JOURNAL OF COMPUTATIONAL PHYSICS 112, 334–363 (1994)

Motion of Multiple Junctions: A Level Set Approach*

BARRY MERRIMAN, JAMES K. BENCE, AND STANLEY J. OSHER

Department of Mathematics, UCLA, Los Angeles, California 90024

Received June 15, 1993

A coupled level set method for the motion of multiple junctions is proposed. The new method extends the "Hamilton–Jacobi" level set formulation of Osher and Sethian. It retains the feature of tracking fronts by following level sets and allows the specification of arbitrary velocities on each front. The diffusion equation is shown to generate curvature dependent motion and this is used to develop an algorithm to move multiple junctions with curvature-dependent speed. Systems of reaction diffusion equations are shown to possess inherent properties which prohibit efficient numerical solutions when applied to curvature-dependent motion. © 1994 Academic Press, Inc.

1. INTRODUCTION

This article explores algorithms for the motion of multiple junctions. In many situations, e.g., crystal growth, a material is composed of three or more phases. The interfaces between the phases move according to some law. If the material is a metal and its grain orientation is different in each region, then an isotropic surface energy means that the velocity is the mean curvature of the interface. Or the velocities of the interfaces may depend on the pair of phases in contact; e.g., a different constant velocity on each interface, as in Fig. 1.

When there are only two regions and one interface separating them, the "Hamilton–Jacobi" level set formulation introduced by Osher and Sethian [15] applies. The resulting numerical formulation allows fronts to self-intersect, develop singularities, and change topology. The Osher–Sethian algorithm, if used in Fig. 1, would produce a boundary between regions that has a non-empty interior; it would create what appears to be a new region. Also the Osher–Sethian formulation uses a continuous velocity and Fig. 1 requires a discontinuous velocity function since each interface moves at a different rate.

There has been little work done on the motion of multiple junctions. Only Taylor [17] and Bronsard and Reitich [2] address this problem. Bronsard and Reitich propose a system of reaction–diffusion equations to model the motion of triple junctions. They show that interfaces in the solution to their reaction–diffusion system move with a velocity proportional to the curvature of the interface. Taylor's work is based on a direct application of Huygens' principle and applies only to constant velocities. Thus, neither method allows arbitrary,. physically consistent velocities to be prescribed.

In Section 2 we exploit the link between diffusion and curvature to derive a method for curvature-dependent motion of multiple junctions. These ideas motivate Section 3, in which we extend the Osher–Sethian algorithm to handle Fig. 1, and, in general, multiple junctions with specified, physically consistent velocities.

Our new method has several advantages over those of [17, 2]. First, it is easy to program. Taylor's approach involves the manipulation of geometric objects. As the dimension of the problem increases, these objects are increasingly difficult to manipulate. The interfaces in Fig. 1 would be considered the union of small line segments. In three dimensions small planar segments are used. The original Osher–Sethian algorithm extends immediately to n dimensions, and so does the new one introduced here. The line segments or planar segments must also interact with each other. At various points a decision must be made to delete or insert segments. Visualization of the possible interactions to determine the conditions under which segments are deleted/inserted is difficult (if not impossible) in higher dimensions. No such decisions are required with our new method. It retains a key feature of the level set approach; we simply use a contour plotter to find the front. All interactions are handled by the underlying PDE.

The reaction–diffusion system possesses a small positive parameter ε. This parameter causes difficulties in the numerical solution of the system when $\varepsilon \ll \Delta x$, where Δx is the spacing on the grid. The difficulties are inherent in the system, not the numerical method. This is important because such systems are employed in modeling dendritic crystal growth [11], and the development of the dendrites is linked to the size of ε. The only remedy to the numerical problems, which we will see in Section 4, is to take $\Delta x/\varepsilon \ll 1$ which is impractical numerically.

* Research was supported by DARPA/ONR-N00014-92-J-1890, ARO DAA L03-91-G0162, Grant NSF DMS-91-03104, and by Department of Commerce Grant DOC 60NANB1D1125.

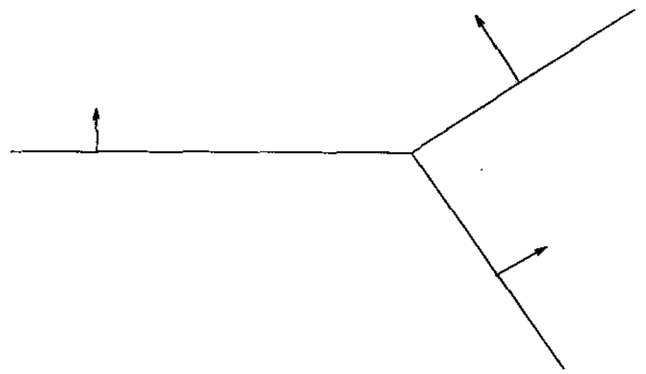


FIG. 1. Triple junction, with prescribed velocities.

2. DIFFUSION GENERATED CURVATURE DEPENDENT MOTION

It is well known that a link exists between curvature and diffusion [13, 19]. We first show how a splitting method applied to a reaction–diffusion equation leads to an algorithm for propagating interfaces with curvature dependent speed. Then we apply the method to multiple junctions.

2.1. Splitting a Reaction–Diffusion Equation

Consider a splitting method applied to

$$\begin{aligned} u_t &= \varepsilon\,\Delta u - \frac{1}{\varepsilon} f_u(u) \\ u&: \mathbb{R}^2 \times \mathbb{R}^+ \mapsto \mathbb{R}, \end{aligned} \tag{1}$$

where $\varepsilon > 0$ is small. For example, if we have an iterate u^n, then first solve

$$\begin{aligned} \bar{u}_t &= \varepsilon\,\Delta \bar{u} \\ \bar{u}(x, 0) &= u^n \end{aligned} \tag{2}$$

until some time T_d and then solve

$$\begin{aligned} u_t &= -\frac{1}{\varepsilon} f_u(u) \\ u(x, 0) &= \bar{u}(x, T_d) \end{aligned} \tag{3}$$

until some time T_r and finally set $u^{n+1} = u(x, T_r)$. The key question is can (and if so *how* do) we choose T_d and T_r to obtain the right solution? A qualitative description of the solution of (1) is offered in [16]. The solution approaches a piecewise constant function whose values are the stable zeros of $f(u)$. There are sharp transitions between the regions in which u is constant, and these interfaces move. If $f(u)$ is bistable and the wells are of equal depth, then the velocity of an interface is $\varepsilon\kappa$, where κ is the curvature of the interface. This is the case in which we are interested.

The splitting method (2), (3) is attractive because the individual problems are well understood. We can always describe the qualitative behavior of both problems. We can write down exact solutions for (2) immediately. The same may be done for (3) under conditions on the initial data [9].

Suppose $f_u(u) = u(u+1)(u-1)$. The phase plane diagram in Fig. 2 clearly describes the qualitative behavior of (3): Values less than zero are driven towards -1 and values greater than zero are driven to 1. There are an infinite number of pairs (T_d, T_r) we may use in our splitting method. However, note that for T_r large enough, we can write the solution to (3) down immediately, based on the phase plane. For $T_r = \infty$, we have that if initially $u(x, 0) < 0$, then $u(x, \infty) = -1$ and if $u(x, 0) > 0$, then $u(x, \infty) = +1$.

So the solution to (3) can be computed rapidly if we take $T_r = \infty$. This seems excessive but note that if we do so, then on the next iteration of the splitting method (2), (3) the diffusion equation will be applied to piecewise constant initial data. Let us consider the effect of this. For simplicity, we examine the special case where the initial data is the characteristic function χ of a connected domain D with a smooth boundary. At a point P on the boundary (Fig. 3), the local geometry of the boundary is completely determined by the circle of curvature there. χ has a local cylindrical symmetry about the center of the circle of curvature at P. Express the diffusion equation in cylindrical coordinates with origin at C in Fig. 3. Near P, χ depends only the radial coordinate so we find

$$\frac{\partial \chi}{\partial t} = \frac{1}{R}\frac{\partial}{\partial R}\left(R\,\frac{\partial \chi}{\partial R}\right) = \frac{1}{R}\frac{\partial \chi}{\partial R} + \frac{\partial^2 \chi}{\partial R^2}. \tag{4}$$

This equation has the form of an advection–diffusion equation in the radial direction, with advective velocity $1/R$. The initial data for this equation is a step function as depicted in Fig. 4. The advective term will simply propagate the initial data, preserving its shape. The diffusive term, for this initial data, produces the solution (assuming that the position of the step is given by $x = 0$)

$$\frac{1}{2}\left(1 - \operatorname{erf}\left(\frac{x}{2t^{1/2}}\right)\right).$$

Observe the effect of both terms on the level set $\chi = 0.5$. The velocity of the propagation is $1/R$, that is, the curvature of χ at P. The diffusive term does not affect this level set, as the initial position of the level set $\{\chi = 0.5\}$ corresponds to $x = 0$. Thus, for a short time the level set $\{\chi = 0.5\}$ will move with velocity equal to the curvature.

Note that this argument applies to piecewise constant functions after a scaling and/or translation. This means there is no need to consider specific bistable $f_u(u)$; it is the relative depths of the wells of $f(u)$ that determines the velocity [16],

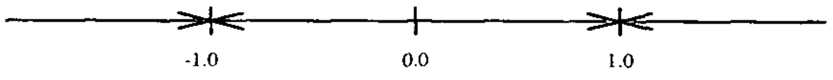


FIG. 2. Phase plane for (3) with $f_u(u) = u^3 - u$.

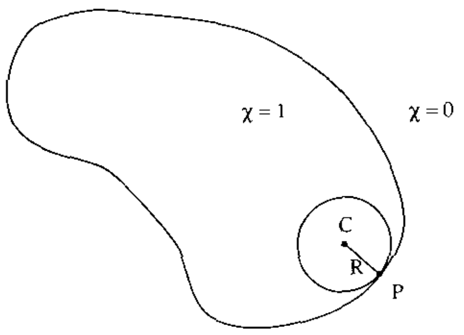


FIG. 3. Circle of curvature at P.

not the values attained within the wells. Therefore it is simplest to discard $f_u(u)$ completely and apply diffusion to characteristic function instead. However, we must still solve the equivalent of (3); that is, how do we proceed after the characteristic function has diffused for a short time? Since the level set $\{\chi = 0.5\}$ represents the boundary, the simplest course is to create a new characteristic function whose boundary is the set $\{\chi = 0.5\}$. We can collect these observations together into a numerical method.

Suppose we want to propagate the boundary of a region D with curvature–dependent speed. The diffusion generated curvature dependent motion algorithm is:

ALGORITHM DGCDM.

(1) Initialize: $\chi = \chi_D$

(2) Apply diffusion to χ for some time T_{chop}

(3) "Sharpen" the diffused function (e.g., if $\chi < \frac{1}{2}$ then set $\chi = 0$ else set $\chi = 1$) and then begin again.

The location of the interface is given by the level set $\{\chi = \frac{1}{2}\}$.

An important consideration is the size of T_{chop}. The neglected angular term in (4) will pollute the approximation at nearby points after some time. We can make a rough estimate of the time in which the velocity is a good approximation to curvature. Certainly, we must let the diffusion proceed for a long enough time that the level set $\chi = 0.5$ moves at least one grid point (otherwise, the chopping step would keep the front stationary), so we require $\kappa T_{\text{chop}} \gg \Delta x$. The neglected angular term will become important once the diffusive information has traveled a distance on the order of the local radius of curvature, so we need $T_{\text{chop}}^{1/2} \ll R$ (recall $\kappa = 1/R$). Combining these gives

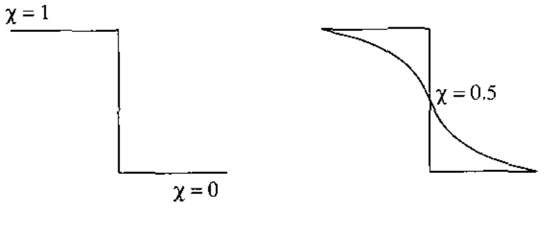


FIG. 4. Side view of Fig. 3 near P.

$$\frac{R}{\Delta x} \ll \frac{T_{\text{chop}}}{\Delta x^2} \ll \left(\frac{R}{\Delta x}\right)^2$$

and, so as long as the grid resolves the smallest radius of curvature on the grid, there are acceptable T_{chop}. Note that during a computation, T_{chop} may be varied in response to changes in the character of the solution.

This method gives qualitatively correct results. The quantitative results are also good. We can compute the exact velocity in the case of a circle, because the curvature at any point on a circle of radius r is $1/r$. Starting with a circle of radius 0.15, the computed velocity is within 5% of the true velocity while the grid resolves the radius of the circle (a circle shrinks under this motion, and the grid is not adaptive, so we must eventually fail to resolve the curvature). In addition, it has been shown [1, 7, 14] that the above algorithm converges to mean curvature motion in a certain sense. Briefly, Evans defines an operator $\mathscr{H}(t)$ as follows: given a compact set $C_0 \in \mathbb{R}^n$, solve

$$u_t = \Delta u$$
$$u(x, 0) = \chi_{C_0}.$$

At time $t > 0$ define $C_t = \{x : u(x, t) \geq \frac{1}{2}\}$. Then we write $C_t = \mathscr{H}(t)\, C_0$. The set $\{\mathscr{H}(t)\}_{t \geq 0}$ is described as "heat diffusion flow." Then the "mean curvature flow semigroup" is defined. Through nonlinear semigroup theory, it is proven that

$$\lim_{m \to \infty} \mathscr{H}(t/m)^m\, g = \mathscr{M}(t)\, g \qquad \text{for} \quad g \in C(\mathbb{R}^n), \qquad t > 0.$$

Thus, as the heat equation is iterated over smaller and smaller times, we obtain motion by mean curvature.

2.2. Multiple Junctions

As stated, the method does not apply to curves with multiple junctions, but it can be extended in a straightforward way. Any extension we make should reduce to the original algorithm when there is only one region. First, observe that for one region R it does not matter if we use χ_R or $1 - \chi_R$ to compute the motion since curvature is invariant under a change of sign. So we could use two characteristic functions, χ_R and χ_{R^c} to compute the motion of ∂R. We would apply the algorithm to each χ independently and at the chopping step we find the sets $\{\chi > \frac{1}{2}\}$ and $\{\chi_{R^c} > \frac{1}{2}\}$. Note that these two sets are equivalent to the sets $\chi_R > \chi_{R^c}$ and $\chi_{R^c} > \chi_R$, and this guides us when there are more than two regions to an extension of the DGCDM algorithm:

(1) Given a partition of $\mathbb{R}^m$ into regions R_i, $i = 1, ..., n$,

(2) For $i = 1, ..., n$ construct χ_i, the characteristic function of R_i

(3) For $i = 1, ..., n$ diffuse each χ_i independently for some time T_{chop}

(4) For $i=1, ..., n$ set $\chi_i = \{x : \chi_i(x) \geqslant \chi_j(x), j=1, ..., n\}$

(5) Go to 3.

The interfaces are given by $\bigcup_{i=1,...,n} \{\chi_i = \frac{1}{2}\}$. This new algorithm reduces to the original when there are only two regions.

The choice of the update after each χ_i is diffused affects the stable triple junctions that develop. For the above update step, the stable triple junctions are symmetric: ones whose angles are all 120°. Initially, and after each update step, the χ_i form a partition of $\mathbb{R}^m$ and so we see that at these times the vector $(\chi_1(x), \chi_2(x), ..., \chi_n(x))$ is one of the usual basis vectors for $\mathbb{R}^n$; e.g., for $n=3$ it is one of (0, 0, 1), (0, 1, 0), or (1, 0, 0). The diffusion process is linear and, since initially $\sum \chi_i(x) = 1$, this is true for all times. Therefore, at each time step, every point on the grid is associated with a point on the surface $\sum_{i=1}^{m} x_i = 1$. This is best visualized in $\mathbb{R}^3$; see Fig. 5.

For $n=3$ each point on the grid is associated with a vertex of the plane $x+y+z=1$ in the first octant. As each χ_i diffuses, the points move out into the interior of the plane. Taking as the update step the maximum χ_i, we divide the plane into three regions as shown in Fig. 5. By choosing a different configuration on the plane, we can select a different shape, as in Fig. 6, which leaves a 90° angle fixed. Calculations with both configurations follow in figures on the next few pages.

2.3. Summary

The idea underlying the DGCDM algorithm, that diffusion generates curvature, has been known for some time [13, 19], and it seems to have been considered not for computing curvature-dependent motion but rather for image enhancement. Found lacking in this respect, it has not been pursued further. The DGCDM algorithm is known in image processing technology as an iterated median filter. In [5], it is noted that "computer vision researchers know quite well that a median filter, iterated on a grid, remains steady after some iterations." It would seem that the derivation of the appropriate time to run the filter has not been carried out. This "well-known" behavior is caused by iterating the filter too rapidly, i.e., taking T_{chop} too small, which prevents motion.

The DGCDM algorithm is not limited to curvature-dependent motion. By using a variable coefficient diffusion equation, we can obtain anisotropic velocities. It is also possible to obtain constant velocities by following different level sets as the algorithm progresses [14]. Specifically, if we track the level set $\frac{1}{2} - v(t/4\pi\varepsilon)^{1/2}$ then the speed of the front will be $v + \varepsilon\kappa$. However, note that by doing this we lose symmetry in that we may no longer compute with either χ or $1-\chi$. If we wish to compute with $1-\chi$ then we must follow the level set $\frac{1}{2} + v(t/4\pi\varepsilon)^{1/2}$; otherwise the level set will move in the opposite direction (refer to Fig. 4). This requirement prevents the application of this method to multiple junctions when the velocities are constant, as will be made clear in the next section.

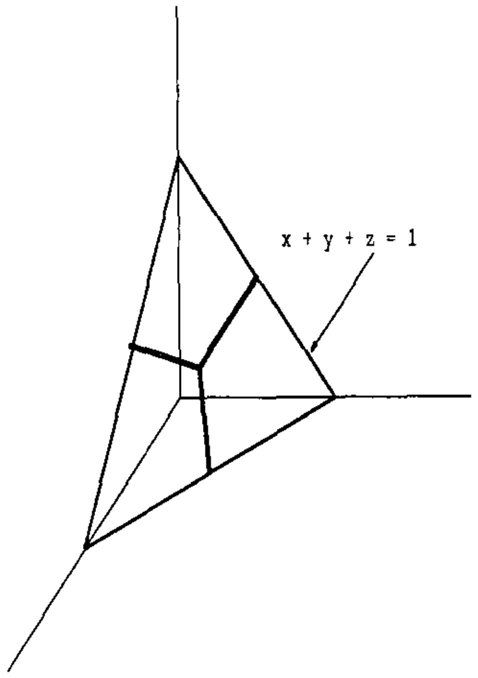


FIG. 5. Stable shape "Y" for update step in which maximum is taken.

For configurations on $x+y+z=1$ in which the meeting point of the regions is near the centroid point, triple points in the corresponding computation approach the same shape as the meeting point. But as the meeting point moves towards the edges of the plane this is no longer true. The precise connection between the configuration on the plane and the stable angles in the computation has not been established (see Figs. 7–16).

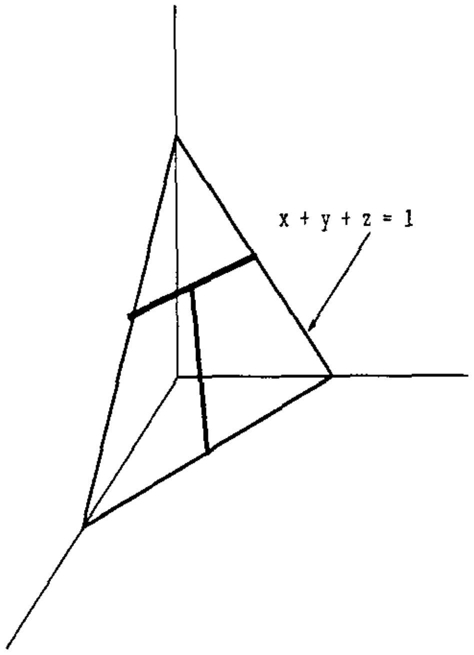


FIG. 6. Alternative update method has "T" shaped stable configuration.

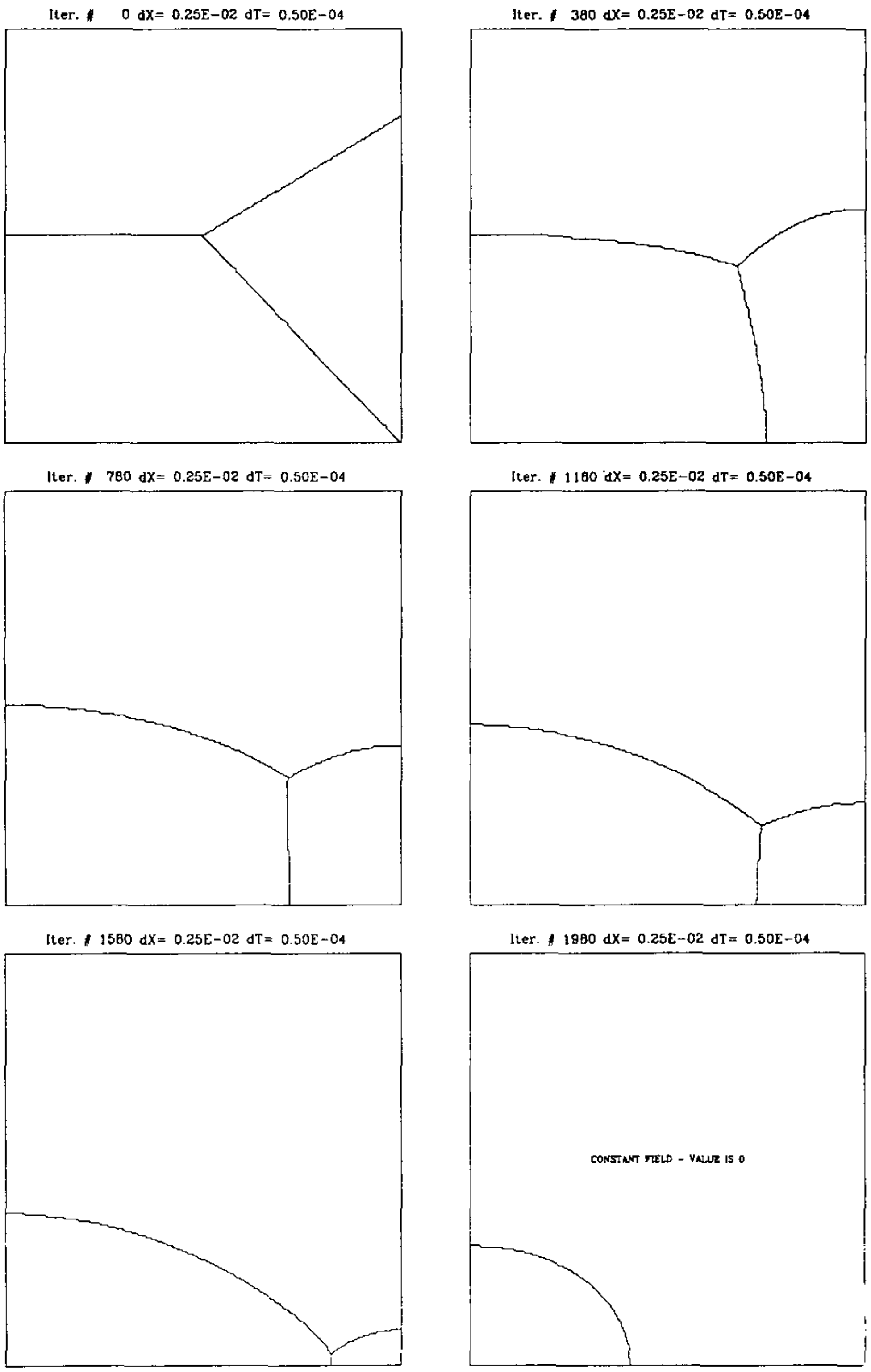


FIG. 7. Motion of triple junction under curvature.

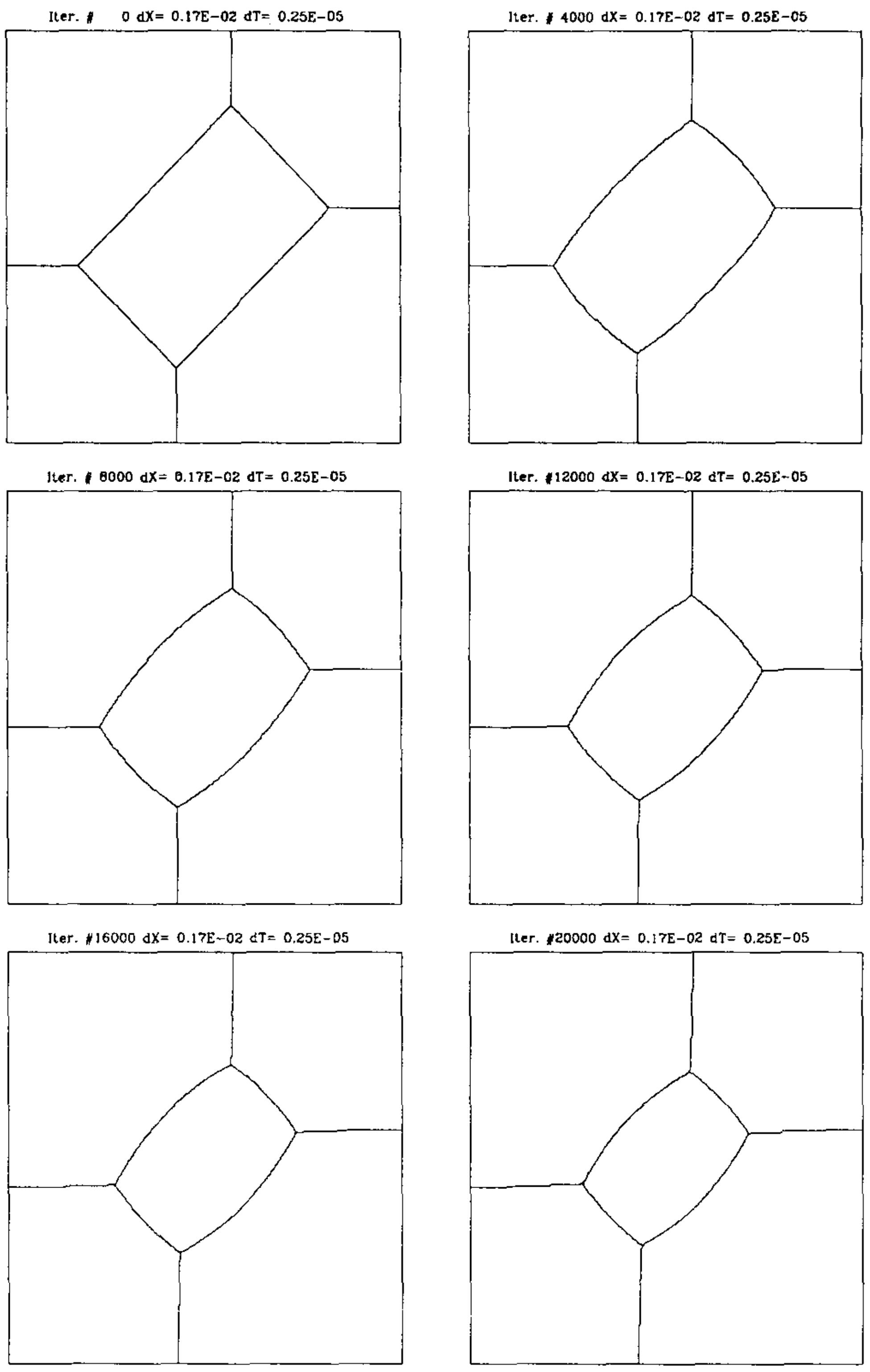

Iter. # 0 dX= 0.17E-02 dT= 0.25E-05
Iter. # 4000 dX= 0.17E-02 dT= 0.25E-05
Iter. # 8000 dX= 0.17E-02 dT= 0.25E-05
Iter. #12000 dX= 0.17E-02 dT= 0.25E-05
Iter. #16000 dX= 0.17E-02 dT= 0.25E-05
Iter. #20000 dX= 0.17E-02 dT= 0.25E-05

FIG. 8. Motion of triple junction under curvature.

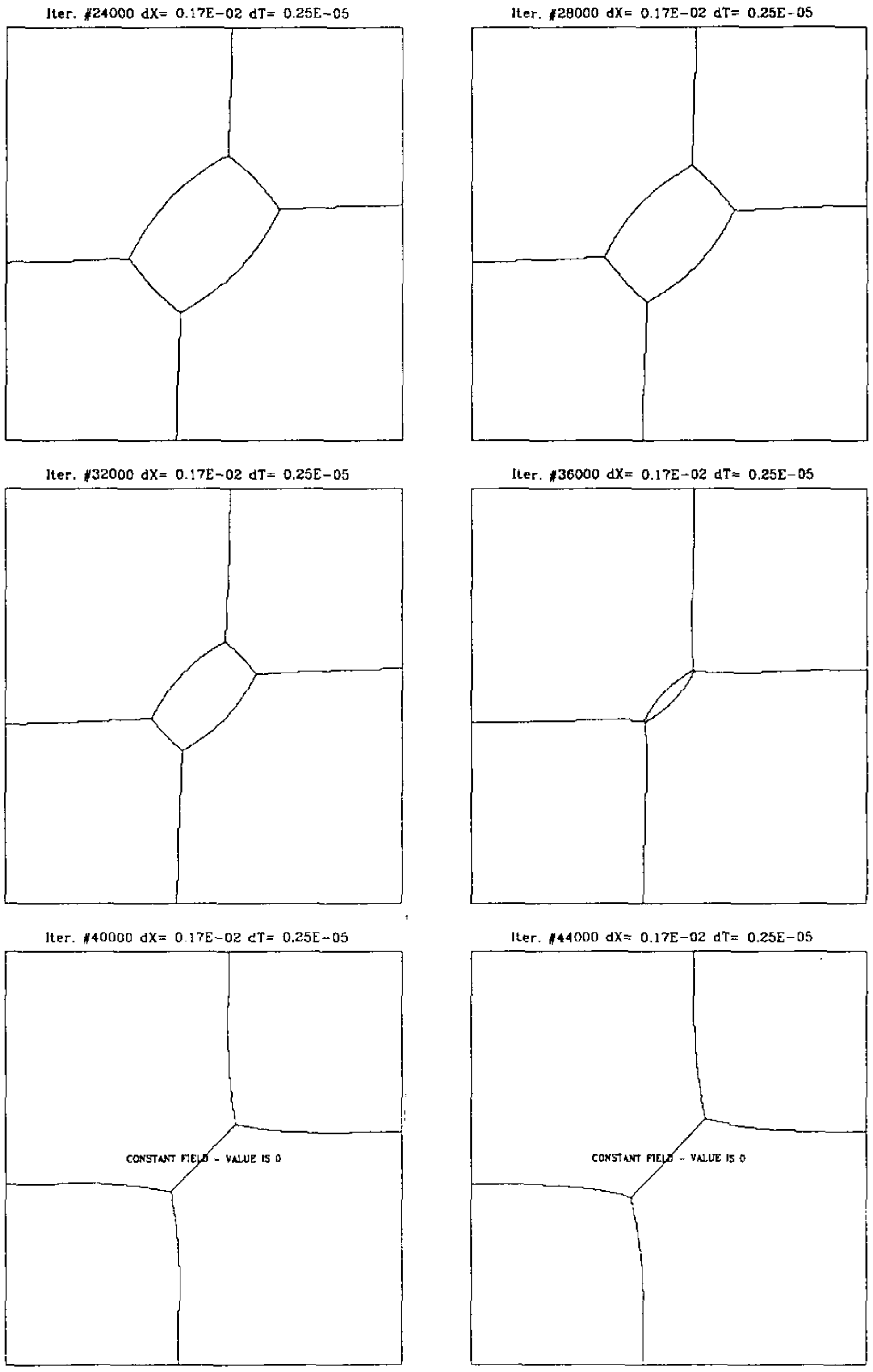

Iter. #24000 dX= 0.17E-02 dT= 0.25E-05
Iter. #28000 dX= 0.17E-02 dT= 0.25E-05
Iter. #32000 dX= 0.17E-02 dT= 0.25E-05
Iter. #36000 dX= 0.17E-02 dT= 0.25E-05
Iter. #40000 dX= 0.17E-02 dT= 0.25E-05
CONSTANT FIELD - VALUE IS 0
Iter. #44000 dX= 0.17E-02 dT= 0.25E-05
CONSTANT FIELD - VALUE IS 0

FIG. 9. Motion of triple junctions under curvature (cont.).

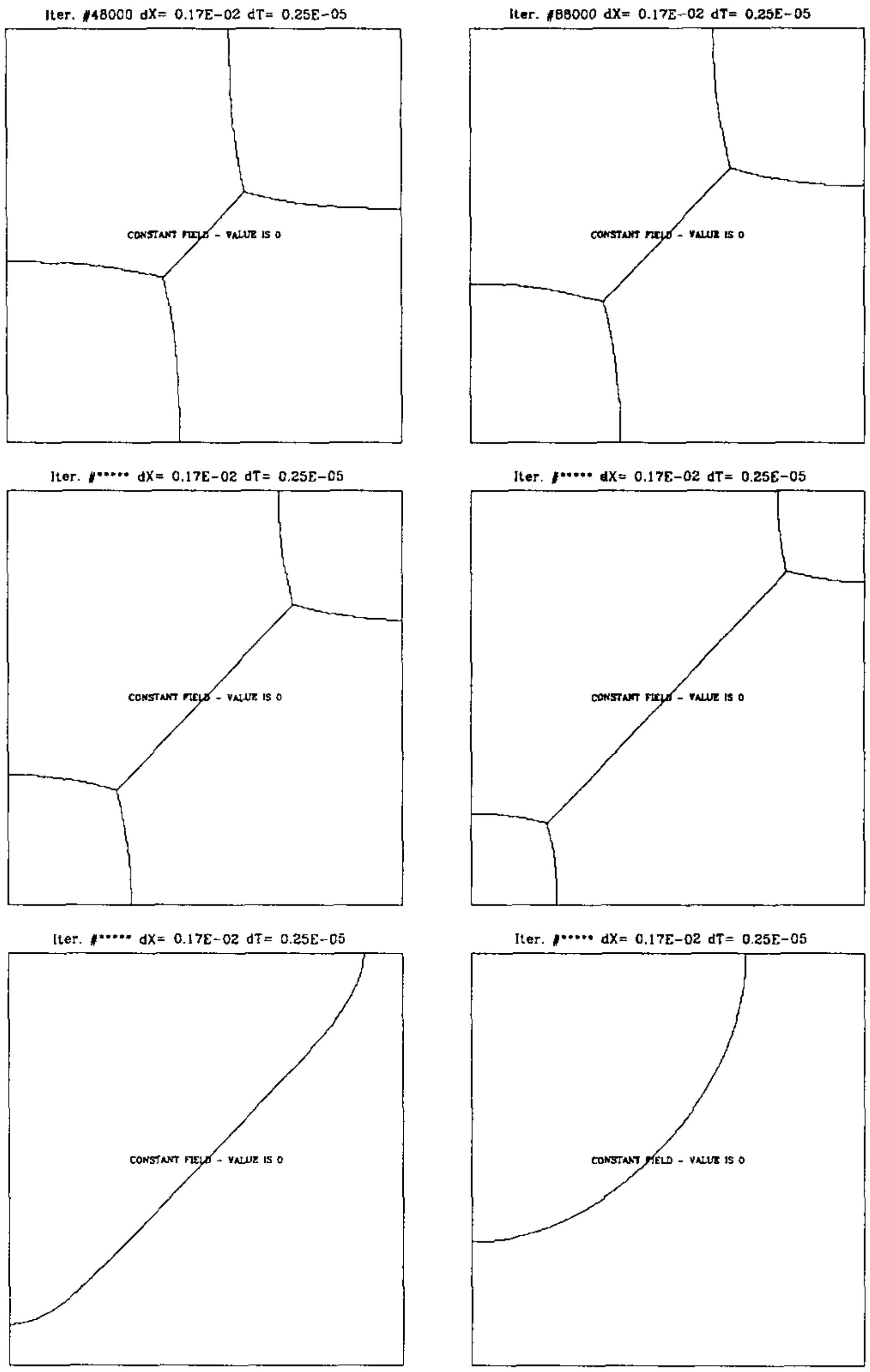


FIG. 10. Motion of triple junctions under curvature (cont.).

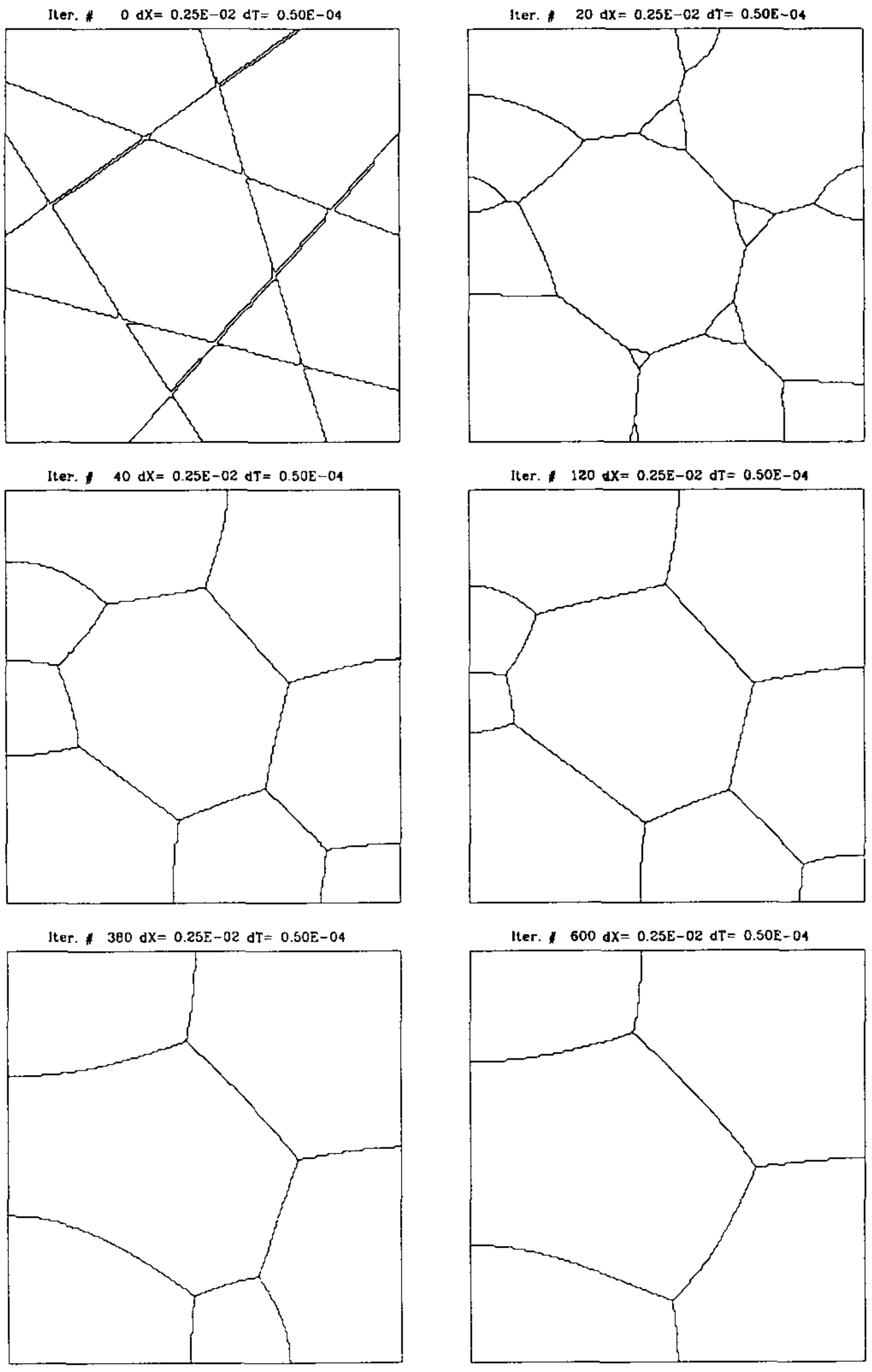


FIG. 11. Motion of several regions under curvature.

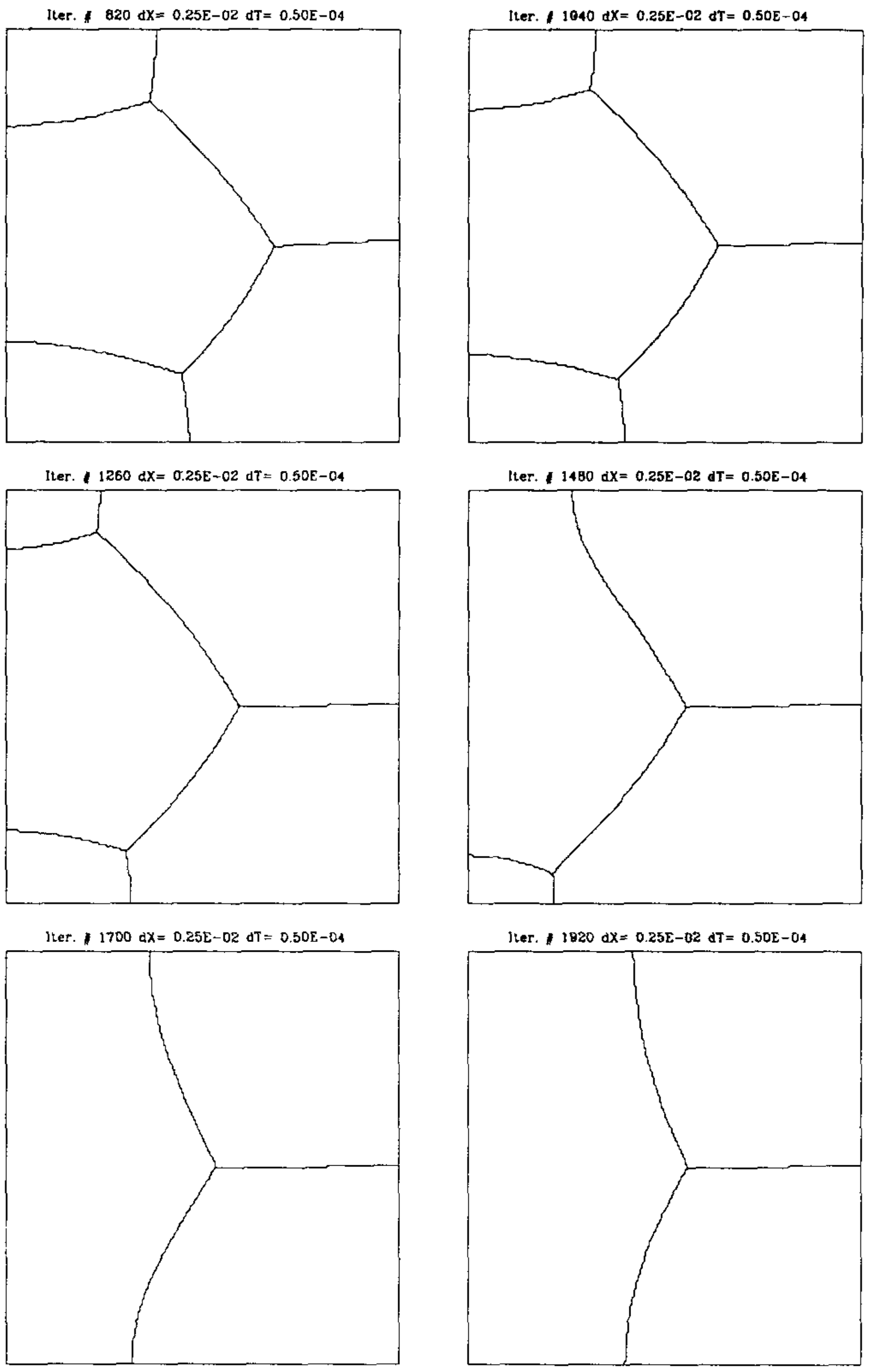

Iter. # 820 dX= 0.25E-02 dT= 0.50E-04
Iter. # 1040 dX= 0.25E-02 dT= 0.50E-04
Iter. # 1260 dX= 0.25E-02 dT= 0.50E-04
Iter. # 1480 dX= 0.25E-02 dT= 0.50E-04
Iter. # 1700 dX= 0.25E-02 dT= 0.50E-04
Iter. # 1920 dX= 0.25E-02 dT= 0.50E-04

FIG. 12. Motion of several regions under curvature (cont.).

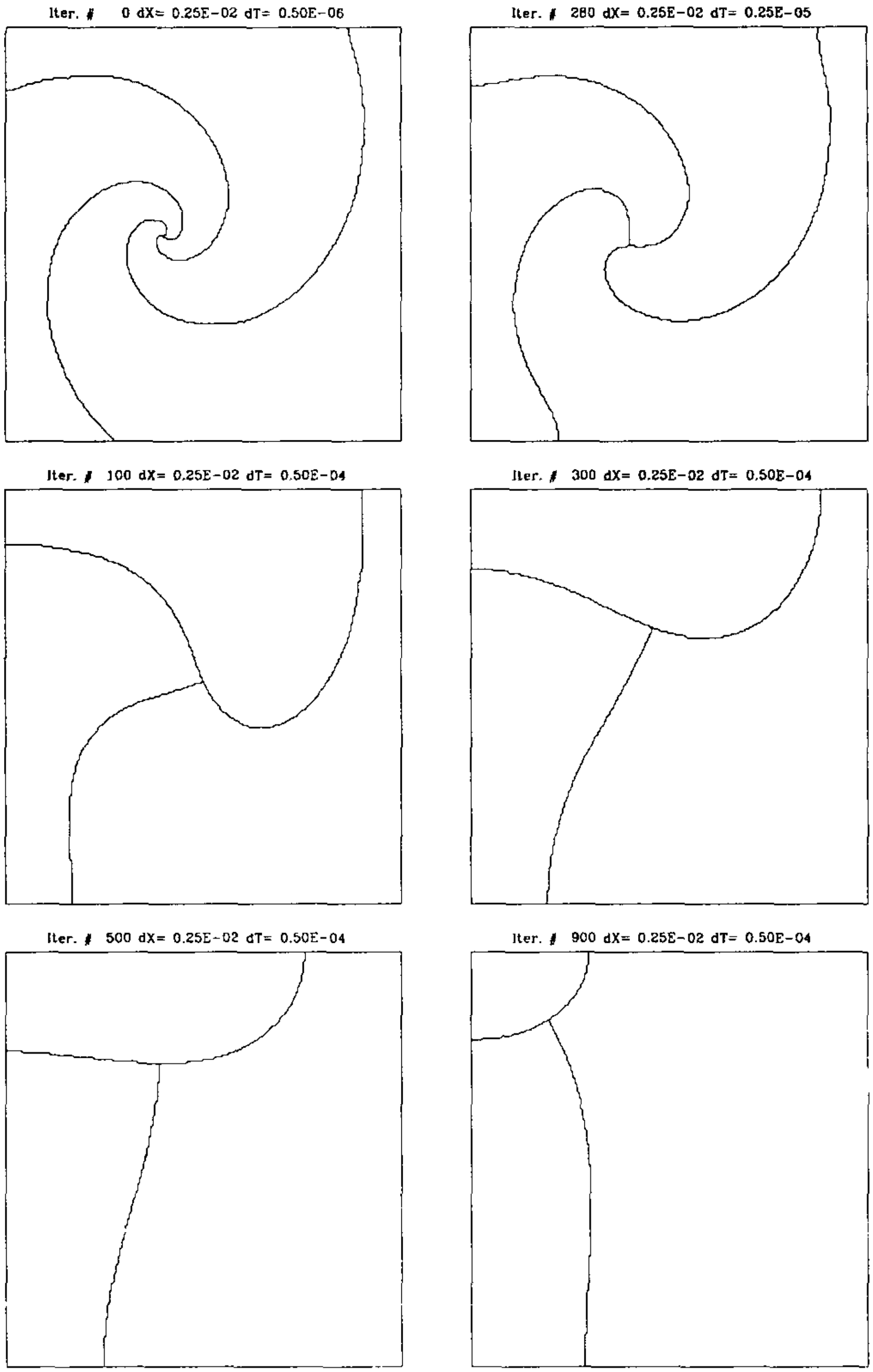

Iter. # 0 dX= 0.25E-02 dT= 0.50E-06
Iter. # 280 dX= 0.25E-02 dT= 0.25E-05
Iter. # 100 dX= 0.25E-02 dT= 0.50E-04
Iter. # 300 dX= 0.25E-02 dT= 0.50E-04
Iter. # 500 dX= 0.25E-02 dT= 0.50E-04
Iter. # 900 dX= 0.25E-02 dT= 0.50E-04

FIG. 13. Motion of spiral with "T" stable shape.

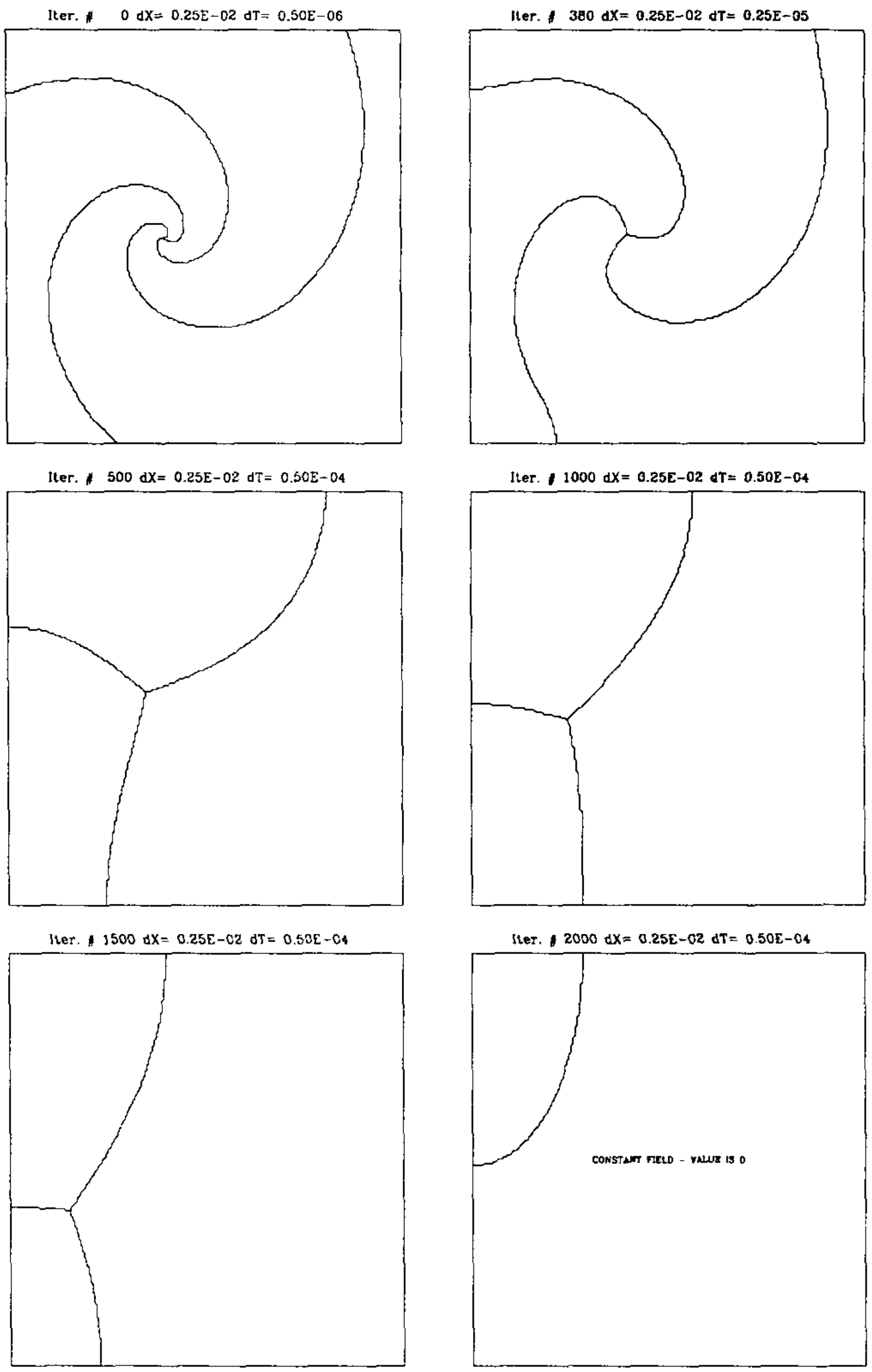


FIG. 14. Motion of spiral with "Y" stable shape.

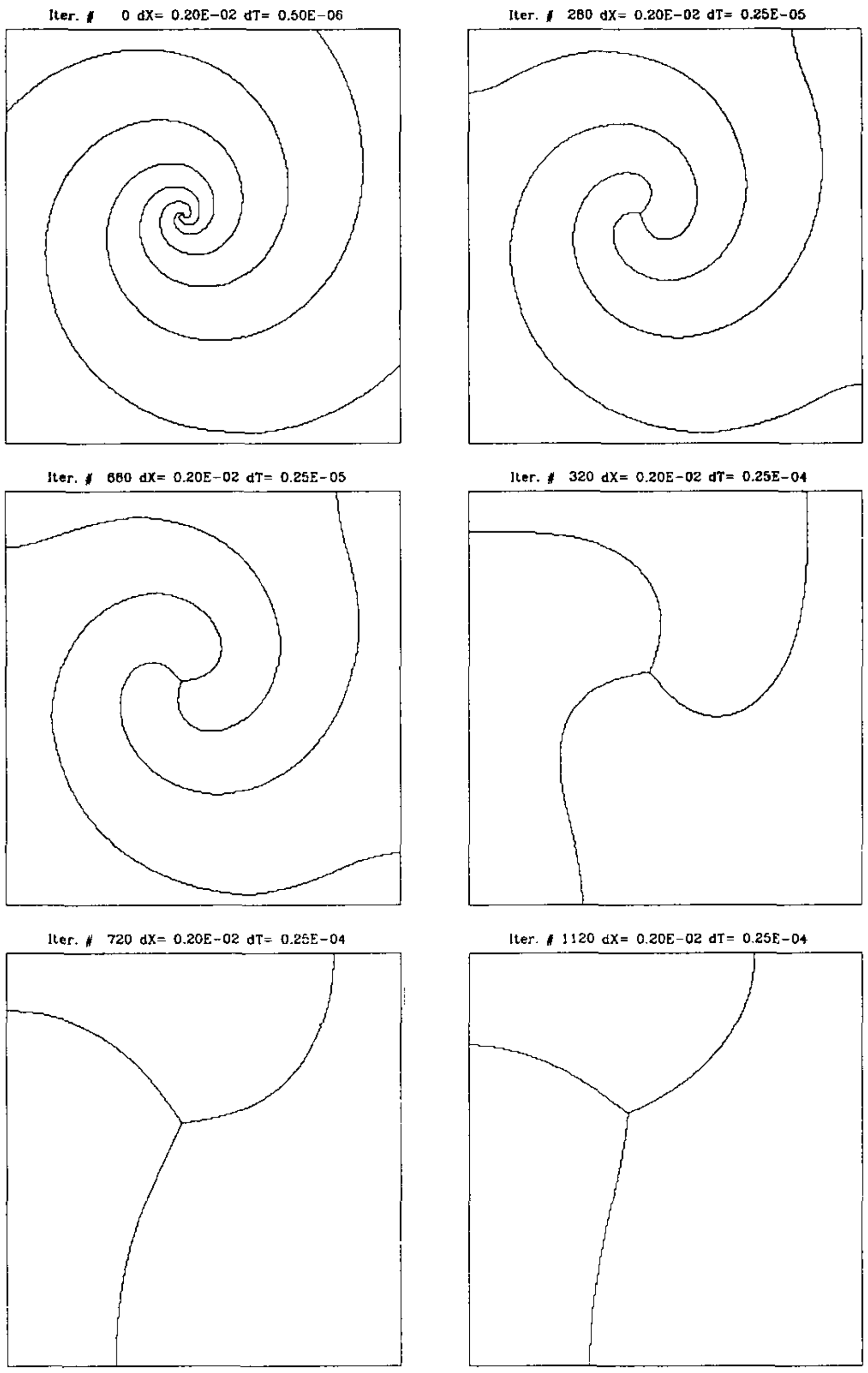

Iter. # 0 dX= 0.20E-02 dT= 0.50E-06
Iter. # 280 dX= 0.20E-02 dT= 0.25E-05
Iter. # 660 dX= 0.20E-02 dT= 0.25E-05
Iter. # 320 dX= 0.20E-02 dT= 0.25E-04
Iter. # 720 dX= 0.20E-02 dT= 0.25E-04
Iter. # 1120 dX= 0.20E-02 dT= 0.25E-04

FIG. 15. Motion of double spiral with "Y" stable shape.

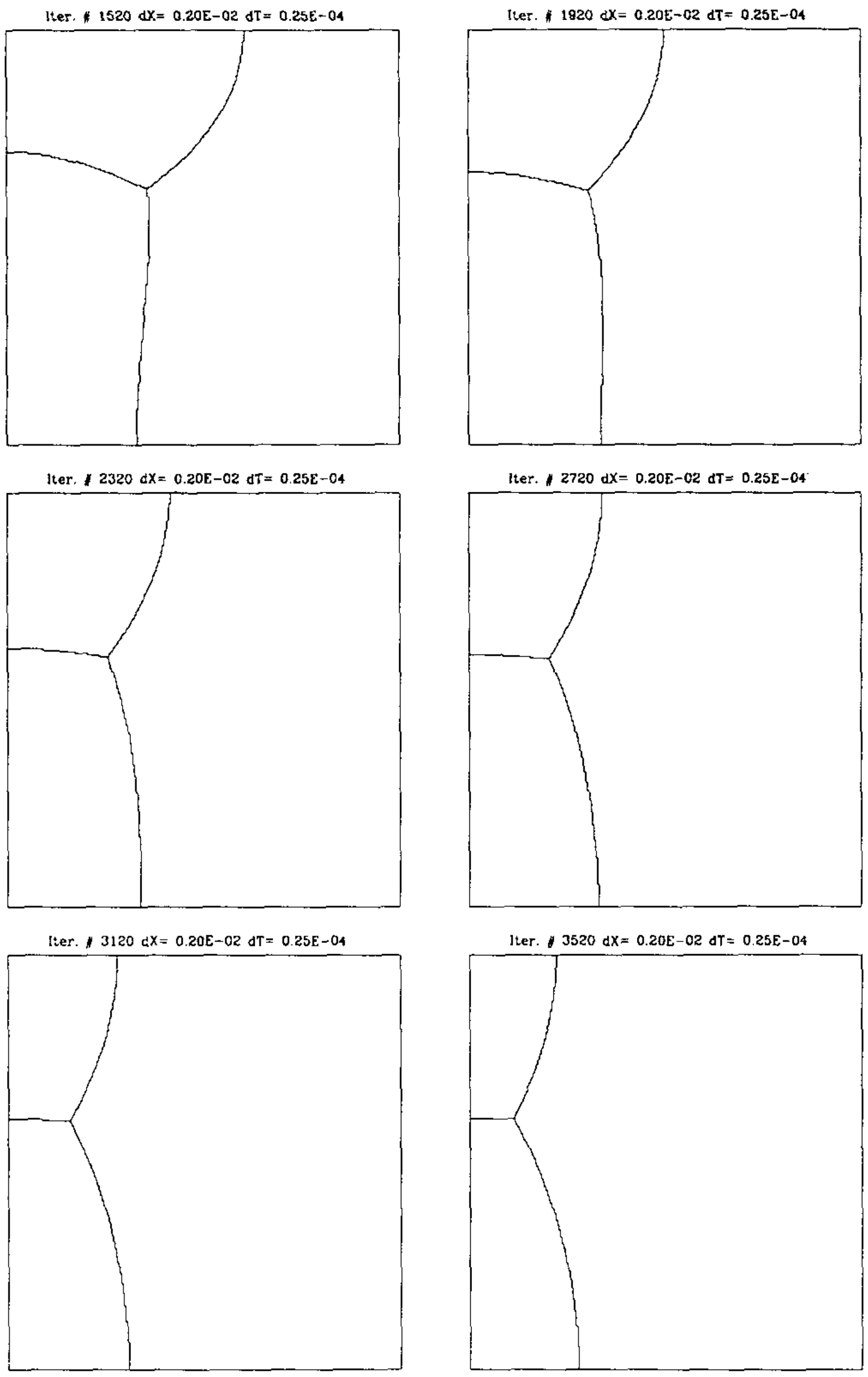

Iter. # 1520 dX= 0.20E-02 dT= 0.25E-04
Iter. # 1920 dX= 0.20E-02 dT= 0.25E-04
Iter. # 2320 dX= 0.20E-02 dT= 0.25E-04
Iter. # 2720 dX= 0.20E-02 dT= 0.25E-04
Iter. # 3120 dX= 0.20E-02 dT= 0.25E-04
Iter. # 3520 dX= 0.20E-02 dT= 0.25E-04

FIG. 16. Motion of double spiral with "Y" stable shape.

3. A COUPLED OSHER–SETHIAN METHOD

3.1. Original Osher–Sethian

The original Osher–Sethian algorithm [15] is as follows: given an initial hypersurface Γ_0, choose a continuous $\psi: \mathbb{R}^n \mapsto \mathbb{R}$ such that

$$\Gamma_0 = \{x \in \mathbb{R}^n : \psi(x) = 0\};$$

then solve

$$\begin{aligned} \phi_t &= F\,|\nabla\phi| \\ \phi(x, 0) &= \psi \end{aligned} \qquad (5)$$

and define

$$\Gamma(t) = \{x : \phi(x, t) = 0\}.$$

The normal velocity of $\Gamma(t)$ is given by F. It has been proven [4, 8] that if $F = \kappa$ then the above definition of $\Gamma(t)$ agrees with the classical notion of motion by mean curvature, as long as it exists. It has also been shown that any continuous ψ may be used; a typical choice is distance to Γ_0:

$$\psi(x) = \begin{cases} \operatorname{dist}(x, \Gamma_0) & \text{if } x \text{ "inside" } \Gamma_0 \\ -\operatorname{dist}(x, \Gamma_0) & \text{if } x \text{ "outside" } \Gamma_0. \end{cases}$$

We use this initialization for all our computations.

This approach is very appealing because of the ease with which it handles difficult numerical situations—topological merging, breaking, etc. No special action need be taken in the event of such topological changes; a contour plotter finds $\Gamma(t)$ as it evolves. It does, however, require that there be no more than two distinct regions involved. That is, there must be an "inside" and an "outside" which the interface separates. If this is not true, then we cannot choose an appropriate ψ as above. As an example, consider the initial data in Fig. 17. Suppose we choose ψ to be distance, having the sign indicated in the figure. The Osher–Sethian algorithm produces a $\Gamma(t)$ as on the right. We see that $\Gamma(t)$ develops an interior, which we do not want. So as described Osher–Sethian does not apply to triple junctions.

3.2. Coupling

It seems reasonable that we could assign each region a separate function ϕ_i and then evolve with Osher–Sethian. There are two approaches we might take initially. First, we could write down a system of coupled Hamilton–Jacobi equations for the ϕ_i. Second, we could evolve each ϕ_i *independently* according to the original Osher–Sethian idea, and then periodically interact the values of the ϕ_i in some way, as was done for the DGCDM algorithm. The first approach was not successful.

The second method requires us to decide how the ϕ_i should interact. At the very least, any interaction we choose must not move the initial configuration in Fig. 18 when the velocity is proportional to the curvature. We use this fact as a guide in deciding upon an interaction step for the ϕ_i.

If we apply Osher–Sethian to each individual ϕ_i in the picture on the left in Fig. 18, then after a time the boundaries will look like the picture on the right in the same figure. Near the edges where the individual boundaries nearly touch, we have that some pair of ϕ_i are nearly equal, while the remaining ϕ is much smaller. For example, near point A, we have $\phi_1 \approx \phi_2$, and we are far from the boundary of ϕ_3 so ϕ_3 is much smaller than either ϕ_1 or ϕ_2. The same is true at points B and C, where $\phi_2 \approx \phi_3$, and $\phi_1 \approx \phi_3$, respectively. The dotted line represents the original positions of the zero-level sets, and we would like these to be the zero level sets after the interaction is complete. Along the dotted line, we have $\phi_1 = \phi_2$ near A, $\phi_1 = \phi_3$ near C, and $\phi_2 = \phi_3$ near B because each ϕ_i represents distance from the zero-level set. Therefore a correct interaction in this case would be

$$\phi_i = \phi_i - \max_{i \neq j} \phi_j. \qquad (6)$$

This forces each ϕ_i to have the zero-level set that it started with, which holds the triple point in place. So the numerical method is

ALGORITHM A.

(1) For each of n regions, initialize ϕ_i, $i = 1, ..., n$, with distance to the boundary of the ith region.

(2) Solve (5) with ϕ_i, $i = 1, ..., n$, as initial data up to time T^*.

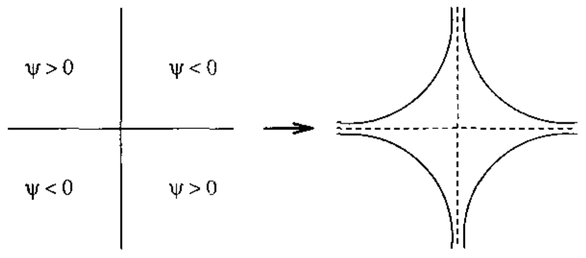


FIG. 17. Development of an interior.

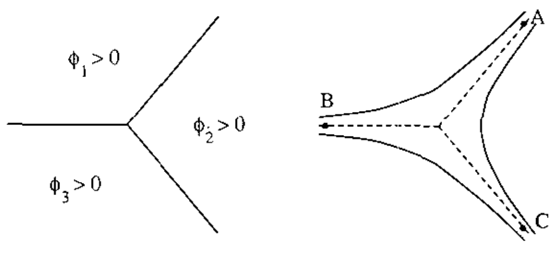


FIG. 18. Deriving the interaction step.

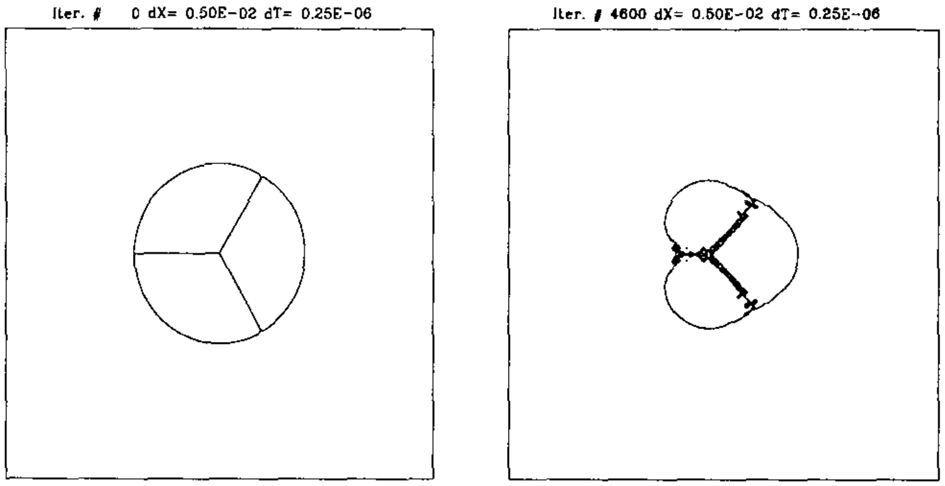


FIG. 19. Applying Algorithm A.

(3) For $i = 1, ..., n$, compute

$$\phi_i^{\text{new}} = \phi_i - \max_{i \neq j} \phi_j.$$

(4) For $i = 1, ..., n$, set $\phi_i = \phi_i^{\text{new}}$.

(5) Return to step 2.

We retain the generality of the original Osher–Sethian method in that we may easily specify the velocity of the level sets. The previous methods can theoretically be modified to provide different velocities, but the procedure is not direct, as the Osher–Sethian method is. Also, no errors are made in specifying T^*. That is, the DGCDM algorithm relies on an approximation whose accuracy is controlled by the choice of T_{chop}, whereas (5) actually moves level sets with normal velocity F.

3.3. Degenerate Level Sets

Algorithm A has a side effect. It is clearly seen by considering an example. Look at Fig. 19 in which the initial configuration is shown on the left. After 23 steps of Algorithm A, the picture on the right results. There appears to be some kind of instability in the algorithm. The problem is that, although (6) correctly locates the zero-level sets of the ϕ_i, it also changes the character of each ϕ_i near the zero-level set.

Consider ϕ_4 in Fig. 20. Since ϕ_4 represents the exterior of the shape, ϕ_4 is a cone. We have $\phi_4 < 0$ within the circle and $\phi_4 > 0$ outside the circle. When we apply (6) to ϕ_4, we subtract larger values from ϕ_4 at points A, B, C than at points D, E, F. In fact, at D, E, F we have $\max_{1 \leqslant i \leqslant 3} \phi_i = 0$ while at A, B, C we have $\max_{1 \leqslant i \leqslant 3} \phi_i > 0$, since each point is within a region where some $\phi_i > 0$. The unintended effect of (6) is to create new local extrema in ϕ_4.

The cumulative effect of Algorithm A is to create a level set in ϕ_4 that is shaped like the original figure. This level set is very close to the zero level set of ϕ_4. Thus, ϕ_4 eventually acquires a level set similar to that in Fig. 17 and Osher–Sethian no longer applies.

We can avoid this problem by observing that we do not have to set $\phi_i = \phi_i^{\text{new}}$ in the algorithm. Instead, all we need to do is set ϕ_i equal to a continuous function whose zero-level set coincides with that of ϕ_i^{new}. The simplest such function is again distance. That is, we replace $\phi_i = \phi_i^{\text{new}}$ in Algorithm A by

(1) For $i = 1, ..., n$

- Generate a discrete representation of the zero level set of ϕ_i^{new}.
- for each point on the grid, compute the distance to the zero-level set using the above representation.
- set ϕ_i equal to the computed distance.

With this new algorithm, the initial data in Fig. 19 evolves as in Fig. 21 (see also Fig. 22).

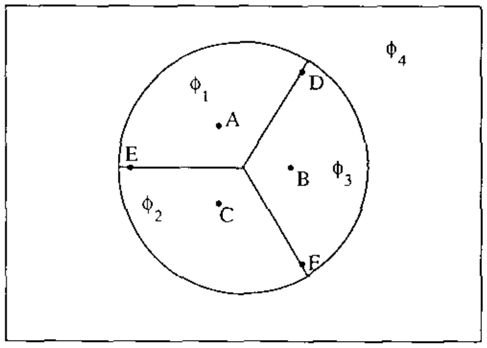


FIG. 20. Figure 19, with ϕ_i marked.

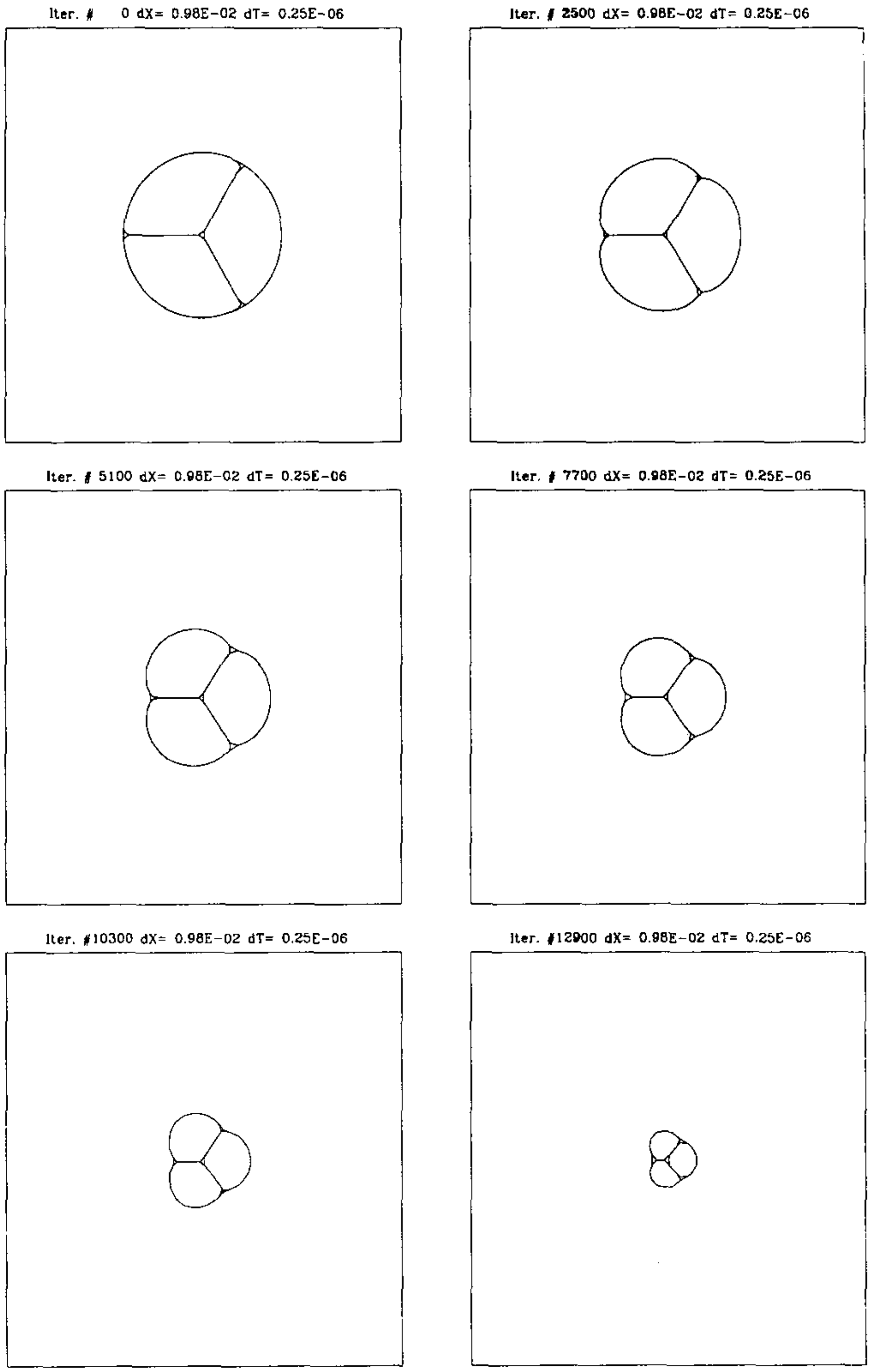


FIG. 21. Evolution under new algorithm.

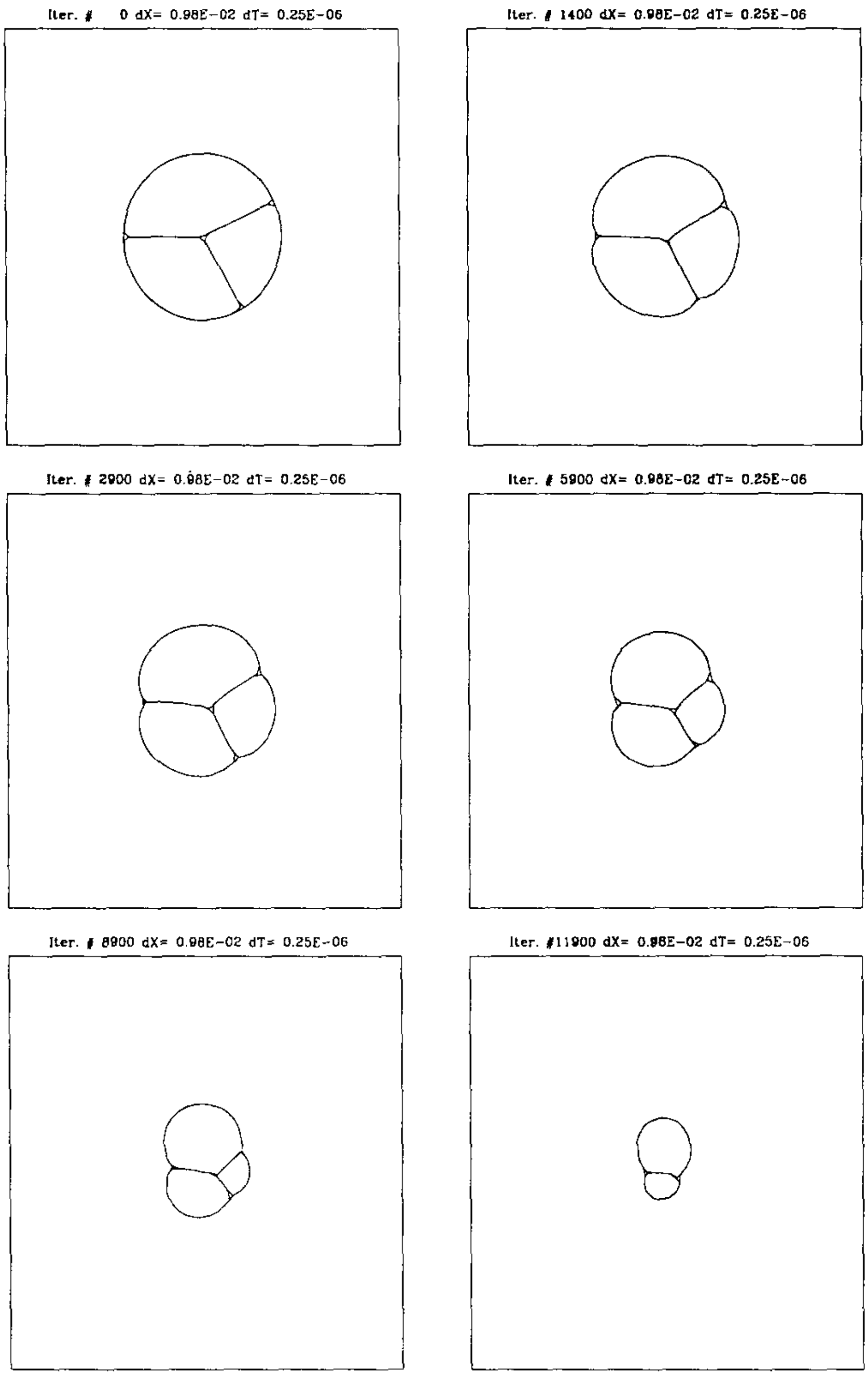

Iter. # 0 dX= 0.98E-02 dT= 0.25E-06
Iter. # 1400 dX= 0.98E-02 dT= 0.25E-06
Iter. # 2900 dX= 0.98E-02 dT= 0.25E-06
Iter. # 5900 dX= 0.98E-02 dT= 0.25E-06
Iter. # 8900 dX= 0.98E-02 dT= 0.25E-06
Iter. #11900 dX= 0.98E-02 dT= 0.25E-06

FIG. 22. Evolution under new algorithm.

3.4. Constant Velocities

The choice of curvature as velocity is a special one because the curvature of ϕ is equal to the curvature of $-\phi$. Therefore it does not matter which sign of ϕ we choose to denote the "inner" region. So in the case of a circle shrinking under its curvature, we may use either $\phi < 0$ or $\phi > 0$ inside the circle and the computed motion is the same.

This is not true for general velocities. If we substitute $-\phi$ into (5), then we must have $F(-\phi) = -F(\phi)$ for (5) to be invariant under the substitution. In particular, if the velocity is a positive constant, then a circle will expand if $\phi > 0$ within the circle and shrink if $\phi < 0$ there. So, for example, if we want to use the method of the previous section to propagate a line at a constant velocity c we need ϕ_1 and ϕ_2, one for each side of the line, and we must use $F_1 = c$ for ϕ_1 and $F_2 = -c$ for ϕ_2 when solving (5). Otherwise, the zero-level sets will move in opposite directions; the update step will not produce the correct position of the line.

Now suppose we want to assign constant velocities to each of the arms of a triple junction (see Fig. 23). Since the velocities are usually unequal we immediately see that we must use a discontinuous F_i for each ϕ_i. In addition there is the question of how to implement the transition from one velocity to the next at the triple junction.

A discontinuous F in (5) produces a discontinuous ϕ. The original Osher–Sethian algorithm expects a continuous velocity. But we can still apply Algorithm B because of the reconstructive step introduced in the previous section. Any discontinuities introduced in ϕ will be eliminated when we reinitialize it. We need merely choose an implementation for a discontinuous F; e.g., for Fig. 23 we can solve (5) with

$$F_2 = \begin{cases} c_1 & \text{if } \phi_1 < \phi_3 \\ c_2 & \text{if } \phi_3 < \phi_1 \end{cases}$$

as the velocity function for ϕ_2. The dotted line represents the position at which $\phi_1 = \phi_3$; this is the line along which F_2 is discontinuous.

In general, the velocity for ϕ_i depends on the relative sizes of the remaining ϕ:

$$F_i = F_{ij} \quad \text{if } \phi_j \text{ greater than all other } \phi.$$

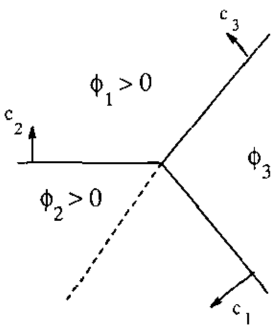


FIG. 23. ϕ_2 requires a discontinuous velocity function.

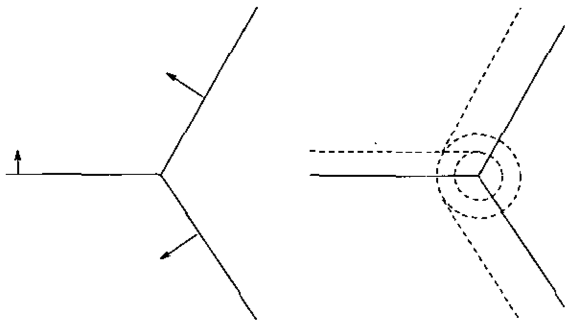
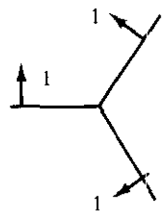
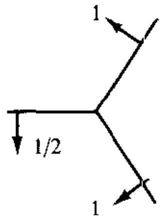
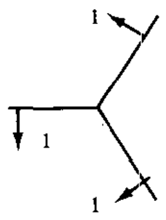


FIG. 24. Applying Huygens' principle.

The motion of triple junctions under constant velocities has been studied by Taylor [17]. She develops a method which is an expression of Huygens' principle. Consider Fig. 24. For the initial configuration on the left, Taylor proceeds by constructing a representation of the position of the front at time t. Each respresentation is constructed independently of the others by first translating the ith arm by a distance $v_i t$ and then extending the translation into a circle of radius $v_i t$, as shown in the figure. Then for each pair of regions a point that represents the meeting of the regions after time t has elapsed is found. If the velocities are consistent there will be a common point after each pair of regions

TABLE I

Assigned Velocities for Figs. 25–27

Fig. 25	
Fig. 26	
Fig. 27	

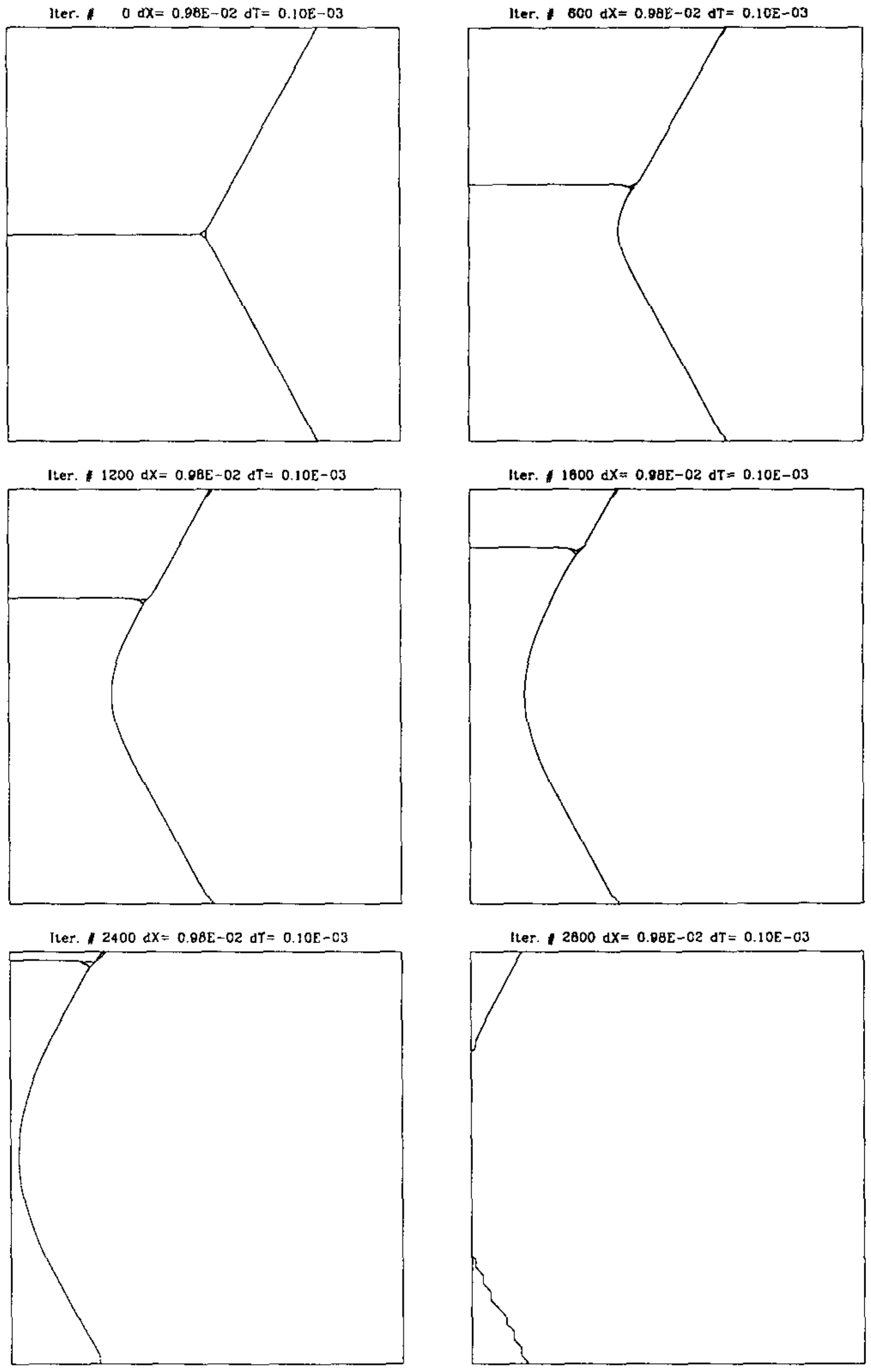


FIG. 25. Evolution under new algorithm.

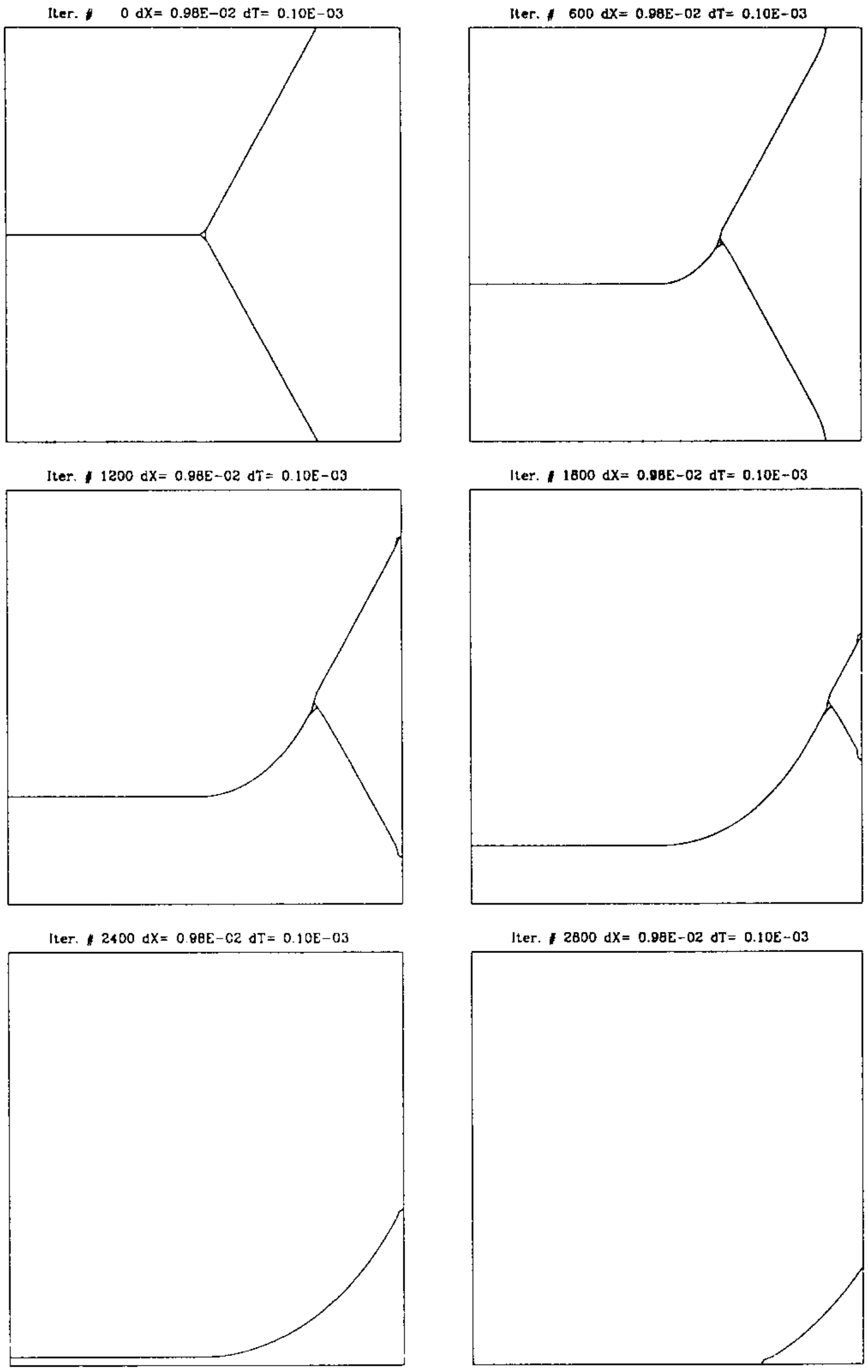

Iter. # 0 dX= 0.98E-02 dT= 0.10E-03
Iter. # 600 dX= 0.98E-02 dT= 0.10E-03
Iter. # 1200 dX= 0.98E-02 dT= 0.10E-03
Iter. # 1800 dX= 0.98E-02 dT= 0.10E-03
Iter. # 2400 dX= 0.98E-02 dT= 0.10E-03
Iter. # 2800 dX= 0.98E-02 dT= 0.10E-03

FIG. 26. Evolution under new algorithm.

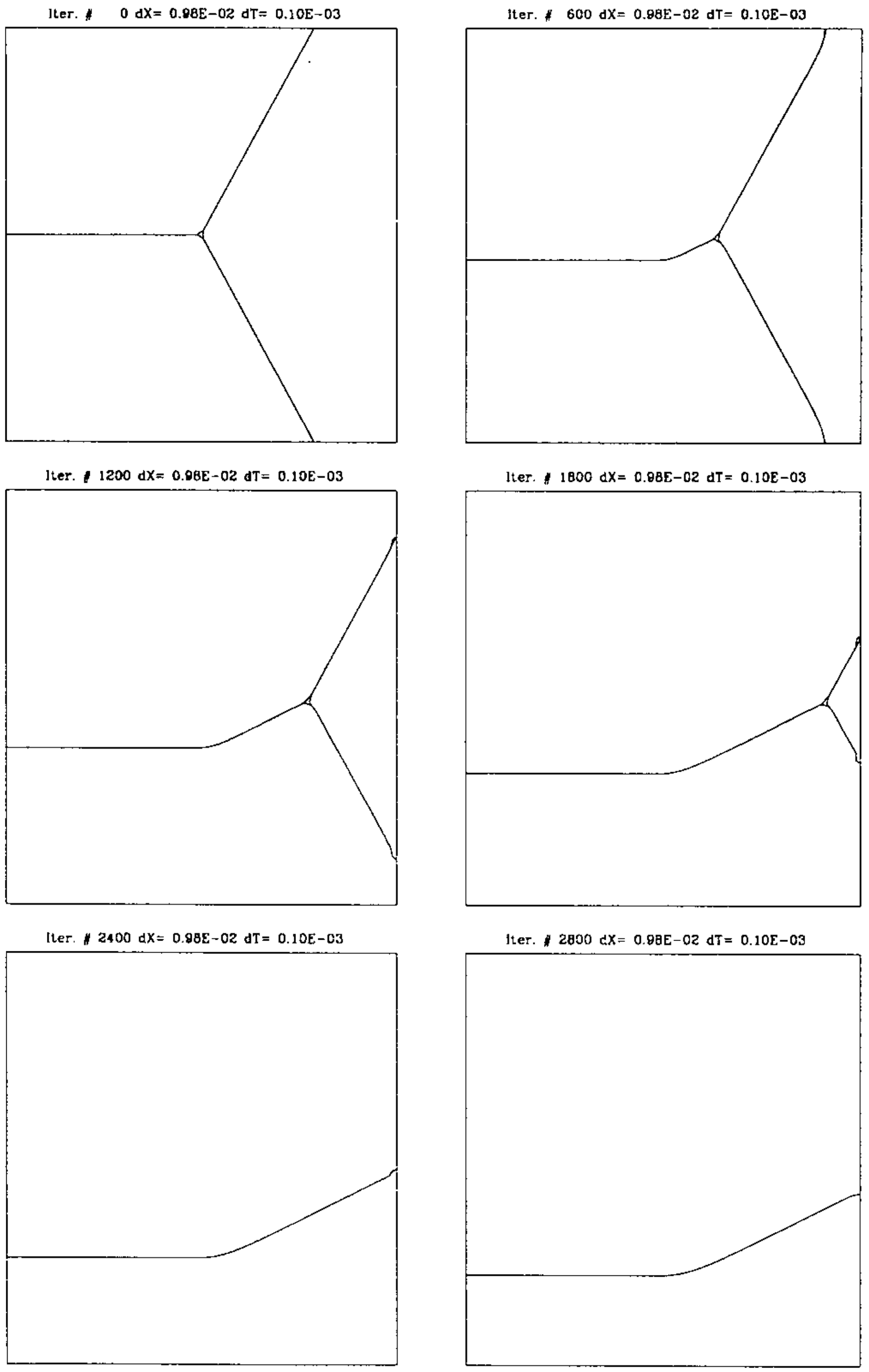


FIG. 27. Evolution under new algorithm.

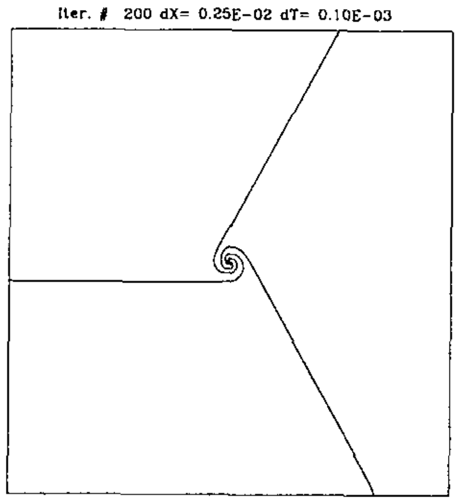


FIG. 28. Development of a spiral.

is examined. The construction of these points follows the idea that the maximal intrusion wins. That is, of the possible candidates for the meeting point, the one that represents the greatest penetration of the faster phase is the point chosen. But this is precisely what (6), the update step, expresses. Our computations agree with the hand drawn results of Taylor's algorithm [17]. Figure 25 shows a computation corresponding to Fig. 24; iteration 600 is in close agreement with Fig. 24. Other computations follow on succeeding pages. The prescribed velocities for the computations are given in Table I (see also Figs. 26 and 27).

Taylor also claims that certain velocities do not produce a well-posed problem. One such situation is when all velocities are equal and (counter)clockwise. We may still attempt a computation in this case, and we find that a spiral develops (Fig. 28). The computation may not proceed farther for the spiral will tighten beyond the resolution of the grid. The resulting figure is interesting and further study is needed to understand how perturbations of the velocities would affect the formation of the spiral.

4. SYSTEMS OF REACTION–DIFFUSION EQUATIONS

Recently attention has been given to fast reaction, slow diffusion equations as a means of moving fronts or as part of a system describing dynamical behavior (e.g., crystal growth, see [11]). An example of this type of equation in one dimension is

$$u_t = \varepsilon\, \Delta u - \frac{1}{\varepsilon} f_u(u), \tag{7}$$

where $\varepsilon > 0$ is a small parameter.

In Section 4.1, we examine a system of reaction–diffusion equations proposed to model triple junctions. Some computational difficulties are noted, and then in Section 4.3 we study Eq. (7) with various initial data to gain insight into the numerical problems with a system of such equations.

4.1. A Proposed Model

Bronsard and Reitich [2] propose the following model for the study of triple junctions:

$$\begin{gathered} u_t = \varepsilon\, \Delta u - \frac{1}{\varepsilon} \nabla_u W(u) \\ \text{on} \quad \Omega \subset \mathbb{R}^n \\ u: \Omega \times \mathbb{R}^+ \mapsto \mathbb{R}^m. \end{gathered} \tag{8}$$

Boundary conditions are of Dirichlet or homogeneous Neumann type. Here, $W: \mathbb{R}^m \mapsto \mathbb{R}$ is a non-negative function which has three minima **a**, **b**, **c**, at which $W = 0$. An example is $W(\mathbf{u}) = |\mathbf{u}-\mathbf{a}|^2\, |\mathbf{u}-\mathbf{b}|^2\, |\mathbf{u}-\mathbf{c}|^2$. W is a "triple well potential"; it has three wells, one for each phase. A contour plot of the example W above with selected wells is shown in Fig. 29. Sample initial data for $\mathbf{u}(x, t)$ in the case of Fig. 1 is shown in Fig. 30.

Bronsard and Reitich suggest via formal asymptotics that $\mathbf{u}(x, t)$ separates Ω into three regions in which $\mathbf{u} \approx \mathbf{a}, \mathbf{b}, \mathbf{c}$, that there is a sharp transition layer between each region, and that each transition layer moves with normal velocity $\varepsilon\bar{\kappa}$. They also derive an expression which determines the angles at which the interfaces meet at the triple junction:

$$\frac{\sin(\theta_1)}{\Phi^{ba}} = \frac{\sin(\theta_2)}{\Phi^{bc}} = \frac{\sin(\theta_3)}{\Phi^{ca}}. \tag{9}$$

Here, Φ^{xy} represents the energy required to make the transition from phase x to phase y. Each Φ is the solution of a minimization problem involving an integral of $W(u)$; the

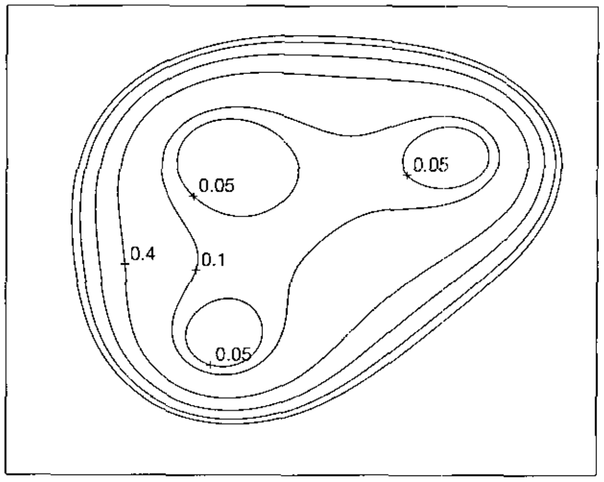


FIG. 29. Contours of $W(\mathbf{u})$; triple well potential.

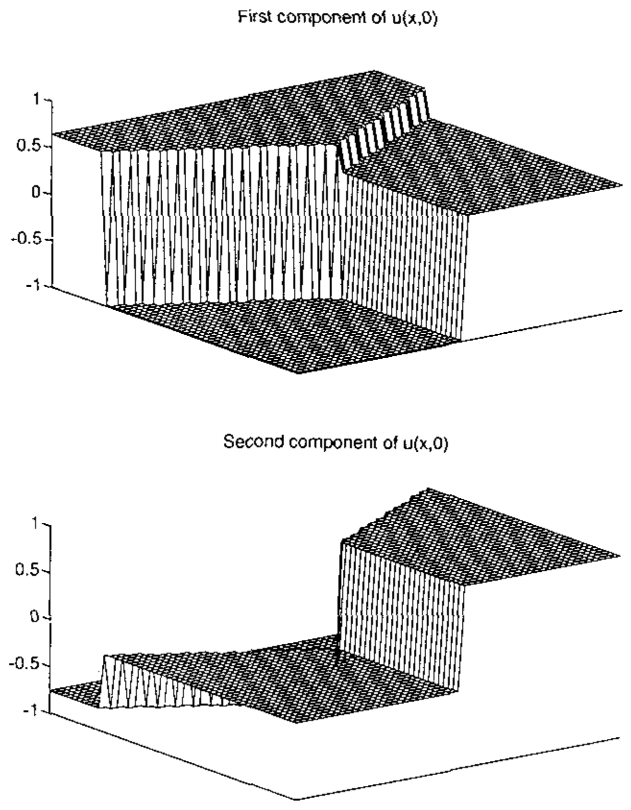


FIG. 30. Initial data for Eq. (8), Fig. 1.

minimization is over all C^1 paths in $\mathbb{R}^m$ connecting two minima of W. The key observation for our purposes here is that if $W(u)$ is symmetric then all Φ^{xy} are equal and hence we must have $\theta_1 = \theta_2 = \theta_3 = 120°$.

Thus, if we choose

$$\mathbf{a} = \left(\cos\left(\frac{\pi}{4}\right), \sin\left(\frac{\pi}{4}\right)\right)$$
$$\mathbf{b} = \left(\cos\left(\frac{11\pi}{12}\right), \sin\left(\frac{11\pi}{12}\right)\right)$$
$$\mathbf{c} = \left(\cos\left(\frac{19\pi}{12}\right), \sin\left(\frac{19\pi}{12}\right)\right)$$

with the example W given earlier, then all angles are equal to 120°.

Equation (8) consists of competing processes: the diffusive term will widen the transition regions, while the reactive term will narrow them. The interfaces will move when points are able to make the transition from one minimum to another.

4.2. Computations

The grid chosen was $\Omega = [-0.1, 0.1] \times [-0.1, 0.1]$; the discretization is uniform in both directions. A smaller Ω was chosen to achieve smaller stepsizes and also to focus attention on the motion of the junction. The velocities here are small; a focused grid shortens the time required to make a computation. We use a basic five-point discretization of the laplacian and the explicit Euler method in time. No special discretization is needed for W.

Boundary conditions will affect the motion of the triple point. If we use homogeneous Neumann conditions, then curvature will be introduced at the boundary since the initial interfaces typically do not satisfy the boundary conditions. If we use Dirichlet conditions and the velocities are constants, then the interfaces will be unable to move along the boundary. In most of our experiments we will use homogeneous Neumann conditions. This allows the interfaces to move along the boundary when the velocities are constants. Also, there is still a time interval in which the motion of the triple point is not corrupted by the boundary-generated curvature.

Our first experiment verifies a steady state solution from earlier sections. Dirichlet boundary conditions are used. For initial data, we used Fig. 30; the initial angles at the triple point are 120°. Since the curvature along each arm is zero and the triple junction initially satisfies Eq. (9), we do not expect any motion. This is indeed the case for the coarsest grid and largest ε: $(\frac{1}{250}, \frac{1}{100})$ as well as the finest grid and smallest ε: $(\frac{1}{1000}, \frac{1}{1000})$.

The next experiment starts with angles that are out of equilibrium. Here we use $\Delta x = \frac{1}{1000}$. The angles each initial interface makes with the x-axis are 180°, 30°, and 315°. Intuition suggests that the initial direction of the triple junction's motion should be in the direction of the vector sum of the interfaces' initial meeting. This is also the case as the triple point moves down and to the right. It continues down and eventually moves off the grid.

Now we set $\varepsilon = 0.0005$ and repeat the above computation. In this case, we expect that the motion should be slower, but we actually observe that the motion terminates. Calculations are carried out in double precision, and a non-constant steady state solution of Eq. (8) is obtained in which all three interfaces clearly possess non-zero curvature, in violation of the theory.

4.3. A Study of One Reaction–Diffusion Equation

In an attempt to better understand the behavior of the numerical solution of (8), we study a simpler problem first. In this section, we examine in detail a single equation, (7), with $f_u = u^3 - u$. For certain initial data, an exact solution may be found to which we may compare our computations.

4.3.1. *Numerical Experiments*

For numerical experiments, we have $f_u(u) = (u+1)(u)(u-1)$. The computational grid is $\Omega = [-\frac{1}{4}, \frac{1}{4}] \times$

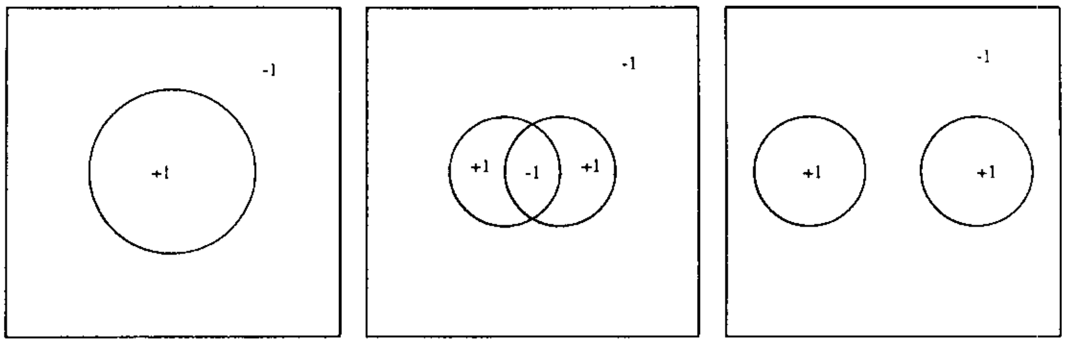


FIG. 31. Types of initial data.

$[-\frac{1}{4}, \frac{1}{4}]$. We use a larger Ω here because we can compute an exact solution below in which the interface will move quickly and we want to study its motion over a longer time than would be possible with the Ω of Section 4.2. We worked with three different types of initial data—a single circle, two disjoint circles, and two overlapping circles (see Fig. 31).

The initial data assumes the values in the regions indicated. We began by employing a straightforward discretization of (7):

$$u_{ij}^{n+1} = u_{ij}^n + \frac{\varepsilon\,\Delta t}{\Delta x^2}[\Delta_+^x \Delta_-^x + \Delta_+^y \Delta_-^y]\, u_{ij}^n - \frac{\Delta t}{\varepsilon} f_u(u_{ij}^n). \tag{10}$$

From [3], we know that the velocity is, up to $\mathcal{O}(\varepsilon^2)$, $\varepsilon\kappa$, where κ is the curvature of the interface. The qualitative behavior of regions like the initial data in Fig. 31 under this flow is well understood [10]. The regions in which $u \approx 1$ should shrink and eventually vanish, leaving $u \approx -1$ throughout Ω.

Our emphasis is on the case of a single circle, for if we let $r(t)$ denote the radius of the circle at time t, and let $r_0 = r(0)$ denote the initial radius, then $dr/dr = -\varepsilon r^{-1}$, and so $r(t) = r_0^2 - 2\varepsilon t^{1/2}$. Thus we can compare both the computed velocity and position of the interface with the exact values. The position of the interface is determined by linear interpolation and the velocity is approximated by a central difference of the interface position.

We chose $\varepsilon = 0.01$, $\Delta t = 0.00005$, $\Delta x = \frac{1}{50}, \frac{1}{100}, \frac{1}{200}$ and computed the solution until it was constant. We found that for $\Delta x = \frac{1}{50}$ the solution never became constant; instead it reached a non-constant steady state, which simply should not happen. This occurred for each of the initial configurations shown in Fig. 31.

Three questions come to mind: first, does this behavior (particularly the frozen profile) persist for smaller ε? Second, does this behavior persist for more complicated geometries (e.g., Fig. 32)? Third, does this behavior persist for other finite difference methods? The answer to these questions is yes.

A variety of alternative methods were employed to solve (7). Table II lists the approaches used in the space and time variables. The terms "5 pt" and "9 pt" refer to the number of points used to discretize the Laplacian. The "9 pt" stencil is the usual fourth-order accurate approximation. The discretizations are explicit except where noted. The first entry in the table is just (10). The NL-SOR method is an implicit, non-linear, successive over-relaxation scheme, which is described in detail below.

4.3.2. *Implicit NL-SOR*

Treating the laplacian in (7) implicitly is easily accomplished by a variety of methods. However, it is possible to include the reaction term $(\Delta t/\varepsilon)\, f_u(u_{ij}^n)$ in the implicit formulation.

This gives rise to a system of non-linear equations to solve at each time step. The system has the form

$$u_{ij}^{n+1} - \varepsilon\,\Delta t[g(n+1, \Delta_\pm^x, \Delta_\pm^y)]^{n+1} + \frac{\Delta t}{\varepsilon} f_u(u_{ij}^{n+1}) = u_{ij}^n \tag{11}$$

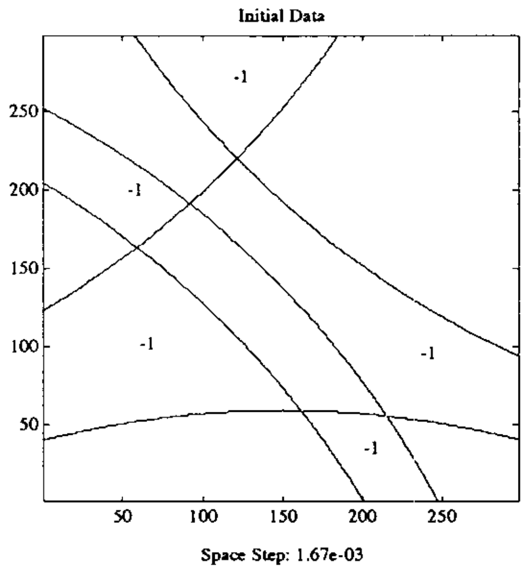


FIG. 32. Initial data with complicated geometry.

TABLE II

Methods Used to Solve (7)

Space	Time	Accuracy
5 pt	Forward Euler	(2, 1)
9 pt	Forward Euler	(2, 1)
5 pt	Heun's method	(2, 2)
9 pt	Heun's method	(4, 2)
Implicit (5 pt)	Forward Euler	(2, 1)
Implicit (9 pt)	Forward Euler	(4, 1)
Implicit NL-SOR (5 pt)	Forward Euler	(2, 1)
Implicit NL-SOR (9 pt)	Forward Euler	(4, 1)

where $g(n, \Delta^x_\pm, \Delta^y_\pm)$ is some approximation to the laplacian at time level n. We then solve (11) via the following (assuming that $g(n, \Delta^x_\pm, \Delta^y_\pm)$ is the five-point approximation to the Laplacian):

(1) Let $v^{(0)}_{ij} = u^n_{ij}$, set $k = 0$.

(2) For $j = 1, ..., N$
For $i = 1, ..., N$
Solve

$$z - \frac{\varepsilon\,\Delta t}{\Delta x^2}\left[v^{(r+1)}_{i,j-1} + v^{(r+1)}_{i-1,j} + v^{(r)}_{i+1,j} + v^{(r)}_{i,j+1} - 4z\right] + \frac{\Delta t}{\varepsilon} f_u(z) = u^n_{ij} \tag{12}$$

for z, and set

$$v^{(k+1)}_{ij} = (1-\omega)\, v^{(k)}_{ij} + \omega z. \tag{13}$$

(3) Compute the residual of (12); call this r.

(4) If $r < TOL$, then take $u^{n+1}_{ij} = v^{(k+1)}_{ij}$ else $k = k+1$ and return to (2).

The solution of the non-linear equation (12) is accomplished via Newton's method with an initial guess $z = v^{(k)}_{i,j}$; typically, two iterations suffice. With this method, much larger time steps may be taken; for Figs. 33 through 35, we used $\Delta t = \frac{1}{100}$.

4.3.3. *Results*

It turns out that the behavior of the interface is highly dependent upon the relative values of Δx and ε. For $\Delta x/\varepsilon$ near 2.0, the interface did not move at all after a short time in all three cases. It was not unitil $\Delta x/\varepsilon$ came near 0.5 that correct motion was observed.

Figs. 33, 34, 35 show the computed interface position and exact position for $\varepsilon = 0.01$, $\Delta t = 0.01$, $\Delta x = \frac{1}{50}, \frac{1}{100}, \frac{1}{200}$ and the NL-SOR method. Figs. 33, 34, 35 show the corresponding computed and exact velocity.

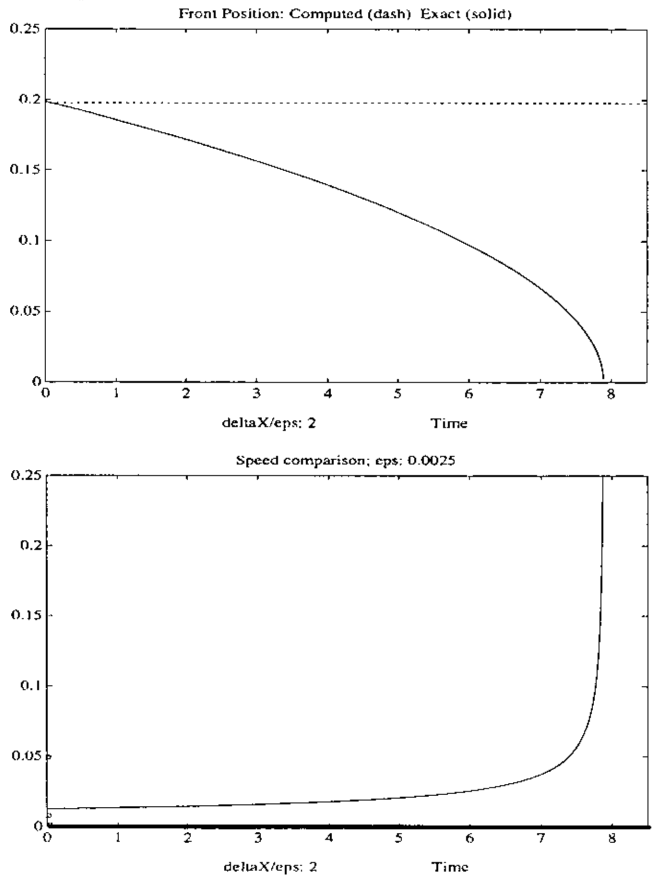


FIG. 33. Position and velocity comparisons, I.

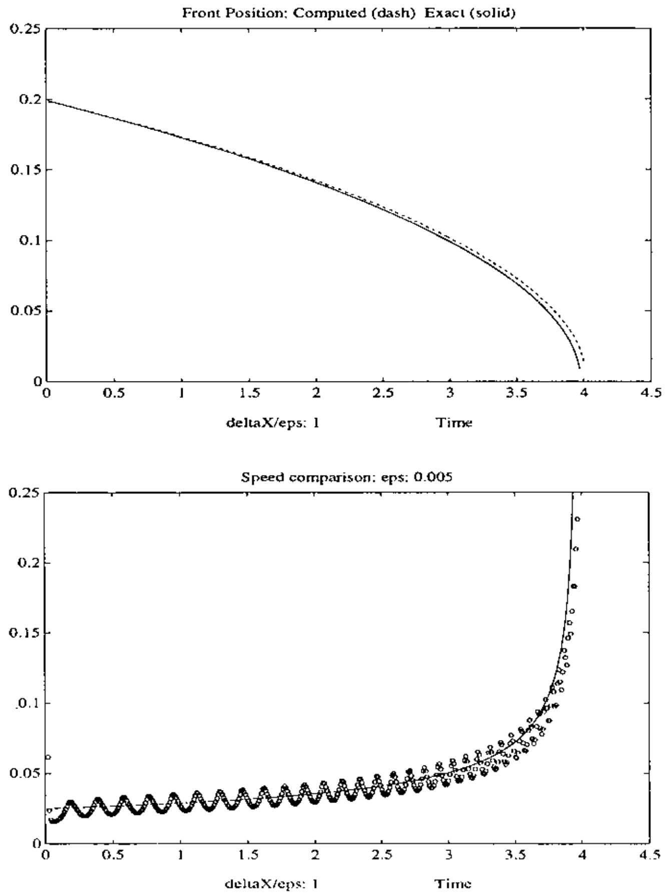


FIG. 34. Position and velocity comparisons, II.

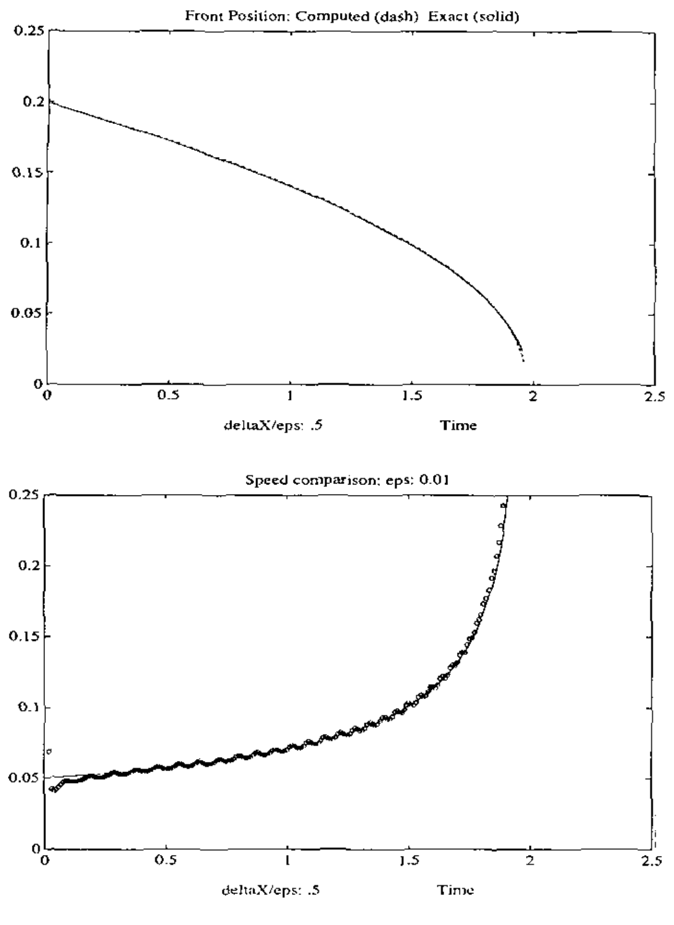


FIG. 35. Position and velocity comparisons, III.

Note the case where $\Delta x/\varepsilon = 2.0$ (Fig. 33); the interface has stopped moving, which is qualitatively incorrect. The other cases, $\Delta x/\varepsilon = 1.0$ and 0.5, are much better. (An explanation of the oscillatory nature of the computed velocity is given in Section 4.4.) Most disturbing, however, is Fig. 34. Here the front behaves qualitatively as we expect, but the velocity is off by fairly large margin. It is a cause for concern because we cannot tell, in a situation where we do not know the correct qualitative behavior, if the velocity is correct. Yet the incorrect velocities may persist beyond the point of feasible grid refinement.

We have not included pictures for smaller ε because they yield no new information; they are essentially the same. The frozen profile occurs for $\Delta x/\varepsilon \approx 2.0$ in both of the other initial configurations and for more complicated geometries (Fig. 32).

4.4. Discussion

It is convenient for the following discussion to write (7) as

$$T_t = \varepsilon\, \Delta T - \frac{1}{\varepsilon} R(T), \qquad (14)$$

where $T(x, t)$ and $R(T)$ are interpreted as temperature and reaction rate, respectively. Now consider a semi-discrete approximation to (14),

$$\frac{\partial T_{ij}}{\partial t} = \frac{\varepsilon}{\Delta x^2} [\Delta_+^x \Delta_-^x + \Delta_+^y \Delta_-^y] \cdot T_{ij} - \frac{1}{\varepsilon} R(T_{ij}). \qquad (15)$$

This approximation yields a simple intuition. The idea is that (15) describes a physical system which is a grid of "fuel elements" connected by thermal conductors (see Fig. 36).

Each "fuel element" (gridpoint) has a temperature T_{ij}. We think of the interfaces that develop as "burning" fronts which separate "hot" points (points near one of the stable zeros of $R(T)$) from "cold" points (points near the other stable zero of $R(T)$). Points within the interface are "igniting" (making the transition from "cold" to "hot").

Within this framework we can reformulate our basic question: why do we obtain frozen profiles? A frozen profile in the language above is a "fire" which has stopped "burning." That is, "fuel elements" are no longer "igniting," even when they are near points already "burning." Consider the most extreme case, where a "cold" element is completely surrounded by "hot" neighbors. Let T_{hot} and T_{cold} denote the stable points of $R(T)$, so that $T_{\text{hot}} > T_{\text{cold}}$.

The "cold" center point receives heat via conduction along the thermal conductors (gridlines). This is the contribution of the term $\varepsilon\, \Delta T$. The maximum rate at which heat comes in is, using (15),

$$\frac{4\varepsilon}{\Delta x^2} (T_{\text{hot}} - T_{\text{cold}}). \qquad (16)$$

If the reaction, $(1/\varepsilon)\, R(T)$, can absorb heat at this rate (or faster) then the center point will not "ignite." Comparing these two expressions would give a relationship between ε and Δx to avoid freezing in this case. However, we can make

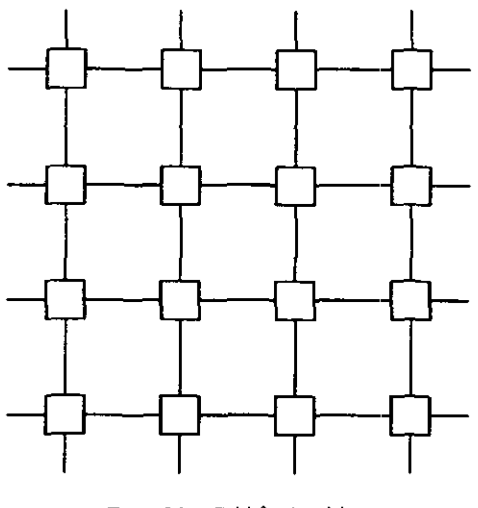


FIG. 36. Grid for intuition.

a more general statement. Note that our intuitive approach relies only upon the reaction $R(T)$ and heat conductivity along the grid. Boundary conditions, time discretization, etc. play no role in producing the frozen profile. Consider a general discretization of (7),

$$\frac{\partial T_\alpha}{\partial t} = \varepsilon \sum_{\beta \in G} K_{\alpha\beta}(T_\beta - T_\alpha) - \frac{1}{\varepsilon} R(T_\alpha), \tag{17}$$

where $K_{\alpha\beta} \geqslant 0$ and $\{T_\alpha\}_{\alpha \in G}$ are values on a grid G. Make the following

Assumption 1 (Maximum principle). *If the initial and boundary data for* (17) *satisfy* $T_{\text{cold}} \leqslant T_\alpha(0) \leqslant T_{\text{hot}}$, *then* $T_{\text{cold}} \leqslant T_\alpha(t) \leqslant T_{\text{hot}}$ *for* $t > 0$.

This is not unreasonable. For discretizations like (17) the principle would have to be violated at the boundary first, which reasonably discretized Neumann or Dirichlet conditions will not do. Now we can show that if conduction is too weak, frozen profiles appear.

Theorem 1. *Assume that the maximum principle holds and assume that the initial data takes on only the values* T_{hot} *and* T_{cold}. *Then no* T_α *will ever cross the threshold value* T_0 *if*

$$\frac{\max_{T_{\text{cold}} \leqslant T \leqslant T_0} (1/\varepsilon) R(T)}{\varepsilon \sum_{\beta \in G} K_{\alpha\beta}(T_\beta - T_\alpha)} > 1 \tag{18}$$

$$\frac{|\min_{T_0 \leqslant T \leqslant T_{\text{hot}}} (1/\varepsilon) R(T)|}{\varepsilon \sum_{\beta \in G} K_{\alpha\beta}(T_\beta - T_\alpha)} > 1. \tag{19}$$

Proof. Consider a point T_α such that $T_\alpha = T_{\text{cold}}$. Then

$$\frac{\partial T_\alpha}{\partial t} = \varepsilon \sum_{\beta \in G} K_{\alpha\beta}(T_\beta - T_\alpha) - \frac{1}{\varepsilon} R(T_\alpha) \tag{20}$$

$$\leqslant \varepsilon \sum_{\beta \in G} K_{\alpha\beta}(T_{\text{hot}} - T_{\text{cold}}) - \frac{1}{\varepsilon} R(T_\alpha) \tag{21}$$

by the maximum principle.

Now, if the right-hand side of this inequality is negative at some T_α^* satisfying $T_{\text{cold}} \leqslant T_\alpha^* \leqslant T_0$, then $T_\alpha(t)$, which has $T_\alpha(0) = T_{\text{cold}}$, can never exceed T_α^*. But this is the same as saying that

$$\max_{T_{\text{cold}} \leqslant T \leqslant T_0} \left[\varepsilon \sum_{\beta \in G} K_{\alpha\beta}(T_{\text{hot}} - T_{\text{cold}}) - \frac{1}{\varepsilon} R(T_\alpha) \right] < 0$$

and this is just (18).

The other condition follows from considering a point T_α such that $T_\alpha(0) = T_{\text{hot}}$ and looking for a positive right-hand side to the inequality obtained from the maximum principle. ∎

Now we can apply Theorem 1 to (10). Here $K_{\alpha\beta} = 1/\Delta x^2$, and solving the conditions (18), (19) for $\Delta x/\varepsilon$ we obtain

$$\frac{\Delta x}{\varepsilon} > \sqrt{(4(T_{\text{hot}} - T_{\text{cold}}))/\max_{T_{\text{cold}} \leqslant T \leqslant T_0} R(T)}$$

$$\frac{\Delta x}{\varepsilon} > \sqrt{(4(T_{\text{hot}} - T_{\text{cold}}))/|\min_{T_{\text{cold}} \leqslant T \leqslant T_0} R(T)|}.$$

For the choice of $R(T)$ in our numerical experiments, $T_{\text{hot}} = 1$, $T_{\text{cold}} = -1$, and with $T_0 = 0$, the above expressions are identical, giving

$$\frac{\Delta x}{\varepsilon} > \sqrt{4(2)/(2/3\sqrt{3})} \approx 4.5 \tag{22}$$

as the condition for frozen profiles. As we noted in Section 4.3.1, the critical ratio is 2.0 so the above estimate is off by a significant amount.

The explanation is that the theorem takes a worst-case approach. It describes the ratio for which motion cannot possibly occur, regardless of the configuration of the initial data. Intuitively, different initial configurations lead to different freezing ratios. One would suspect that it is less likely that a single "hot" point with "cold" points on all four sides (Fig. 37) will freeze than the same point with "cold" points on only two sides. The motivation is that the center point will receive more "heat" in the former case. This is borne out via numerical experiments.

For the initial data in Fig. 37, the computed freezing ratio is 4.0, which is much closer to that predicted by the theorem.

As an aside, we see that the oscillation in the computed velocities in Figs. 33–35 is expected. The interface moves in a jerky fashion, the reaction term pulling values toward T_{hot} and T_{cold} and the diffusion term spreading them out. The initially sharp front spreads, becoming less sharp, and then the reaction term "snaps" points back to the stable values. So motion of the front will occur only if enough heat is conducted so that a point can get close enough to the other stable state so that the reaction term pulls it in.

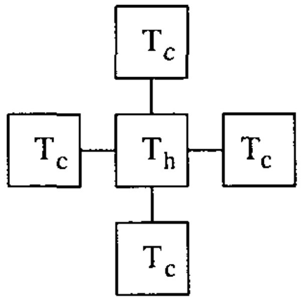


FIG. 37. Extreme case for freezing.

4.5. Summary of Reaction–Diffusion Approach

Using a reaction–diffusion equation is an attractive idea in that curvature-dependent motion can be calculated without the need for computing a numerical approximation to the curvature itself. In two dimensions, the curvature of $\phi(x, y)$ can be expressed as

$$\frac{\phi_{xx}\phi_y^2 - 2\phi_{xy}\phi_x\phi_y + \phi_{yy}\phi_x^2}{(\phi_x^2+\phi_y^2)^{3/2}} \tag{23}$$

and in three dimensions the Gaussian curvature is

$$\frac{1}{(\phi_x^2+\phi_y^2+\phi_z^2)^{3/2}} \det\begin{pmatrix} \phi_{xx} & \phi_{xy} & \phi_{xz} \\ \phi_{yx} & \phi_{yy} & \phi_{yz} \\ \phi_{zx} & \phi_{zy} & \phi_{zz} \end{pmatrix} \tag{24}$$

with the expression for mean curvature no more tractable. The denominators above can be difficult to deal with numerically. There is also the issue of which discretizations to use for the needed derivatives. Avoiding the approximations of (23), (24) is desirable, aside from computation time saved.

However, there is an inherent problem in the numerical solution of these reaction—diffusion equations: the grid must resolve the width of the transition region. More unsettling is the example of Section 4.2, which showed that a significant error is made even if $\Delta x/\varepsilon \approx 1$. For our problem, $\Delta x/e \ll 1$ is required, and the computational expense therein implies that we must employ other methods.

It should be noted that detailed studies of spurious solutions to nonlinear differential equations were conducted in [8, 18]. The emphasis in [18] is on computing steady state solutions to ODEs and PDEs containing nonlinear source terms via iteration in time. They show that a spatial discretization can produce spurious steady state solutions even when the original equation has only one steady solution, even when used with linear multistep methods in time. They also note the misconception that simply taking a smaller time step will alleviate the problems. Our equation does possess a steady state solution for particular initial data, but none of the examples we have computed fall into that class. Therefore, our results are different although clearly related. The work in [6] considers a more general problem and shows that insufficient spatial resolution can yield smooth spurious steady solutions.

5. SUMMARY

This work began because of the amazing pictures of [11, 12]; see Fig. 39. The system solved there possesses an equation like (7), and our experiments suggest that a basic finite difference method on a regular grid (as was used in [11, 12]) provides an inadequate solution unless one takes $\Delta x \ll \varepsilon$, which is impractical numerically. More problematic is that the system of equations is intended to model dendritic crystal growth, and the size of ε affects the development of the dendrites. To correctly capture the dendrite growth for small ε with a fast reaction, slow diffusion equation is numerically infeasible. The analysis of Section 4.4 shows that this is unavoidable and purely a result of the numerical method used. These results suggest that coupling the Osher–Sethian algorithm to equations describing the phenomena in [11, 12] is the best approach numerically.

There are several areas requiring further investigation. First, is there a more efficient way to perform the reinitialization step of Section 3.3? Osher, Smereka, and Sussman are currently implementing a method that avoids

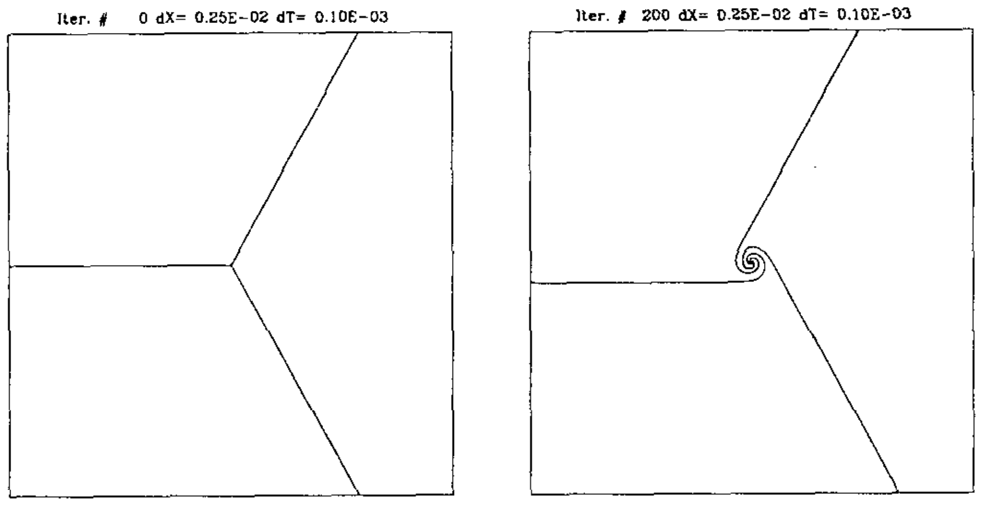

Iter. # 0 dX= 0.25E-02 dT= 0.10E-03
Iter. # 200 dX= 0.25E-02 dT= 0.10E-03

FIG. 38. Tightening spiral; velocities equal, all clockwise.

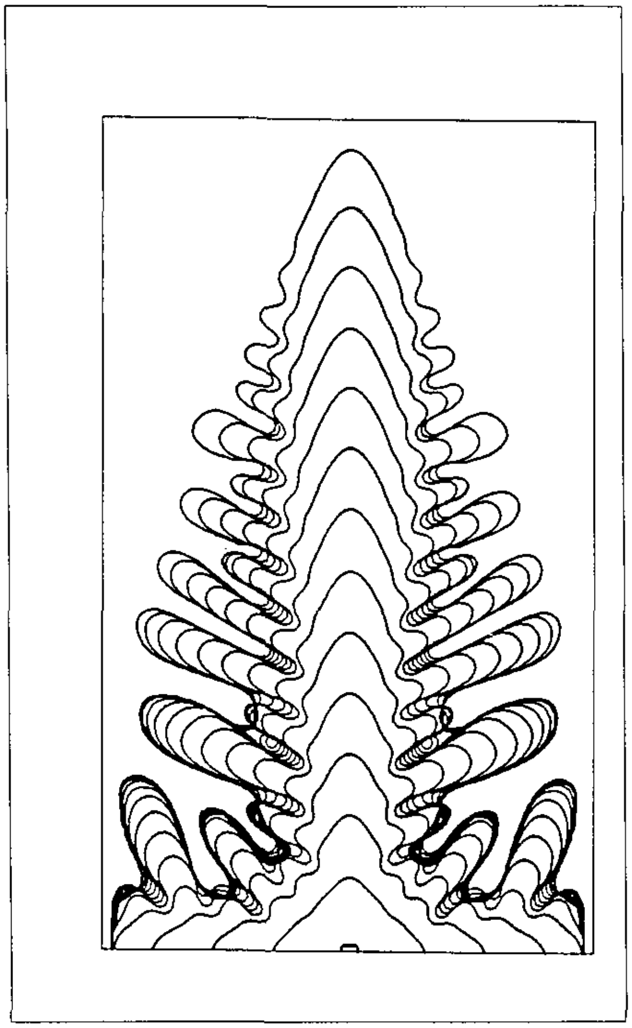


FIG. 39. Intriguing slide from [11].

explicit location of the zero-level set. They solve a PDE whose solution converges to the distance function used in initializing the method. This has the advantages of being faster and applicable more frequently than the chopping step, so that discontinuities do not develop as rapidly. Under the current algorithm discontinuities develop and are then removed. Second, the behavior of a triple junction as the velocities approach the same magnitude and clockwise orientation is unclear. As noted in Section 3, if the velocities are set in this manner, we find after the first update step that we have a spiral (see Fig. 38). The triple point has reformed at the center of the spiral, and if we were to continue the computation the spiral would wind up further, beyond the resolution of the grid. This phenomenon needs to be understood. Third, what other stable shapes might we discover if a different update step were taken? Rather than using

$$\phi_i = \phi_i - \max_{j \neq i} \phi_j,$$

we might imitate the alternative chopping schemes of the diffusion algorithm in Section 2.2, by using some convex combination of the surrounding ϕ_i rather than the maximum. Fourth, the connection between the shape on the plane $x+y+z=1$ and the stable shape exhibited in computations by the diffusion algorithm needs to be clarified. Finally, the DGCDM algorithm may lead to a method for computing convex hulls. Given a set S, set the diffusion coefficient D on that set to be zero and start diffusion-generated motion on some large ball containing S. The ball will collapse down onto the $D=0$ set and equilibrate at its convex hull.

There are currently only two algorithms for moving multiple junctions. Each method has limitations which our new method does not. Our method handles general velocities directly and easily; the others either do not or they require lengthy asymptotics to do so. There are no numerical difficulties, as in the reaction–diffusion system and the method is easy to implement in many dimensions. The coupled Osher–Sethian method is a powerful new tool for moving multiple junctions. The one advantage of the reaction–diffusion system proposed by [2] is in the area of analysis. Their approach may provide an avenue for understanding the theory behind the motion of multiple junctions, whereas ours will efficiently compute it.

REFERENCES

1. G. Barles and C. Georgelin, preprint, September 1992.
2. L. Bronsard and F. Reitich, Research Report 92-NA-022, Carnegie Mellon Mathematics Department, Carnegie-Mellon University, Pittsburgh, PA 15213-3890, July 1992.
3. G. Caginalp and P. C. Fife, *SIAM J. Appl. Math.* **48**, 506 (1988).
4. Y. G. Chen, Y. Giga, and S. Goto, *J. Differential Geom.* **23**, 749 (1986).
5. T. Cohignac, F. Eve, F. Guichard, C. Lopez, and J. Morel, Technical Report, November 1992.
6. C. M. Elliott and A. M. Stuart, *SIAM J. Numer. Anal.*, submitted.
7. L. C. Evans, preprint, September 1992.
8. L. C. Evans and J. Spruck, *J. Differential Geom.* **23**, 69 (1986).
9. P. C. Fife, *Mathematical Aspects of Reacting and Diffusing Systems*, Lect. Notes in Biomathematics, Vol. 28, Springer-Verlag, New York/Berlin, 1979.
10. M. Grayson, *J. Differential Geom.* **26**, 285 (1987).
11. R. Kobayashi, Department of Applied Math and Informatics 12, Ryukuko University, Seta, Ohtsu, Japan, July 1991.
12. R. Kobayashi and M. Mimura, videotape, 1989.
13. J. J. Koenderink and A. J. van Doorn, *Biol. Cybernet.* **1**(53), 383 (1986).
14. P. Mascarenhas, Cam report, UCLA Math. Dept., 1992.
15. S. J. Osher and J. A. Sethian, *J. Comput. Phys.* **79**, 12 (1988).
16. J. Rubenstein, P. Sternberg, and J. Keller, *SIAM J. Appl. Math.* **49**, 116 (1989).
17. J. Taylor, preprint, 1990.
18. H. C. Yee, P. K. Sweby, and D. F. Griffiths, Tm-102820, NASA, 1990 (unpublished).
19. A. Yuille, in *Proceedings, Second International Conference on Computer Vision, Tampa, FL, 1988*, p. 685.