

Fast Marching Methods and Level Set Methods for Propagating Interfaces

J.A. Sethian *

Department of Mathematics
University of California, Berkeley, California 94720
sethian@math.berkeley.edu

Abstract

In this set of notes, we review past work on Fast Marching Methods, introduced in [55], and Level Set Methods, introduced in [46], for tracking propagating interfaces in two and three space dimensions. Both sets of techniques are based on a partial differential equations view of interface motion, and rely on the use of the theory of viscosity solutions, upwind schemes for hyperbolic conservation laws, and the theory of curve and surface evolution developed in [51]. Both sets of techniques require an adaptive methodology to obtain computational efficiency. In the case of Fast Marching methods, this leads to the use of heap sort schemes borrowed from discrete network algorithms; in the case of level set methods, this leads to the Narrow Band Method given in [1]. We review the optimal implementation of these methods, and discuss a series of applications, including those from seismic simulations, combustion and fluid mechanics, robotic navigation, and etching and deposition in semi-conductor manufacturing.

*Supported in part by the Applied Mathematics Subprogram of the Office of Energy Research under contract DE-AC03-76SF00098, and the National Science Foundation.

Citation Information:
J.A. Sethian,
von Karman Institute Lecture Series, Computational Fluid Mechanics, 1998

Theory and Algorithms

1 Introduction

Fast Marching Methods and Level Set Methods are computational techniques for tracking propagating interfaces. They share the virtues of working in an arbitrary number of space dimensions with no change, handle topological merger and splitting with no special procedures, and accurately and efficiently compute the motion of fronts moving under complex speed laws, including large variations in velocities and sharp discontinuities.

Fast Marching Methods, introduced by Sethian in [55], approximate the solution of a boundary value partial differential equations view of propagating interface, while level set methods, introduced by Osher and Sethian in [46], approximate the solution of an initial value partial differential equation. At the core, both techniques rely on viscosity solutions for Hamilton-Jacobi equations, upwind schemes for hyperbolic conservation laws, and the theory of curve and surface evolution developed by Sethian in [51]. They have been used in a large variety of applications, including problems in fluid interface motion, combustion, dendritic solidification, etching and deposition in semi-conductor manufacturing, robotic navigation and path planning, image segmentation in medical imaging scans, computation of seismic travel times, and aspects of computational geometry and computer vision.

Both sets of techniques require an adaptive methodology to obtain computational efficiency. In the case of Fast Marching methods, this leads to the use of heap sort schemes borrowed from discrete network algorithms; in the case of level set methods, this leads to the Narrow Band Method given in [1].

These notes review some recent work in these areas, with a focus on implementation and applications. A overview and summary of current work on Fast Marching and level set methods is given in a recent cumulative introduction and resouce book on level set and Fast Marching Methods [59]; an additional resource is the web site

http://math.berkeley.edu/~sethian/level_set.html

2 Initial and Boundary Value Formulations of Front Propagation

2.1 A Boundary Value Formulation

Imagine a closed curve Γ in the plane propagating normal to itself with speed F . Furthermore, assume that $F > 0$, hence the front always moves “outwards”. One way to characterize the position of this expanding front is to compute the arrival time $T(x, y)$ of the front as it crosses each point (x, y) , as shown in Figure 1.

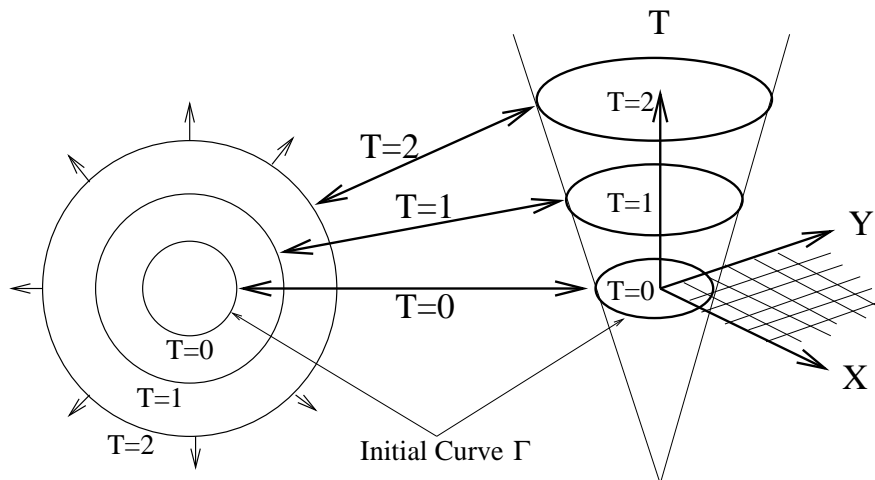


Figure 1: Transformation of Front Motion into Boundary Value Problem

The equation that describes this arrival surface $T(x, y)$ is easily derived. Using *distance = rate * time* (see Figure 2), we have

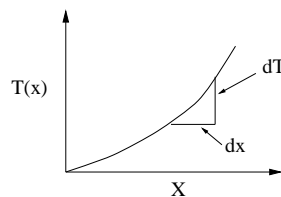


Figure 2: Setup for Boundary Value Formulation

$$dx = F (dT)$$

and hence

$$1 = F \frac{dT}{dx}.$$

In multi-dimensions, the spatial derivative of the solution surface T becomes the gradient, and hence we have

$$|\nabla T|F = 1 \quad T = 0 \text{ on } \Gamma. \quad (1)$$

Thus, the front motion is characterized as the solution to a boundary value problem; if the speed F depends only on position, then the equation reduces to the familiar Eikonal equation.

2.2 Initial Value Formulation

Conversely, suppose we embed the initial position of the front as the zero level set of a higher dimension function ϕ . We can then identify the evolution of this function ϕ with the propagation of the front itself through a time-dependent initial value problem. At any time, the front is given by the zero level set of the time-dependent level set function ϕ , see Figure 3.

In order to derive an equation of the motion for this level set function ϕ , we note that the stipulation that the zero level set of the evolving function ϕ always match the propagating hypersurface means that

$$\phi(x(t), t) = 0. \quad (2)$$

By the chain rule,

$$\phi_t + \nabla\phi(x(t), t) \cdot x'(t) = 0. \quad (3)$$

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

$$\phi_t + F|\nabla\phi| = 0, \quad (4)$$

$$\text{given } \phi(x, t = 0). \quad (5)$$

This is the level set equation introduced by Osher and Sethian [46]. For certain forms of the speed function F , one obtains a standard Hamilton–Jacobi equation.

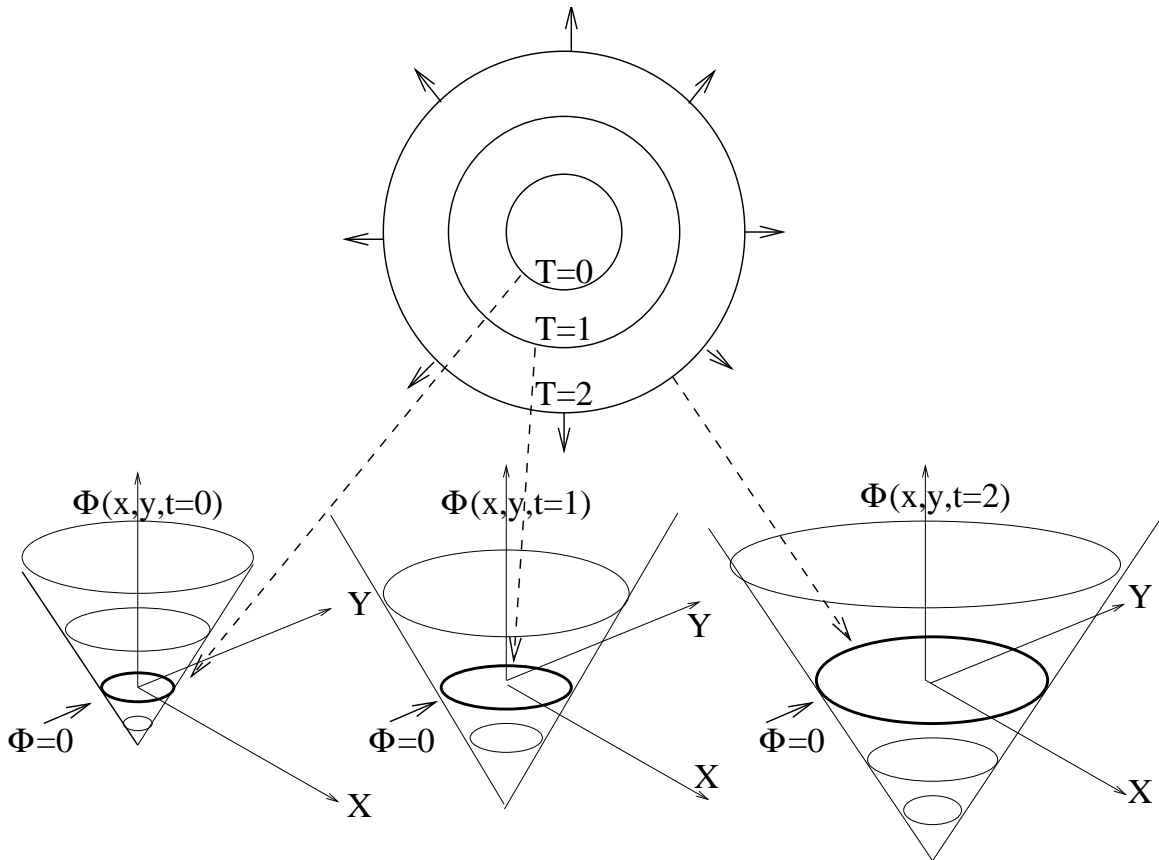


Figure 3: Transformation of Front Motion into Initial Value Problem

This equation describes the time evolution of the level surface function ϕ in such a way that the zero level set of this evolving function is always identified with the propagating interface; see Figure 3.

Thus, we wish to solve

Initial Value Formulation	Boundary Value Formulation
$\phi_t + F \nabla\phi = 0$	$ \nabla T F = 1$
Front = $\Gamma(t) = \{(x, y) \phi(x, y, t) = 0\}$	Front = $\Gamma(t) = \{(x, y) T(x, y) = t\}$
Applies for arbitrary F	Requires $F > 0$

(6)

2.3 Advantages of These Perspectives

There are certain advantages associated with these two perspectives on propagating interfaces.

- Both are unchanged in higher dimensions; that is, for surfaces propagating in three dimensions and higher.
- Topological changes in the evolving front Γ are handled naturally. The position of the front at time t is given either by the zero level set $\phi(x, y, t) = 0$ of the evolving level set function or the contour $T = t$ of the boundary value solution. This set need not be connected, and can break and merge as t advances. In both cases, the key fact is that the boundary value solution $T(x, y)$ and the level set function ϕ remain single-valued.
- Both rely on viscosity solutions of the associated partial differential equations in order to guarantee that the unique, entropy-satisfying weak solution will be obtained.
- Both can be converted into computational schemes by exploiting schemes borrowed from the numerical solutions of hyperbolic conservation laws.
- Both are made efficient through the use of adaptive computational strategies.

At the same time, there are significant differences between the two approaches.

- The most obvious distinction between the two views is that the initial value level set formulation allows for both positive and negative speed functions F ; the front may move forwards and backwards as it evolves. The boundary value perspective is restricted to fronts that always move in the same direction, because it requires a single crossing time T at each grid point, and hence a point cannot be revisited.
- Approximation of more complex speed functions F , such as those including curvature, are most naturally done in the initial value level set perspective, because of the ease of approximating such terms. For

example, at any point the normal vector is given in the level set formulation by

$$\vec{n} = \frac{\nabla\phi}{|\nabla\phi|}, \quad (7)$$

and the curvature of each level set is easily obtained from the divergence of the unit normal vector to the front, i.e.,

$$\kappa = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|} = \frac{\phi_{xx}\phi_y^2 - 2\phi_x\phi_y\phi_{xy} + \phi_{yy}\phi_x^2}{(\phi_x^2 + \phi_y^2)^{3/2}}. \quad (8)$$

- In contrast, speed functions F which depend on position and/or vary widely are best handled through the boundary value perspective approximated through the use of Fast Marching Methods. This is because
 1. Fast Marching Methods employ no time step, and hence are not subject to CFL conditions, unlike level set methods.
 2. Through the use of heap sort algorithms, Fast Marching Methods can be made extremely computationally efficient, far eclipsing level set methods.

2.4 Solutions to the Boundary Value and Initial Value Perspective

We can be slightly informal and include both formulations under the general partial differential equation

$$\alpha u_t + H(Du, x) = 0. \quad (9)$$

Here, Du represents the partials of u in each variable, for example, u_x and u_y . For example, in the case of Eikonal equation, $\alpha = 0$, and the function H reduces to $H = F|\nabla\phi|$.

One of the main difficulties in solving the above equations is that the solution need not be differentiable, even with arbitrarily smooth boundary data. This non-differentiability is intimately connected to the notion of appropriate weak solutions; our goal is construct numerical techniques which naturally account for this non-differentiability in the construction of accurate and efficient approximation schemes, and admit physically correct non-smooth solutions. This is the goal of the next section.

3 Singularity Development, Viscosity Solutions, and Numerical Approximations

3.1 Fundamental Equations and Non-Differentiability

We begin with an example based on the simple Eikonal equation

$$|\nabla u| = f(x, y), \tag{10}$$

with $f = 1$. The solution may be non-differentiable, even with smooth initial data. As an illustration, consider the case of

$$|\nabla u| = 1 \text{ outside the unit circle} \quad u = 0 \text{ on the unit circle.} \tag{11}$$

It is readily checked the function $u(x, y) = (x^2 + y^2)^{1/2} - 1$ corresponding to the distance function from the unit circle solves the given Eikonal equation. Indeed, specifying that the right-hand-side $f(x) = 1$ and providing zero as the boundary condition implies that the solution is just the distance to the initial curve Γ . The solution is shown as a family of concentric circles in Figure 4. It is, of course, easy to check that the solution is everywhere differentiable.

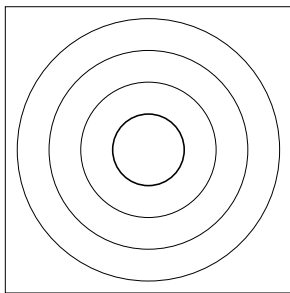


Fig. 4

Figure 4: Distance Function Solution to Eikonal Equation $|\nabla u| = 1$

In contrast, consider now boundary data given by the parabola $y = x^2$, and suppose we try to find the distance from any point in the domain above the parabola to the boundary curve. One constructive approach is to put a particle at each point of the boundary curve and move each particle away

from the curve in a normal direction with unit speed. At any time C , the position of these particles gives a set of all points a “distance” C away from the boundary curve. This solution is given by the swallowtail shown in Figure 5a, obtained by drawing the normal to each point on the boundary data; and proceeding out a distance u along each of these normals.

While this is certainly a possible solution, it does not correspond to the closest points of the boundary data. The above construction builds the set of points which are “locally” a distance C from the boundary curve. A different approach, which yields the true set of closest points, comes from asking for the set of all points located a given distance C away. One way to build this solution (shown on the right) is through a Huyghens principle construction; the solution is developed by imagining wavefronts emanating with unit speed from each point of the boundary data; the envelope of these wave fronts always corresponds to the “first arrivals”, and will automatically produce the solution given on the right in Figure 5b. This is the approach taken in [51]; there, an entropy condition is introduced for propagating interface on the basis of Huyghens principle and the limiting solution which develops the ideas of curve and surface evolution under various speed laws, including non-constant speeds and curvature-based motion.

We note that there will be a vertical ridge along which two points on the boundary curve are the same distance away, as shown in Figure 5b. Along this ridge, the solution is non-differentiable, and the gradient ∇u is not defined.

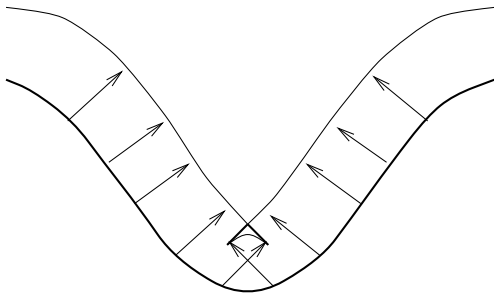


Fig. 5a

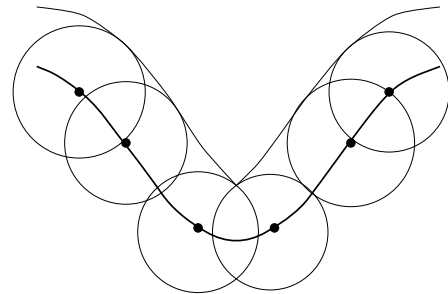


Fig. 5b

Points a “local distance” from the Boundary Data Globally Closest Points to Boundary Data

Figure 5: Possible Solutions to Eikonal Equation $|\nabla u| = 1$

Both of the above constructions can be thought of as solutions to the Eikonal equation. However, the solution that we want, corresponding to the

shortest distance or “first arrival”, is the one obtained through the Huyghens construction. Another way to obtain the solution is through the notion of an entropy condition; as defined in [50, 51], we imagine the boundary curve as a source for a propagating flame, and the expanding flame satisfies the requirement that once a point in the domain is ignited by the expanding front, it stays burnt. This construction yields the entropy-satisfying Huyghens construction given in Fig. 5b.

Another way to obtain this entropy-satisfying Huyghens construction comes from adding a smoothing term to the equation. Consider the associated “viscous” partial differential equation given by

$$|\nabla u(x)| = f(x) + \epsilon \nabla^2 u. \quad (12)$$

It can be shown that the viscous term $\epsilon \nabla^2 u$ acts to smooth out sharp corners in the solution, and guarantees that the solution stays smooth in the entire domain Ω (see [51, 59]). As ϵ goes to zero, the solution converges to the first arrival solution given in Figure 5b. We then have the limiting cases given in Figure 6 below.

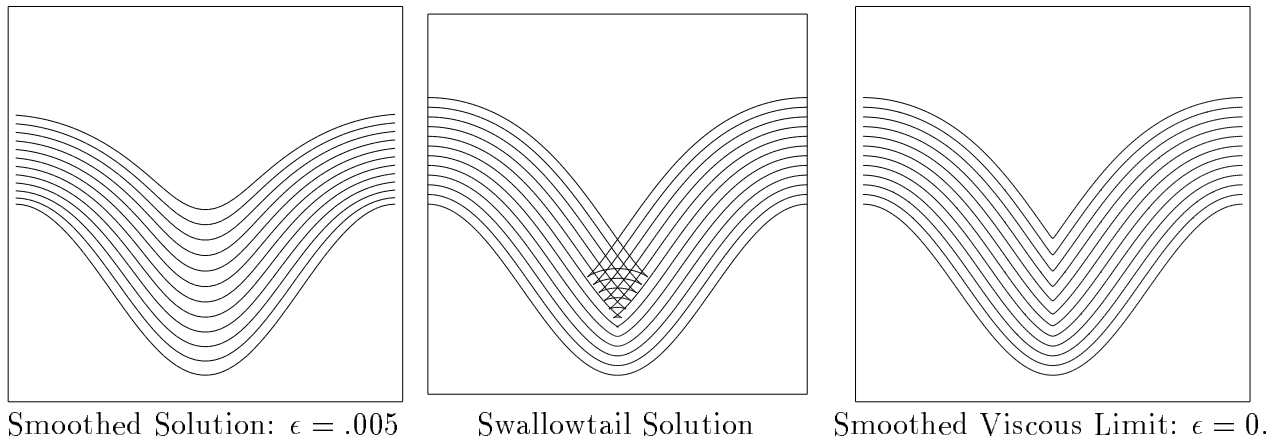


Figure 6: Viscous limit as $\epsilon \rightarrow 0$ for $|\nabla u(x)| = f(x) + \epsilon \nabla^2 u$

3.2 Viscosity Solutions of Hamilton-Jacobi Equations

Thus, our goal is to build numerical methods that automatically extract this viscous limit. Before doing so, we note that the formal way of defining this

viscous limit is through the idea of viscosity solutions. Instead of defining the solution as the viscous limit, one instead analyzes the behavior of potential solutions when measured against possible test functions. Briefly, and following the definitions of Crandall and Lions, [20],

Definition:

u is said to be a **viscosity solution** of

$$u_t + H(Du, x) = 0$$

if for all smooth test functions v ,

1. if $u - v$ has a local maximum at a point (x_o, t_o) then

$$v_t(x_o, t_o) + H(Dv(x_o, t_o), x_o) \leq 0$$

2. if $u - v$ has a local minimum at a point (x_o, t_o) then

$$v_t(x_o, t_o) + H(Dv(x_o, t_o), x_o) \geq 0.$$

We begin by noting that first, the definition does not differentiate u , only v , and second, it follows the usual trick of integration by parts, and passes all the derivatives off of u and onto v . This turns out to be a reasonable and valuable definition of the solution, because one can show the following:

- *If u is a smooth solution of the Hamilton–Jacobi equation, then it is a viscosity solution.* In other words, any classical solution that stays smooth for all time satisfies the two inequalities in the above definition.
- *If a viscosity solution u is differentiable at some point, then it satisfies the Hamilton–Jacobi equation there.* In other words, where the viscosity solution is smooth, it gives the same answer as the classical solution.
- *The above viscosity solution is unique, given appropriate initial conditions.* That is, there is only one viscosity solution satisfying the above definition.

- *The solution produced by taking the limit of the smooth solutions u_ϵ as ϵ goes to zero is a viscosity solution; by uniqueness, this solution must be the one given above.*

Thus, one proves that the viscosity solution is the limiting solution as the smoothing term $\epsilon \nabla^2 u$ goes to zero, and thus automatically constructs the globally “closest” solution to the distance function problem. Armed with this definition, our goal is to develop numerical approximations which correctly select this viscous limit. Since the entropy condition is similar to the one for hyperbolic conservation laws, it suggests using the associated numerical methodologies to solve the equations of motion, as discussed in [52]. Our goal now is to solve the Eikonal equation with right-hand-side F , using an “entropy-satisfying” approximation.

3.3 Upwind Schemes and Numerical Approximations

3.3.1 Upwind Schemes: Quadrature

As motivation for the use of upwind schemes for approximating the gradient operator, consider the one-dimensional Eikonal equation given by

$$u_x^2 = f^2(x) \quad u(0) = 0. \quad (13)$$

Here, the right-hand-side $f(x) > 0$ is given, and the goal is to construct $u(x)$ away from the boundary condition that $u(0) = 0$. We note immediately that the solution to even this problem is not unique; if $v(x)$ solves the problem, then so does $-v(x)$. Hence, we further restrict ourselves to non-negative solutions u .

We can imagine building the solution “outwards” along the positive and negative x -axis from the origin, and in fact solve each problem separately. Using a first order approximation, we simply try to solve the ordinary differential equations:

$$\begin{aligned} \frac{du}{dx} &= \sqrt{f(x)} & u(0) &= 0 & x &\geq 0 \\ \frac{du}{dx} &= -\sqrt{f(x)} & u(0) &= 0 & x &\leq 0 \end{aligned} \quad (14)$$

Since the right-hand-side of this o.d.e. is only a function of x , we are essentially performing numerical quadrature. Using the standard finite difference

notion that $u_i \approx u(i\Delta x)$, and $f_i = f(i\Delta x)$, we can approximate each of these two problems using Euler's method, namely

$$\begin{aligned} \frac{u_{i+1}-u_i}{\Delta x} &= \sqrt{f_i} \quad i > 0 \\ \frac{u_i-u_{i-1}}{\Delta x} &= -\sqrt{f_i} \quad i < 0 \end{aligned} \tag{15}$$

where $u_0 = 0$. This is an upwind scheme; we are computing derivatives using points “upwind” or towards the boundary condition. Another way to look at this is that we are solving each ordinary differential equation away from the boundary condition.

3.3.2 Upwind Schemes: Evolving Interfaces

Further motivation for approximating the gradient using upwind differences comes from the instructive example of an evolving curve whose position can always be described as the graph of a function. Consider the initial front given by the graph of $g(x)$, with g and g' periodic on $[0, 1]$, and suppose that (1) the front propagates with speed $F = 1$ in its normal direction and (2) remains a function for all time. Let ψ be the height of the propagating function at time t , thus $\psi(x, 0) = g(x)$. We then consider the initial value problem

$$\psi_t = (1 + \psi_x^2)^{1/2}, \quad \psi(x, 0) = g(x) = \begin{cases} 1/2 - x & x \leq 1/2 \\ x - 1/2 & x > 1/2 \end{cases}. \tag{16}$$

The initial front is a “V” formed by rays meeting at $(1/2, 0)$. By the entropy condition and Huyghens principle construction, the solution at any time t is the set of all points located a distance t from the initial “V”. Our goal is to show that the choice of numerical approximations for the “gradient term” $(1 + \psi_x^2)^{1/2}$ has some subtlety.

One approach is to divide the interval $[0, 1]$ into $2M - 1$ points, and form the central difference approximation to the spatial derivative ψ_x in Eqn. 16, namely

$$\psi_t \approx \frac{\psi_i^{n+1} - \psi_i^n}{\Delta t} = [1 + [\frac{\psi_{i+1}^n - \psi_{i-1}^n}{2\Delta x}]^2]^{1/2} = [1 + [D_i^{0x}\psi]^2]^{1/2}, \tag{17}$$

where in the last expression we have used standard notation for the central difference.

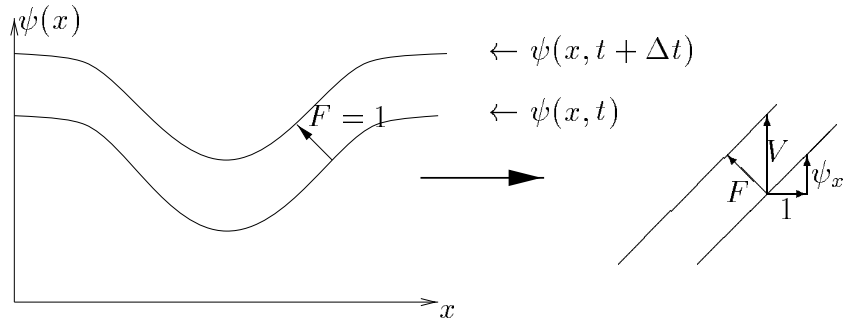


Figure 7: Variables for propagating graph

Since $x_M = 1/2$, by symmetry, $\psi_{M+1} = \psi_{M-1}$, thus the right-hand-side is 1. However, for all $x \neq 1/2$, ψ_t is correctly calculated to be $\sqrt{2}$, since the graph is linear on either side of the corner and thus the central difference approximation is exact. Note that this has nothing to do with the size of the space step Δx or the time step Δt . *No matter how small we take the numerical parameters, as long as we use an odd number of points, the approximation to ψ_t at $x = 1/2$ gets no better.* It is simply due to the way in which the derivative ψ_x is approximated. In Figure 8 we show results using this scheme, with the time derivative ψ_t replaced by a forward difference scheme.

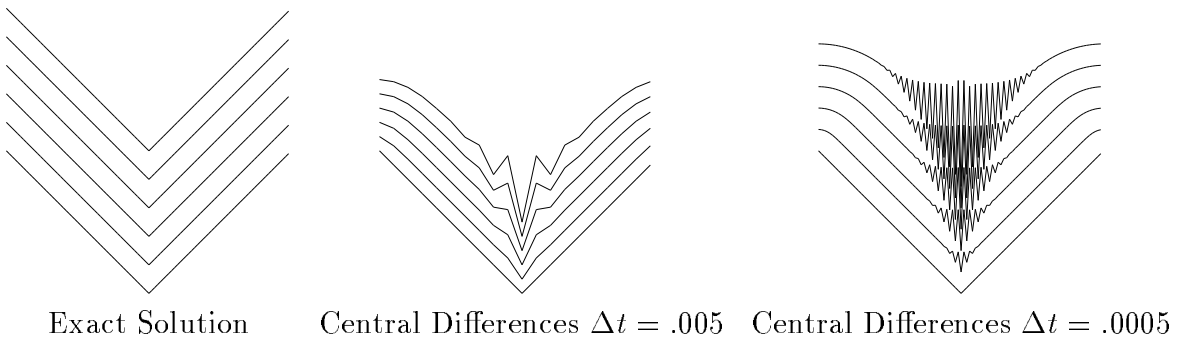


Figure 8: Central difference approximation to Gradient

It is easy to see what has gone wrong. In the exact solution, $\psi_t = \sqrt{2}$

for all $x \neq 1/2$. This should also hold at $x = 1/2$ where the slope is not defined; the Huyghens construction sets $\psi_t(x = 1/2, t)$ equal to $\lim_{x \rightarrow 1/2} \psi_t$. Unfortunately, the central difference approximation chooses a different (and, for our purpose, wrong) limiting solution. It sets the undefined slope ψ_x equal to the average of the left and right slopes. As the calculation progresses, this miscalculation of the slope propagates outwards from the spike as wild oscillations. Eventually, these oscillations cause blowup in the code. For details, see [59].

3.4 Schemes for Viscosity Solutions

Continuing with the example of an evolving interface, we now focus on the gradient term $(1 + \psi_x^2)$. Consider now the following finite difference approximation introduced in [46]

$$\psi_x^2 \approx (\max(D_i^{+x}\psi, 0)^2 + \min(D_i^{-x}\psi, 0)^2) \quad (18)$$

where again we have used standard finite difference notation that

$$D_i^{-x}\psi = \frac{\psi_i - \psi_{i-1}}{h} \quad D_i^{+x}\psi = \frac{\psi_{i+1} - \psi_i}{h} \quad (19)$$

where ψ_i is the value of ψ on a grid at the point ih with grid spacing h .

Eqn. 18 is an “upwind” scheme (see [59]); it chooses grid points in the approximation in terms of the direction of the flow of information. Intuitively, upwind means that if a wave progresses from left to right, then one should use a difference scheme which reaches *upwind* to the left in order to get information to construct the solution *downwind* to the right (see [59]). If we consider our propagating “V” curve from the example above, we see that at the symmetric point, the symmetry of the scheme is changed, and a non-zero value is chosen. In Figure 9, we show what happens if we use the scheme given in Eqn. 18. The exact answer is shown, together with two simulations. The first uses the entropy-satisfying scheme with only 20 points (Figure 9(b)), the second (Figure 9(c)) with 100 points. In the first approximation, the entropy condition is satisfied, but the corner is somewhat smoothed due to the small number of points used. In the more refined calculation, the corner remains sharp, and the exact solution is very closely approximated. Thus we see that this scheme does a correct job of satisfying the entropy-condition.

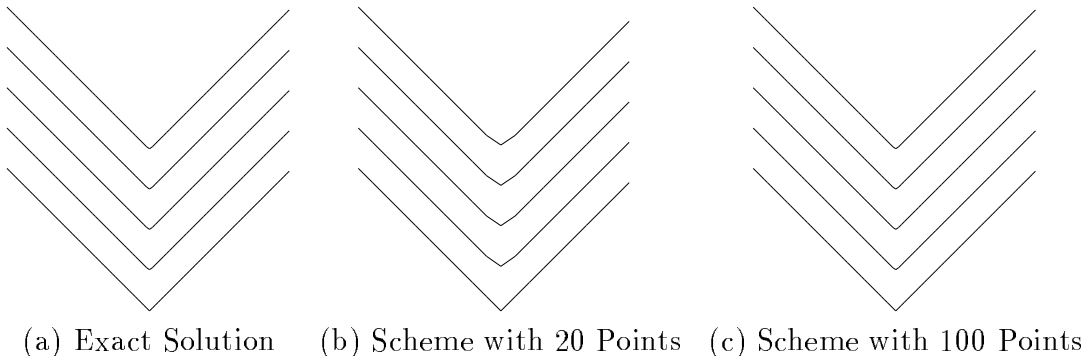


Figure 9: Upwind, entropy-satisfying approximations to the Gradient

While a vast array of other upwind, entropy-satisfying schemes are available to approximate the gradient, for our purposes, the above approximation (and one small variation) will be sufficient; details on other schemes may be found in [56, 59].

3.5 Upwind, Entropy-Satisfying Viscosity Schemes for the Boundary Value and Initial Value Formulations

We can use the above ideas about upwind schemes to construct appropriate schemes for both the boundary value formulation and initial value formulation of the equations of motion for a propagating front.

3.5.1 The Boundary Value Formulation

Recall that we wish to solve

$$|\nabla u(x, y, z)| = f(x, y, z). \quad (20)$$

Extending the above ideas of upwind approximations to the gradient to multi-dimensions, we have the scheme

$$\begin{aligned} |\nabla u| &\approx (\max(D_{ijk}^{-x}u, 0)^2 + \min(D_{ijk}^{+x}T, 0)^2 \\ &\quad + \max(D_{ijk}^{-y}u, 0)^2 + \min(D_{ijk}^{+y}T, 0)^2 \\ &\quad + \max(D_{ijk}^{-z}u, 0)^2 + \min(D_{ijk}^{+z}T, 0)^2)^{1/2} \\ &= f_{ijk}. \end{aligned} \quad (21)$$

The forward and backwards operators D^{-y} , D^{+y} , D^{-z} , and D^{+z} in the other coordinate directions are similar to the one defined earlier for the x direction.

A slightly different upwind scheme, given in [48], which will turn out to be more convenient, is given by

$$\left[\begin{array}{c} \max(D_{ijk}^{-x}u, -D_{ijk}^{+x}u, 0)^2 + \\ \max(D_{ijk}^{-y}u, -D_{ijk}^{+y}u, 0)^2 + \\ \max(D_{ijk}^{-z}u, -D_{ijk}^{+z}u, 0)^2 \end{array} \right]^{1/2} = f_{ijk}, \quad (22)$$

where we use the same forward and backward operators D^- and D^+ and f_{ijk} is the slowness at the gridpoint ijk .

3.5.2 The Initial Value Formulation

An entropy-satisfying viscosity scheme for initial value formulation was introduced in [46], leading to the numerical method known as the level set method, namely

$$\phi_{ijk}^{n+1} = \phi_{ijk}^n - \Delta t [\max(F_{ijk}, 0)\nabla^+ + \min(F_{ijk}, 0)\nabla^-], \quad (23)$$

where

$$\begin{aligned} \nabla^+ = & [\max(D_{ijk}^{-x}, 0)^2 + \min(D_{ijk}^{+x}, 0)^2 + \\ & \max(D_{ijk}^{-y}, 0)^2 + \min(D_{ijk}^{+y}, 0)^2 + \\ & \max(D_{ijk}^{-z}, 0)^2 + \min(D_{ijk}^{+z}, 0)^2]^{1/2} \end{aligned} \quad (24)$$

$$\begin{aligned} \nabla^- = & [\max(D_{ijk}^{+x}, 0)^2 + \min(D_{ijk}^{-x}, 0)^2 + \\ & \max(D_{ijk}^{+y}, 0)^2 + \min(D_{ijk}^{-y}, 0)^2 + \\ & \max(D_{ijk}^{+z}, 0)^2 + \min(D_{ijk}^{-z}, 0)^2]^{1/2}. \end{aligned} \quad (25)$$

Here, we have used a short-hand notation in which $D^{+x}\phi_i^n$ is written as D_i^{+x} , etc. Higher order schemes are available, some based on the ENO formulations introduced in [33]. For details, see [46, 59] and [60].

4 Adaptivity:

The schemes given for both the boundary value and initial value formulations are computationally inefficient. In this section, we make both schemes optimal through the use of variations on adaptivity and causality.

4.1 Schemes and Operation Counts

4.1.1 The Level Set Scheme

Equation 23 is an explicit scheme, and hence can be solved directly. The time step requirement depends on the nature of the speed function F ; for an F that depends only on position, the time step behaves like $\frac{\Delta t}{\Delta x} F \leq 1$. In the case when the speed function F depends on curvature terms (for example, $F = -\kappa$), the equation has a parabolic component, and hence the time step requirement resembles that of a non-linear heat equation; the time step depends roughly on $\frac{\Delta y}{\Delta x^2}$.

This level set formulation reveals two central embeddings.

1. First, in the initialization step, the signed distance function is used to build a function ϕ which corresponds to the interface at the level set $\phi = 0$. This step is known as “initialization”; when performed at some later point in the calculation beyond $t = 0$, it is referred to as “re-initialization”. The need for re-initialization in level set methods was first discussed by Chopp in his work on minimal surfaces, see [16], and has since been employed in detail in many level set simulations.
2. Second, the construction of the initial value PDE given in Eqn. 23 means that the velocity F is now defined for **all** the level sets, not just the zero level set corresponding to the interface itself. We can be more precise by rewriting the level set equation as

$$\phi_t + F_{ext} |\nabla \phi| = 0 \tag{26}$$

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

$$F_{ext} = F \text{ on } \phi = 0 \tag{27}$$

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

The implications of this is that the level set method presented in [46] require that the level set function ϕ^n be updated at every grid point in the computational domain, using a speed function F that has been defined at every point in the computational domain. In some problems, such as curvature flow ($F = -\kappa$), an appropriate definition for the extension velocity is straightforward; the curvature of the level set passing through grid point (i, j) is used as the speed function at that point in the update. However, in many problems (such as those involving fluid interfaces and combustion), the appropriate definition of the extension velocity off of the front is not so straightforward; while considerable flexibility exists, one must nonetheless construct the extension velocity at all grid points.

We can make a rough operation count of the level set method. Assume N grid points in each space dimension of a three-dimensional problem. We consider a simple problem of straightforward propagation of a front with speed $F = 1$; assuming that it takes roughly N time steps for the front to propagate through the domain (here, the CFL condition is taken almost equal to unity), this produces an $O(N^4)$ method.

4.1.2 The Boundary Value Perspective

In contrast, Equation 21 is an implicit equation. One solution, as given in [48], is through iteration. Consider a stencil of a grid point and its six neighbors, as shown in Fig. 10.

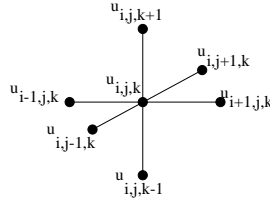


Figure 10: Updating a Grid Point

Observe that the scheme

$$\left[\begin{array}{l} \max(D_{ijk}^{-x}u, -D_{ijk}^{+x}u, 0)^2 + \\ \max(D_{ijk}^{-y}u, -D_{ijk}^{+y}u, 0)^2 + \\ \max(D_{ijk}^{-z}u, -D_{ijk}^{+z}u, 0)^2 \end{array} \right]^{1/2} = f_{ijk},$$

is a quadratic equation for u_{ijk} , assuming that the neighboring grid values for u are given. Thus, one solution comes from updating the value of u at each grid point according to this quadratic until a solution is reached:

```

For iter=1,n;
  For i,j,k=1,Dim
    Solve Quadratic for  $u_{ijk}$ , given
       $u_{i-1,j,k}, u_{i+1,j,k}, u_{i,j-1,k}, u_{i,j+1,k}, u_{i,j,k-1}, u_{i,j,k+1}$       (28)
  EndFor
EndFor

```

An operation count on this approach, assuming N points in each direction, also yields an $O(N^4)$ technique, assuming roughly N steps to converge.

4.2 The Narrow Band Level Set Method

Considerable computational speedup in the level set method comes from the use of the “Narrow Band Level Set Method”, introduced in [1]. It is clear that performing calculations over the entire computational domain is wasteful. Instead, an efficient modification is to perform work only in a neighborhood of the zero level set; this is known as the *Narrow Band Approach*. There are three reasons to choose this method of implementing the initial value level set perspective:

- First, in this case, the operation count in three dimensions drops to $O(kN^2)$, where k is the number of cells in the narrow band, a significant cost reduction.
- Second, construction of extension velocities need only be done to points lying in the narrow band, as opposed to all points in the computational domain. This can greatly reduce the computational labor.
- Third, the full-matrix approach requires a time step that satisfies a CFL condition with regard to the maximum velocity over the entire domain, not simply in response to the speed of the front itself. In a narrow band

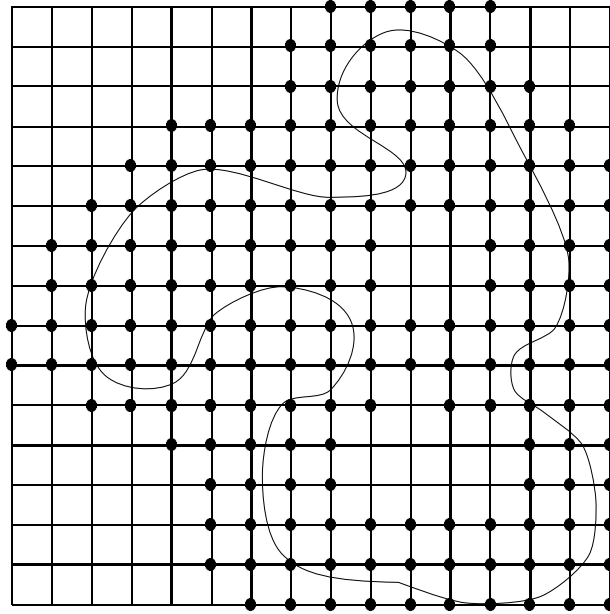


Figure 11: Dark grid points are members of narrow band

implementation, the time step can be adaptively chosen in response to the maximum velocity field only within the narrow band. This is advantageous when the front speed changes substantially as it moves (such as in curvature flow). In such problems, the CFL restriction for the velocity field for *all* the level sets may be much more stringent than the one for those sets within the narrow band.

This “Narrow Band Method” was introduced in Chopp [16], used in recovering shapes from images in Malladi, Sethian and Vemuri [37], and analyzed extensively by Adalsteinsson and Sethian in [1]. The idea is straightforward, and can be best understood by means of two figures.

Figure 11 shows the placement of a narrow band around the familiar initial front. The entire two-dimensional grid of data is stored in a square array. A one-dimensional object is then used to keep track of the points in this array (dark grid points in Figure 11 are located in a narrow band around the front of a user-defined width) (see Figure 12). Only the values of ϕ at such points within the tube are updated. Values of ϕ at grid points on the boundary of the narrow band are frozen. When the front moves near

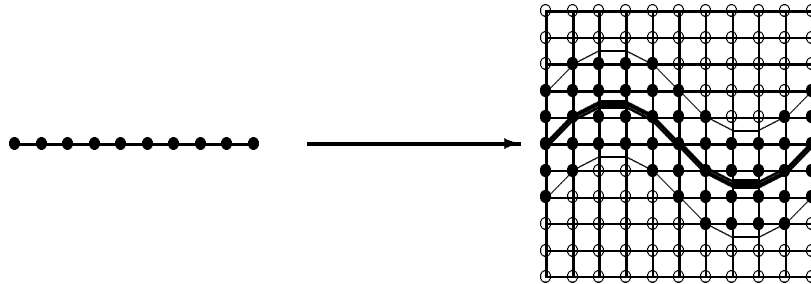


Figure 12: Pointer array tags interior and boundary band points

the edge of the tube boundary, the calculation is stopped, and a new tube is built with the zero level set interface boundary at the center. This rebuilding process is known as “re-initialization”.

Thus, the narrow band method consists of the following loop:

- Tag “Alive” points in narrow band.
- Build “Land Mines” to indicate near edge.
- Initialize “Far Away” points outside (inside) narrow band with large positive (negative) values.
- Solve level set equation until land mine hit.
- Rebuild, loop.

Use of narrow bands leads to level set front advancement algorithms that are computationally equivalent in terms of complexity to traditional marker methods and cell techniques, while maintaining the advantages of topological merger, accuracy, and easy extension to multi-dimensions. Typically, the speed associated with the narrow band method is about ten times faster on a 160×160 grid than the full matrix method. Such a speed-up is substantial; in three-dimensional simulations, it can make the difference between computationally intensive problems and those that can be done with relative ease. Details on the accuracy, typical tube sizes, and number of times a tube must be rebuilt may be found in Adalsteinsson and Sethian [1].

4.3 The Fast Marching Method

The boundary value formulation can also be made fast, in fact, much faster than the Narrow Band Level Set Method, if we focus on the iterative loop given above. The central idea behind the Fast Marching Method, introduced in [55], is to systematically construct the solution in a “downwind” fashion to produce the solution u . The key is the observation that the upwind difference structure of Equation (22) means that information propagates “one way”, that is, from smaller values of u to larger values. Hence, the Fast Marching algorithm rests on “solving” Equation (22) by building the solution outwards from the smallest u value.

In fact, this is behind our quadrature view of upwinding described earlier; we can step the solution outwards from the boundary condition in a downwind direction. The algorithm is again made fast by confining the “building zone” to a narrow band around the front. The idea is to sweep the front ahead in an upwind fashion by considering a set of points in narrow band around the existing front, and to march this narrow band forward, freezing the values of existing points and bringing new ones into the narrow band structure. The key is in the selection of *which* grid point in the narrow band to update, which we now discuss.

Consider a two-dimensional version of the Eikonal equation, in which the boundary value is known at the origin; this is shown schematically in Figure 13. The black sphere at $u_{0,0}$ signifies a grid point where the value of u is known (in this case, the initial value), and the light grey spheres are grid points where the solution value is unknown.

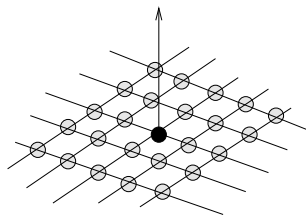


Figure 13: Beginning of Fast Marching Method

We may start the algorithm by marching “downwind” from the known value, computing new values at each of the four neighboring grid points; as shown in Figure 15. This provides possible values for u at each grid point $u_{-1,0}$, $u_{1,0}$, $u_{0,-1}$, $u_{0,1}$, and are shown as dark grey spheres in Figure 15.

Now, we would like to march downwind from these values given at the dark grey spheres, but we don't know which one to choose. The answer lies in the observation that the *smallest u value at these dark grey spheres must be correct*; because of upwinding, no point can be affected by grid points containing larger values of u . Thus, we may freeze the value of u at this smallest dark grey sphere¹, and proceed ahead with the algorithm; this is shown schematically in Figure 15.

This algorithm works because the process of recomputing the T values at upwind neighboring points cannot yield a value smaller than any of the accepted points. Thus, we can march the solution outwards, always selecting the narrow band grid point with minimum trial value for T , and readjusting neighbors, (see Figure 14).

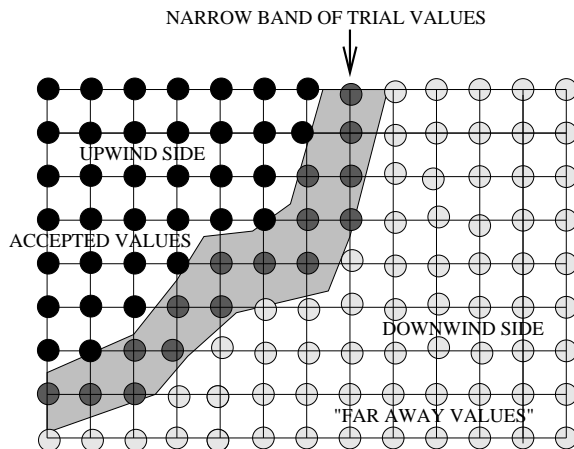
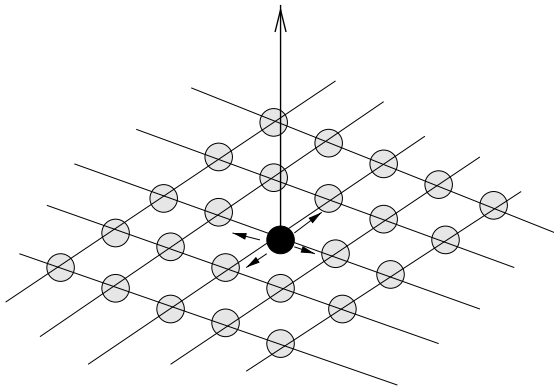
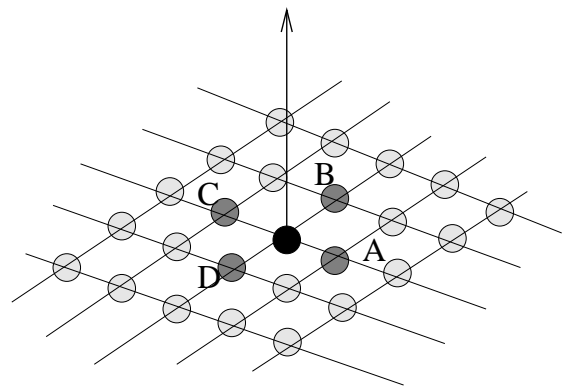


Figure 14: Upwind construction of Accepted Values

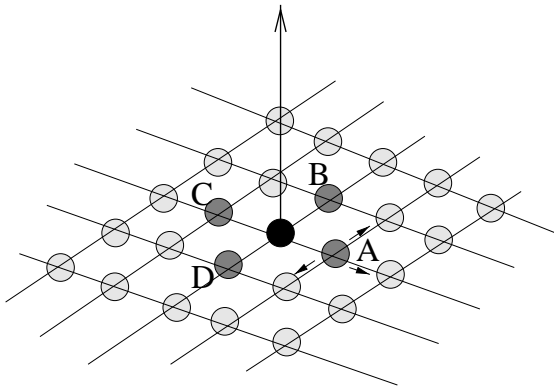
¹that is, turn it into a black sphere and consider its value known.



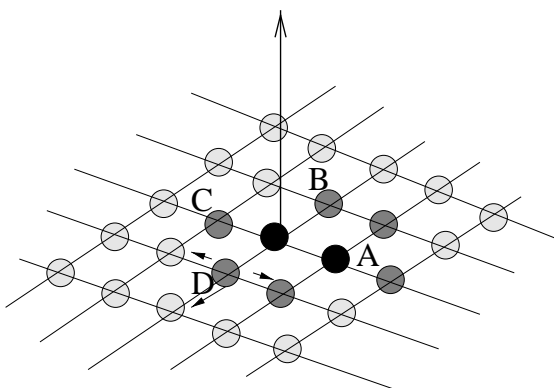
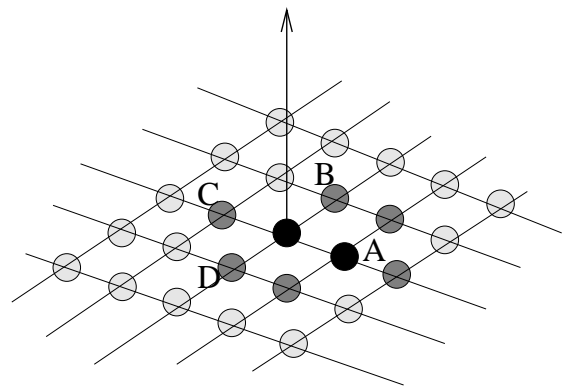
Update “downwind”



Compute New Possible Values



Choose smallest dark gray sphere (for example, “A”) Freeze value at A, update neighboring downwind points



Choose smallest dark gray sphere (for example, “D”) Freeze value at D, update neighboring downwind points

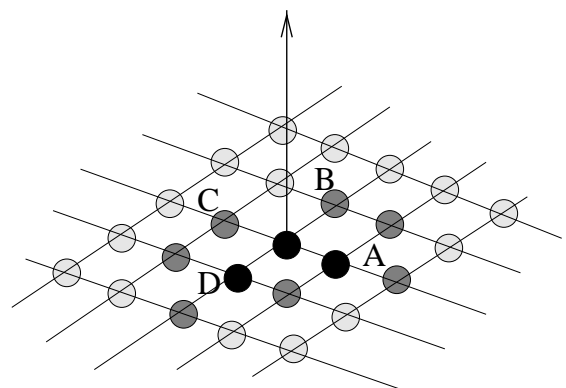


Figure 15: Update Procedure for Fast Marching Method

Another way to look at this is that each minimum trial value begins an application of Huyghen's principle, and the expanding wave front touches and updates all others. For more details, see [56, 55, 59].

Thus, the Fast Marching Method is as follows: First, tag points in the initial conditions as *Alive*. Then tag as *Close* all points one grid point away. Finally, tag as *Far* all other grid points. Then the loop is

1. Begin Loop: Let *Trial* be the point in *Close* with the smallest T value.
2. Add the point *Trial* to *Alive*; remove it from *Close*
3. Tag as *Close* all neighbors of *Trial* that are not *Alive*. If the neighbor is in *Far* remove it from that list and add it to the set *Close*.
4. Recompute the values of T at all neighbors according to Eqn. 22 by solving the quadratic equation.
5. Return to top of Loop;

4.4 Heap Sorts and Computational Efficiency

The key to an efficient version of the above technique lies in a fast way of locating the grid point in the narrow band with the smallest value for u . An efficient scheme to do so is discussed in detail in [59]; here we follow that discussion.²

We use a variation on a heap algorithm (see Sedgewick [49]) with back pointers to store the u values. Specifically, we use a min-heap data structure. In an abstract sense, a min-heap is a “complete binary tree” with a property that the value at any given node is less than or equal to the values at its children. In practice, it is more efficient to represent a heap sequentially as an array by storing a node at location k and its children at locations $2k$ and $2k + 1$. From this definition, the parent of a given node at k is located at $k/2$. Therefore, the root which contains the smallest element is stored at location $k = 1$ in the array. Finding the parent or children of a given element are simple array accesses which take $O(1)$ time.

The values of u are stored, together with the indices which give their location in the grid structure. The marching algorithm works by first looking for the smallest element in the *NarrowBand*; this `FindSmallest` operation involves deleting the root and one sweep of `DownHeap` to ensure that the remaining elements satisfy the heap property. The algorithm proceeds by tagging the neighboring points that are not *Alive*. The *FarAway* neighbors are added to the heap using an `Insert` operation and values at the remaining points are updated using equation (22). `Insert` works by increasing the heap size by one and trickling the new element upward to its correct location using an `UpHeap` operation. Lastly, to ensure that the updated u values do not violate the heap property, we need to perform an `UpHeap` operation starting at that location and proceeding up the tree.

The `DownHeap` and `UpHeap` operations (in the worst case) carry an element all the way from root to bottom or vice versa. Therefore, this takes $O(\log N)$ time assuming there are N elements in the heap. It is important to note that the heap, which is a complete binary tree, is always guaranteed to remain balanced. All that remains is the operation of searching for the *NarrowBand* neighbors of the smallest element in the heap. This can be made $O(1)$ in time by maintaining back pointers from the grid to the heap array. Without

²The work and contributions of Dr. Ravikanth Malladi were invaluable in the optimal programming of the initial version of the Fast Marching Method.

the back pointers, the above search takes $O(N)$ in the worst case.

As an example, Figure 16 shows a typical heap structure and an UpHeap operation after the element at location $(2, 7)$ gets updated from 3.1 to 2.0.

Thus, since the total work in changing the value of one element of the heap and bubbling its value upwards is $O(\log M)$, where M is the size of the heap, this produces a total operation count of $M \log M$ for the Fast Marching Method on a grid of total M points. Thus, if we imagine a three-dimensional grid of N points in each direction, the Fast Marching Method reduces the total operation count from N^4 to $N^3 \log N$; essentially, each grid point is visited once to compute its arrival time value.

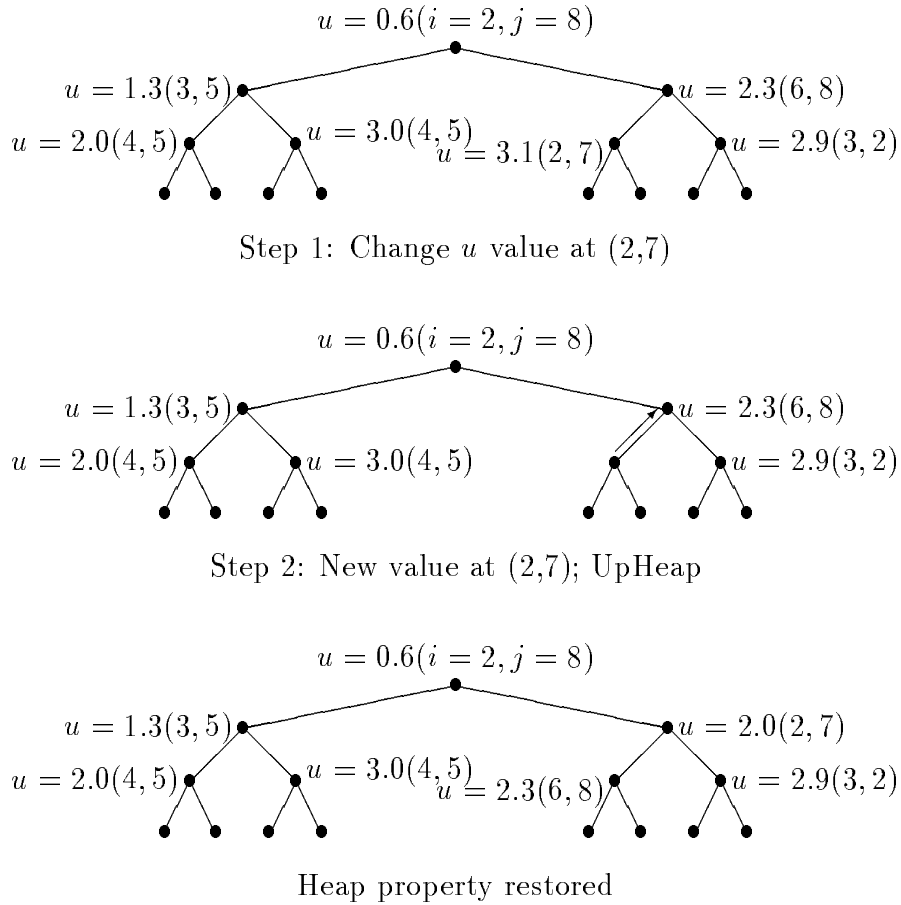


Figure 16: Heap structure and UpHeap+ operation

4.5 Flow Chart of Methods

Schematically, the relationship between the two methods and their adaptive versions is shown in Figure 17.

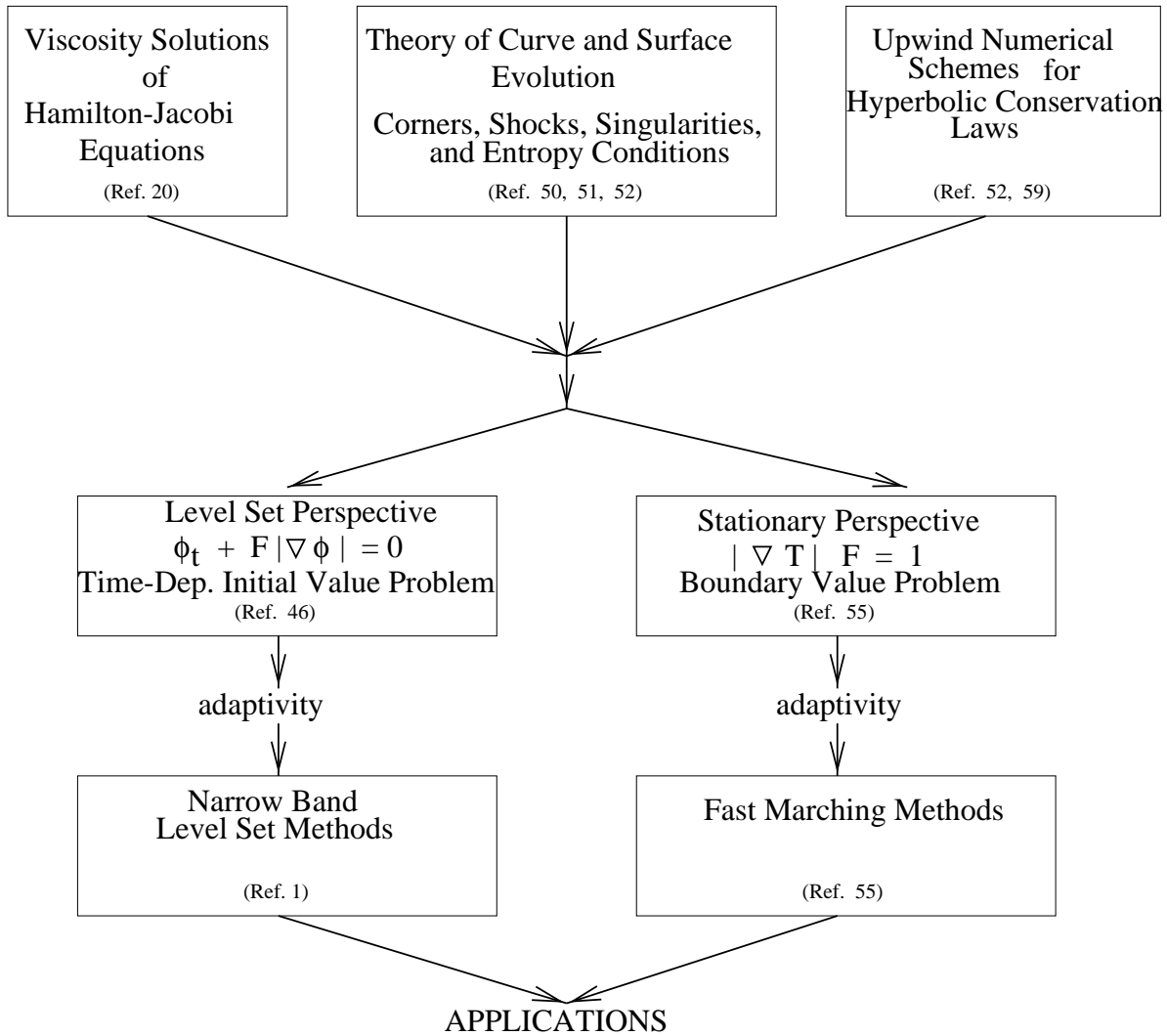


Figure 17: Relation of Fast Marching Methods and Level Set Methods

Applications

5 Building Applications

In this part of the notes, we discuss a collection of applications which involve level set methods and Fast Marching methods. We shall jump between several different applications, in order to convey some feeling for the breadth of applications possible. Complete details on these and a wide variety of other level set and Fast Marching Applications may be found in [59].

5.1 Geometry: Level Set Methods

Probably the most straightforward application level set methods is in the area of pure geometry problems. Here, the goal is to track the evolution of a curve or surface as it propagates under a speed law that depends only on the shape of the front; for example, the local curvature. These problems are particularly straightforward because the velocity field for *each* level set is given by the geometry of that level set.

The first demonstration models some of the remarkable work of Gage and Grayson. First, Gage [27, 28] showed that any convex curve moving under such a motion remains convex and must shrink to a point. Grayson [31] followed this work with the beautiful result that *all* simple closed curves must shrink to a round point, regardless of their initial shape.

In Figure 18, an odd-shaped initial curve is viewed as the zero level set of a function defined in all of R^2 . Here, for illustration, black corresponds to $\phi < 0$, while white corresponds to $\phi > 0$, thus the zero level set is the boundary between the two. As the level curves flow under curvature, the ensuing motion carries each to a circle that then disappears. In the evolution of the front, one clearly sees that the large oscillations disappear quickly, and then as the front becomes circular, motion slows, and the front eventually disappears.

This theorem is not true in three space dimensions, where here the curvature is replaced by the mean curvature; a typical counterexample is the dumbbell described in [32]. Figure 19, taken from Chopp and Sethian [17], shows two connected dumbbells. As the intersection point collapses, the



Figure 18: $F(\kappa) = -\kappa$

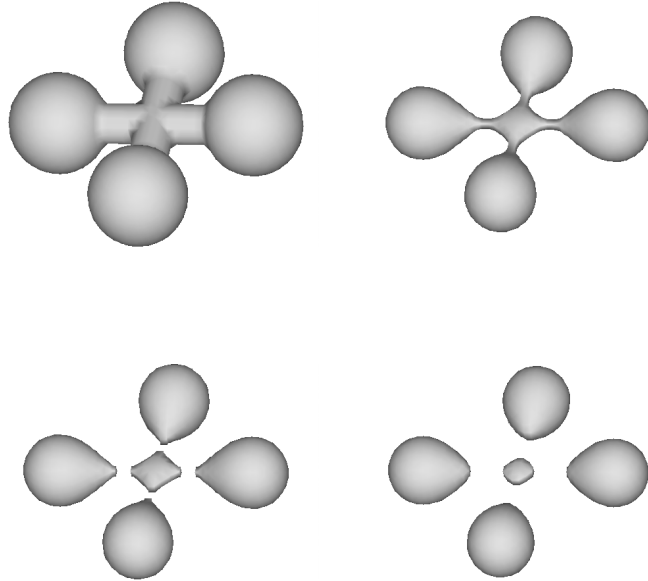


Figure 19: Collapse of two-handled dumbbell

necks break off and leave a remaining “pillow”-like region behind. This pillow region collapses as well, and eventually all five regions disappear.

5.2 Optimal Path Planning: Fast Marching Methods

5.2.1 Statement of Problem

The next application is one of the Fast Marching Method to problems in optimal path planning; these were first developed by Kimmel and Sethian in [35]. Those calculations provided the optimal way of solving path planning problem in problems with relatively low degrees of freedom. Here, we include some of those results.

Given a cost function $F(x_1, x_2, \dots, x_n)$, and a starting point A in R^n , one goal in path planning is to construct the path $\gamma(\tau) : [0, \infty) \rightarrow R^n$ from A to any point B in R^n which minimizes the integral

$$\int_{A=\gamma(0)}^{B=\gamma(L)} F(\gamma(\tau)) d\tau, \quad (29)$$

where τ is the arclength parameterization of γ ; namely $|\gamma_\tau| = 1$, and L is the total length of γ .

More specifically, in two dimension, suppose we are given the cost function $F(x, y)$, and a starting point A . Let $T(x, y)$ be the minimal cost required to travel from A to the point (x, y) , that is

$$T(x, y) = \min_{\gamma} \int_A^{(x,y)} F(\gamma(\tau)) d\tau, \quad (30)$$

with $|\gamma_\tau| = 1$. The level set $T(x, y) = C$ is the set of all points in R^2 that can be reached with minimal cost C , and the minimal cost paths are orthogonal to the level curves, hence we have

$$|\nabla T| = F(x, y). \quad (31)$$

We can use the Fast Marching Method can be used to solve this Eikonal equation to produce $T(x, y)$ in all of R^2 . Then, given a point B in R^2 , explicit construction of the shortest path comes through back propagation from B to A via the solution of the ordinary differential equation

$$X_t = -\nabla T \quad \text{given} \quad X(0) = B, \quad (32)$$

until we reach the starting point A .

5.2.2 Two-Dimensional Results

We begin with a straightforward application of the techniques, namely to two-dimensional path planning with constraints. In Figure 20, we show the optimal path from point A to point B in a dual-valued domain; the speed F on the left is half the value as the speed on the right. In Fig. 20a we show the initial and end points, together with the two domains; in Fig. 20b we show the equal time T solutions obtained using the Fast Marching Method together with the optimal path obtained by solving the O.D.E. The results, as expected, show the effects of Snell's law in the shortest path between two differing media.

In Figure 21 we add constraints to the motion, namely rectangles which are impenetrable. We set the speed function F to zero inside these regions, and the corresponding alteration in the optimal path is shown.

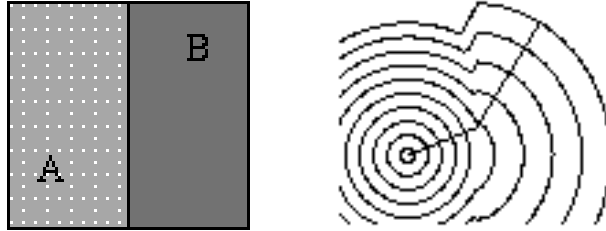


Figure 20: Two-dimensional Navigation on Variable Domain

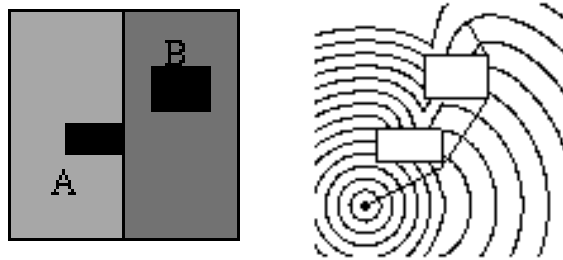


Figure 21: Two-dimensional Navigation with Constraints on Variable Domain

5.2.3 Three Degrees of Freedom

We next turn to two-dimensional problems with three degrees of freedom. Consider now the problem of navigation with constraints and rotation. Instead of a robot as a single point, we imagine a two-dimensional rectangle with a given width and length; thus the initial position A of the robot in configuration space is specified by the position of the center of the rectangle, plus an angle θ between 0 and 2π . The final configuration B is similarly specified, and the goal is to construct the optimal path from A to B .

In the absence of obstacles, a completely straightforward application of the Fast Marching Method is possible: one discretizes the configuration space into a three-dimensional grid, that is, we similarly grid both R^2 and θ between 0 and 2π , employing periodic boundary conditions in θ . Thus, we solve the Eikonal equation

$$\left[u_x^2 + u_y^2 + u_\theta^2 \right]^{1/2} = 1. \quad (33)$$

In the presence of obstacles, we take the following approach; see [36]. Rather

than maneuver an oddly shaped robot, we instead consider the robot as a point, and, for every discretized angle θ_i , alter the shape of the obstacles corresponding to that angle. For example, imagine that one of the obstacles is a square of unit width and height in the domain, suppose the rectangular robot has length .5, and consider the discretization of configuration space at the angle $\theta_i = 0$. Then, in this direction the center of the robot can only come within .25 of the square in the positive or negative x direction; hence we may consider, for this angle, the equivalent problem of a point robot avoiding an obstacle which has height 1 and length 1.5. In Figure 22 we show several examples of a two-dimensional robot with rotational angle navigating in two-dimensional space around obstacles.

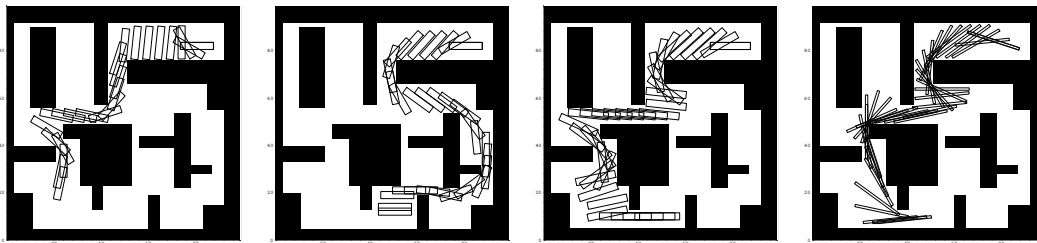


Figure 22: Two-Dimensional Navigation around Obstacles with Rotation

5.2.4 Four Degrees of Freedom

In Figure 23, we consider a motion problem which contains four degrees of freedom. We consider a stick fixed at the lower point, with three hinged segments. The goal is to move the stick from one position (shown in light grey) to the other position. Figure 23 shows the intermediate stages, left to right and top to bottom, as the stick is moved from the initial position to the final position.

Complete details about applications of Fast Marching Methods to path planning and robotic navigation and commentary about additional issues may be found in [35].

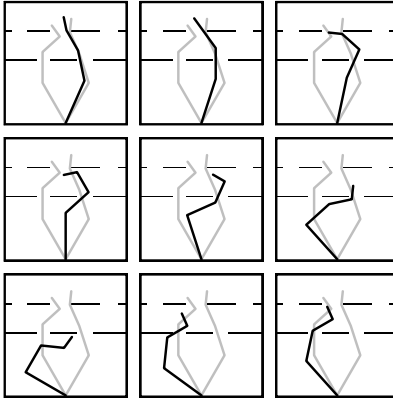


Figure 23: Optimal motion involving four degrees of freedom

5.3 Computing First Arrivals in Seismic Travel Times: Fast Marching Methods

Another application area which shares the Eikonal equation is in three-dimensional prestack migration of surface seismic data. This is a tool for imaging the Earth's subsurface when complex geological structures and velocity fields are present. The most commonly used imaging techniques applied to 3-D prestack surveys are methods based on the Kirchhoff integral, because of its flexibility in imaging irregularly sampled data and its relative computational efficiency. In order to perform this Kirchhoff migration, one approximately solves the wave equation with a boundary integral method. The reflectivity at every point of the Earth's interior is computed by summing the recorded data on multidimensional surfaces; the shape of the summation surfaces and the summation weights are computed from the Green's functions of the single scattering wave-propagation experiment (Schneider, 1978; Berkhout 1982).

In some more detail, the essence of 3-D prestack migration is expressed

by the following integral equation:

$$\text{Image}(\mathbf{x}) = \int \int_{\mathbf{x}_s} \int_{\mathbf{x}_r} G(\mathbf{x}_s, \mathbf{x}, \omega) G(\mathbf{x}, \mathbf{x}_r, \omega) \text{Data}(\mathbf{x}_s, \mathbf{x}_r, \omega) d\mathbf{x}_r d\mathbf{x}_s d\omega, \quad (34)$$

where \mathbf{x} is the image output location, and $\mathbf{x}_s, \mathbf{x}_r$ are the data source and receiver coordinates, and ω is angular frequency. The Green's functions $G(\mathbf{x}_s, \mathbf{x}, \omega)$ and $G(\mathbf{x}, \mathbf{x}_r, \omega)$ parameterize propagation from source to image point and from image point to receiver, respectively. In most implementations, the calculation is often done instead in the time domain, and can be expressed as the summation

$$\text{Image}(\mathbf{x}) = \sum_{\mathbf{x}_s} \sum_{\mathbf{x}_r} A_s A_r \text{Input}(\mathbf{x}_s, \mathbf{x}_r, t_s + t_r), \quad (35)$$

where Input is a filtered version of the input data, and the Green's functions are parameterized by the amplitude terms A_s, A_r and traveltimes t_s, t_r .

For 3-D prestack Kirchhoff depth migration, the Green's functions are represented by five-dimensional tables; these tables are functions of the source/receiver surface locations (x, y) and of the reflector position (x, y, z) in the Earth's interior. This Green's function parameterization is usually based on the assumption of acoustic propagation. This Kirchhoff prestack migration process consists of two stages. First, traveltimes tables are computed and stored. Second, the migrated image is formed by convolving the prestack data with migration operators derived from the traveltimes tables. Both phases present challenges from the perspective of the geophysical accuracy and of the computer implementation.

The key element of 3-D prestack Kirchhoff depth migration is the calculation of traveltimes tables used to parameterize the asymptotic Green's functions. An efficient traveltimes calculation method is required to generate the 5-D traveltimes tables. Traveltimes computation methods have a long history; the past ten years have seen considerable new advancements, particularly those aimed at a finite difference approach, see Vidale ([66]). Prior to this work, traveltimes were typically computed using ray tracing; while these ray tracing methods offer a high degree of accuracy, they also pose interpolation problems in shadow areas and areas where multiple caustics develop. The use of finite difference traveltimes ameliorates these interpolating problems in shadow zones, at the price of issues of stability and the detection of most energetic arrivals versus the first arrivals.

A broad spectrum of traveltimes computation methods were developed in the early 90's, see [67, 65]. At the core, the problem of computing first arrival times requires the solution of the Eikonal equation, with the goal of accurately and robustly dealing with the formation of cusps and corners, topological changes in the solution, and singularities; Fast Marching Methods, both the first and second order versions, provide viable approaches.

As illustration, in Figure 24 we show a horizontal traveltimes slice through a SEG/EAGE Salt Model, taken from [62]. Shown are the equi-arrival lines from a point source computed using the Fast Marching Method on a rectangular mesh.

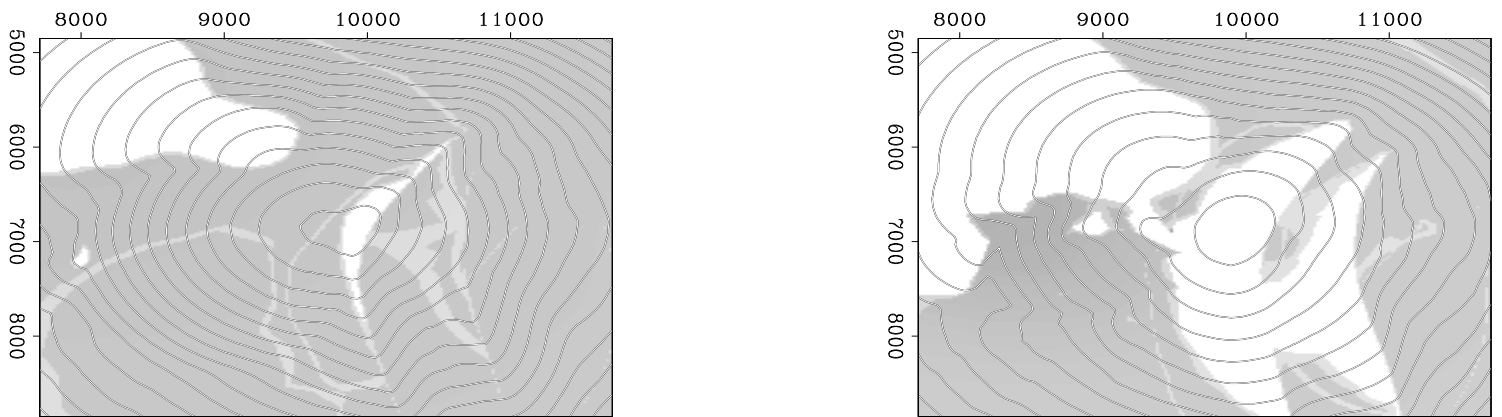


Figure 24: Equi-arrival curves for two-dimensional slices through SEG/EAGE Salt Model.

5.4 Grid Generation: Level Set Methods

We return now to an application using level set methods with a brief section on grid generation. As is well-known, one of the most perplexing problems in computational fluid mechanics is the generation of appropriate grids around complex structures. Non-structured meshes are versatile, but often lack high accuracy; body-fitted cartesian meshes allow high order schemes, but can be

difficult to construct near intricate physical boundaries where gradients are large are require fine, close-fitting meshes.

In [54], narrow band level set methods were used to generate body-fitted cartesian orthogonal grids around bodies. The general idea is to exploit the geometric nature of the problem and view the body itself as the initial position of an interface that must be advanced outwards away from the body. The initial position of the interface and its position at later times forms one set of grid lines; its orthogonal set forms the other. This technique roughly falls into the category of a hyperbolic solver. However, by solving the correct evolution equation for an advancing front, the difficulties of shock formation and colliding characteristics that plague most hyperbolic techniques are avoided. User intervention is kept to a minimum; for the most part, grids are generated automatically without the need to adjust parameters.

In some more detail, first an interface is propagated outwards using the speed function $F(\kappa) = 1 - \epsilon\kappa$; this creates a concentric set of body-fitted coordinate lines. To construct transverse lines, we follow trajectories normal to evolving outward grid lines; this corresponds to following the gradient of the level function ϕ . Place N nodes at points X_i , $I = 1, N$ on the initial body and solve the N ordinary differential equations

$$\frac{dX_i}{dt} = F(\kappa) \frac{\nabla\phi}{|\nabla\phi|}, \quad (36)$$

where $X_i(t)$ is the location of the i th grid point at time t ; a second order (Heun's) method can be used.

Some of the results of this approach are shown in Figure 25 and Figure 26.

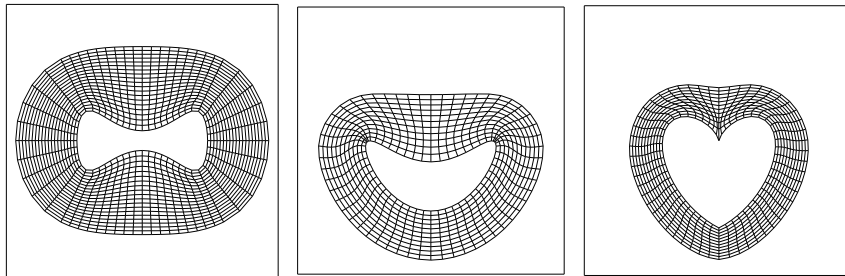


Figure 25: Body-fitted grids generated using level set approach

We note that the last figure in Fig. 26 is a three-dimensional grid; the technique carries over easily to three dimensions.

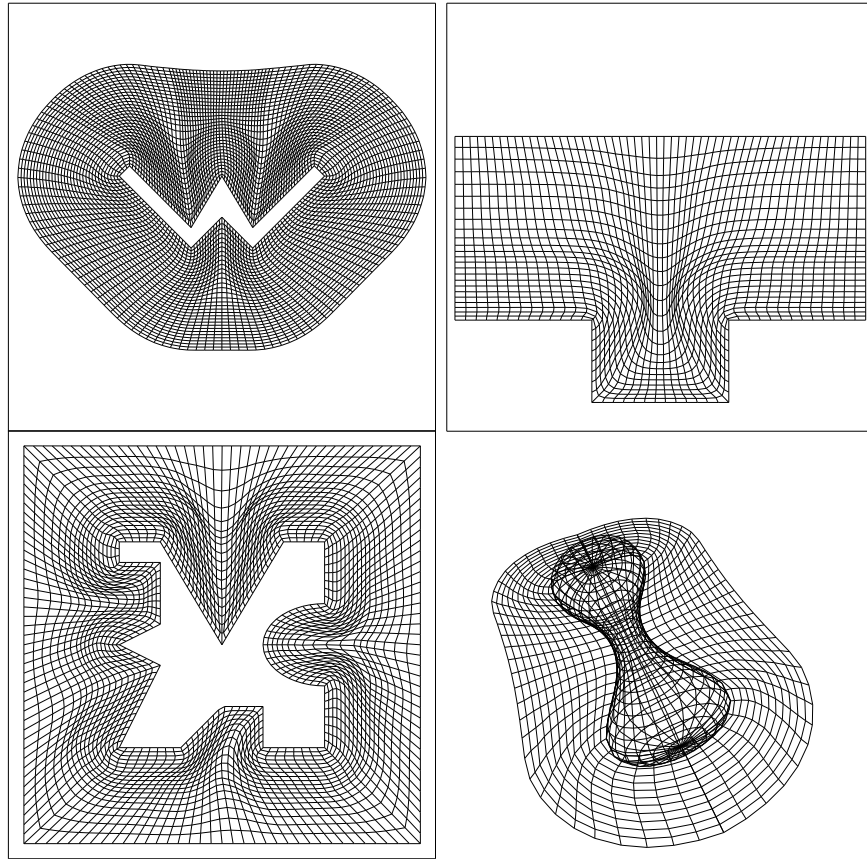


Figure 26: Body-fitted grids generated using level set approach

What is not being proposed here is a general technique for constructing complete grids far from bodies. First, in the case of highly non-convex bodies, the above technique will not work; sides of the body will grow together before the front has “time” to escape; in this case, some sort of domain decomposition is required. Second, the use of more standard techniques for modifying the grid, once the basic design is achieved, might prove fruitful. In particular, the grids generated using this approach may make excellent initial grids for variational and algebraic methods. Finally, extension of this work to multiple bodies requires hybrid techniques in which single grids are patched together or allowed to overlap, similar to those commonly used in other schemes. An attractive approach currently under study is the use of these level set techniques for grids near bodies, coupled with a transition to

cartesian grids away from the body; this may be of particular use in fluid dynamics calculations where body-fitted coordinates are attractive to resolve boundary layer calculations, while far-field cartesian grids are appropriate to connect complex geometries.

5.5 The Construction of Extension Velocity Fields: Two-Phase Flow and Combustion: Level Set Methods and Fast Marching Methods

In a variety of physical problems, such as two phase flow and combustion dynamics, the interface/boundary is physical object, and its motion is determined by solving a set of partial differential equations off of the front. For example, in following an interface which separates two immiscible fluids of differing densities, the interface position determines the density locations, which then controls the solution of the Navier-Stokes equations on either side of the interface. Surface tension along the interface is transformed into a forcing term on the right-hand-side. In combustion problems, the flame acts a source of both exothermicity and vorticity, as well as propagates with a flame speed that depends on the local curvature. We explore both applications in a little more depth.

Much of the technology implementing level set methods in two phase flow was introduced in [64] and [14]. In those papers, surface tension term is taken normal to the fluid interface, and is proportional to the curvature, due to a balance of force argument between the pressure on each side of the interface. This leads to the relation

$$\text{Surface Tension} = \sigma \kappa \delta(d) n, \quad (37)$$

where σ is the coefficient of surface tension, κ is the curvature, n is the normal to the interface, $\delta(d)$ is the Dirac delta function, and d is the distance to the front. The level set function ϕ is used to define the location of the interface, and evaluate the curvature and normal vector. Thus, the effect of surface tension is to act as an additional forcing term in the direction normal to the fluid interface. The forcing term is added to the right-hand-side of the Navier-Stokes equations by smearing the delta function over the grid. The equations are then typically approximated through variants of Chorin's projection scheme, see [19].

Level set methods applied to complex combustion simulations were first performed in [47]; again the location of the front is transferred to an underlying grid through the use of smeared delta functions. In the case, the physics is more intricate, since one must account for both an exothermic velocity, a flame stretch term which depends on a tangential derivative, and a vorticity production term which again is exhibited through a delta function on the burning interface. For details, see [47].

We note that in both cases, once the effect of the interface is transferred to the appropriate differential equations, one still must update the front itself. It is tempting to use the fluid velocity itself to act as the velocity for any given level set. However, as discussed in considerable detail in [8], such a choice of extension velocity F_{ext} may not be wise. This is because in some simulations, such as two phase flow and combustion, the fluid undergoes a jump in the normal velocity across the interface, due to the presence of the delta function source terms. The jump in this velocity across the interface drives a discontinuity in the gradient of the level set function, leading to bunching of level set functions, which then forces constant re-initialization. This can lead to a loss of accuracy during this re-positioning step.

The preferred approach instead, as discussed in [8], is to build the extension velocity as follows. First, the velocity of the front itself F is determined through the appropriate physics. Then, using the Fast Marching Method, we construct an extension velocity F_{Ext} satisfying

$$\nabla F_{ext} \cdot \nabla \phi = 0. \tag{38}$$

with the boundary condition that $F_{ext} = F$ on the zero level set. This produces an extension velocity which equals the correct one on the zero level set corresponding to the front, and smoothly varies away from it. Equation 38 can be solved extremely rapidly using the Fast Marching Method; as the signed distance function ϕ is constructed, the extension field is simultaneously built using the same causality rules. This provides the optimal way to construct extension velocities in delicate problems such as those in computational fluid mechanics and combustion simulations. For details, see [8].

5.6 Etching and Deposition in Semi-conductor Manufacturing: Fast Marching Methods and Level Set Methods

We end with a special section on etching and deposition in semi-conductor manufacturing. This is one of the most detailed and complex set of simulations performed using using level set and Fast Marching Methods; their intricacy lies in the need to blend the two methods, construct extension velocities, and to assemble a speed law that depends on a large collection of terms. We shall discuss this application in some detail.

The goal of numerical simulations in microfabrication is to model the process by which silicon devices are manufactured. Here, we briefly summarize some of the stages involved. First, a single crystal ingot of silicon is extracted from molten pure silicon. This silicon ingot is then sliced into several hundred thin wafers, each of which is then polished to a smooth finish. A thin layer of crystalline is then oxidized, a light-sensitive “photoresist” that is sensitive to light is applied, and then the wafer is covered with a pattern mask that shields part of the photoresist. This pattern mask contains the layout of the circuit itself. Under exposure to a light or an electron beam, the exposed photoresist polymerizes and hardens, leaving an unexposed material that is then etched away in a dry etch process, revealing a bare silicon dioxide layer. Ionized impurity atoms such as boron, phosphorus, and argon are then implanted into the pattern of the exposed silicon wafer, and silicon dioxide is deposited at reduced pressure in a plasma discharge from gas mixtures at a low temperature. Finally, thin films such as aluminum are deposited by processes such as plasma sputtering, and contacts to the electrical components and component interconnections are established. The result is a device that carries the desired electrical properties.

The above processes produce considerable changes in the surface profile as it undergoes various effects of etching and deposition. This problem is known as the “surface topography problem” in microfabrication, and is controlled by a large collection by physical effects, including the visibility of the etching/deposition source at each point of the evolving profile, surface diffusion along the front, non-convex sputter laws that produce faceting, shocks and rarefactions, material-dependent discontinuous etch rates, and masking profiles.

The underlying physics and chemistry that contributes to the motion of

the interface profile is very much an area of active research. Nonetheless, once empirical models are formulated, the problem ultimately becomes the familiar one of tracking an interface moving under a speed function F . Simulations in this chapter are taken from Adalsteinsson and Sethian [2, 3, 4]; complete details may be found therein.

Several factors contribute to the ultimate shape of the profile. These include

- Deposition: Particles are deposited on the surface, which causes build-up in the profile.
- Etching: Particles are etched away from the surface, causing subtraction of material.
- Re-deposition: particles that are expelled during the etching process land elsewhere. The fraction of particles expelled depends on a “sticking coefficient” β .
- Re-emission: particle that are deposited during the deposition process may in fact not stick, and be sent off to land elsewhere. The fraction of particles re-deposited depends on a “sticking coefficient” β .
- Visibility Effects: Whether or not a point on the surface is visible from another point on the surface controls whether or not particles can be travel between the two points; this factor is known as the visibility effect, and can be quite time consuming to evaluate.
- Surface Diffusion: The surface profile is altered by surface diffusion along the front; this corresponds to motion by the second derivative of curvature, which is itself a highly delicate calculation.

These terms may be put together in the following form:

$$F = \left\{ \begin{array}{l} F_{Isotropic}^{Etching} + F_{Direct}^{Etching} + F_{Isotropic}^{Deposition} + \\ F_{Direct}^{Deposition} + F_{Re-Deposition}^{Deposition} + F_{Re-Emission}^{Deposition} + F_{surface_diffusion} \end{array} \right\} \quad (39)$$

The two isotropic terms are evaluated at a point x by simply evaluating the strengths at that point. The two direct terms are evaluated at a point

x on the profile by first computing the visibility to each point of the source, and then evaluating the flux function; thus these terms require computing an integral over the entire source. To compute the fifth term at a point x , we must consider the contributions of every point on the profile to check for re-deposition particles arising from the etching process, thus this term requires computing an integral over the profile itself. The sixth term, $F_{Re-Emission}$ is more problematic; since every point on the front can act as a deposition source of re-emitted particles that do not stick, the total flux function deposition function comes from evaluating an integral equation along the entire profile. The seventh term is a pure geometry term.

In more detail, let Ω be the set of points on the evolving profile at time t , and let $Source$ be the external source. Let r, ψ, γ represent spherical coordinates, and let θ be the angle between the normal and the source ray. Given two points x and x' , let $\Upsilon(x, x')$ be one if the points are visible from one another and zero otherwise. Let r be the distance from x to x' , \vec{n} be the unit normal vector at the point x , and finally, let $\vec{\alpha}$ be the unit vector at the point x' on the source pointing towards the point x on the profile. Then we may refine the above terms as:

$$F = \left[\begin{array}{c} \text{Flux}_{Isotropic}^{Etching} \\ + \\ \int_{Source} \text{Flux}_{Direct}^{Etching}(r, \psi, \gamma, \theta, x) \Upsilon(x, x') (\vec{n} \cdot \vec{\alpha}) dx' \\ + \\ \text{Flux}_{Isotropic}^{Deposition} \\ + \\ \int_{Source} \text{Flux}_{Direct}^{Deposition}(r, \psi, \gamma, \theta, x) \Upsilon(x, x') (\vec{n} \cdot \vec{\alpha}) dx' \\ + \\ \int_{\Omega} (1 - \beta_{re-deposition}) \text{Flux}_{Re-Deposition}^{Deposition}(r, \psi, \gamma, \theta, x) \Upsilon(x, x') (\vec{n} \cdot \vec{\alpha}) dx' \\ + \\ \int_{\Omega} (1 - \beta_{re-emission}) \text{Flux}_{Re-emission}^{Deposition}(r, \psi, \gamma, \theta, x) \Upsilon(x, x') (\vec{n} \cdot \vec{\alpha}) dx' \\ + \\ F_{surface_diffusion} \end{array} \right] \quad (40)$$

The integrals are performed in a straightforward manner. The front is located by constructing the zero level set of ϕ ; in two dimensions it is represented by a collection of line segments; in three-dimensions by a collection of

voxel elements, see [2, 3]. The centroid of each element is taken as the control point, and the individual flux terms are evaluated at each control point. In the case of the two isotropic terms, the flux is immediately found. In the case of the two integrals over sources, the source is suitably discretized and the contributions summed. In the fifth term, corresponding to re-deposition, the integral over the entire profile is calculated by computing the visibility to all other control points and the corresponding re-deposition term is produced by the effect of direct deposition. Thus, as presented, the fifth term requires N^2 evaluations, where N is the number of control points which approximate the front.

There are four subtleties in solving for the speed function F ; these involve the re-emission term, computing the visibility, extending the speed function to the other level sets in the narrow band, and evaluating the surface diffusion term. We address each in turn:

5.6.1 Re-emission

The re-emission term requires evaluation of the flux contribution $Flux_{Direct/Re-Deposition}$ from each point of the interface, each of which depends on the contribution from all other points. Thus, this is an integral equation which must be solved to produce the total deposition flux at any point.

In [4], an iterative relation was built to solve this integral equation; here, we follow that discussion. Imagine a particle at node i that lands on a part of the profile, and either sticks or bounces with a given sticking probability. Define the reflected intensity $I_{R,k}$ after the k -th bounce, namely

$$I_{R,1}^i = (1 - \beta_0)I_S^i, \quad (41)$$

Then,

$$I_{R,k+1}^i = (1 - \beta) \sum_{j, j \neq i} I_{R,k}^j \frac{2 \cos(\theta_j^i) \cos(\theta_j^j)}{\pi r_{ij}} \Upsilon_{ij} l_j. \quad (42)$$

In a matrix form, this becomes

$$I_{R,0} = (1 - \beta_0)I_S, \quad (43)$$

$$I_{R,k+1} = (1 - \beta)\Omega I_{R,k}, \quad (44)$$

where Ω is the matrix given by

$$\Omega_{ij} = 2 \frac{\cos(\theta_j^i) \cos(\theta_i^j)}{\pi r_{ij}} \Upsilon_{ij} l_j, \quad (45)$$

when $i \neq j$, and 0 if $i = j$. Define $I_{S,k}$ to be the portion that sticks at the k -th bounce. We then have that

$$I_{S,0} = \beta_0 I_S, \quad (46)$$

$$I_{S,1} = \beta \Omega (1 - \beta_0) I_S, \quad (47)$$

and, in general,

$$I_{S,k+1} = \frac{\beta}{1 - \beta} I_{R,k+1} = (1 - \beta) \Omega I_{S,k}. \quad (48)$$

Therefore, we have

$$I_{S,k} = \beta (1 - \beta)^{k-1} (1 - \beta_0) \Omega^k I_S, \quad (49)$$

and thus the total intensity after N applications is given by

$$I_N = \beta (1 - \beta_0) \left[\sum_{k=1}^N (1 - \beta)^{k-1} \Omega^k \right] I_S + \beta_0 I_S. \quad (50)$$

Each application of the operator may be viewed as either an additional term in the expansion or an additional included bounce. The situation is simplified by the existence of a recurrence relation, namely

$$\begin{aligned} I_{N+1} &= \beta (1 - \beta_0) \left[\sum_{k=1}^{N+1} (1 - \beta)^{k-1} \Omega^k \right] I_S + \beta_0 I_S \\ &= \beta (1 - \beta_0) \left[(1 - \beta) \Omega \sum_{k=1}^N (1 - \beta)^{k-1} \Omega^k + \Omega \right] I_S + \beta_0 I_S \\ &= (1 - \beta) \Omega (I_N - \beta_0 I_S) + \beta (1 - \beta_0) \Omega I_S + \beta_0 I_S \\ &= (1 - \beta) \Omega I_N + (\beta - \beta_0) \Omega I_S + \beta_0 I_S \end{aligned}$$

This allows an extremely fast summation expression for the total received flux, consisting only of matrix/vector multiplies.

5.6.2 Visibility

In order to perform this integral, we must evaluate the visibility between any two points. This may be done using the level set function ϕ in a straightforward manner, since it provides a clear test of the sign; if ϕ is always positive between two points on the surface, then the two points are mutually visible.

5.6.3 Extension of the Speed Function

Both the total flux and the visibility are quantities that only have meaning on the front itself. In this case, the front extension methodology described earlier is required in order to build values for the speed function throughout the narrow band.

5.6.4 Surface diffusion

Here, we need to add the additional term of the form

$$F = 1 + \epsilon \kappa_{\alpha\alpha}, \quad (51)$$

where α is an arc-length parameterization. This term models the effects of surface diffusion, which relates to the motion of metal boundaries; see [13, 34] for experimental evidence.

The problem is delicate because Eqn. 51 is a time-dependent fourth order partial differential equation, and the presence of the fourth derivative requires an exceedingly small time step for stability in an explicit scheme; the linear fourth order heat equation has a stability time step requirement of the form $O(\Delta t / \Delta h^4)$. Such schemes can in fact be made implicit to allow a larger time step; see [18] for further discussion.

Following the discussion in [18], we have

$$\phi_t + (1 + \epsilon \kappa_{\alpha\alpha}) |\nabla \phi| = 0 \quad (52)$$

$$\phi_t + |\nabla \phi| = -\epsilon \left[\nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} \right]_{\alpha\alpha} \quad (53)$$

$$\phi_t + |\nabla \phi| = -\epsilon \nabla \left[\nabla \left[\nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} \right] \cdot \frac{\phi_y, -\phi_x}{|\nabla \phi|} \right] \cdot \frac{\phi_y, -\phi_x}{|\nabla \phi|} \quad (54)$$

We use a central difference approximation for the derivatives, and regularize the denominator to avoid the singularity associated with vanishing gradients

5.6.5 Results

We begin with a three-dimensional example in Figure 27, using the initial shape shown in Fig. 27a. It was shown in [2], that certain forms for deposition/etching profiles lead to non-convex flux laws in the resulting Hamilton-

Jacobi equation. By this, we mean that the normal speed of the profile depends on the angle of incidence between the surface normal and the incoming beam, with the maximum etching/deposition strength occurring away from the normal direction. These non-convex flux laws must be approximated through different schemes than the ones presented above; for such schemes applied to semi-conductor manufacturing, see [2, 3, 4, 59]. We consider a combination of two cosine flux deposition sources. That is, the initial flux at each point is given by

$$Flux(x) = \cos^5(\theta_1) \cos(\theta_2) + \cos(\theta_1) \cos(\theta_2); \quad (55)$$

in addition, the second deposition term is given a sticking coefficient of .1, thus we also consider the effects of re-deposition. We see that in this case, the void does not form; results are shown after some time evolution in Fig. 27b; a two-dimensional cross-sectional cut is shown in Figure 27c.

Next, we show results in which several effects are combined. We imagine an initial block, in which a mask covers a substrate, and envision a simultaneous etch and deposition process. We imagine that one material (which will be shown as light gray) is isotropically deposited on both the mask (shown in dark gray) and substrate (shown in black). At the same time that this material is being deposited, it is being etched under an ion-milling/sputter law, such that the etch rate in the substrate is twice as fast as the etch rate in the mask. Thus, we have

$$F = F_{IsotropicDeposition} + F_{SputterEtching} \quad (56)$$

$$F_{IsotropicDeposition} = .5 \quad F_{SputterEtching} = Factor(x) * (1. + 4 \sin^2 \theta) \cos \theta \quad (57)$$

- $Factor(x) = 1.0$ if in dark gray material (Mask),
- $Factor(x) = 2.0$ if in black material (Substrate).

In Figure 28, we show the sequence of profile evolution under these effects. We note the development of the thin nano-layer which covers the side walls, but is fully etched away along the top and the bottom; we also note the existence of evolving points where several fronts touch. The presence of an ion-milling sputter etching speed law promotes faceting due to the presence of non-convex Hamiltonians; see [2, 3]. Here, we note the rounding of the

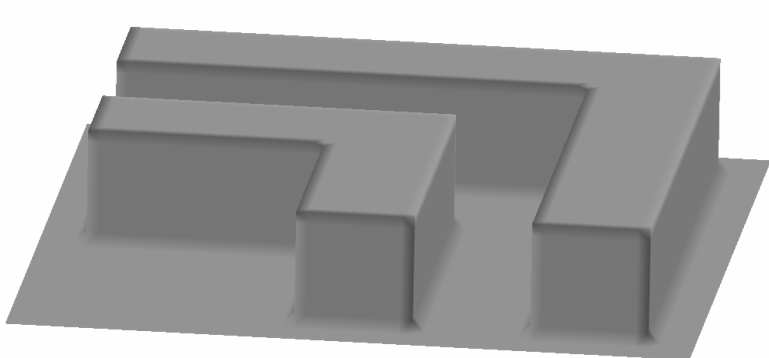


Fig. 27a: Initial Position

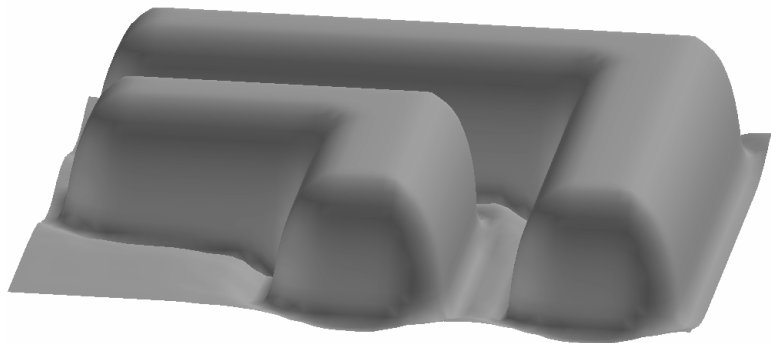


Fig. 27b: Time Evolution

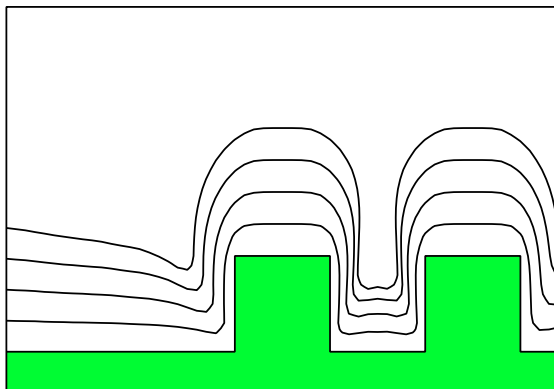


Fig. 27c: 2d Cross-Section

Figure 27: Three-dimensional Evolution under Cosine Source Distribution with Sticking Coefficient .1

side wall layers, as well as the existence of multiple fronts and thin layer structures.

We stress that the grid used for this calculation is significantly larger than the size of the nano-layer; thus our algorithms provide for significant sub-grid resolution without resorting to adaptive mesh technology, see [59] for details of this technology.

Finally, we show some results of a surface diffusion calculation. We consider a speed function which contains isotropic deposition and an ion-milling non-convex etch function together with surface diffusion, that is, in Figure 29 we consider the speed function

$$F = 4 - (1 + 4 \sin^2 \theta) + \epsilon \kappa_{\alpha\alpha} \quad (60)$$

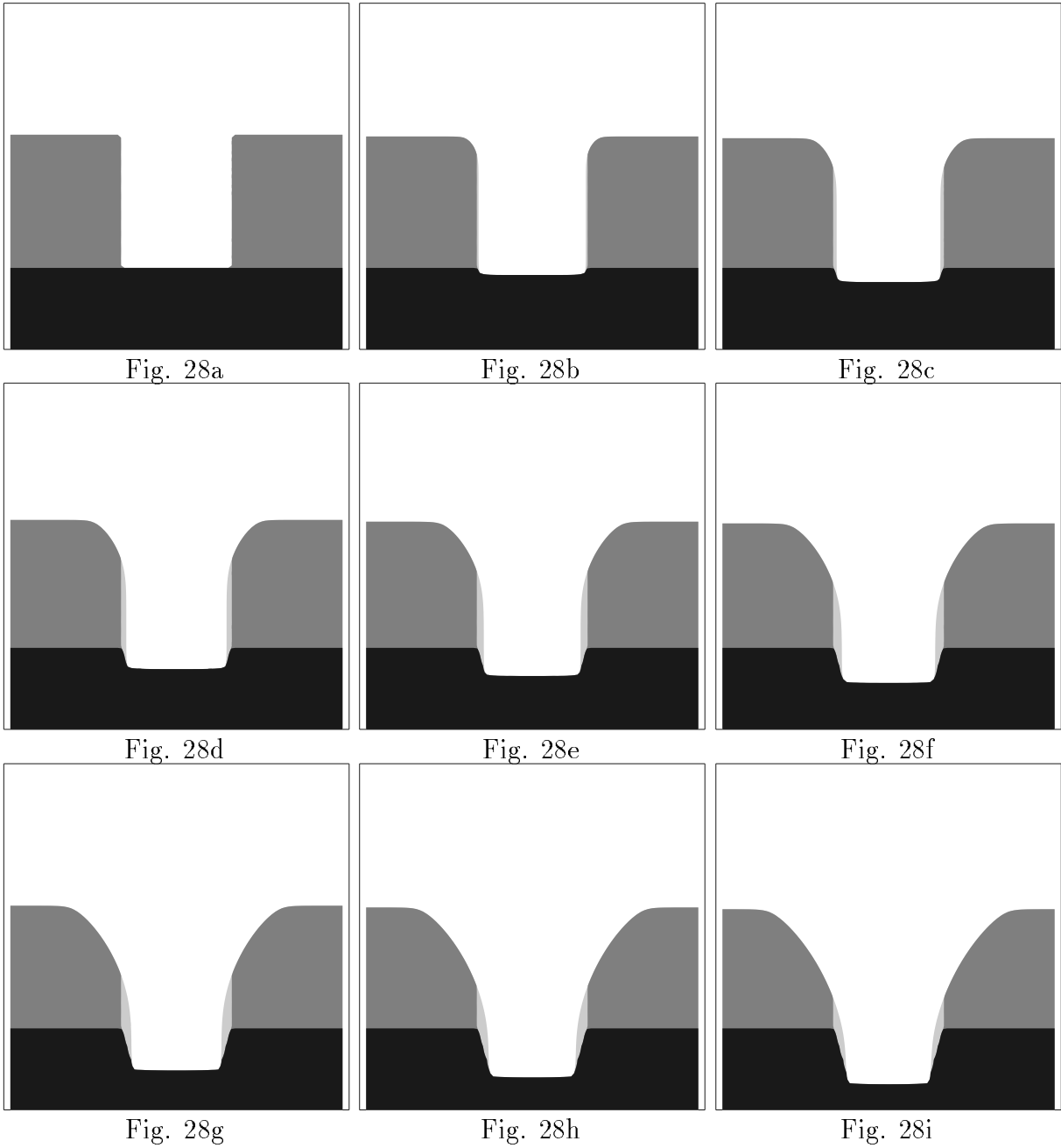
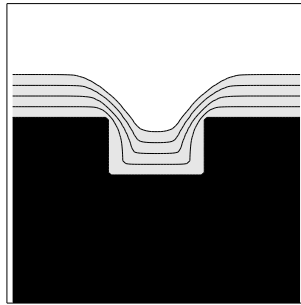


Figure 28: Combination of Isotropic Deposition of Thin Layer and Non-Convex Sputter Etching of Materials: Time Sequence

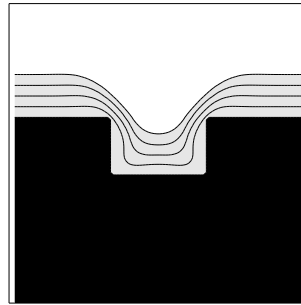
$$F = F_{IsotropicDeposition} + F_{SputterEtching} \quad (58)$$

$$F_{IsotropicDeposition} = .5 \quad F_{SputterEtching} = Factor(x) * (1. + 4sin^2\theta) \cos \theta \quad (59)$$

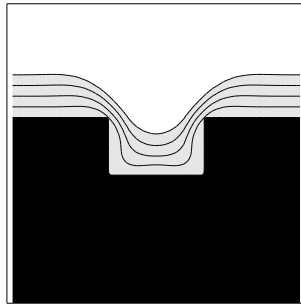
$Factor(x) = 1.0$ if in Mask (dark gray) / $Factor(x) = 2.0$ if in Substrate(black)



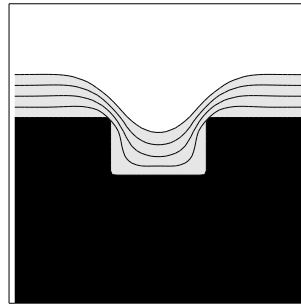
$$F = 4 - (1 + 4 \sin^2 \theta)$$



$$F = 4 - (1 + 4 \sin^2 \theta) + .0005\kappa_{\alpha\alpha}$$



$$F = 4 - (1 + 4 \sin^2 \theta) + .001\kappa_{\alpha\alpha}$$



$$F = 4 - (1 + 4 \sin^2 \theta) + .002\kappa_{\alpha\alpha}$$

Figure 29: Effects of Surface Diffusion on Deposition Plus Ion-Milling: $F = 4 - (1 + 4 \sin^2 \theta) + \epsilon\kappa_{\alpha\alpha}$

For a large number of additional results, as well as detailed explanation of the application of level set methodology to etching and deposition, see [2, 3, 4].

6 Additional Comments

The range of Fast Marching and level set techniques extends far beyond the work covered here. Here, we point the reader to some additional topics.

On the theoretical side, considerable analysis has been performed in recent years; see, for example, Brakke [12], Ecker and Huisken [22], Evans and Spruck [23, 24, 25, 26], Chen, Giga, Goto, and Ishii [15, 29, 30], and Ambrosio and Soner [9]. These works have concentrated on many aspects, including questions of existence and uniqueness, pathological cases, extensions of these ideas to fronts of co-dimension greater than one (such as evolving curves in

three dimensions), coupling with diffusion equations, links between the level set technique, and Brakke's groundbreaking original varifold approach.

A large number of phenomena and simulations have been performed with level set and Fast Marching Methods at this point; here, we have reviewed only a subset of the literature. We refer the interested reader to the fluid bubble calculations in [14, 64], work on a variational level set method for multiple interface in [68] and triple points [11], combustion and flame propagation [47, 70, 69], crystal growth and dendritic solidification [63], applications to photolithography development [57, 56], application to medical imaging and image segmentation [37, 38, 39, 42, 44, 45], general algorithms for interfaces in computational fluid dynamics and materials sciences [58], image denoising [40, 41, 43], semi-conductor manufacturing [2, 3, 4, 5, 61, 6, 7], and level set methods on triangulated domains [10].

We again refer the interested reader to a resource book on level set and Fast Marching Methods [59] and the additional web site resource

http://math.berkeley.edu/~sethian/level_set.html

All calculations were performed at the University of California, Berkeley, and the Lawrence Berkeley National Laboratory. We would like to thank D. Adalsteinsson, T. Barth, R. Kimmel, R. Malladi for many useful discussions.

References

- [1] Adalsteinsson, D., and Sethian, J.A., *A Fast Level Set Method for Propagating Interfaces*, J. Comp. Phys., 118, 2, pp. 269–277, 1995.
- [2] Adalsteinsson, D., and Sethian, J.A., *A Unified Level Set Approach to Etching, Deposition and Lithography I: Algorithms and Two-dimensional Simulations*, 120, 1, pp. 128–144, 1995.

- [3] Adalsteinsson, D., and Sethian, J.A., *A Unified Level Set Approach to Etching, Deposition and Lithography II: Three-dimensional Simulations*, 122, 2, pp. 348–366, 1995.
- [4] Adalsteinsson, D., and Sethian, J.A., *A Unified Level Set Approach to Etching, Deposition and Lithography III: Complex Simulations and Multiple Effects*, J. Comput. Phys., 138, 1, pp. 193-223, 1997.
- [5] Adalsteinsson, D., Sethian, J.A., Rey, J., *High Density Plasma Deposition Modeling Using Level Set Methods*, Proceedings Second International Dielectrics for VLSI/ULCI Multilevel Interconnection Conference, Santa Clara, California, pp. 116-123, Feb. 1996.
- [6] Adalsteinsson, D., Sethian, J.A., Rey, J., *Etching and Deposition Modeling Using Level Set Methods*, 13th International VMIC Meeting, Santa Clara, California, June 18th-20th, 1996.
- [7] Adalsteinsson, D., Sethian, J.A., Rey, J., *Void Development in Plasma Enhanced CVD Models*, in press, Proceedings Third International Dielectrics for VLSI/ULCI Multilevel Interconnection Conference, Santa Clara, California, Feb., 1997.
- [8] Adalsteinsson, D., Sethian, J.A., *The Fast Construction of Extension Velocities in Level Set Methods*, submitted for publication, J. Comp. Phys., 1997.
- [9] Ambrosio, L., and Soner, H.M., *Level Set Approach to Mean Curvature Flow in Arbitrary Co-dimension*, in press, J. Diff. Geom, 1995.
- [10] Barth, T.J., and Sethian, J.A., *Numerical Schemes for the Hamilton-Jacobi and Level Set Equations on Triangulated Domains*, submitted for publication, J. Comp. Phys., Sept. 1997
- [11] Bence, J., Merriman, B., and Osher, S.J., *Motion of Multiple Junctions: A Level Set Approach*, Jour. Comp. Phys., 112, 2, pp. 334–363, 1994.
- [12] Brakke, K.A., *The Motion of a Surface by Its Mean Curvature*, Princeton University Press, Princeton, NJ, 1978.
- [13] Cale, T.S., Jain, M.K., Tracy, C.J., and Duffin, R., submitted for publication, J. Vac. Sci, Tech, B, 1996,

- [14] Chang, Y.C., Hou, T.Y., Merriman, B., and Osher, S.J., *A Level Set Formulation of Eulerian Interface Capturing Methods for Incompressible Fluid Flows*, Jour. Comp. Phys., 124, pp. 449-464, 1996.
- [15] Chen, Y., Giga, Y., and Goto, S., *Uniqueness and Existence of Viscosity Solutions of Generalized Mean Curvature Flow Equations*, J. Diff. Geom, 33, 749, 1991.
- [16] Chopp, D.L., *Computing Minimal Surfaces via Level Set Curvature Flow*, Jour. of Comp. Phys., 106, pp. 77-91, 1993.
- [17] Chopp, D.L., and Sethian, J.A., *Flow Under Curvature: Singularity Formation, Minimal Surfaces, and Geodesics*, Jour. Exper. Math., 2, 4, pp. 235-255, 1993.
- [18] Chopp, D.L., and Sethian, J.A., *A Level Set Approach to the Numerical Simulation of Viscous Sintering*, work in progress, 1998.
- [19] Chorin, A.J., *Numerical Solution of the Navier-Stokes Equations*, Math. Comp., 22, pp. 745, 1968.
- [20] Crandall, M.G., and Lions, P-L., *Viscosity Solutions of Hamilton-Jacobi Equations*, Tran. AMS, 277, pp. 1-43, 1983.
- [21] Dijkstra, E.W., *A Note on Two Problems in Connection with Graphs*, Numerische Mathematic, 1:269-271, 1959.
- [22] Ecker, K., Huisman, G., *Interior Estimates for Hypersurfaces Moving by Mean Curvature*, Inventiones Mathematica, 105, 3, pp. 547-569, 1991.
- [23] Evans, L.C., and Spruck, J., *Motion of Level Sets by Mean Curvature I*, J. Diff. Geom, 33, 635, 1991.
- [24] Evans, L.C., and Spruck, J., *Motion of Level Sets by Mean Curvature II*, Transactions of the American Mathematical Society, 330, 1, pp. 321-332, 1992.
- [25] Evans, L.C., and Spruck, J., *Motion of Level Sets by Mean Curvature III*, J. Geom. Anal. 2, pp. 121-150, 1992.

- [26] Evans, L.C., and Spruck, J., *Motion of Level Sets by Mean Curvature IV*, J. Geom. Anal., 5, 1, pp. 77–114, 1995.
- [27] Gage, M., *Curve Shortening Makes Convex Curves Circular*, Inventiones Mathematica, 76, pp. 357, 1984.
- [28] Gage, M., and Hamilton, R., *The Equation Shrinking Convex Planes Curves*, J. Diff. Geom, 23, pp. 69, 1986.
- [29] Giga, Y., and Goto, S., *Motion of Hypersurfaces and Geometric Equations*, Journal of the Mathematical Society of Japan, 44, pp. 99, 1992.
- [30] Giga, Y., Goto, S., Ishii, H., *Global Existence of Weak Solutions for Interface Equations Coupled with Diffusion Equations*, SIAM J. Math. Anal., 23, N4, pp. 821–835, 1992.
- [31] Grayson, M., *The Heat Equation Shrinks Embedded Plane Curves to Round Points*, J. Diff. Geom., 26, pp. 285, 1987.
- [32] Grayson, M., *A Short Note on the Evolution of Surfaces Via Mean Curvatures*, J. Diff. Geom., 58, pp. 555, 1989.
- [33] Harten, A., Engquist, B., Osher, S., and Chakravarthy, S., *Uniformly High Order Accurate Essentially Non-oscillatory Schemes. III*, J. Comp. Phys., 71, 2, pp. 231–303, 1987.
- [34] Jain, M.K., Cale, T.S., Tracy, C.J., and Duffin, R.L., *Curvature Driven Surface Diffusion of Aluminum-(1.5)Copper During Sputter Deposition*, Proceedings, 11th VMIC Conference, 1992.
- [35] Kimmel, R., and Sethian, J.A., *Application of Fast Marching Methods to Path Planning*, submitted for publication, IEEE Transactions on Robotics and Automation, 1996.
- [36] Latombe, J.C., *Robot motion planning*, Kluwer Academic Publishers, Boston, MA, 1991.
- [37] Malladi, R., Sethian, J.A., and Vemuri, B.C., *Evolutionary Fronts for Topology-independent Shape Modeling and Recovery*, in Proceedings of Third European Conference on Computer Vision, Stockholm, Sweden, Lecture Notes in Computer Science, 800:3–13, 1994.

- [38] Malladi, R., Sethian, J.A., and Vemuri, B.C., *Shape Modeling with Front Propagation: A Level Set Approach*, IEEE Trans. on Pattern Analysis and Machine Intelligence, 17, 2, pp. 158–175, 1995.
- [39] Malladi, R., Sethian, J.A., and Vemuri, B.C., *A Fast Level Set based Algorithm for Topology-Independent Shape Modeling* J. Math. Imaging and Vision, 6, 2/3, pp. 269–290, 1996.
- [40] Malladi, R., and Sethian, J.A., *Image Processing via Level Set Curvature Flow*, Proc. Natl. Acad. of Sci., 92, 15, pp. 7046–7050, 1995.
- [41] Malladi, R., and Sethian, J.A., *Image Processing: Flows under Min/Max Curvature and Mean Curvature*, Graphical Models in Image Processing, 58(2) pp. 127-141, March, 1996.
- [42] Malladi, R., and Sethian, J.A., *An $O(N \log N)$ Algorithm for Shape Modeling*, Proc. Nat. Acad. Sci., Vol. 93, 1996.
- [43] Malladi R., and Sethian, J.A., *A Unified Approach to Noise Removal, Image Enhancement, and Shape Recovery*, IEEE Transactions on Image Processing, 5, 11, pp. 1154-1168, 1996.
- [44] Malladi R., and Sethian, J.A., *Flows under Min/Max Curvature and Mean Curvature: Applications in Image Processing*, Proceedings of the Fourth European Conference on Computer Vision, LNCS Vol. 1064, pp. 251-261, University of Cambridge, Cambridge, England, April 1996.
- [45] Malladi R., and Sethian, J.A., *A Real-Time Algorithm for Medical Shape Visualization and Measurement*, *Computer Aided Geometric Design*, submitted for publication, special issue on Medical Imaging, January 1997.
- [46] Osher, S., and Sethian, J.A., *Fronts Propagating with Curvature Dependent speed: Algorithms Based on Hamilton-Jacobi Formulation*, J. Comp. Phys., 79:12-49, 1988.
- [47] Rhee, C., Talbot, L., and Sethian, J.A., *Dynamical Study of a Premixed V flame*, Jour. Fluid Mech., 300, pp. 87–115, 1995.
- [48] Rouy, E. and Tourin, A., *A Viscosity Solutions Approach to Shape-From-Shading*, SIAM J. Num. Anal., 29, 3, pp. 867–884, 1992.

- [49] Sedgewick, R., *Algorithms*, Addison-Wesley, Reading, MA, 1988.
- [50] Sethian, J.A., *An Analysis of Flame Propagation*, Ph.D. Dissertation, Dept. of Mathematics, University of California, Berkeley, CA, 1982.
- [51] Sethian, J.A., *Curvature and the Evolution of Fronts*, Comm. Math. Phys., 101:487–499, 1985.
- [52] Sethian, J.A., *Numerical Methods for Propagating Fronts*, in Variational Methods for Free Surface Interfaces, Eds. P. Concus and R. Finn, Springer-Verlag, NY, 1987.
- [53] Sethian, J.A., *Numerical Algorithms for Propagating Interfaces: Hamilton–Jacobi Equations and Conservation Laws*, Journal of Differential Geometry, 31, pp. 131–161, 1990.
- [54] Sethian, J.A., *Curvature Flow and Entropy Conditions Applied to Grid Generation*, J. Comp. Phys., 115, pp. 440–454, 1994.
- [55] Sethian, J.A., *A Marching Level Set Method for Monotonically Advancing Fronts*, Proc. Nat. Acad. Sci., 93(4): 1591–1595, 1996.
- [56] Sethian, J.A., *Applications of Level Set Methods for Propagating Interfaces*, Acta Numerica, 1996.
- [57] Sethian, J.A., *Fast Marching Level Set Methods for Three-Dimensional Photolithography Development*, Proceedings, SPIE 1996 International Symposium on Microlithography, Santa Clara, California, March, 1996.
- [58] Sethian, J.A., *Algorithms for Tracking Interfaces in CFD and Material Science*, Annual Review of Computational Fluid Mechanics, 1995.
- [59] Sethian, J.A., *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Material Science*, Cambridge University Press, 1996.
- [60] Sethian, J.A., *Fast Marching Methods*, to appear, SIAM Review, 1998.
- [61] Sethian, J.A., and Adalsteinsson, D., *An Overview of Level Set Methods for Etching, Deposition, and Lithography Development*, IEEE Transactions on Semiconductor Devices, 1996. 10, 1, pp.167-184, 1997.

- [62] Sethian, J.A., and Popovici, A.M., *Three dimensional traveltimes computation using the Fast Marching Method*, submitted for publication, Geophysics, 1997.
- [63] Sethian, J.A. and Strain, J.D., *Crystal Growth and Dendritic Solidification* J. Comp. Phys., 98, pp. 231–253, 1992.
- [64] Sussman, M., Smereka, P. and Osher, S.J., *A Level Set Method for Computing Solutions to Incompressible Two-Phase Flow*, J. Comp. Phys. 114, pp. 146–159, 1994.
- [65] van Trier, J., and Symes, W.W., *Upwind Finite-difference Calculations of Traveltimes*, Geophysics, 56, 6, pp. 812–821, 1991.
- [66] Vidale, J., *Finite-Difference Calculation of Travel Times*, Bull. of Seism. Soc. of Amer., 78, 6, pp. 2062–2076, 1988.
- [67] Vidale, J., *Finite-difference calculation of traveltimes in three dimensions*, *Geophysics*, **55**, 521–526., 1990.
- [68] Zhao, H-K, Chan, T., Merriman, B. Osher, S., *A variational level set approach to multiphase motion*, J. Comp. Phys., 127, pp. 179-95, 1996.
- [69] Zhu, J., and Ronney, P.D., *Simulation of Front Propagation at Large Non-dimensional Flow Disturbance Intensities*, Comb. Sci. Tech., 100, pp. 183–201, 1995.
- [70] Zhu, J., and Sethian, J.A., *Projection Methods Coupled to Level Set Interface Techniques*, J. Comp. Phys., 102, pp. 128–138, 1992.