UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

A Fixed Grid Method for Capturing the Motion of Self-Intersecting Interfaces and Related PDEs

Steven J. Ruuth
Barry Merriman
Stanley Osher

July 1999

CAM Report 99-22

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90095-1555

http://www.math.ucla.edu/applied/cam/index.html

A Fixed Grid Method for Capturing the Motion of Self-Intersecting Interfaces and Related PDEs

Steven J. Ruuth*, Barry Merriman† and Stanley Osher‡

July 1, 1999

Abstract

Moving interfaces that self-intersect arise naturally in the geometric optics model of wavefront motion. Ray tracing techniques can be used to compute these motions, but they lose resolution as rays diverge. In this paper we develop a new numerical method that maintains uniform spatial resolution of the front at all times. Our approach is a fixed grid, interface capturing formulation based on the Dynamic Surface Extension method of Steinhoff and Fan [10]. The new methods can treat arbitrarily complicated self intersecting fronts, as well as refraction, reflection and focusing. We also further extend this approach to curvature dependent front motions, and the motion of filaments. We validate the new methods with numerical experiments.

*Department of Mathematics, University of California at Los Angeles. (ruuth@math.ucla.edu).
The work of this author was partially supported by AFOSR STTR FQ8671-9801346.

†Department of Mathematics, University of California at Los Angeles. (barry@math.ucla.edu).
The work of this author was partially supported by AFOSR STTR FQ8671-9801346.

‡Department of Mathematics, University of California at Los Angeles. (sjo@math.ucla.edu).
The work of this author was partially supported by AFOSR STTR FQ8671-9801346.

1 Introduction

In the limit of short wavelengths, it is well known that a wavefront moving through a medium can be described as a moving surface with a normal velocity that depends on position,

$$\vec{v} = c(x)\hat{n}$$

where $\hat{n}$ is the local normal to the front and $c(x)$ is the local wave speed. Notable examples include the short wavelength approximation of seismic and electromagnetic pulses, as well as the familiar example of ripples moving on the surface of a pond. An important feature of this idealized wavefront motion is that intersecting wavefronts pass through each other, and also that they reflect and refract off boundaries.

Many interesting numerical methods have been developed to compute these complex motions (see, e.g., [3, 4, 5, 6, 12, 13]). The most detailed approach is to discretize the governing wave equations directly (e.g., [13]). Unfortunately, this approach is often impractical because it requires that the discretization resolve the short wavelengths, which may be thousands of times smaller than the length scale of interest.

At the other extreme, ray tracing can be used to evolve wavefronts according to geometrical optics (e.g., [5]). Here, the front is represented using a number of markers which are moved independently. This approach has the advantage of simplicity, but the markers tend to diverge which leads to loss of resolution and aliasing of the front.

To maintain a uniform resolution of the interface, it is natural to consider a fixed grid, interface capturing formulation such as the Level Set method [8]. Here, the wavefront is represented as the zero contour of a smooth function ϕ, which in turn evolves according to the level set equation

$$\phi_t + c(x)|\nabla\phi| = 0.$$

This can be solved accurately and efficiently using numerical Partial Differential Equation (PDE) techniques. Unfortunately, the basic level set method is inappropriate for treating evolving wavefronts because the solutions to this PDE will have fronts merge upon colliding, rather than pass through one another.

To obtain a fixed grid method appropriate for capturing wavefront self-intersection Steinhoff and Fan proposed Dynamic Surface Extension (DSE)

methods [9, 10]. These schemes start from some spatially distributed representation of the interface (similar to, but more general than, the level set ϕ representation), and the motion is achieved by alternating between two steps: a simple short time evolution comparable to ray tracing, and an extension step that updates the distributed representation to reflect the new front location. DSE methods automatically give a uniform resolution of expanding fronts by using a fixed spatial grid, and the fronts automatically pass through one another rather than merging.

The original DSE methods are not well suited to certain fundamental self-intersection problems such as the formation of swallowtails. In this paper, we show how to generalize a basic DSE scheme (the Closest Point Method) to handle this fundamental problem, as well as all other complex intersections. We further generalize our approach to reflecting and refracting wavefronts. We also discuss new extensions for propagating intensity values, for treating curvature-dependent flows, and for treating the motion of filaments (or more generally, objects of co-dimension > 1).

The outline of the paper follows. In Section 2, we review the Closest Point Method and discuss its key properties. In Section 3, we give a generalization of the method (the Arrival Time Method) which produces a much more uniform representation of the interface. Sections 4 and 5 explain how to extend our method to problems involving refraction and reflection. In Section 6, we describe how to treat intensity by retaining other attributes of the wavefront, such as wavefront curvature. Section 7 extends our approach to more general curvature-dependent motions. Finally, in Section 8 we summarize our results and outline some potential areas for future research. Throughout the paper, numerical experiments are provided to validate our methods.

2 The Closest Point Method

To evolve self-intersecting wavefronts on a fixed grid, Steinhoff and Fan [9] proposed Dynamic Surface Extension methods. The cornerstone of this approach is to choose a suitable distributed representation of the wavefront surface. The form of this representation depends on the problem, and is dictated by the information required to accurately and efficiently evolve the surface. But generally speaking, the idea is to store at each point in space x

a representation of the surface near some "tracked point" $TP(x)$ located on the surface. A two-step scheme is then used to evolve this distributed representation for a short time Δt. First, the *Evolution* Step updates the local surface representation at each x based on the surface motion law at $TP(x)$. During this evolution, the representation may develop inconsistencies or become less well behaved, but this is repaired in the second step, which *extends* the representation from near the wavefront (where it is most accurate) to points further away, perhaps reassigning new tracked points $TP(x)$ to each x in the process. Note that non-geometric surface properties such as optical intensity may considered as part of the "representation", and these can be evolved and extended off the interface in this manner as well [10].

A particularly instructive DSE scheme is the Closest Point Method (see [9, 10]). In this section, we describe the Closest Point Method and discuss its key properties. Improvements and extensions to this basic method will be the focus of subsequent sections.

2.1 The Method

To construct a DSE scheme for wavefront propagation we must first select an appropriate distributed surface representation, i.e. one well suited to representing self-intersecting surfaces. For contrast, note that the level set method relies on a particular distributed surface representation, namely the level set function $\phi(x)$. This is often taken to be the (shortest) distance from x to the surface, with suitable signs. In the language of DSE, this can be viewed as letting the tracked point at x, $TP(x)$, be the point on the surface closest to x, $CP(x)$, and the only local information we retain about the surface near this point is its distance to x (with a sign). However, this representation (i.e. ϕ) leads to unstable kinks, which naturally produce mergings (or curve annihilation) rather than allowing curves to pass through one another.

A better behaved and more convenient alternative is to store the coordinates of the closest point itself, rather than just its distance, i.e. we represent the surface by the value of the vector valued function $CP(x)$. This is smooth on the interface near self-intersections (although it does have discontinuities off the interface, at points that are equidistant between different parts of the surface). Further, this choice can equally well represent geometries for

which the distance function would be highly singular, such as surfaces with boundaries (i.e. a non-closed surface 3D, or a curve with endpoints in 2D) or objects of any codimension (e.g. points, curves or surfaces in 3D). Also, note that a closest point representation is constant *normal* to a surface whereas a distance function representation (such as that used in level set methods) is constant *tangential* to a surface [10].

For moving wavefronts, we must specify some additional piece of information since there are at least two possible normal directions (more at kinks) for any curve or surface. We prefer to store a unit vector in the direction of propagation because this choice automatically generalizes to objects of arbitrary codimension. Using this representation in the Dynamic Surface Extension approach gives the Closest Point Method for moving a surface $\Gamma \subset R^n$ normal to itself with a speed c (which may depend on position) (cf. [10]):

The **Closest Point Method**:
Initialize. For each point $x \in R^n$: Set the initial tracked point $TP(x)$ equal to the closest point (to x) on the initial surface Γ_0. Set $\hat{n}$ equal to the surface normal at the tracked point $TP(x)$, and let c denote the wavefront speed at the tracked point.

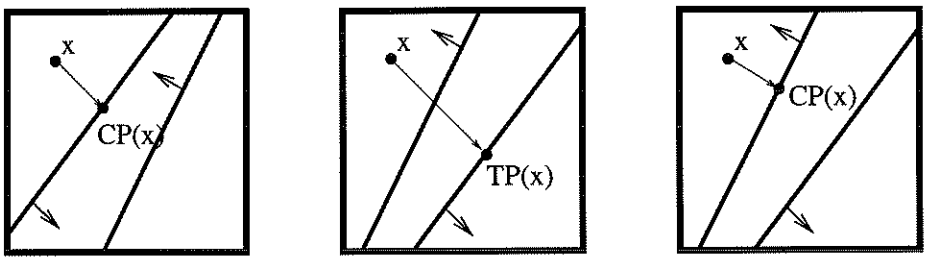
Repeat for all steps:

(1) *Evolve* the tracked point $TP(x)$ according to the local dynamics for a time Δt: $TP(x)_t = c\hat{n}$.

(2) *Extend* the surface representation by resetting each tracked point $TP(x)$ equal to the true closest point on the updated surface Γ, where Γ is defined to be the locus of all tracked points, ie, $\Gamma = \{TP(x) | x \in R^n\}$. Replace each $\hat{n}(x)$ by the normal at the updated $TP(x)$.

End.

Intuitively, the manner in which this method treats self-intersection is most easily understood by considering how it treats two colliding, planar waves. Initially, each nodal tracked point value is set equal to the closest point on the nearest wavefront (Figure 1a). These tracked points are updated during the Evolution Step according to $TP(x)^{new} = TP(x)^{original} + c\hat{n}\Delta t$ (Figure 1b). Notice that the updated tracked points are no longer the true closest points. Finally, the Extension Step resets each nodal value to be a true closest point (Figure 1c).



(a) Initialization (b) Evolve (c) Extend

Figure 1: Two colliding planar waves and a sample grid node x. (a) To initialize, the closest point $CP(x)$ of the nearest wavefront is stored. (b) Evolution is carried out pointwise according to $TP(x)_t = c\hat{n}$. (c) During the Extension Step, nodal values are set equal to true closest point values.

We now direct our attention to the implementation of the Closest Point Method.

2.2 Implementation

In practice, the Initialization Step of the Closest Point Method can often be handled analytically in simple problems. More complicated wavefronts can be treated using fast tree-based algorithms [11]. Implementation of the Evolution Step is also straightforward because each tracked point is just updated according to $TP(x)^{new} = TP(x)^{original} + c\hat{n}\Delta t$. The final Extension Step is more complicated and is typically divided into two parts, a search step and an interpolation step (cf. [10]).

In the search step, the updated value for a node is taken to be the closest of all tracked points (localization of this step is possible—see Section 3.3) [10]. This gives an improved approximation of the closest point representation. Unfortunately, this process cannot create any new tracked points so diverging wavefronts will lose resolution. Thus, a second interpolation step is needed in order to maintain a uniform representation.

Steinhoff, Fan and Wang carry out this interpolation by averaging over nearby nodes [10]. This very simple approach is effective for a variety of interesting problems [10], but it can produce spurious wavefronts in certain

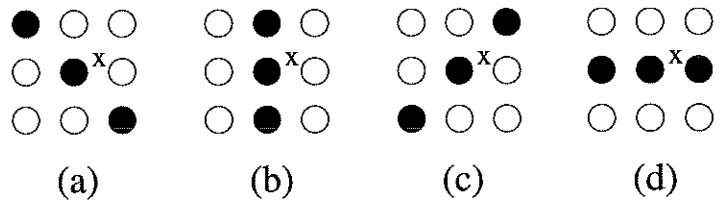


Figure 2: For a node x we interpolate using 3 points (solid) which are roughly parallel to the interface. Taking θ to be the angle $\hat{n}(x)$ forms with the horizontal axis we find 4 cases in 2D: (a) $\frac{\pi}{8} \leq \theta < \frac{3\pi}{8}$ or $\frac{9\pi}{8} \leq \theta < \frac{11\pi}{8}$, (b) $-\frac{\pi}{8} \leq \theta < \frac{\pi}{8}$ or $\frac{7\pi}{8} \leq \theta < \frac{9\pi}{8}$, (c) $\frac{5\pi}{8} \leq \theta < \frac{7\pi}{8}$ or $\frac{13\pi}{8} \leq \theta < \frac{15\pi}{8}$, (d) $\frac{3\pi}{8} \leq \theta < \frac{5\pi}{8}$ or $\frac{11\pi}{8} \leq \theta < \frac{13\pi}{8}$.

cases and is low order accurate. For these reasons, we consider a higher order interpolation based on nearby neighbors. These neighbors (call them y and z) are chosen so that x, y and z are collinear and roughly parallel to the interface (see Figure 2). If the tracked points for x and y are distinct and lie on the same smooth curve, then an improved estimate for the closest point to x can be generated using the nodal values at x and y (see Figure 3). Similarly, an improved closest point estimate can be attempted using the nodal values at x and z. The closest of these two results to x is taken to be the updated nodal value.

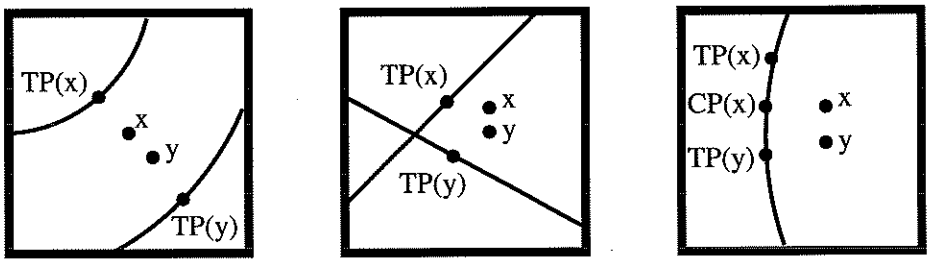
Notice that this Extension Step does not yield a true closest point representation. However, closest point values are expected *near* the interface. Furthermore, this extension has the useful property that every nodal value represents some tracked point on the interface.

We now direct our attention to how the Closest Point Method treats two prototype problems in wave propagation: rarefaction fans and swallowtails.

2.3 Numerical Experiments

We now apply the Closest Point Method to the problem of evolving wavefronts according to a constant normal velocity, $\vec{v} = c\hat{n}$. In these experiments, the fronts are plotted simply as the locus of all tracked points at a given time, $\{TP(x)|x \in G\}$, where G is a uniform grid of points on the domain.

First, consider the motion of a square curve moving outward with unit speed, as is shown in Figure 4. Using the Closest Point Method, a rarefaction



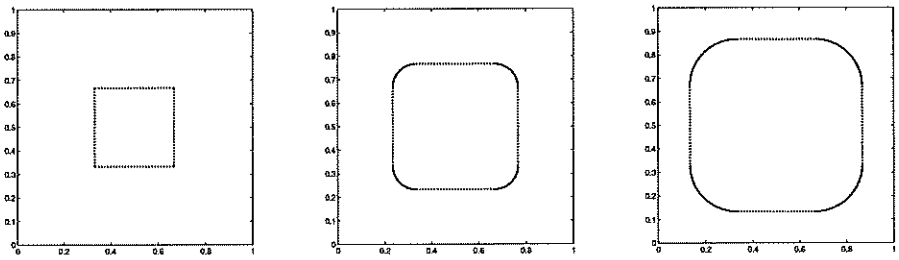
(a) Points far apart (b) Normals disagree (c) Consistent case

Figure 3: (a) If the tracked points $TP(x)$ and $TP(y)$ for nodes x and y are far apart ($||TP(x) - TP(y)|| > 2\Delta x$) then we take $TP(x)$ to be the best estimate for the closest point to x. (b) If the corresponding normals are inconsistent, we take $TP(x)$ to be the closest point estimate to x. Here, we assume $TP(x)$ and $TP(y)$ lie on different wavefronts whenever the angle between their normals is greater than 0.2. (c) Otherwise, an arc is drawn between $TP(x)$ and $TP(y)$ based on $TP(x)$, $TP(y)$ and $\hat{n}(TP(x))$. The desired estimate is given by the closest point on the arc to x.

fan is automatically and uniformly generated. Notice that the Evolution Step always yields a closest point representation, so the Extension Step does not change the tracked point $TP(x)$. Thus, the overall error is comprised entirely of roundoff errors generated from the Initialization and Evolution Steps.

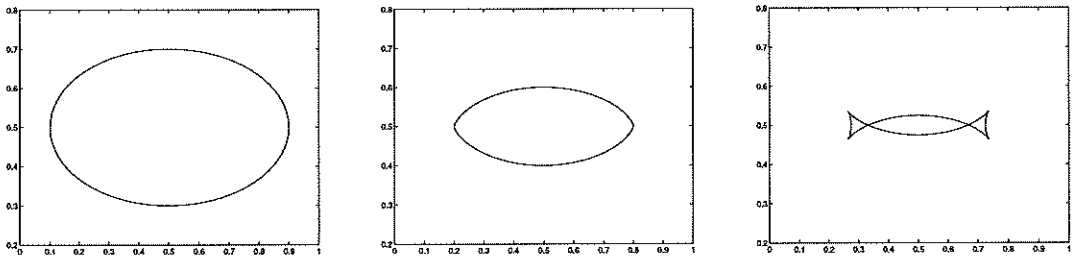
An entirely different, swallowtail solution also occurs in many problems. For example, consider an ellipse moving inward with unit speed (e.g., Figure 5). Here, the front forms two kinks (Figure 5b) and passes through itself to form a swallowtail (Figure 5c). Unfortunately, this swallowtail solution is not adequately reproduced using the Closest Point Method. This flaw causes gaps in the interface which propagate and grow (see Figure 6).

Thus the Closest Point Method is inadequate for treating the prototype swallowtail problem. Fortunately, a generalization based on the idea of first arrival times is possible. This approach will be the focus of the next section.



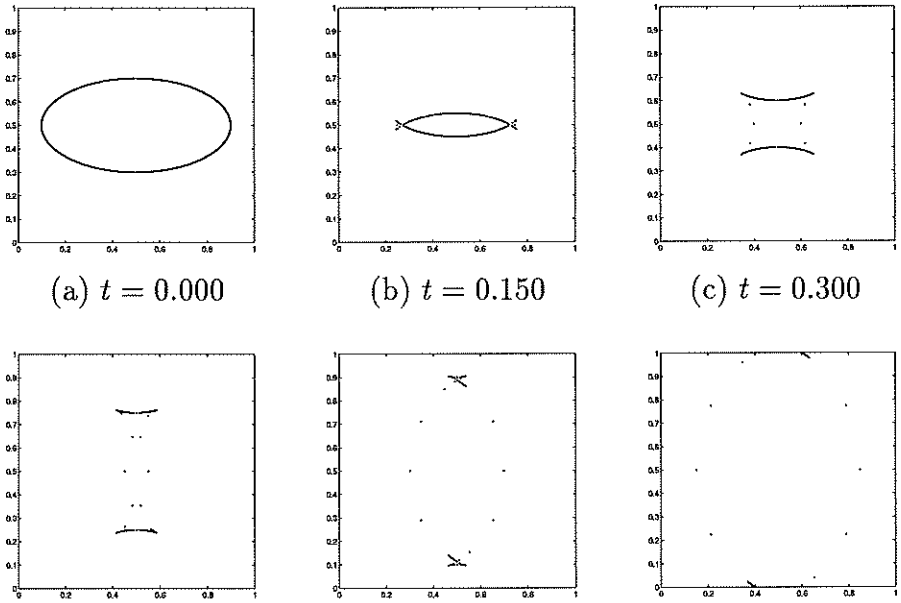
(a) $t = 0.0$ (b) $t = 0.1$ (c) $t = 0.2$

Figure 4: Rarefaction fans are given exactly by the Closest Point Method. Here, a wavespeed equal to 1 was considered. Discretization step sizes of $\Delta x = 1/80$ and $\Delta t = 1/40$ were used throughout the calculation. In this example (and others throughout the report), wavefronts are visualized by plotting all nodal values. Notice that this simple visualization can produce a dotted effect when the interface is aligned with the grid.



(a) $t = 0.0$ (b) $t = 0.1$ (c) $t = 0.175$

Figure 5: An ellipse evolving inward with unit speed: (a) Initial ellipse, (b) Kinks form, (c) Wavefront passes through itself to form a swallowtail.



(a) $t = 0.000$ (b) $t = 0.150$ (c) $t = 0.300$

(d) $t = 0.450$ (e) $t = 0.600$ (f) $t = 0.750$

Figure 6: Using the Closest Point Method, swallowtails are not accurately reproduced. Gaps in the interface form, and these propagate and grow. Here, a wavespeed equal to 1 was considered. Discretization step sizes of $\Delta x = 1/80$ and $\Delta t = 1/40$ were used throughout the calculation.

3 First Arrival Times

As demonstrated in the previous section, the Closest Point Method can produce gaps in the interface. We now discuss a method which gives a much more uniform representation of the interface and validate our approach with numerical experiments.

3.1 The Method

The formation of gaps for the Closest Point Method is most easily understood by considering how swallowtails are represented.

Consider, for example, the swallowtail representation shown in Figure 7a. When the swallowtail is small, nodal values over-represent corners. Since large regions are used to represent corner points, few grid points are available to represent the end of the swallowtail (see Figure 7b). This uneven treatment leads to gaps in the interface which propagate and grow.

To obtain an improved result, a more uniform representation is needed. For example, we can set each nodal value to be the point on the interface with the *minimal arrival time* rather than the minimal distance. In a homogeneous medium, without reflection, this just means that each nodal value is set equal to the closest point on the interface that propagates directly to or from the node. Using this representation, we find that redundancy is largely eliminated, and a greatly improved approximation of the swallowtail is obtained (see Figure 8). Unfortunately, arrival times are often difficult and expensive to evaluate in the variable index of refraction case or when reflections occur. Furthermore, even in a homogeneous medium, this approach requires a more intricate search step since nodal values can only be updated when a nearby tracked point travels directly towards the node (which rarely occurs).

Of course, we are not limited to representations that minimize distance or arrival times—a minimization based on some combination of distance and direction of motion can also be carried out. A particularly interesting choice arises when nodal values are set equal to the "Minimizing Point"

$$MP(x) = \min_{y \in \text{Interface}} \gamma |(x-y) \cdot \hat{n}^{\perp}(y)| + ||x-y||^2 \qquad (1)$$

for $\gamma > 0$ since then a good agreement with the minimal arrival time representation is found near the interface. Using this idea leads to the following modification of the Closest Point Method:

The **Arrival Time Method**:
Initialize. For each point x: Set the tracked point $TP(x)$ equal to the minimizing point $MP(x)$ on the initial surface Γ_0 for the minimization in Eq. (1). Set $\hat{n}$ equal to the normal at $MP(x)$.

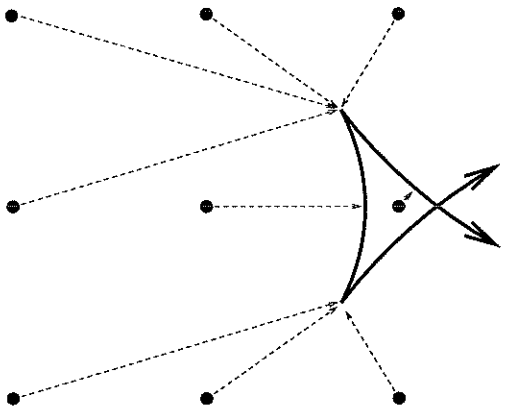
Repeat for all steps:
(1) *Evolve* the tracked point $TP(x)$ according to the local dynamics for a time Δt: $TP(x)_t = c\hat{n}$.
(2) *Extend* the surface representation by replacing each tracked point $TP(x)$ by the point $MP(x)$ on the updated surface $\Gamma = \{TP(x) | x \in R^n\}$ that minimizes Eq. (1). Replace each $\hat{n}$ by the normal at the updated $MP(x)$.
End.

As we shall see next, this simple approach gives a much more uniform representation and naturally treats the prototype swallowtail problem.

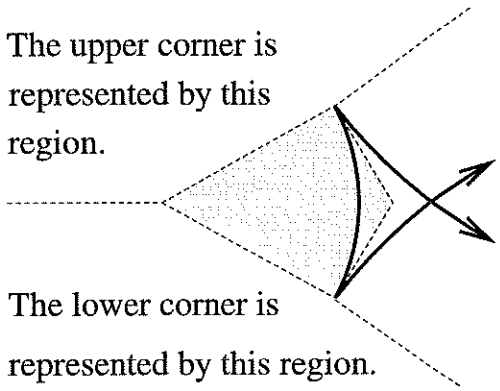
3.2 Numerical Experiments

Consider an ellipse evolving inward with unit speed as is shown in Figure 5. As discussed in the previous section, the Closest Point Method produces large gaps in the interface. A much more uniform representation of the swallowtail is derived using the Arrival Time Method (see, e.g., Figure 9). Over large times, this improvement leads to dramatically superior results, as can be seen by comparing Figures 6 and 10.

Of course, we also want an estimate of how closely each tracked point approximates the true wavefront. Analytically, an $O((\Delta x)^3/\Delta t)$ error should be produced over the length of the computation because time steps are carried out exactly (to within roundoff) and a quadratic interpolation step is used. In practice, we find that tracked points remain close to the true solution surface. For example, in the test problem of Figure 10f a very small error (measured as the L_1-distance of the tracked points $TP(x)$ from the true solution surface Γ) was produced that declined rapidly with Δx. See Table I.



(a) Representation of a swallowtail.



(b) Regional representation of the swallowtail.

Figure 7: (a) When a swallowtail is small, the nodal values over-represent corners. (b) Because large regions are used to represent corners, only the small shaded region is left to capture the end of the swallowtail. This small region will contain few (or no) grid points after the swallowtail is first formed.

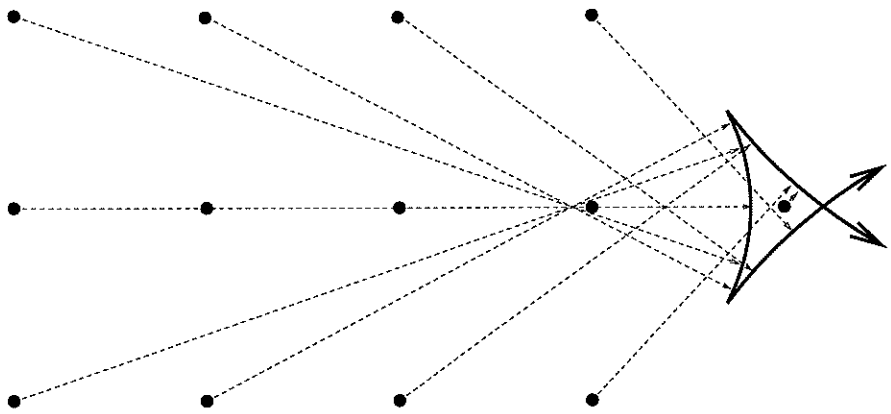
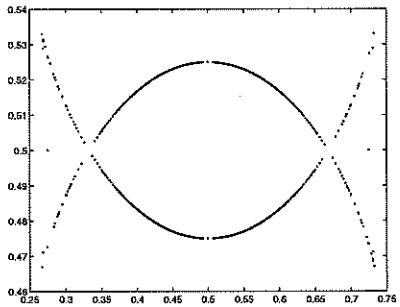
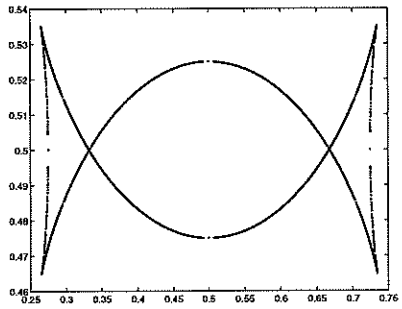


Figure 8: Minimizing arrival times rather than distance gives a more uniform representation of the interface



(a) Closest Point Method.



(b) Arrival Time Method.

Figure 9: (a) Using the Closest Point Method, the end of the swallowtail is lost. (b) By minimizing first arrival times, a good representation of the entire wavefront is obtained.

Δx	*Error*
1/20	1.4e-6
1/40	3.7e-7
1/80	1.5e-8

Table I. Errors for an initial ellipse, measured as L_1-distance of tracked points from exact solution surface.

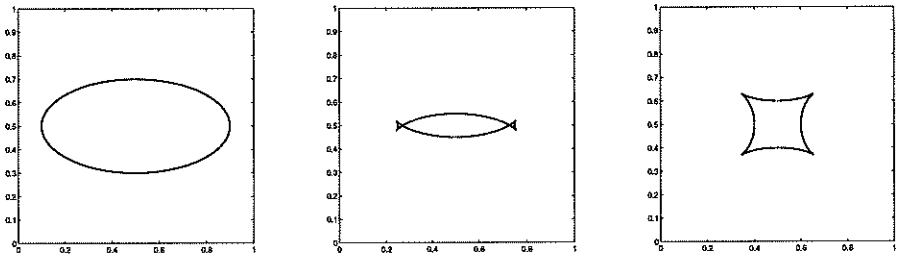
We now direct our attention to localization methods for improving the efficiency of DSE schemes.

3.3 Localization

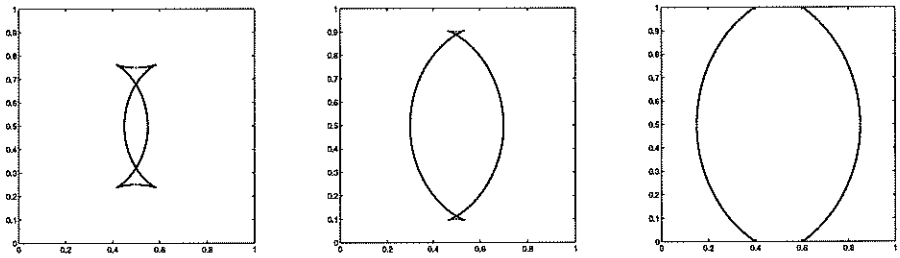
In previous sections, the search step was carried out globally. Although simple, this approach can be expensive because at each grid point, all other tracked points must be searched for a new minimizing point, leading to $O(n^2)$ operations per time step, where n is the total number of grid nodes.

Alternatively, a local search can be used over a radius R of each grid node [10] to achieve an $O(n)$ operation count per step. Notice that in this case we must choose

$$R > c\Delta t \tag{2}$$



(a) $t = 0.000$ (b) $t = 0.150$ (c) $t = 0.300$



(d) $t = 0.450$ (e) $t = 0.600$ (f) $t = 0.750$

Figure 10: Using an approximation to first arrival times, a uniform representation is achieved. Here, a wavespeed equal to 1 was considered. Discretization step sizes of $\Delta x = 1/80$ and $\Delta t = 1/40$ were used throughout the calculation.

because information about the interface should propagate more quickly than the interface itself [10]. Of course, this is a necessary and not a sufficient condition. In practice, larger values of R are often needed to treat interesting problems involving swallowtails.

In our formulation, each grid node value represents some tracked point on the interface. This fact allows us to design an algorithm that searches *all* tracked points in a neighborhood of the interface rather than a few tracked points in a neighborhood of each node. We proceed as follows:

1. Initialize the updated nodal values $TP(x)^{new} = MP(x)$ for each node x.

2. Evaluate

$$F(TP(x), p) = |(TP(x) - p) \cdot \hat{n}^{\perp}(TP(x))| + ||TP(x) - p||^2 \quad (3)$$

for each tracked point $TP(x)$ and each node p which lies inside a disc of radius r centered at the interface point $TP(x)$. Whenever $F(TP(x), p) < F(TP(x)^{new}, p)$ an update is made to the new nodal values: $TP(x)^{new} = TP(x)$.

Notice that this approach has the advantage that all tracked point information near the interface is instantaneously propagated globally away from the interface, so the propagation speed requirement (2) no longer applies and r can be selected independently of c and Δt.

This localization naturally leads to some modifications of the Extension Step. First, only nodes which are within a distance r of the interface should be used for interpolations since only these nodes are updated during the search. Also note that searching according to Equation (1) can cause neighboring nodes to represent different wavefronts (see Figure 11b) which makes interpolation impossible. Because the Closest Point Method does not exhibit this shortcoming (see Figure 11c), we carry out the Extension Step twice in the localized algorithm—once with a search that minimizes distance and once using Equation (1). Whichever result minimizes expression (3) is used as the updated value at each node.

We have found that this simple, fast approach gives excellent results in a wide variety of problems. In particular, the examples in the next three

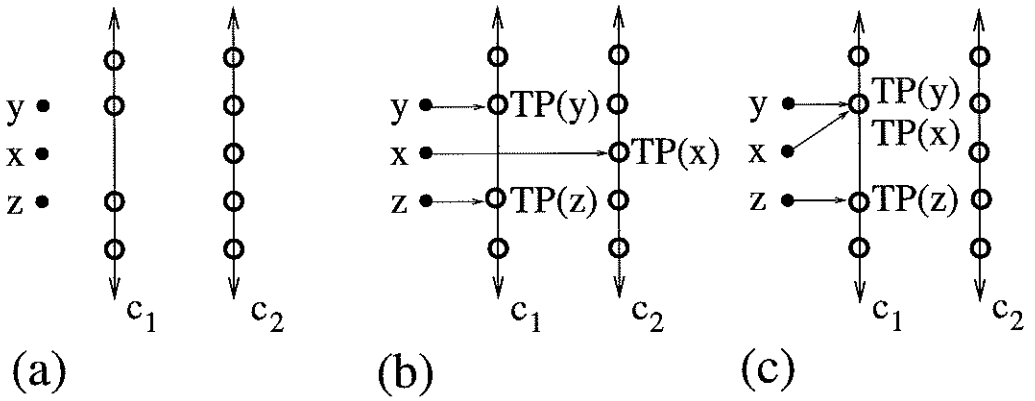


Figure 11: (a) Before the search step. Consider 3 nodes x, y and z and several neighboring tracked points: **o** (b) After the search step. If we search according to Equation 1, the nodal value at x will belong to the second wavefront c_2 since $|(x - TP(x)) \cdot \hat{n}^{\perp}(TP(x))| \ll |(x - TP(y)) \cdot \hat{n}^{\perp}(TP(y))|$. Neighboring values will belong to the first wavefront. Because $TP(x)$ belongs to a different wavefront than its neighbors, no interpolation will occur at x and resolution may be lost. (c) After the search step. If we search according to distance, an interpolation at node x is possible based on the tracked points $TP(x)$ and $TP(z)$.

sections are carried out using this localization[1].

4 Refraction

The Arrival Time Method described in the previous section applies to homogeneous media. In problems where the wavespeed is piecewise constant, the direction of propagation will change as the wavefront moves from one material to another. Specifically, the angle of refraction will be given by Snell's Law

$$\mathrm{Sn}(\theta) = \arcsin\left(\frac{c_b}{c_a}\sin(\theta)\right) \tag{4}$$

where θ is the angle of incidence of the ray and c_a and c_b are the wavespeeds in the original and final media (see, e.g., [7]).

To extend the Arrival Time Method to this refractive case, we must take Snell's Law into account in the Evolution Step. The Extension Step of the algorithm remains unchanged. Figure 12 gives a simple example of a refracting wavefront treated using this approach. For the variable index of refraction case, we proceed in a similar fashion, except now the Evolution Step is governed by the ray equations (see [7]),

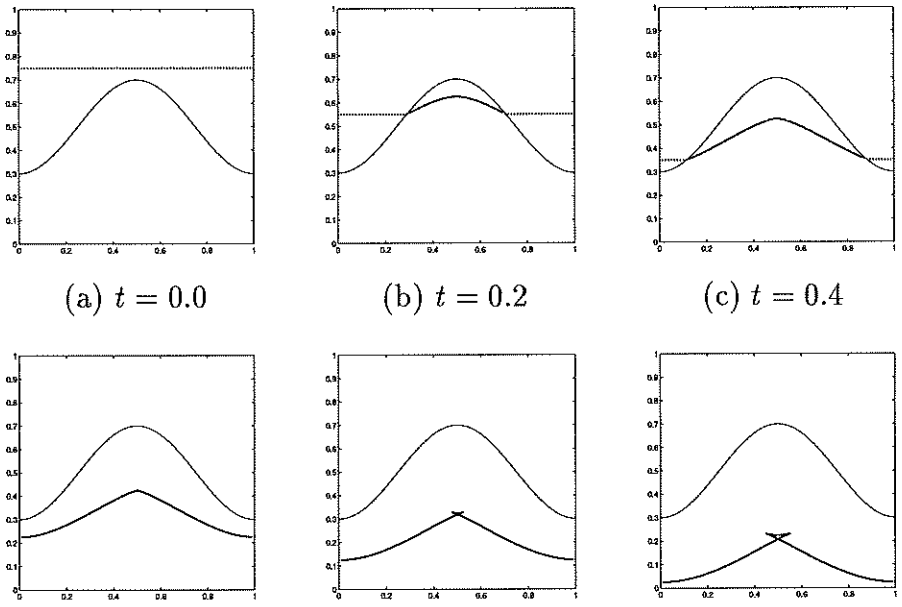
$$\frac{d^2X}{dt^2} = \nabla_X\left(\frac{n^2}{2}\right) \tag{5}$$

$$||X_t||^2 = n(X)^2 \tag{6}$$

where X is the coordinate of the ray being traced (or in our language, the coordinate of the tracked point) and $n(X) = 1/c(X)$ is the variable index of refraction.

Of course, physically, we expect that a reflected wave will also be produced when a wavefront passes from one material to another. Fortunately, these reflected components are straightforward to treat using methods discussed in the next section.

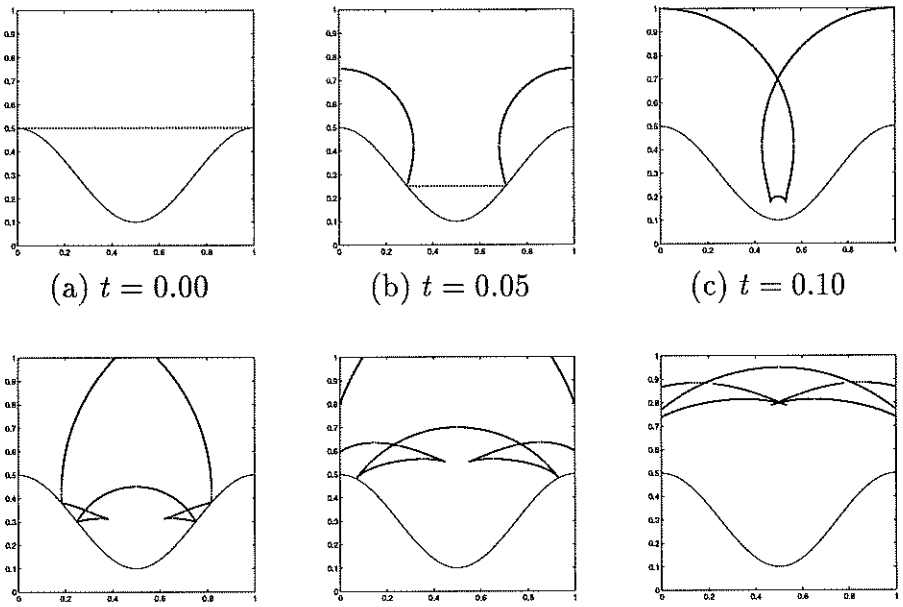
[1] We use a radius of four cells throughout. This is somewhat arbitrary: Other values appear equally effective. For example, using a radius of just two cells a solution of the swallowtail problem can be computed that essentially coincides with the global result shown in Figure 10.



(a) $t = 0.0$ (b) $t = 0.2$ (c) $t = 0.4$

(d) $t = 0.6$ (e) $t = 0.8$ (f) $t = 1.0$

Figure 12: Refraction is handled by applying Snell's Law in the Evolution Step. Here, the wavespeed is 1 in the upper region $\{(x, y) : y > 1/2 - 0.2\cos(2\pi x)\}$ and $1/2$ in the lower region. Discretization step sizes of $\Delta x = 1/80$ and $\Delta t = 1/20$ were used throughout the calculation.



(a) $t = 0.00$ (b) $t = 0.05$ (c) $t = 0.10$

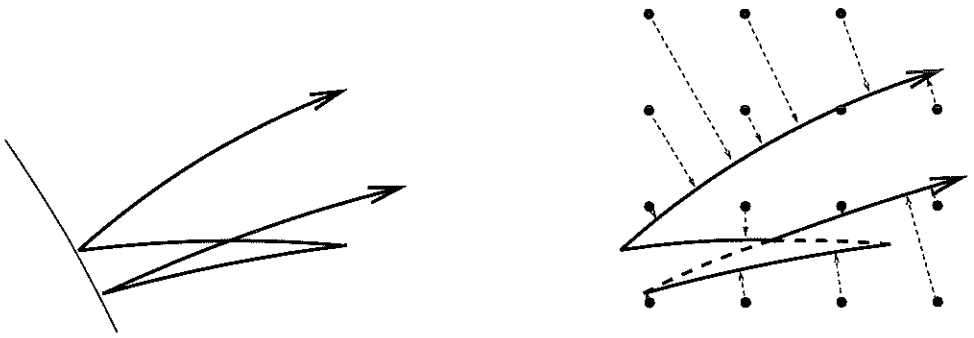
(d) $t = 0.15$ (e) $t = 0.20$ (f) $t = 0.25$

Figure 13: Reflections are handled by applying the Law of Reflection in the Evolution Step. Here, a wavespeed equal to 1 was considered. Discretization step sizes of $\Delta x = 1/80$ and $\Delta t = 1/20$ were used throughout the calculation.

5 Reflection

When a ray traveling in a medium encounters a boundary, part of the incident ray is reflected back into the medium. Very often, the direction of propagation of the reflected wave will be given by the Law of Reflection: *the angle of reflection equals the angle of incidence.*

To extend the Arrival Time Method to the reflective case, we must take the Law of Reflection into account in the Evolution Step. The Extension Step of the algorithm remains unchanged. Figure 13 gives an interesting example of a reflecting wavefront treated using this approach. Although this simple method gives a very good representation of the interface, small gaps occasionally form where wavefronts cross (see Figure 13f). These gaps arise when too few grid points are used to represent complicated reflecting wavefronts. See Figure 14.



(a) Reflected wavefront.

(b) For each node, a point on the interface is stored.

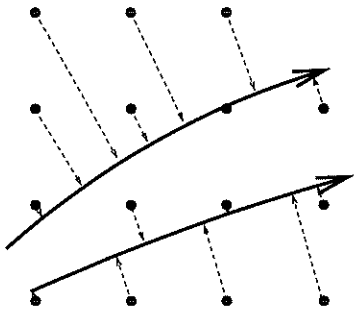
Figure 14: (a) When a kink reflects from a boundary, a rather complicated wavefront develops. (b) There are too few grid points to represent dashed segments, so gaps form in the interface.

Fortunately, this problem can usually be overcome simply by refining the mesh (see Figure 16a). In more complicated problems, two tracked points may be stored at each node—one for parts of the wavefront that have reflected an even number of times, the other for parts that have reflected an odd number of times. As shown in Figure 15, this approach gives a more uniform representation of reflected kinks and an improved treatment of complicated wavefronts.

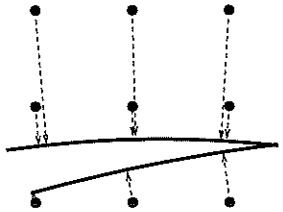
We now direct our attention to another important property of wave propagation: the intensity.

6 Intensity

Previous sections evolved wavefronts by propagating out both position and normal values. In this section, we describe how to treat intensity by retaining other attributes of the wavefront, such as wavefront curvature. Numerical experiments are also carried out to validate our approach.

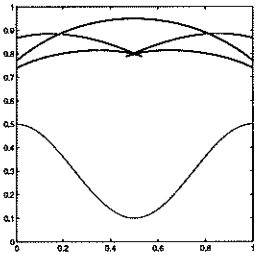


(a) Even reflections.

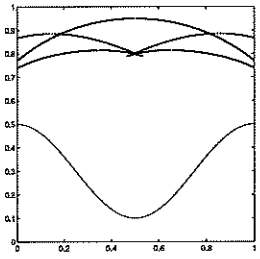


(b) Odd reflections.

Figure 15: If even (a) and odd (b) reflections are represented separately, then a more uniform treatment of the wavefront is obtained.



(a) $t = 0.25$



(b) $t = 0.25$

Figure 16: Small gaps that form may be eliminated by (a) refining the mesh (here we have taken $\Delta x = 1/115$) or (b) treating odd and even reflections separately (here $\Delta x = 1/80$).

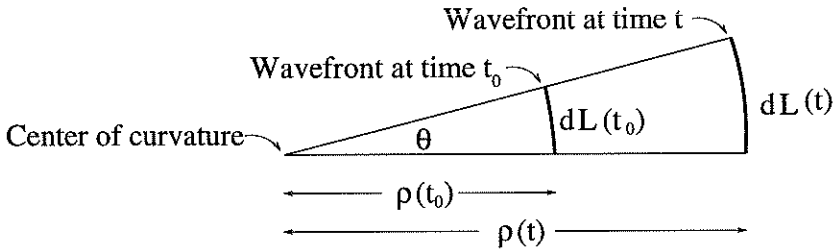


Figure 17: In two dimensions, the expansion ratio is the initial length of a wavefront element divided by the final length. For a homogeneous medium, $\xi(t_0,t) = \frac{dL(t_0)}{dL(t)} = \frac{\theta\rho(t_0)}{\theta\rho(t)} = \frac{\rho(t_0)}{\rho(t)}$.

6.1 The Method

To develop a method for evolving intensity values, we make use of a simple observation: Away from degenerate cases, the intensity of a ray at time t is given by the intensity at an earlier time, t_0, multiplied by the *expansion ratio*[2], $\xi(t_0,t)$. As shown in Figure 17, the two dimensional expansion ratio for a homogeneous medium is just the initial radius of curvature divided by the final radius of curvature. I.e.,

$$\xi(t_0,t) = \frac{\rho(t_0)}{\rho(t)}. \tag{7}$$

In three dimensions, it is easily shown [7] that the expansion ratio for a homogeneous medium becomes

$$\xi(t_0,t) = \frac{\rho_1(t_0)\rho_2(t_0)}{\rho_1(t)\rho_2(t)}. \tag{8}$$

where ρ_1 and ρ_2 are the principal curvatures of the wavefront surface.

Thus, intensity values may be propagated along a ray using just the initial intensity, time and principal curvature values. For example, in two dimensions the intensity is given by

$$I(t) = I_0\left(\frac{\rho(t_0)}{\rho(t_0)+c(t-t_0)}\right) \tag{9}$$

[2]The *expansion ratio* is a measure of the expansion of the cross-section of a tube of rays. See Figure 17 for a derivation of the expansion ratio in two dimensions and reference [7] for a derivation in the general case.

in terms of these quantities. Notice that this simple analytical approach applies even when the intensity is infinite (e.g., at a focus) at some intermediate time. Indeed, even degenerate cases (i.e., the radius of curvature is initially zero) may be treated analytically. See [7] for further details.

When a wavefront is reflected or refracted, however, curvature and intensity values can change and Equation (9) cannot be used. Fortunately, updated values for these quantities are easily calculated. After reflection, intensity is unchanged and the curvature of a 2D wavefront is given by

$$\kappa_{reflected} = \kappa_{incident} - \frac{2}{\cos(\theta)}\kappa_{boundary} \tag{10}$$

where $\kappa_{incident}$ is the curvature of the incident wavefront, θ is the angle of incidence and $\kappa_{boundary}$ is the curvature of the reflective surface. After refraction, the curvature of a 2D wavefront is given by

$$\kappa_{refracted} = \frac{\mathrm{Sn}'(\theta)}{\sqrt{1+(1-(\frac{c_b}{c_a})^2)\tan^2(\theta)}}\kappa_{incident} + \frac{\mathrm{Sn}'(\theta)-1}{\sqrt{1-(\frac{c_b}{c_a}\sin(\theta))^2}}\kappa_{boundary} \tag{11}$$

where $\mathrm{Sn}'(\theta)$ is the derivative of Snell's Law (4), $\kappa_{boundary}$ is the curvature of the boundary between media and c_a and c_b are the propagation speeds in the original and final media. Updated intensity values after refraction are given by

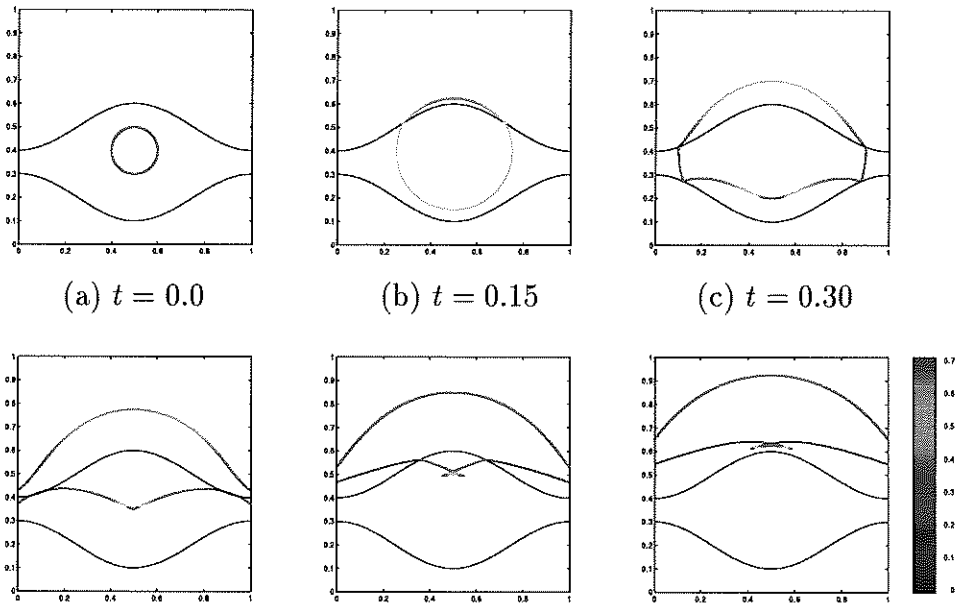
$$\frac{I_{refracted}}{I_{incident}} = \frac{\cos(\theta)}{\cos(\mathrm{Sn}(\theta))} \tag{12}$$

Thus, we treat intensity by storing curvature and intensity values at an initial time t_0 (which is also stored). These stored values are updated (using Equations (10-12)) whenever a reflection or refraction occurs, to give an explicit formula (9) for intensity.

We now use this approach to evolve the intensity of a propagating, two dimensional wavefront and compare our results to a front tracking method.

6.2 Numerical Experiments

Consider the evolution of the initially circular wavefront given in Figure 18a. Using the methods of the previous section, intensity values are determined



(a) $t = 0.0$ (b) $t = 0.15$ (c) $t = 0.30$

(d) $t = 0.45$ (e) $t = 0.60$ (f) $t = 0.75$

Figure 18: Intensity is calculated by propagating out curvature values. Here, the wavespeed is 1 in the middle region $\{(x, y) : 0.2 + 0.1\cos(2\pi x) \leq y < 0.5 - 0.1\cos(2\pi x)\}$ and $1/2$ in the upper region. Discretization step sizes of $\Delta x = 1/80$ and $\Delta t = 1/20$ were used throughout the calculation.

for each tracked point after a variety of reflections and refractions. A color visualization of these intensity values is given in Figure 18. Plotting the intensity as a function of arclength for the lower curve at $t = 0.75$ gives the results in Figure 19. These intensity values agree well with the exact solution since they have less than a 2 percent relative error (as measured in the maximum norm) when compared to a well resolved front tracking calculation.

In all our calculations, intensity and curvature values for new (i.e., interpolated) points are derived by linearly interpolating the values of neighboring points at the current time. Higher order interpolations may be preferred when very accurate results are sought.

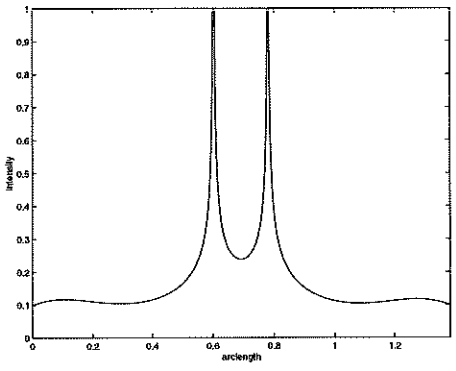


Figure 19: Intensity as a function of arclength for the lower curve of Figure 18f.

7 Curvature-Dependent Motions

In previous sections, we described methods for evolving interfaces according to position-dependent velocities. However, asymptotic models for physical processes often yield equations of motion for a surface moving with a velocity that is a function of its local geometry, i.e., a function of local normal, curvature and derivatives of these quantities. Toward this end, we now consider an extension of the Closest Point method to more general curvature-dependent motions and validate our approach with numerical experiments.

7.1 The Method

A particularly fundamental motion arises when the normal velocity equals the mean curvature of the surface. For this important case, a simple DSE scheme can be constructed. Moreover, this scheme applies to motion by mean curvature for objects of any codimension. To illustrate this, we will develop the model for the arbitrary codimension case, and apply it to a curve in three dimensions with a velocity equal to its vector curvature $\kappa\hat{n}$.

We begin by noting that the vector mean curvature for a surface of arbitrary codimension is given by

$$\kappa\hat{n} = -\Delta\nabla\left(\frac{d^2}{2}\right) \tag{13}$$

where κ is the local mean curvature and d is the distance to the surface (see, e.g., [1]). Taking into account that

$$d\nabla d = x - CP(x)$$

where $CP(x)$ is the closest point to x on the surface we obtain a very simple expression for vector mean curvature:

$$\kappa\hat{n} = -\Delta(x - CP(x)) = \Delta CP(x).$$

Thus, motion by mean curvature for surfaces of arbitrary codimension can be achieved by replacing the Evolution Step of the Closest Point Method by

$$TP(x)_t = \Delta TP(x)|_{TP(x)}.$$

Other curvature dependent velocities are possible by replacing the Evolution Step by

$$TP(x)_t = F(\Delta TP(x)|_{TP(x)} \cdot \hat{n})\ \hat{n}$$

since $\kappa = \Delta CP(x)|_{CP(x)} \cdot \hat{n}$. Of course, it is crucial to evaluate the Laplacian term at the closest point since Equation (13) is only valid *at the interface.*

We now validate our algorithm with some numerical experiments.

7.2 Numerical Experiments

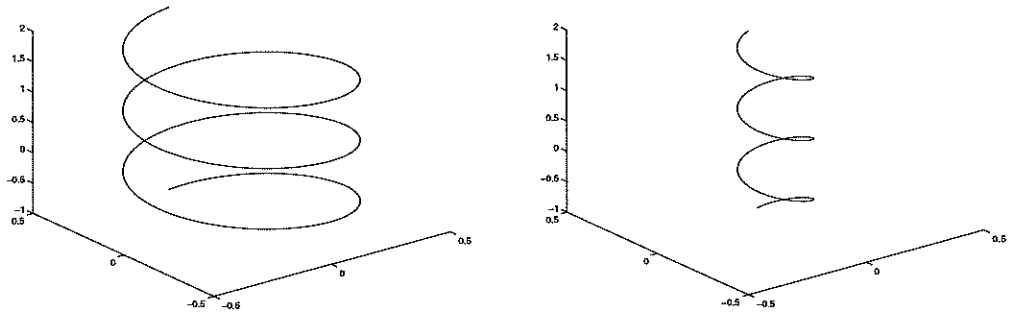
For our final example, consider the curvature motion of a periodic spiral in three dimensions,

$$\begin{aligned} x &= 0.4\sin(2\pi s), \\ y &= 0.4\cos(2\pi s), \\ z &= s. \end{aligned}$$

Here, it is easily shown that the exact solution is also a spiral, but with a radius that shrinks according to the ordinary differential equation,

$$\dot{r} = -\frac{r}{r^2 + \frac{1}{4\pi^2}}.$$

Using the algorithm of the previous section, the maximum error in the position of the filament at time $t = 0.1$ was computed for several Δx (Δt was taken to be $\frac{1}{4}(\Delta x)^2$). The results for a number of experiments are reported in Table II, below.



Initial Filament *After a time t=.1*

Figure 20: The curvature motion of a filament, $v_n = \kappa \hat{n}$.

Δx	*Error*	*Conv. Rate*
1/10	0.00688	-
1/20	0.00159	2.1
1/30	0.00071	2.0

Table II. Errors for the curvature motion of a filament.

These results are suggestive of an approximately second order error in the position of the front.

In all calculations, the standard second order finite difference approximation of the Laplacian was used with forward Euler time stepping. Interpolated values of the Laplacian were derived using linear interpolation. Extension Steps were carried out by fitting a spline to the filament and reinitializing the closest point representation at each step.

8 Summary and Topics for Future Research

In this work, we have considered a class of Dynamic Surface Extension methods based on an approximation of first arrival times instead of closest points. Our approach uses a spatially distributed representation of the surface that

maintains a uniform and balanced resolution during the formation of complicated self-intersecting fronts (such as swallowtails) which are not well treated by the original Closest Point Method. Simple extensions for treating refraction, reflection and intensity are also provided and validated with numerical experiments. Finally, a surprisingly simple generalization of our method is given for interesting geometric motions, such as mean curvature flow. As with other DSE schemes our methods automatically treat points, filaments and surfaces of arbitrary codimension, which we have illustrated with curvature motion of a filament in three dimensions.

A variety of interesting topics for future research are still open. For example, notice that our approach requires a thresholding step to determine when interpolations occur (see Section 2.2). As we saw throughout the paper, it is often adequate to base this thresholding on the position and normal values of tracked points. Of course, no single threshold is adequate for all problems. It would be interesting to investigate robust decisions based on propagating out other quantities. For example, one might consider propagating initial position and normal values along each ray (cf. [12]), since interpolations can only occur when these quantities are approximately equal.

Other potential areas for future research include studies of the method's three dimensional performance (cf. [10]), especially for optical intensity calculations. It would also be very interesting to extend the method to include the effects of geometrical diffraction (cf. [7]), i.e. the bending of the wavefront as it passes by an obstacle.

There is also a great deal to explore in terms of using this type of representation to move objects of more complex topology and geometry, such as surfaces with boundaries (or curves with endpoints), objects of composite topology (such as a filament attached to a sheet), and surfaces or curves with triple point junctions.

Another major direction would be to couple these self-intersecting surface representations to physical processes occuring off the interface. For example, it would be interesting to treat the case of multiply intersecting shock wave fronts coupled to surrounding gas dynamics.

Finally, further work in the area of curvature dependent motions is also possible. Computationally, the construction of fast extension methods would be of great practical importance. Theoretically, it would be interesting to study the convergence properties of the method. It would be particularly interesting to determine if surfaces fatten (or develop interiors) when mergers

occur. See [2] for a detailed discussion on the "fattening phenomenon."

9 Acknowledgments

We thank John Steinhoff for many helpful discussions on Dynamic Surface Extension methods. We also thank Ron Fedkiw for interesting discussions on applications of intensity propagation.

References

[1] L. Ambrosio and H. Soner. Level set approach to mean curvature flow in arbitrary codimension. *J. Differential Geometry*, 43:693–737, 1996.

[2] G. Bellettini, M. Novaga, and M. Paolini. An example of three dimensional fattening for linked space curves evolving by curvature. *Comm. Partial Differential Equations*, 23(9-10):1475–1492, 1998.

[3] J. Benamou. Big ray tracing: Multivalued travel time field computation using viscosity solutions of the eikonal equation. *J. Comput. Phys.*, 128(2):463–474, 1996.

[4] Y. Brenier and L. Corrias. Capturing multivalued solutions of the eikonal equation. Technical Report, INRIA, 1995.

[5] M. Brown. Numerical considerations in ray tracing and ray expansions of the acoustic wavefield. *J. of Acoustical Society of America*, August 15, 1984.

[6] E. Fatemi, B. Engquist, and S. Osher. Numerical solution of the high frequency asymptotic expansion for the scalar wave equation. *J. Comput. Phys.*, 120:145–155, 1995.

[7] J.B. Keller and R.M. Lewis. Asymptotic methods for partial differential equations: The reduced wave equation and Maxwell's equations. In J.B. Keller, D.W. McLaughlin, and G.C. Papanicolaou, editors, *Surveys in Applied Mathematics*, pages 1–82. Plenum, New York, 1995.

[8] S. Osher and J.A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79:12–49, 1988.

[9] J. Steinhoff and M. Fan. Eulerian computation of evolving surfaces, curves and discontinuous fields. Technical Report, University of Tennessee Space Institute, Tullahoma, TN 37388, 1998.

[10] J. Steinhoff, M. Fan, and L. Wang. A new Eulerian method for the computation of propagating short acoustic and electromagnetic pulses. Technical Report, University of Tennessee Space Institute, Tullahoma, TN 37388, 1998.

[11] J. Strain. Fast tree-based redistancing for level set computations. *J. Comput. Phys.*, 152:648–666, 1999.

[12] W. Symes. A slowness matching finite difference method for traveltimes beyond transmission caustics. Technical Report, Dept. of Computational and Applied Mathematics, Rice University, Houston, Texas 77005, 1996.

[13] C. Tam. Computational aeroacoustics: Issues and methods. *AIAA J.*, 33(10):1788–1796, 1995.