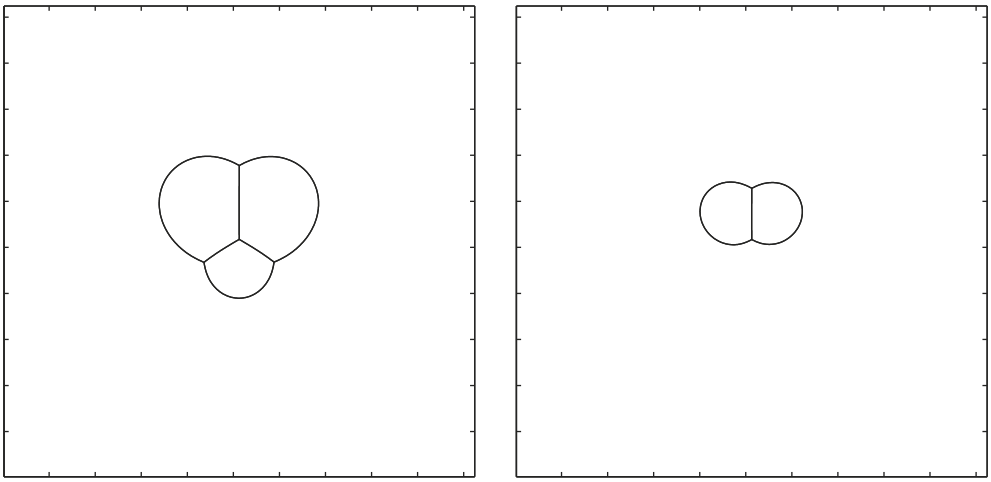

Fig. 14. Further evolution of four phase initial condition from Fig. 13 at a 512×512 resolution. One of the phases disappears at an intermediate time; the evolution then seamlessly proceeds as a three phase flow.

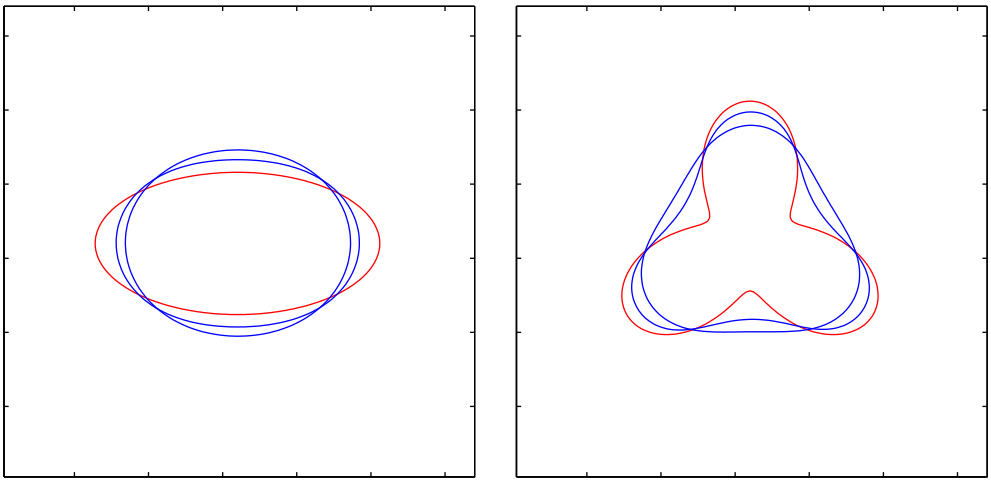

Fig. 15. Evolution of two curves under the tentative algorithm for surface diffusion of Section 5.4. The red curve is the initial condition in each case. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Finally, we expect that the rate of convergence of the algorithm can be improved, if needed, by the Richardson extrapolation type ideas used in Section 5.1, as was done in [31] for multiphase motion in the context of threshold dynamics.

6.4. High order motions

Here we present a couple of simple numerical tests of the tentative algorithm for surface diffusion suggested in Section 5.4. The examples are intended merely as a qualitative check. The first plot of Fig. 15 shows the evolution of an ellipse under this algorithm towards a circle at times 1.25×10^{-6} (reached with 2000 time steps) and 2.5×10^{-6} (4000 time steps), computed on the modest grid size of 128×128. Surface diffusion flow preserves area; the change in area in the computed solution at final time is $\approx 4.25\%$. The second plot of Fig. 15 shows the evolution of a slightly more interesting, initially flower shaped curve.

The scheme appears to be stable under much larger time steps, too, but then the error becomes large rather quickly. Perhaps the Richardson extrapolation idea used in Section 5.1 can be applied also here to improve the accuracy.

Acknowledgments

Selim Esedoglu was supported by NSF DMS-0748333, NSF DMS-0713767, an Alfred P. Sloan Foundation fellowship, and a University of Michigan Rackham faculty grant. Steve Ruuth was supported by an NSERC Discovery Grant. Richard Tsai was supported by NSF DMS-0714612 and an Alfred P. Sloan Foundation fellowship. The authors thank Matt Elsey, who provided important corrections to a previous version of the paper. They also thank the Banff International Research Station (BIRS) for hosting the Research in Teams Event 06rit314 where some of the early work on this project was completed.

References

[1] F. Almgren, J.E. Taylor, L.-H. Wang, Curvature-driven flows: a variational approach, SIAM Journal on Control and Optimization 31 (2) (1993) 387–438.
[2] S. Angenent, G. Sapiro, A. Tannenbaum, On the affine invariant heat equation for nonconvex curves, Journal of the American Mathematical Society 11 (1998) 601–634.
[3] S. Baldo, Minimal interface criterion for phase transitions in mixtures of Cahn–Hilliard fluids, Annual I.H.P. Analyse Nonlineaire 7 (1990) 37–65.
[4] G. Barles, C. Georgelin, A simple proof of convergence for an approximation scheme for computing motions by mean curvature, SIAM Journal of Numerical Analysis 32 (1995) 484–500.
[5] K. Brakke, The surface evolver, Experimental Mathematics 1 (2) (1992) 141–165.
[6] L. Bronsard, F. Reitich, On three-phase boundary motion and the singular limit of vector-valued Ginzburg–Landau equation, Archive for Rational Mechanics and Analysis 124 (1993) 355–379.
[7] L. Bronsard, B. Wetton, A numerical method for tracking curve networks moving with curvature motion, Journal of Computational Physics 120 (1) (1995) 66–87.
[8] A. Chambolle, An algorithm for mean curvature motion, Interfaces and Free Boundaries 6 (2) (2004) 195–218.
[9] A. Chambolle, An algorithm for total variation minimization and applications, Journal of Mathematical Imaging and Vision 20 (2004) 89–97.
[10] A. Chambolle, M. Novaga, Approximation of the anisotropic mean curvature flow, Mathematical Models and Methods in Applied Sciences 17 (6) (2007) 833–844.
[11] L.-T. Cheng, Y.-H. Tsai, Redistancing by flow of time dependent Eikonal equation, Journal of Computational Physics 227 (8) (2008) 4002–4017.
[12] M.C. Delfour, J.-P. Zolesio, Shapes and Geometries. Analysis, Differential Calculus, and Optimization. Advances in Design and Control, SIAM, 2001.
[13] M. Elsey, S. Esedoglu, P. Smereka, Diffusion generated motion for grain growth in two and three dimensions, Journal of Computational Physics 228 (21) (2009) 8015–8033.
[14] S. Esedoglu, S. Ruuth, Y.-H. Tsai, Threshold dynamics for shape reconstruction and disocclusion, in: Proceedings of the ICIP, 2005.
[15] S. Esedoglu, S. Ruuth, Y.-H. Tsai, Threshold dynamics for high order geometric motions, Interfaces and Free Boundaries 10 (3) (2008) 263–282.
[16] S. Esedoglu, P. Smereka, A variational formulation for a level set representation of multiphase flow and area preserving curvature flow, Communications in Mathematical Sciences 6 (1) (2008) 125–148.
[17] L.C. Evans, Convergence of an algorithm for mean curvature motion, Indiana University Mathematics Journal 42 (1993) 553–557.
[18] H. Federer, Curvature measures, Transactions of the American Mathematical Society 93 (1959) 418–491.
[19] H. Garcke, B. Nestler, B. Stoth, A multiphase filed concept: numerical simulationsof moving phase boundaries and multiple junctions, SIAM Journal of Applied Mathematics 60 (1999) 295–315.
[20] R. Grzhibovskis, A. Heintz, A convolution thresholding scheme for the Willmore flow, Interfaces and Free Boundaries 10 (2) (2008) 139–153.
[21] C. Herring, Surface Tension as a Motivation for Sintering, McGraw Hill, 1951. pp. 143–179.
[22] M. Kimura, H. Notsu, A level set method using the signed distance function, Japan Journal of Industrial and Applied Mathematics 19 (2002) 226–415.
[23] D. Kinderlehrer, I. Livshitz, S. Taasan, A variational approach to modeling and simulation of grain growth, SIAM Journal on Scientific Computing 28 (5) (2006) 1694–1715.
[24] P. Mascarenhas, Diffusion generated motion by mean curvature. CAM Report 92-33, UCLA, July 1992, URL: <http://www.math.ucla.edu/applied/cam/index.html>.
[25] B. Merriman, J.K. Bence, S. Osher, Motion of multiple junctions: a level set approach, Journal of Computational Physics 112 (2) (1994) 334–363.
[26] W.W. Mullins, Two dimensional motion of idealized grain boundaries, Journal of Applied Physics 27 (1956) 900–904.
[27] S. Osher, J. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulation, Journal of Computational Physics 79 (1988) 12–49.
[28] L. Rudin, S. Osher, E. Fatemi, Nonlinear total variation based noise removal algorithms, Physica D 60 (1992) 259–268.
[29] G. Russo, P. Smereka, A remark on computing distance functions, Journal of Computational Physics 163 (2000) 51–67.
[30] S.J. Ruuth, A diffusion generated approach to multiphase motion, Journal of Computational Physics 145 (1998) 166–192.
[31] S.J. Ruuth, Efficient algorithms for diffusion-generated motion by mean curvature, Journal of Computational Physics 144 (1998) 603–625.
[32] S.J. Ruuth, B. Merriman, Convolution generated motion and generalized Huygens' principles for interface motion, SIAM Journal on Applied Mathematics 60 (2000) 868–890.
[33] G. Sapiro, A. Tannenbaum, Affine invariant scale-space, International Journal of Computer Vision 11 (1993) 25–44.
[34] G. Sapiro, A. Tannenbaum, On affine plane curve evolution, Journal of Functional Analysis 119 (1994) 79–120.
[35] J. Sethian, A fast marching level set method for monotonically advancing fronts, Proceedings of the National Academy of Sciences 93 (4) (1996) 1591–1595.

[36] P. Smereka, Semi-imlpicit level set methods for curvature flow and motion by surface diffusion, Journal of Scientific Computing 19 (2003) 439–456.
[37] K.A. Smith, F.J. Solis, D.L. Chopp, A projection method for motion of triple junctions by level sets, Interfaces and Free Boundaries 4 (3) (2002) 263–276.
[38] J.E. Taylor, A variational approach to crystalline triple-junction motion, Journal of Statistical Physics 95 (1999) 1221–1244.
[39] Y.-H. Tsai, L.T. Cheng, S. Osher, H.-K. Zhao, Fast sweeping methods for a class of Hamilton–Jacobi equations, SIAM Journal on Numerical Analysis 41 (2) (2003) 673–694.
[40] J. Tsitsiklis, Efficient algorithms for globally optimal trajectories, IEEE Transactions on Automatic Control 40 (1995) 1528–1538.
[41] J. von Neumann, Metal Interfaces, American Society for Metals, Cleveland, OH, 1952, pp. 108–110.
[42] X. Zhang, J.-S. Chen, S. Osher, A multiple level set method for modeling grain boundary evolution of polycrystalline materials, CAM Report 06-69, UCLA, December 2006, URL: <http://www.math.ucla.edu/applied/cam/index.html>.
[43] H. Zhao, A fast sweeping method for Eikonal equations, Mathematics of Computation 74 (250) (2005) 603–627.
[44] H.K. Zhao, T.F. Chan, B. Merriman, S. Osher, A variational level set approach to multiphase motion, Journal of Computational Physics (1996) 179–195.