SIAM J. SCI. COMPUT.
Vol. 23, No. 1, pp. 230–244

SOME IMPROVEMENTS OF THE FAST MARCHING METHOD*

DAVID L. CHOPP†

Abstract. The fast marching method published by Sethian [*Proc. Natl. Acad. Sci. USA*, 93 (1996), pp. 1591–1595] is an optimally efficient algorithm for solving problems of front evolution where the front speed is monotonic. It has been used in a wide variety of applications such as robotic path planning [R. Kimmel and J. Sethian, *Fast Marching Methods for Computing Distance Maps and Shortest Paths*, Tech. Report 669, CPAM, University of California, Berkeley, 1996], crack propagation [M. Stolarska et al., *Internat. J. Numer. Methods Engrg.*, 51 (2001), pp. 943–960; N. Sukumar, D. L. Chopp, and B. Moran, *Extended finite element method and fast marching method for three-dimensional fatigue crack propagation*, J. Comput. Phys., submitted], seismology [J. Sethian and A. Popovici, *Geophysics*, 64 (1999), pp. 516–523], photolithography [J. Sethian, *Fast marching level set methods for three-dimensional photolithography development*, in Proceedings of the SPIE 1996 International Symposium on Microlithography, Santa Clara, CA, 1996], and medical imaging [R. Malladi and J. Sethian, *Proc. Natl. Acad. Sci. USA*, 93 (1996), pp. 9389–9392]. It has also been a valuable tool for the implementation of modern level set methods where it is used to efficiently compute the distance to the front and/or an extended velocity function.

In this paper, we improve upon the second order fast marching method of Sethian [*SIAM Rev.*, 41 (1999), pp. 199–235] by constructing a second order approximation of the interface generated from local data on the mesh. The data is interpolated on a single box of the mesh using a bicubic approximation. The distance to the front is then calculated by using a variant of Newton's method to solve both the level curve equation and the orthogonality condition for the nearest point to a given node. The result is a second order approximation of the distance to the interface which can then be used to produce second order accurate initial conditions for the fast marching method and a third order fast marching method.

Key words. fast marching method, level set method, bicubic interpolation, reinitialization

AMS subject classifications. 65M06, 65D05, 65D10

PII. S106482750037617X

1. Introduction.

The fast marching method published by Sethian [12] is an optimally efficient algorithm for solving general problems of front evolution where the front speed is monotonic. It has been used in a wide variety of applications such as robotic path planning [7], crack propagation [15, 16], seismology [14], photolithography [10], and medical imaging [8]. It has also been a valuable tool for the implementation of modern level set methods where it is used to efficiently compute the distance to the front and/or an extended velocity function. The fast marching method can trace its roots back to graph theoretical results first published by Dijkstra [4], and a review of the fast marching method can be found in Sethian's review article [13].

The fast marching method, presented in [12], was first order accurate, using a first order approximation of the derivatives. In [13], a second order fast marching method is constructed by using corresponding higher order approximations of the derivatives. For one to achieve higher order methods, the initial data must be more accurate. This is possible if the initial distance to the front is given explicitly, but it is more complicated if the initial front is given as a level curve of a function on the mesh, as

*Received by the editors August 4, 2000; accepted for publication (in revised form) February 20, 2001; published electronically June 19, 2001. This work was supported by the NSF/DARPA VIP program under award 96-15877.

http://www.siam.org/journals/sisc/23-1/37617.html

†Engineering Sciences and Applied Mathematics Dept., Northwestern University, Evanston, IL 60208-3125 (chopp@northwestern.edu).

commonly arises in applications of the level set method. The method for initializing the fast marching data in [13] uses first order accurate linear interpolation limiting the global error to second order at best.

The level set method is a numerical method which has gained significant popularity in recent years for solving complex problems of evolving interfaces, particularly where the topology of the interface may change. The key advantages of the method are that it can change topology without special cases, can propagate sharp corners and cusps in the interface, and is easily extendible to any number of spatial dimensions. This compares with marker particle-type methods (see, e.g., [5]), where topology changes require collision detection and interface surgery, and the mesh itself requires node point redistribution for stability (and also smooths sharp edges formed by the flow). On the other hand, the interface is known only implicitly in a level set formulation, while it is known explicitly for a marker particle method. The results of this paper offer a way to close this latter gap between the two types of interface representations. Both methods have their uses, and a complete comparison between these two methods is beyond the scope of this paper. For a complete discussion of the level set method and reinitialization, see [9, 11].

One common application of the fast marching method is for implementing reinitialization within the framework of the level set method. Reinitialization, introduced in [3], is where the signed distance map to the zero level set of a function is computed. One common criticism of the level set method is its purported problem with mass conservation. Reinitialization techniques, such as those given in [3] and [18], are well known to have difficulty preserving mass. This is primarily due to the poor accuracy around the zero level set.

In this paper, we improve upon the second order fast marching method of Sethian [13] by constructing a second order approximation of the interface generated from local data on the mesh. The data is interpolated on a single box of the mesh using a bicubic approximation. The distance to the front is then calculated by using a variant of Newton's method to solve both the level curve equation and the orthogonality condition for the nearest point to a given node. The result is a second order approximation of the distance to the interface which can then be used to produce second order accurate initial conditions for the fast marching method and a third order fast marching method. We show with computed examples that the method appears to be second order accurate locally around the zero level set, can be used to build a globally third order accurate fast marching method, preserves mass markedly better during reinitialization than existing methods, and easily generalizes to any number of spatial dimensions—all without increasing the computational complexity of the fast marching method.

In section 2, we describe the fast marching method as presented in [12, 13]. In section 3 we describe the bicubic interpolation and how it is used to construct a second order approximation to the distance map. We also give the formulation for implementing a third order accurate fast marching method. In section 4 we compare the new bicubic approximation method with the second order method in [13] with a series of examples. Finally, in section 5 we summarize our results, discuss how this local approximation can be used for velocity extensions, and briefly describe a current application in crack propagation where this method is already successfully in use.

2. The fast marching method. The fast marching method is an optimal method for solving an equation of the form

$$\|\nabla\phi\| = 1/F(\mathbf{x}), \tag{2.1}$$

where $F(\mathbf{x})$ is a monotonic speed function for an advancing interface. If $\phi^{-1}(0)$ represents the initial interface, and ϕ solves (2.1), then $\phi^{-1}(t)$ gives the location of the interface evolving with normal velocity $F(\mathbf{x}) > 0$ at time t. The advantage of this method over the original level set method is that the entire evolution of the front is computed in one pass over the mesh with an operation count of $O(N \log N)$ for N mesh points. It is also advantageous over other front tracking algorithms in that it uses techniques borrowed from hyperbolic conservation laws to properly advance fronts with sharp corners and cusps. We present here a basic description of the method; the interested reader is referred to [11, 12, 13].

In the fast marching method, (2.1) is solved numerically by using upwind finite differences to approximate $\nabla\phi$. The use of upwind finite differences indicates a causality, or a direction for the flow of information propagating from the initial contour $\phi^{-1}(0)$ outward to larger values of ϕ. This causality means that the value of $\phi(\mathbf{x})$ depends only on values of $\phi(\mathbf{y})$ for which $\phi(\mathbf{y}) \leq \phi(\mathbf{x})$. Thus, if we solve for the values of ϕ in a monotonically increasing fashion, then the upwind differences are always valid and all the mesh points are eventually computed. This sequential procession through the mesh points is maintained by a heap sort which controls the order in which the mesh points are computed.

To begin, the mesh points are separated into three disjoint sets: the set of *accepted* points A, the set of *tentative* points T, and the set of *distant* points D. The mesh points in the set A are considered computed and are always closer to the initial interface than any of the remaining mesh points. The mesh points in T are all potential candidates to be the next mesh point to be added to the set A. The mesh points in T are always kept sorted in a heap sort so that the best candidate is always easily found. The mesh points in D are considered too far from the initial interface to be possible candidates for inclusion in A. Thus, if $\mathbf{x} \in A$, $\mathbf{y} \in T$, and $\mathbf{z} \in D$, then $\phi(\mathbf{x}) < \phi(\mathbf{y}) < \phi(\mathbf{z})$. Figure 2.1 shows the relationship between the different sets of mesh points.

To describe the main algorithm for the fast marching method, we will use the notation for discrete derivatives given by

$$\begin{aligned}
\phi_{i,j} &= \phi(\mathbf{x}_{i,j}),\\
D_x^+\phi_{i,j} &= \frac{1}{\Delta x}(\phi_{i+1,j} - \phi_{i,j}),\\
D_x^-\phi_{i,j} &= \frac{1}{\Delta x}(\phi_{i,j} - \phi_{i-1,j}),\\
D_y^+\phi_{i,j} &= \frac{1}{\Delta y}(\phi_{i,j+1} - \phi_{i,j}),\\
D_y^-\phi_{i,j} &= \frac{1}{\Delta y}(\phi_{i,j} - \phi_{i,j-1}),
\end{aligned}$$

where Δx, Δy are the space step sizes in the x and y directions, respectively. One of the key components in the fast marching method is the computation of the estimate of ϕ for points in T. Suppose, for example, mesh points $\mathbf{x}_{i-1,j}$, $\mathbf{x}_{i,j+1} \in A$ and $\mathbf{x}_{i,j} \in T$. Given the values of $\phi_{i-1,j}$, $\phi_{i,j+1}$, we must estimate the value of $\phi_{i,j}$. This is accomplished by looking at the discretization of (2.1) given by

$$(D_x^-\phi_{i,j})^2 + (D_y^+\phi_{i,j})^2 = \frac{1}{F_{i,j}^2}. \tag{2.2}$$

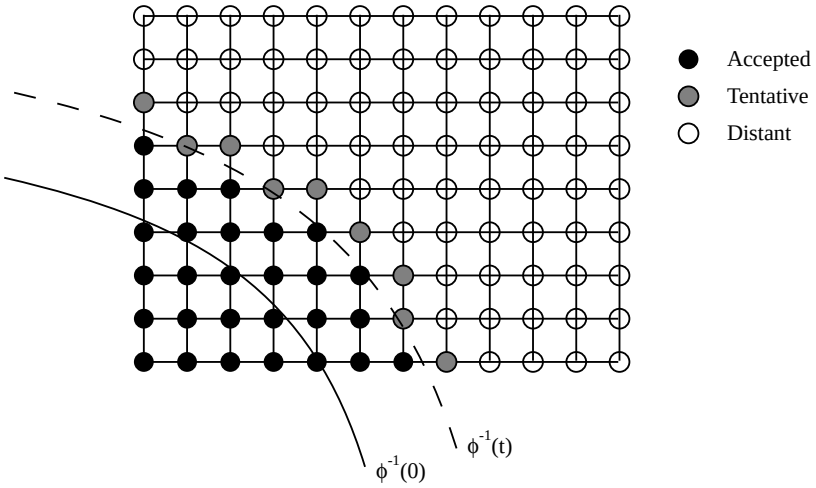

FIG. 2.1. *Illustration of the sets A, T, and D.*

Equation (2.2) reduces to a quadratic equation in $\phi_{i,j}$ given by

$$\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)\phi_{i,j}^2 - 2\left(\frac{\phi_{i-1,j}}{\Delta x^2}+\frac{\phi_{i,j+1}}{\Delta y^2}\right)\phi_{i,j} + \frac{\phi_{i-1,j}^2}{\Delta x^2}+\frac{\phi_{i,j+1}^2}{\Delta y^2} - \frac{1}{F^2} = 0. \tag{2.3}$$

The new estimate for $\phi_{i,j}$ is given by the largest of the two roots of (2.3). The remaining configurations and the resulting quadratic equations can be derived in a similar fashion.

Note that (2.2) is a first order approximation of (2.1). The general second and third order approximations of (2.1) requires the use of switches determined by which points are in A:

$$\begin{aligned}\max\Bigg(&D_x^-\phi_{i,j} + s_{x,-1}\frac{\Delta x}{2}D_x^-D_x^-\phi_{i,j} + s_{x,-1}s_{x,-2}\frac{\Delta x^2}{6}D_x^-D_x^-D_x^-\phi_{i,j},\\ &-D_x^+\phi_{i,j} + s_{x,1}\frac{\Delta x}{2}D_x^+D_x^+\phi_{i,j} + s_{x,1}s_{x,2}\frac{\Delta x^2}{6}D_x^+D_x^+D_x^+\phi_{i,j}, 0\Bigg)^2\\ +\max\Bigg(&D_y^-\phi_{i,j} + s_{y,-1}\frac{\Delta y}{2}D_y^-D_y^-\phi_{i,j} + s_{y,-1}s_{y,-2}\frac{\Delta y^2}{6}D_y^-D_y^-D_y^-\phi_{i,j},\\ &-D_y^+\phi_{i,j} + s_{y,1}\frac{\Delta y}{2}D_y^+D_y^+\phi_{i,j} + s_{y,1}s_{y,2}\frac{\Delta y^2}{6}D_y^+D_y^+D_y^+\phi_{i,j}, 0\Bigg)^2 = \frac{1}{F_{i,j}^2},\end{aligned} \tag{2.4}$$

where the switches are

$$s_{x,k} = \begin{cases}1, & \mathbf{x}_{i+k,j}\in A,\\ 0 & \text{otherwise},\end{cases}$$

$$s_{y,k} = \begin{cases}1, & \mathbf{x}_{i,j+k}\in A,\\ 0 & \text{otherwise}.\end{cases}$$

Now the fast marching method can be assembled as an algorithm:

1. Initialize all the points adjacent to the initial interface with an initial value; put those points in A. A discussion about initialization follows in section 3. All points $\mathbf{x}_{i,j} \notin A$ but are adjacent to a point in A are given initial estimates for $\phi_{i,j}$ by solving (2.4). These points are tentative points and put in the set T. All remaining points unaccounted for are placed in D and given initial value of $\phi_{i,j} = +\infty$.
2. Choose the point $\mathbf{x}_{i,j} \in T$ which has the smallest value of $\phi_{i,j}$ and move it into A. Any point which is adjacent to $\mathbf{x}_{i,j}$ (i.e., the points $\mathbf{x}_{i-1,j}$, $\mathbf{x}_{i,j-1}$, $\mathbf{x}_{i+1,j}$, $\mathbf{x}_{i,j+1}$) which is in T has its value $\phi_{i,j}$ recalculated using (2.4). Any point adjacent to $\mathbf{x}_{i,j}$ and in D has its value $\phi_{i,j}$ computed using (2.4) and is moved into the set T.
3. If $T \neq \emptyset$, go to step 2.

The algorithm presented in [12] is first order accurate and can be recovered from (2.4) by taking all the switches $s_{\bullet,\bullet} = 0$. The second order accurate method presented in [13] can also be recovered from (2.4) by taking all the switches $s_{\bullet,\pm 2} = 0$.

3. Locally second order approximation of the level set function. In [13], it is shown that the fast marching method behaves as a globally second order accurate method. However, it is also noted in [13] that the method is only first order accurate in the set of nodes immediately adjacent to the initial front. This does not destroy the global accuracy because as the mesh shrinks, so too does the size of the region of first order accuracy. To achieve a fully second order accurate fast marching method, we must ensure that the initial data is at least second order accurate. Furthermore, we can also get a globally third order accurate fast marching method by virtue of having at least second order initial data.

The underpinning of this higher degree of accuracy around the initial front is the use of a bicubic interpolation function p which is a second order accurate local representation of a level set function ϕ (i.e., $p(\mathbf{x}) \approx \phi(\mathbf{x})$). The interpolation function $p(\mathbf{x})$ can serve many purposes including second order accuracy for the distance to the zero level set, subgrid resolution of the shape of the interface, as well as subgrid resolution of the level set function $\phi(\mathbf{x})$ itself.

3.1. The bicubic interpolation. We begin with a description of the bicubic interpolation for a level set function given on a rectangular mesh. The approximation is done locally in a box of the mesh bounded by grid points; call them $\mathbf{x}_{i,j}$, $\mathbf{x}_{i+1,j}$, $\mathbf{x}_{i,j+1}$, and $\mathbf{x}_{i+1,j+1}$, as in Figure 3.1.

A bicubic interpolation $p(\mathbf{x})$ of a function $\phi(\mathbf{x})$ is a function

$$p(\mathbf{x}) = p(x,y) = \sum_{m=0}^{3}\sum_{n=0}^{3} a_{m,n} x^m y^n \tag{3.1}$$

which solves the following set of equations:

$$\begin{aligned} p(\mathbf{x}_{k,\ell}) &= \phi(\mathbf{x}_{k,\ell}), \\ \frac{\partial p}{\partial x}(\mathbf{x}_{k,\ell}) &= \frac{\partial \phi}{\partial x}(\mathbf{x}_{k,\ell}), \\ \frac{\partial p}{\partial y}(\mathbf{x}_{k,\ell}) &= \frac{\partial \phi}{\partial y}(\mathbf{x}_{k,\ell}), \\ \frac{\partial^2 p}{\partial x \partial y}(\mathbf{x}_{k,\ell}) &= \frac{\partial^2 \phi}{\partial x \partial y}(\mathbf{x}_{k,\ell}) \end{aligned}$$

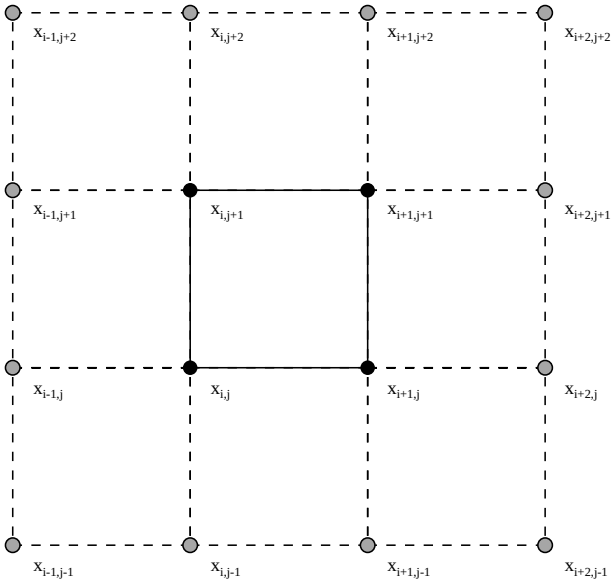

FIG. 3.1. *Sample portion of the mesh where a bicubic interpolation is used.*

for $k = i,\ i+1;\ \ell = j,\ j+1$. This gives sixteen equations for the sixteen unknown coefficients $a_{m,n}$. Solving for the $a_{m,n}$ makes $p(x,y)$ a bicubic interpolating function of $\phi(x,y)$ on the rectangle bounded by the corners $\mathbf{x}_{i,j}$, $\mathbf{x}_{i+1,j}$, $\mathbf{x}_{i,j+1}$, and $\mathbf{x}_{i+1,j+1}$.

Since ϕ is known only on the mesh points, the values for the derivatives of ϕ must be approximated. We use second order finite difference approximations for the derivatives of ϕ:

$$\begin{aligned}
\frac{\partial\phi}{\partial x}(\mathbf{x}_{m,n}) &\approx \frac{1}{2\Delta x}(\phi(\mathbf{x}_{m+1,n}) - \phi(\mathbf{x}_{m-1,n})),\\
\frac{\partial\phi}{\partial y}(\mathbf{x}_{m,n}) &\approx \frac{1}{2\Delta y}(\phi(\mathbf{x}_{m,n+1}) - \phi(x_{m,n-1})),\\
\frac{\partial^2\phi}{\partial x\partial y}(\mathbf{x}_{m,n}) &\approx \frac{1}{4\Delta x\Delta y}(\phi(\mathbf{x}_{m+1,n+1}) - \phi(\mathbf{x}_{m-1,n+1})\\
&\qquad - \phi(\mathbf{x}_{m+1,n-1}) + \phi(\mathbf{x}_{m-1,n-1}))
\end{aligned}$$

for $m = i, i+1$ and $n = j, j+1$. Thus, construction of the interpolant p requires all the points shown in Figure 3.1. The observant reader will note that the accuracy of this method is restricted to second order by our use of second order approximations for the derivatives of ϕ. In principle, higher order local approximations can be made using higher order finite difference approximations and using a larger set of grid points around the box where the interpolant is used.

One useful property of this local interpolation method is that interpolations in neighboring boxes match smoothly along their common edge. To see this, consider two interpolations $p(x,y)$ and $q(x,y)$ interpolating in two adjacent boxes as depicted in Figure 3.2. To show continuity, we must show $p(x_i,y) = q(x_i,y)$ for $y_j \le y \le y_{j+1}$. Now, $p(x_i,y)$ and $q(x_i,y)$ are both cubic polynomials in y. Furthermore, by construction, $p(x_i,y_j) = q(x_i,y_j)$, $p(x_i,y_{j+1}) = q(x_i,y_{j+1})$, $\frac{\partial p}{\partial y}(x_i,y_j) = \frac{\partial q}{\partial y}(x_i,y_j)$,

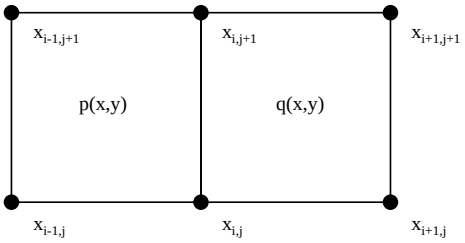

FIG. 3.2. *Two neighboring bicubic interpolants.*

and $\frac{\partial p}{\partial y}(x_i, y_{j+1}) = \frac{\partial q}{\partial y}(x_i, y_{j+1})$. These four conditions uniquely determine the cubic polynomial along the common edge; hence $p(x_i, y) \equiv q(x_i, y)$.

We can go one step further by noting that $\frac{\partial p}{\partial x}(x_i, y)$ and $\frac{\partial q}{\partial x}(x_i, y)$ are also cubic polynomials in y and by construction, $\frac{\partial p}{\partial x}(x_i, y_j) = \frac{\partial q}{\partial x}(x_i, y_j)$, $\frac{\partial p}{\partial x}(x_i, y_{j+1}) = \frac{\partial q}{\partial x}(x_i, y_{j+1})$, $\frac{\partial^2 p}{\partial x \partial y}(x_i, y_j) = \frac{\partial^2 q}{\partial x \partial y}(x_i, y_j)$, and $\frac{\partial^2 p}{\partial x \partial y}(x_i, y_{j+1}) = \frac{\partial^2 q}{\partial x \partial y}(x_i, y_{j+1})$. As above, these four equations uniquely determine the cubic polynomial $\frac{\partial p}{\partial x}(x_i, y) \equiv \frac{\partial q}{\partial x}(x_i, y)$.

If we differentiate the two results $p(x_i, y) \equiv q(x_i, y)$, $\frac{\partial p}{\partial x}(x_i, y) \equiv \frac{\partial q}{\partial x}(x_i, y)$ with respect to y, we get the additional equations $\frac{\partial p}{\partial y}(x_i, y) \equiv \frac{\partial q}{\partial y}(x_i, y)$, $\frac{\partial^2 p}{\partial x \partial y}(x_i, y) \equiv \frac{\partial^2 q}{\partial x \partial y}(x_i, y)$. Therefore, the combined bicubic interpolation over two adjacent boxes is at least $C^{(1)}$.

The approximation $p(x, y)$ is now a local subgrid representation of the level set function ϕ inside the box and can be used as an initializer for a fast marching or velocity extension method.

3.2. Reinitialization. Reinitialization is the construction of the distance map to the zero level set of a given level set function ψ. Thus, given $\psi(\mathbf{x})$, we want to construct $\phi(\mathbf{x})$ such that $\phi^{-1}(0) = \psi^{-1}(0)$ and $\|\nabla\phi\| = 1$. Reinitialization was first introduced in [3], where it was shown to improve the stability of the level set method. It continues to play a crucial role in many level set applications, for example, narrow-band level set methods [1], incompressible fluid flow [18], and computer-aided design [6], to name a few.

One of the main drawbacks of using reinitialization in level set methods is the difficulty in maintaining the original position of the interface, often leading to breakdown in conservation of area/volume. This problem surfaced in the original method proposed in [3] and persists today as observed in [17]. To overcome this problem with the level set method, different solutions have been proposed. Adalsteinsson and Sethian [2] used velocity extensions constructed with the fast marching method to drastically reduce or even eliminate the need for reinitialization in level set method applications. However, their velocity extension method is only first order accurate in a neighborhood of the interface. Sussman and Fatemi [17] proposed a mass correction procedure which artificially enforces mass conservation by modifying the front velocity. While this method forces mass conservation by construction, it comes at the price of modifying the original front velocity with a nonphysical correction term.

In this paper, we focus on reinitialization because it is a stepping stone to initializing the more general fast marching method, and also because it is important

to the level set method on its own. The method for reinitialization we present here using bicubic interpolation has two very important properties that previous reinitialization techniques lack: very good front location preservation and, consequently, very good mass conservation. This will be demonstrated by showing that repeated reinitializations do not significantly degrade the location of the interface compared with existing first order methods. In the next section we will describe how this method of reinitialization can be used to solve more general fast marching method applications.

Given a function $\psi(\mathbf{x})$, we want to construct $\phi(\mathbf{x})$ such that $\phi^{-1}(0) = \psi^{-1}(0)$ and $\|\phi\| = 1$. We begin by passing through the mesh to locate all rectangles bounded by nodes which are not all of the same sign, indicating that the zero level set passes through that rectangle. Suppose the rectangle is bounded by the nodes $\mathbf{x}_{i,j}$, $\mathbf{x}_{i+1,j}$, $\mathbf{x}_{i,j+1}$, $\mathbf{x}_{i+1,j+1}$ as in Figure 3.1. Using the values of ψ at these nodes, we construct the bicubic interpolation function $p(\mathbf{x})$ as described above.

Now, let $\mathbf{x}^0$ be a point in the rectangle where we wish to compute the distance to the set $\psi^{-1}(0) \approx p^{-1}(0)$. Let $\mathbf{y}$ be a point in $p^{-1}(0)$ nearest to $\mathbf{x}^0$; then it follows that

$$p(\mathbf{y}) = 0, \tag{3.2}$$

$$\nabla p(\mathbf{y}) \times (\mathbf{x}^0 - \mathbf{y}) = 0. \tag{3.3}$$

Equation (3.2) is a requirement that $\mathbf{y}$ must be on the zero contour of p and (3.3) is a requirement that the normal to the zero contour, given by $\nabla p(\mathbf{y})$, must be aligned with the line through the points $\mathbf{x}^0$, $\mathbf{y}$.

To solve (3.2)–(3.3), we employ a variant of Newton's method to simultaneously solve the pair of conditions. We compute a sequence of iterates $\mathbf{x}^k$ with initial point $\mathbf{x}^0$. The update for the iterates is given by

$$\boldsymbol{\delta}_1 = -p(\mathbf{x}^k)\frac{\nabla p(\mathbf{x}^k)}{\nabla p(\mathbf{x}^k) \cdot \nabla p(\mathbf{x}^k)}, \tag{3.4}$$

$$\mathbf{x}_{k+1/2} = \mathbf{x}^k + \boldsymbol{\delta}_1, \tag{3.5}$$

$$\boldsymbol{\delta}_2 = (\mathbf{x}^0 - \mathbf{x}^k) - \frac{(\mathbf{x}^0 - \mathbf{x}^k) \cdot \nabla p(\mathbf{x}^k)}{\nabla p(\mathbf{x}^k) \cdot \nabla p(\mathbf{x}^k)} \nabla p(\mathbf{x}_k), \tag{3.6}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_{k+1/2} + \boldsymbol{\delta}_2. \tag{3.7}$$

This method takes advantage of the orthogonality of the solution sets for (3.2) and (3.3). Equation (3.4) is the Newton step for the function $g_1(t) = p(\mathbf{x}^k + t\nabla p(\mathbf{x}^k))$. This step moves in the direction of ∇p, and hence holds the value of the left side of (3.3) approximately fixed. Equation (3.6) is the Newton step for the function $g_2(t) = \mathbf{k} \cdot (\nabla p(\mathbf{x}^k) \times (\mathbf{x}_0 - (\mathbf{x}^k + t(\mathbf{k} \times \nabla p(\mathbf{x}^k)))))$. This step moves in the direction normal to ∇p, and hence holds the value of p approximately fixed. Figure 3.3 illustrates this algorithm.

We found the number of iterations required to obtain convergence using this two-step Newton's method was half the standard Newton step or less, but we do not believe the convergence rate differs from the standard Newton iteration. Should this method prove more generally useful, a more complete study of this method will be undertaken by the author.

For the results presented in section 4, we used the convergence criterion

$$\sqrt{\|\boldsymbol{\delta}_1\|^2 + \|\boldsymbol{\delta}_2\|^2} < 10^{-3}\Delta x \Delta y,$$

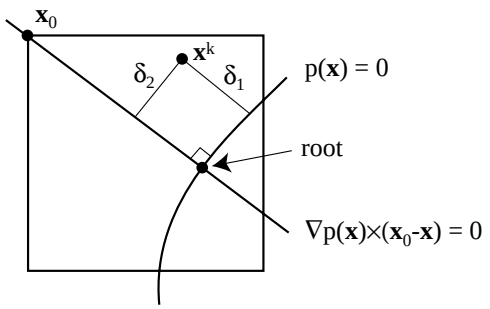

FIG. 3.3. *Example of solution sets for* (3.2) *and* (3.3).

where Δx, Δy are the space step sizes in the x and y directions, respectively. For nearly all evaluations, two or three iterations are necessary to meet the criterion. The distance to the front from the point $\mathbf{x}^0$ is now given by $\phi(\mathbf{x}^0) = \pm\|\mathbf{x}^0 - \mathbf{x}^\infty\|$, where the sign of $\phi(\mathbf{x}^0)$ is taken to be the same as that of $p(\mathbf{x}^0)$.

The above process gives more than just the distance to the zero level set of p; it also gives the location of the nearest point, $\mathbf{x}^\infty$. This will have important consequences later when we discuss the application of this method to velocity extensions and the general fast marching method.

Reinitialization now proceeds by using the above process to calculate the distance to the zero contour from each of the nodes $\mathbf{x}_{i,j}$, $\mathbf{x}_{i+1,j}$, $\mathbf{x}_{i,j+1}$, $\mathbf{x}_{i+1,j+1}$, i.e., each of these nodes plays the role of $\mathbf{x}^0$ above. Now, some of these nodes will be involved in multiple interpolation procedures, so the final value taken for ϕ at those nodes is the minimum of all computed distances to the front. To initialize the fast marching method to complete reinitialization, all nodes whose distance values are constructed using the bicubic interpolation are placed in the set A of accepted points.

The modified Newton iteration will not always converge. Divergence generally can happen when there is no solution for the pair of equations inside the box formed by the nodes $\mathbf{x}_{i,j}$, $\mathbf{x}_{i+1,j}$, $\mathbf{x}_{i,j+1}$, $\mathbf{x}_{i+1,j+1}$. In this event, the calculated distance is taken to be infinite. The iteration rarely requires more than five iterations, and hence twenty iterations is taken to be the maximum before divergence is determined.

3.3. Application to general fast marching method. The above construction was designed to solve the problem of reinitialization. To apply this method for more general fast marching methods, note that the initialization of the general fast marching method requires two components: the local front speed F and the initial distance to the front ϕ. The front speed is determined by the application, and the above process produces the initial distance to the front. Thus, the initial value for a point adjacent to the initial front for the general fast marching method solving (2.1) is ϕ/F. The same set of initially accepted points A are used for general fast marching as for reinitialization.

The fast marching method is also used to construct extension velocity fields for the level set method [2]. Extended velocity fields are used in the level set method when the front velocity F is defined only on the zero level set. In order to advance the level set method, a value of F must exist for the entire domain of the level set function ϕ, not just at the zero contour. The extended velocity field F_{ext} is a speed

function defined on the domain of the level set function which satisfies the conditions

$$F_{\text{ext}}\big|_{\phi^{-1}(0)} = F\big|_{\phi^{-1}(0)}, \tag{3.8}$$

$$\nabla F_{\text{ext}} \cdot \nabla \phi = 0, \tag{3.9}$$

where ϕ is again the signed distance to the front. The solution of (3.9) is accomplished using a technique based on the fast marching method; for details, see [2]. In this application, the speed function F_{ext} must be initialized in the same way as for the fast marching method. In this case, the actual location on the front nearest a given node in the mesh is useful, because we can now set $F_{\text{ext}}(\mathbf{x}^0) = F(\mathbf{x}^\infty)$. This solves (3.9) because F_{ext} is now constant on a line between $\mathbf{x}^\infty$ and $\mathbf{x}^0$, which means that

$$\nabla F_{\text{ext}} \perp (\mathbf{x}^\infty - \mathbf{x}^0).$$

But by construction,

$$(\mathbf{x}^\infty - \mathbf{x}^0) \parallel \nabla \phi(\mathbf{x}^\infty).$$

Therefore,

$$F_{\text{ext}}(\mathbf{x}^0) \cdot \nabla \phi(\mathbf{x}^\infty) = 0.$$

Solving (3.9) at the remainder of the nodes is done using fast marching where the heap sort sorts points according to their ϕ value.

4. Results. In this section, we demonstrate the advantages of this new method for initializing the fast marching method.

4.1. Accuracy of the local distance map. The first test of this method is to show that the initialization method is second order accurate in a neighborhood of the initial zero level set. To test this, we show that the error in the computed distance function to a given zero level set is second order accurate. We measure the error in only those points which are computed via the bicubic interpolation method and exclude all points which are subsequently computed by the fast marching method.

The tests were conducted on three initial contours, which are shown in Figure 4.1. We see in Table 4.1 that the initialization method presented in [13] behaves as a first order approximation, and the bicubic interpolation method shows second order convergence as shown in Table 4.2 for smooth functions. Note that for test cases with corners or cusps (the second two examples), the bicubic interpolation also reduces to first order due to the local discontinuity in the derivatives of the level set function.

The computational cost of the new initialization method compared with the old method increased by 20% for an 80×80 mesh, down to less than a 4% increase for the 320×320 mesh. This is expected because the more refined the mesh, the quicker the iterative step converges on average, and the overall computational cost of the initialization procedure is $O(N)$ with the number of nodes N compared with the fast marching method which increases in cost by $O(N \log N)$.

4.2. Order of accuracy of the fast marching method. Since we now have an initialization procedure which is second order accurate, we should in theory be able to build a third order fast marching method as given in (2.4). For comparison purposes, Table 4.3 shows the results of applying the second order method from [13] with first order initialization. Table 4.4 shows the error measurements for the third

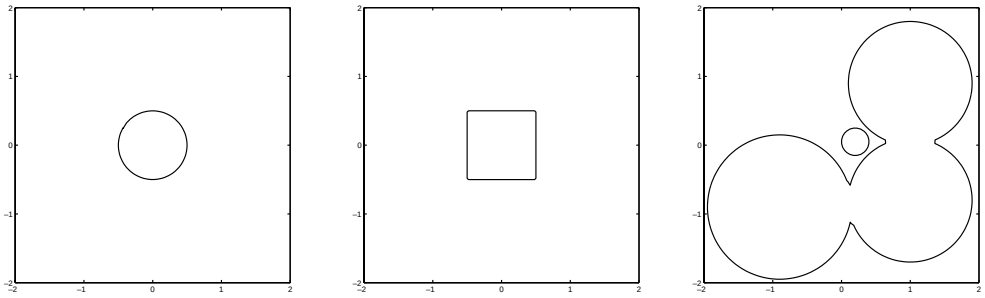

FIG. 4.1. *Three test cases for the new method.*

TABLE 4.1
Error measurements for linear initialization method.

Test case	Mesh size	L_1 error	L_2 error	L_∞ error
Circle	20×20	8.93×10^{-3}	1.05×10^{-2}	1.43×10^{-2}
	40×40	1.41×10^{-3}	1.81×10^{-3}	3.67×10^{-3}
	80×80	3.71×10^{-4}	4.87×10^{-4}	1.19×10^{-3}
	160×160	1.02×10^{-4}	1.36×10^{-4}	2.93×10^{-4}
	320×320	2.47×10^{-5}	3.14×10^{-5}	6.97×10^{-5}
Square	20×20	7.71×10^{-3}	2.04×10^{-3}	5.40×10^{-2}
	40×40	5.93×10^{-4}	2.58×10^{-3}	1.12×10^{-2}
	80×80	1.43×10^{-4}	8.91×10^{-4}	5.56×10^{-3}
	160×160	3.50×10^{-5}	3.11×10^{-4}	2.76×10^{-3}
	320×320	9.58×10^{-7}	2.10×10^{-5}	4.59×10^{-4}
Disks	20×20	7.60×10^{-3}	1.73×10^{-2}	9.14×10^{-2}
	40×40	1.70×10^{-3}	4.82×10^{-3}	4.57×10^{-2}
	80×80	5.71×10^{-4}	2.73×10^{-3}	3.92×10^{-2}
	160×160	1.31×10^{-4}	8.10×10^{-4}	1.43×10^{-2}
	320×320	3.01×10^{-5}	2.39×10^{-4}	6.23×10^{-3}

TABLE 4.2
Error measurements for bicubic initialization.

Test case	Mesh size	L_1 error	L_2 error	L_∞ error
Circle	20×20	5.73×10^{-4}	9.29×10^{-4}	1.81×10^{-3}
	40×40	5.68×10^{-5}	7.17×10^{-5}	1.41×10^{-4}
	80×80	6.07×10^{-6}	7.25×10^{-6}	1.36×10^{-5}
	160×160	6.34×10^{-7}	8.03×10^{-7}	1.96×10^{-6}
	320×320	9.38×10^{-8}	1.20×10^{-7}	2.74×10^{-7}
Square	20×20	1.00×10^{-2}	2.01×10^{-2}	4.94×10^{-2}
	40×40	1.55×10^{-3}	3.98×10^{-3}	1.33×10^{-2}
	80×80	3.82×10^{-4}	1.39×10^{-3}	6.57×10^{-3}
	160×160	9.49×10^{-5}	4.88×10^{-4}	3.26×10^{-3}
	320×320	1.30×10^{-5}	1.81×10^{-4}	2.79×10^{-3}
Disks	20×20	5.74×10^{-3}	1.80×10^{-2}	1.41×10^{-1}
	40×40	2.00×10^{-3}	7.29×10^{-3}	7.63×10^{-2}
	80×80	3.46×10^{-4}	1.43×10^{-3}	1.70×10^{-2}
	160×160	1.11×10^{-4}	1.04×10^{-3}	2.36×10^{-2}
	320×320	2.04×10^{-5}	2.68×10^{-4}	8.47×10^{-3}

order method. This method appears to be third order from the error measurements for the circle example. Note that the higher rate of convergence does not apply to the second two test cases because the exact solution is not smooth. We show them only to illustrate how the method behaves for more difficult problems.

TABLE 4.3
Error measurements for linear initialization method.

Test case	Mesh size	L_1 error	L_2 error	L_∞ error
Circle	20×20	2.65×10^{-2}	4.33×10^{-2}	1.68×10^{-1}
	40×40	4.17×10^{-3}	5.02×10^{-3}	1.78×10^{-2}
	80×80	1.03×10^{-3}	1.30×10^{-3}	9.79×10^{-3}
	160×160	1.60×10^{-4}	2.29×10^{-4}	5.07×10^{-3}
	320×320	3.27×10^{-5}	5.47×10^{-5}	2.57×10^{-3}
Square	20×20	5.40×10^{-2}	6.58×10^{-2}	2.05×10^{-1}
	40×40	9.76×10^{-3}	1.32×10^{-2}	2.72×10^{-2}
	80×80	4.53×10^{-3}	6.58×10^{-3}	1.50×10^{-2}
	160×160	2.17×10^{-3}	3.30×10^{-3}	7.93×10^{-3}
	320×320	2.06×10^{-3}	4.04×10^{-3}	1.60×10^{-2}
Disks	20×20	3.27×10^{-2}	7.28×10^{-2}	3.53×10^{-1}
	40×40	1.30×10^{-2}	3.26×10^{-2}	2.09×10^{-1}
	80×80	4.12×10^{-3}	2.00×10^{-2}	2.71×10^{-1}
	160×160	2.66×10^{-3}	1.59×10^{-2}	2.60×10^{-1}
	320×320	2.21×10^{-3}	1.43×10^{-2}	2.52×10^{-1}

TABLE 4.4
Error measurements for bicubic initialization.

Test case	Mesh size	L_1 error	L_2 error	L_∞ error
Circle	20×20	8.23×10^{-3}	1.13×10^{-2}	3.27×10^{-2}
	40×40	1.00×10^{-3}	1.68×10^{-3}	8.56×10^{-3}
	80×80	1.76×10^{-4}	3.20×10^{-4}	4.71×10^{-3}
	160×160	2.73×10^{-5}	6.88×10^{-5}	2.38×10^{-3}
	320×320	4.38×10^{-6}	1.48×10^{-5}	1.19×10^{-3}
Square	20×20	4.28×10^{-2}	6.15×10^{-2}	1.35×10^{-1}
	40×40	8.48×10^{-3}	1.17×10^{-2}	2.55×10^{-2}
	80×80	3.80×10^{-3}	5.52×10^{-3}	1.29×10^{-2}
	160×160	1.79×10^{-3}	2.67×10^{-3}	6.62×10^{-3}
	320×320	2.32×10^{-4}	9.13×10^{-4}	4.30×10^{-3}
Disks	20×20	9.85×10^{-3}	2.62×10^{-2}	1.61×10^{-1}
	40×40	5.16×10^{-3}	1.79×10^{-2}	1.94×10^{-1}
	80×80	3.50×10^{-3}	1.74×10^{-2}	2.60×10^{-1}
	160×160	2.43×10^{-3}	1.45×10^{-2}	2.54×10^{-1}
	320×320	2.20×10^{-3}	1.40×10^{-2}	2.44×10^{-1}

4.3. Conservation of mass. One common criticism of the level set method concerns the conservation properties of the method. The primary source of variation in mass is due to the process of reinitialization. In some applications of the level set method, such as narrow band methods, for example, frequent reinitializations are used. Repeated reinitializations eventually lead to problems in mass conservation. In Figure 4.2, we show that after twenty reinitializations, the location of the zero level set, and hence the measured value of the area bounded by the zero level set, is very well preserved. Table 4.5 shows the calculated area of the center circle after repeated reinitializations for both the old linear method and the new bicubic method.

4.4. Extension to higher dimensions. Another advantage of the initialization method presented in this paper concerns the extension to higher dimensions. The linear initialization method used does not easily extend to higher dimensions. The method presented in this paper is easily extended to any number of dimensions. For

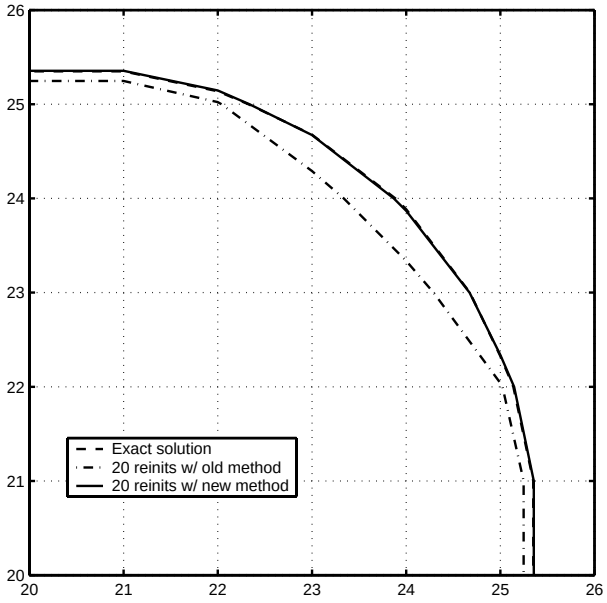

FIG. 4.2. *Deviation after twenty reinitializations.*

TABLE 4.5
Conservation of mass during reinitialization.

Method	Area after 10 reinits	Area after 20 reinits	% error after 20
Exact	0.778683	0.778683	
Linear	0.742338	0.713886	−8%
Bicubic	0.778822	0.779032	0.04%

TABLE 4.6
Error measurements for first order fast marching.

Mesh size	L_1 error	L_2 error	L_∞ error
$20 \times 20 \times 20$	4.49×10^{-2}	5.39×10^{-2}	1.22×10^{-1}
$40 \times 40 \times 40$	2.47×10^{-2}	2.90×10^{-2}	8.04×10^{-2}
$80 \times 80 \times 80$	1.31×10^{-2}	1.53×10^{-2}	5.42×10^{-2}

N dimensions, we use an N-cubic interpolant

$$p(x_1, \ldots, x_N) = \sum_{k_1=0}^{3} \cdots \sum_{k_N=0}^{3} a_{k_1,\ldots,k_N} \prod_{\ell=1}^{N} x_\ell^{k_\ell}.$$

The iterative procedure given by (3.4)–(3.7) is still valid in this higher dimensional framework. We employed this method to reinitialize a sphere in $\mathbb{R}^3$ and used the second order fast marching method to get a second order accurate method including near the zero level set. Table 4.6 shows the errors from the original first order method from [12]. Table 4.7 shows the bicubic initialization method using second order fast marching. Table 4.8 shows the third order fast marching method results.

TABLE 4.7
Error measurements for bicubic initialization method, second order fast marching.

Mesh size	L_1 error	L_2 error	L_∞ error
$20 \times 20 \times 20$	5.00×10^{-3}	6.36×10^{-3}	3.91×10^{-2}
$40 \times 40 \times 40$	1.66×10^{-3}	2.05×10^{-3}	2.54×10^{-2}
$80 \times 80 \times 80$	4.84×10^{-4}	5.87×10^{-4}	1.39×10^{-2}

TABLE 4.8
Error measurements for bicubic initialization method, third order fast marching.

Mesh size	L_1 error	L_2 error	L_∞ error
$20 \times 20 \times 20$	2.15×10^{-3}	2.91×10^{-3}	2.16×10^{-2}
$40 \times 40 \times 40$	3.74×10^{-4}	6.25×10^{-4}	1.22×10^{-2}
$80 \times 80 \times 80$	6.62×10^{-5}	1.37×10^{-4}	6.12×10^{-3}

5. Conclusion. In this paper, we have presented a new interpolation method for initializing the fast marching method. This new method makes higher order accurate fast marching methods possible. Furthermore, the new method extends to higher dimensions much more readily than the old first order method. The new method is more accurate than existing methods and the additional cost is minimal. The additional accuracy also significantly improves the mass conservation of the level set method.

The additional accuracy has already proven useful in practice. We have used this new technique to do both reinitialization and front evolution using fast marching for fatigue crack growth modeling in three dimensions [16]. In this work, this higher order fast marching method is used in three different ways. First, reinitialization is used to construct the signed distance from the crack front. This is required to be as accurate as possible to obtain accurate crack front velocity values. Second, after the crack front velocities are obtained through a finite element calculation, the velocities are extended using the velocity extension method described above. Third, the crack front is advanced using the fast marching method and the extended velocity values. All three steps make use of the bicubic interpolation scheme. This approach is different than most uses of the level set method and is taken because the velocity calculation is by far the most expensive part of the process. By using the fast marching method instead of the level set method, arbitrarily large time steps can be taken to advance the front.

The application presented in [16] limits the crack to a single plane in a three-dimensional solid, and hence all the fast marching applications are done on a two-dimensional domain. When the crack front is allowed to move in arbitrary nonplanar directions, the fast marching method will be done in a three-dimensional domain. In that case, this higher order interpolation method will be critical because the refinement of the level set mesh will by necessity be much coarser than for two dimensions.

The bicubic interpolation presented here will produce at best a third order accurate fast marching method. Higher order fast marching methods are possible by using biquintic interpolation. The construction of the interpolant will require a larger stencil of nodes than shown in Figure 3.1 to approximate the additional derivative conditions, and the finite difference derivative approximations will require additional points to obtain sufficient accuracy. For initialization of the fast marching method, the iterative scheme (3.4)–(3.7) for computing the distance to the zero level set remains the same. The fast marching method would also have to be modified to be higher order by adding more terms and more switches.

There is a limit to the order of accuracy of the fast marching method. It is interesting to note that in one dimension, the fast marching method reduces to a backward difference formula method. These methods are known to be stable only for order less than seven. Convergence of the second and higher order fast marching methods is still not proven, and there is some question about their behavior in very dense meshes. The convergence of the higher order methods and their relationship to backward difference formulas is the subject of future research.

REFERENCES

[1] D. Adalsteinsson and J. Sethian, *A fast level set method for propagating interfaces*, J. Comput. Phys., 118 (1995), pp. 269–277.

[2] D. Adalsteinsson and J. Sethian, *The fast construction of extension velocities in level set methods*, J. Comput. Phys., 48 (1999), pp. 2–22.

[3] D. L. Chopp, *Computing minimal surfaces via level set curvature flow*, J. Comput. Phys., 106 (1993), pp. 77–91.

[4] E. Dijkstra, *A note on two problems in connection with graphs*, Numer. Math., 1 (1959), pp. 269–271.

[5] J. Glimm, J. W. Grove, X. L. Li, and D. C. Tan, *Robust computational algorithms for dynamic interface tracking in three dimensions*, SIAM J. Sci. Comput., 21 (2000), pp. 2240–2256.

[6] R. Kimmel and A. M. Bruckstein, *Shape offset via level sets*, Comput.-Aided Des., 25 (1993), pp. 154–162.

[7] R. Kimmel and J. Sethian, *Fast Marching Methods for Computing Distance Maps and Shortest Paths*, Tech. Report 669, CPAM, University of California, Berkeley, 1996.

[8] R. Malladi and J. Sethian, *An $O(N \log N)$ algorithm for shape modeling*, Proc. Natl. Acad. Sci. USA, 93 (1996), pp. 9389–9392.

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

[10] J. Sethian, *Fast marching level set methods for three-dimensional photolithography development*, in Proceedings of the SPIE 1996 International Symposium on Microlithography, Santa Clara, CA, 1996.

[11] J. Sethian, *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Material Science*, Cambridge University Press, Cambridge, UK, 1996.

[12] J. Sethian, *A marching level set method for monotonically advancing fronts*, Proc. Natl. Acad. Sci. USA, 93 (1996), pp. 1591–1595.

[13] J. Sethian, *Fast marching methods*, SIAM Rev., 41 (1999), pp. 199–235.

[14] J. Sethian and A. Popovici, *Three dimensional traveltimes computation using the fast marching method*, Geophysics, 64 (1999), pp. 516–523.

[15] M. Stolarska, D. L. Chopp, N. Möes, and T. Belytschko, *Modelling crack growth by level sets in the extended finite element method*, Internat. J. Numer. Methods Engrg., 51 (2001), pp. 943–960.

[16] N. Sukumar, D. L. Chopp, and B. Moran, *Extended finite element method and fast marching method for three-dimensional fatigue crack propagation*, J. Comput. Phys., submitted.

[17] M. Sussman and E. Fatemi, *An efficient interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow*, SIAM J. Sci. Comput., 20 (1999), pp. 1165–1191.

[18] M. Sussman, P. Smereka, and S. Osher, *A level set approach for computing solutions to incompressible two-phase flow*, J. Comput. Phys., 94 (1994), pp. 146–159.