Fast methods for the Eikonal and related Hamilton–Jacobi equations on unstructured meshes

J. A. Sethian* and A. Vladimirsky

Department of Mathematics, University of California, Berkeley, CA 94720

The Fast Marching Method is a numerical algorithm for solving the Eikonal equation on a rectangular orthogonal mesh in $O(M \log M)$ steps, where M is the total number of grid points. The scheme relies on an upwind finite difference approximation to the gradient and a resulting causality relationship that lends itself to a Dijkstra-like programming approach. In this paper, we discuss several extensions to this technique, including higher order versions on unstructured meshes in R^n and on manifolds and connections to more general static Hamilton–Jacobi equations.

Fast Marching Methods (1) are numerical algorithms for solving the nonlinear Eikonal equation

$$\|\nabla u(x)\| = F(x) \text{ in } \Omega, F(x) > 0 \qquad [1]$$

$$u = g(x) \text{ on } \Gamma$$

on a Cartesian mesh in $O(M \log M)$ steps, where M is the total number of grid points in the domain. Here, Ω is a domain in R^n, and the right-hand side $F(x) > 0$ is typically supplied as a known input to the equation, as is the boundary condition that u equals a known function $g(x)$ given along a prescribed curve or surface Γ in Ω. The technique hinges on using numerically consistent upwind finite difference approximations to the operators in the Eikonal equation that select the correct viscosity solution. The structure of this finite difference approximation to the gradient yields a resulting causality relationship that lends itself to an efficient programming approach.

The Fast Marching Method is connected to Huyghen's principle, which is a construction involving expanding wavefronts, and Dijkstra's method, which is an algorithm for computing smallest cost paths on a network. The viscosity solution to the Eikonal equation $\|\nabla u(x)\| = F(x)$ can be interpreted through Huyghen's principle in the following way: circular wavefronts are drawn at each point on the boundary, with the radius proportional to $F(x)$. The envelope of these wavefronts is then used to construct a new set of points, and the process is repeated; in the limit the Eikonal solution is obtained. The Fast Marching Method mimics this construction; a computational grid is used to carry the solution u, and an upwind, viscosity-satisfying finite difference scheme is used to approximate the wavefront.

The order in which the grid values produced through these finite difference approximations are obtained is reminiscent of Dijkstra's method, which is a depth-search technique for computing shortest paths on a network (2). Dijkstra's method keeps track of the "current smallest cost" for reaching a grid point and fans out along the network links to touch the adjacent grid points. The Fast Marching Method exploits the same idea in the context of an approximation to the underlying partial differential equation, rather than the discrete network links.

The Fast Marching Method is intertwined with some earlier work on front propagation, including work on curve and surface evolution in ref. 3, the suggestion to use schemes from hyperbolic conservation laws to approximate front motion in ref. 4, the introduction of level set methods by Osher and Sethian (5), and the narrow band level set method (6). In fact, the Fast Marching Method came in part from examining the limit of the narrow band method as the band was reduced to one grid cell. In this paper, we discuss several extensions to Fast Marching Methods, including higher order versions on Cartesian grids and unstructured meshes, on manifolds and in R^n; we also explore the connections to more general static Hamilton–Jacobi equations. For the sake of notational simplicity, the discussion below is limited to R^2, R^3, and two-dimensional manifolds; the results hold for arbitrary dimension.

Fast Marching Methods. We begin by finding a discretized version of the Eikonal Eq. **1** on the Cartesian grid. The easiest way to obtain such a discretization is to replace the gradient by the first-order approximation (see ref. 7):

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

where we have used standard finite difference notation $D_{ij}^{-x}u = u_{i,j} - u_{i-1,j}/h$ and $D_{ij}^{+x}u = u_{i+1,j} - u_{i,j}/h$. Here, u_{ij} is the value of u on a grid at the point ih,jh with grid spacing h. The forward and backwards operators D^{-y} and D^{+y} in the other coordinate direction are similar. This approximation is consistent and stable, and ensures that the viscosity solution is chosen.

One method for solving Eq. **2**, described by Rouy and Tourin (7), is through iteration, which in three dimensions leads to an $O(N^4)$ algorithm, where N is the number of points in each direction. The key to the Fast Marching Methods lies in the observation that the upwind approximation possesses a specific causality relationship. By "causality," we mean that the solution of Eq. **2** at each grid point depends only on the smaller adjacent values. Thus, we can systematically build the solution in the order of increasing values of u.

Suppose, at some time, the solution to the Eikonal equation is known at a set of points (denoted *Accepted* points). For every not-yet accepted grid point, such that it has an accepted neighbor, we compute a trial solution to the above quadratic Eq. **2** by using the given values for u at accepted points and values of ∞ at all other points. We now observe that the smallest of these trial solutions must be correct, because it depends only on accepted values that are themselves smaller. This causality relationship can be exploited to compute the solution efficiently and systematically as follows.

First, tag points in the initial conditions as *Accepted*. Then tag as *Considered* all points one grid point away and compute values at those points by solving the Eq. **2**. Finally, tag as *Far* all other grid points. Then the loop is:

1. Begin loop: Let *Trial* be the point in *Considered* with the smallest value of u.

APPLIED MATHEMATICS

2. Tag as *Considered* all neighbors of *Trial* that are not *Accepted*. If the neighbor is in *Far*, remove it from that set and add it to the set *Considered*.
3. Add *Trial* to *Accepted*; remove from *Considered*.
4. Recompute the values of u at all *Considered* neighbors of *Trial* by solving the piecewise quadratic equation according to Eq. **2**.

This procedure is the Fast Marching Method as described in ref. 1. Some early applications include photolithography (12), a comparison of this approach with volume-of-fluid techniques (13), and a fast algorithm for image segmentation (8); see also ref. 9 for a different Dijkstra-like algorithm, which obtains the viscosity solution through a control-theoretic discretization, which hinges on a causality relationship based on the optimality criterion.

The key to an efficient implementation of the Fast Marching Method lies in a fast way of locating the *Considered* grid point with the smallest value for u. An efficient scheme for doing so, discussed in detail in ref. 10, can be devised by using a min-heap structure, similar to what is done in Dijkstra's method. Given M elements in the heap, this structure allows us to change any element in the heap and reorder the heap in $O(\log M)$ steps. Thus, the computational efficiency of the total Fast Marching Method for the mesh with M total points is $O(M \log M)$; it takes M steps to touch each mesh point, where each step is $O(\log M)$, because the heap has to be reordered each time the values are changed.

Two extensions to the basic technique have added to the capabilities of this method. First, Kimmel and Sethian (11) moved the basic first-order scheme to unstructured acute triangulated meshes in R^2 and on two-dimensional surfaces and used this technique to compute shortest path geodesics on triangulated manifolds. Second, higher order versions of the Cartesian Fast Marching Method were developed by Sethian in ref. 10; these techniques replaced the basic first-order upwind operators by double-backwards second-order approximations to the first derivatives in each coordinate direction in such a way that a one-pass nature of the method was preserved. For details of these second-order extensions, see ref. 10.

In this paper, we build second-order methods for arbitrary unstructured meshes on manifolds and in R^n and show how certain static Hamilton–Jacobi equations in the plane can be recast as Eikonal equations on appropriately chosen manifolds.

Our interest in unstructured meshes stems from our intent to solve Hamilton–Jacobi equations on manifolds.

Unstructured Mesh Methods. *Derivative approximations.* Because there is no natural choice of the coordinate system for an unstructured mesh, we compute the gradient as a linear combination of n directional derivatives. For any grid point x, the difference approximation of the directional derivative is obviously available for each direction $x - x_r$, where x_r is any grid point adjacent to x.

Suppose the directional derivative approximations are available for the directions defined by the linearly independent row unit vectors $P_1, \ldots, P_n$. (If x is a vertex of simplex $xx_1 \ldots x_n$, then we will chose $P_r = x - x_r / \|x - x_r\|$ for $r = 1, \ldots, n$.) Consider the n by n nonsingular matrix P having vectors P_r ($r = 1, \ldots, n$) as its rows. Let $v_r(x)$ be the value of the directional derivative for the direction P_r evaluated at the point x. Assuming that the function u is differentiable at x, we have $P\nabla u(x) = v(x)$, where $v(x)$ is a column vector of $v_r(x)$ values. Combining this equation with the Eikonal equation, we can now write an equation for $v(x)$:

$$v(x)^T (PP^T)^{-1} v(x) = F^2(x). \qquad [3]$$
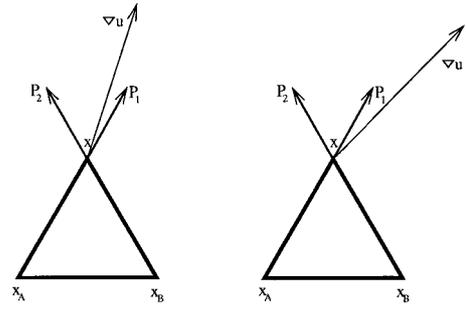
Fig. 1. Examples of acceptable (*Left*) and unacceptable (*Right*) approximations for $\nabla u(x)$. In the latter case, the upwinding requirement is not satisfied, and the update for the point x should be computed by using other simplexes.

The particular difference operator used to approximate v_r will depend on the type of the mesh and will determine the order of convergence of the numerical method. However, if the difference approximations depend linearly on $u(x)$ (i.e., on the value of the function at the point where the approximation is performed), then the resulting discretized version of the Eikonal equation will always be quadratic in terms of $u(x)$ regardless of the number of dimensions.

To obtain the discretized equation, we now replace each v_r with the corresponding difference approximation: $v_r \approx a_r u + b_r$, where b_r linearly depends on values of u (and possibly of ∇u for higher order schemes) at the grid points around x. For convenience let $Q = (PP^T)^{-1}$ and use $v \approx ua + b$. Then, the discretized version of Eq. **3** for the grid point x is the quadratic equation:

$$(a^T Q a)u^2 + (2a^T Q b)u + (b^T Q b) = F^2. \qquad [4]$$

Upwinding criteria. Suppose we are computing the value of $u(x)$ from some simplex having x as one of its vertices. If u is known at the other vertices of that simplex, then $u(x)$ can be computed by solving the quadratic Eq. **4**. Now we can strictly define our *upwinding criterion*: the computed value of $u(x)$ can be accepted, if the update is coming from within that simplex, i.e., only if the computed approximate ($-\nabla u(x)$) lies inside the simplex (see Fig. 1). This restriction is equivalent to requiring that all of the components of the vector $Qv \approx Q(ua + b)$ are nonnegative. If this condition is satisfied, we say that x is *updated* from this simplex and that this simplex is *defining* for x. The upwinding condition for an arbitrary triangle xx_1x_2 is equivalent to two inequalities: $(ua_1 + b_1) \geq (P_1 \cdot P_2^T)(ua_2 + b_2)$ and $(ua_2 + b_2) \geq (P_1 \cdot P_2^T)(ua_1 + b_1)$, where $P_r = x - x_r / \|x - x_r\|$. Thus, for the first-order difference approximation on the equilateral triangle xx_Ax_B, the upwinding requirement means that we will accept $u(x)$ only if $u(x) \geq 2u(x_B) - u(x_A)$ and $u(x) \geq 2u(x_A) - u(x_B)$.

It is possible that selecting a different simplex to define P will result in a different value of $u(x)$. To avoid the difficulty, we always choose the *defining simplex* that produces the smallest new value for $u(x)$; this choice corresponds to a scheme similar to Godunov's method on a Cartesian grid.

Extension to obtuse meshes. The original Fast Marching Method requires solving Eq. **4** for all of the simplexes adjacent to the grid point x. But what should be done if one or more of the vertices are not yet *Accepted*? Previous versions of the Fast Marching Method handled this difficulty by using the values at the *Considered* (as well as *Accepted*) points for updates and/or by computing updates based on the available partial information (values of u and/or ∇u only at the *Accepted* vertices of the simplex; ref. 10). Even though this technique is quite useful for "nice" acute triangulations, it can lead to numerical instability of the scheme when used for arbitrary unstructured meshes (see ref. 11). Computing updates based on partial information (i.e.,
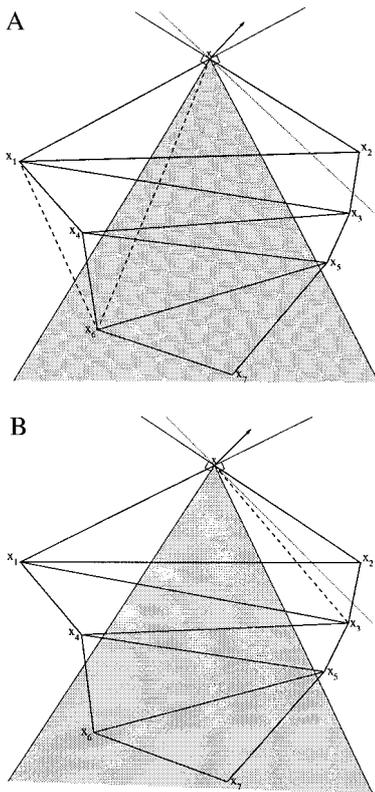
Fig. 2. Splitting strategies. (*a*) Constructing splitting section for an obtuse angle. (*b*) Modified construction of splitting section.

Fig. 3. Obtuse triangulated Fast Marching Method.

using some but not all vertices) can be particularly dangerous for higher accuracy methods, because the obtained estimate for ∇u might be very different from the true gradient.

The causality relationship requires $u(x)$ to depend only on the smaller values of u at the grid points adjacent to x, which means that if $u(x)$ is *updated* from simplex xx_1x_2 [i.e., $u(x)$ is produced by solving Eq. **4** corresponding to that simplex], then both $u(x_1)$ and $u(x_2)$ should be smaller than $u(x)$. This condition is necessary for using the Fast Marching Method, because that method accepts the values at the grid points in ascending order; thus, if either x_1 or x_2 has a larger value than x, then it will not be *Accepted* at the time when we are evaluating x and cannot be used for computing $u(x)$. For the Cartesian grid, the causality relationship is the result of using upwind difference approximations. Previously, we defined the upwinding criteria for an arbitrary triangulated mesh: if $u(x)$ is updated from simplex xx_1x_2, then the vector $-\nabla u(x)$ should point into that simplex. It is easy to show that the causality still follows from the upwinding requirement, provided the simplex has only acute angles.

However, if the triangulated mesh contains simplexes with obtuse angles, then the causality relationship might not hold even in the limit. Consider, for example, the front advancing with unit speed in the direction $(1,1)$ (suppose we are solving $\|\nabla u\| = 1$ in R^2 with the boundary condition $u = 0$ on the line $y = -x$). Consider the simplex xx_1x_2 from Fig. 2. The update for the grid point x should clearly be coming from this simplex, because $-\nabla u(x)$ points into it. However, it is also clear that $u(x_1) < u(x) < u(x_2)$; thus, $u(x_2)$ will not be available for computing $u(x)$.

One possible solution is to build locally numerical support at obtuse angles, as was suggested in ref. 11. For an obtuse angle x_1xx_2 (see Fig. 2*a*), consider its splitting section—an angle such that any ray inside it will split x_1xx_2 into two acute angles. Find the closest *Accepted* grid point in the splitting section and then
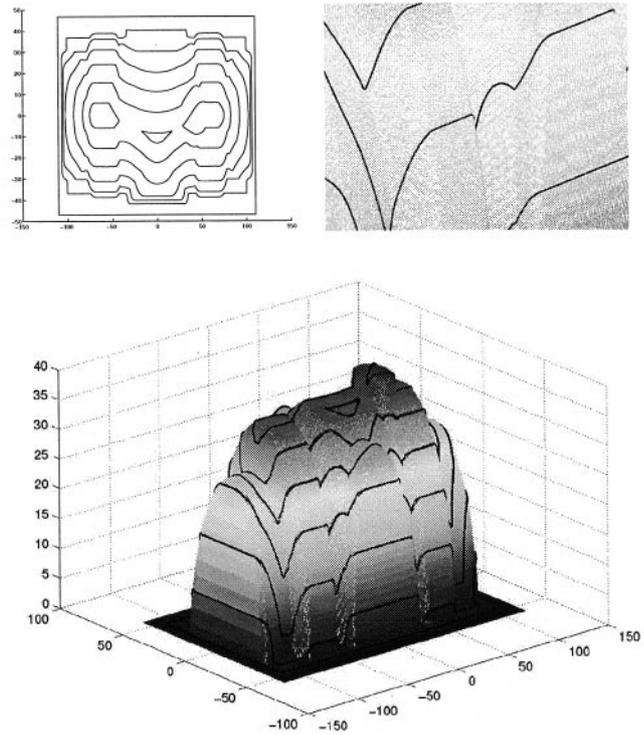
use that point as if it were adjacent to x. Thus, we would use simplex x_1xx_6 in Fig. 2*a*.

There are some disadvantages to this method. First of all, the implementation for higher dimensions is rather cumbersome. Second, implementing it for triangulated surfaces requires an additional step of "unfolding" (11). Third, the method is no longer confined to considering the grid points immediately adjacent to x, because we need to look back for an *Accepted* point in the splitting section. The upper bound for how far back we need to look in the splitting section is available but depends on the maximum angle of the triangulation—the wider angle corresponds to the narrower splitting section that is less likely to contain a grid point near x.

We observe that the splitting section method can be improved further by noting that the cause of the problem is not just the obtuse angle in the *defining simplex* but also the fact that some of the vertices of that simplex are not yet *Accepted*. Thus, it is not necessary to find an *Accepted* node in the splitting section; it is enough to find an *Accepted* node such that the resulting virtual simplex intersects the splitting section. This strategy often allows us to look back much less; thus, in Fig. 2*b*, for example, the grid point x_3 is the first *Accepted* point found such that xx_3 intersects the splitting section. Therefore, the virtual simplex x_1xx_3 will be used to compute the update for $u(x)$.

We note that our construction works equally well on manifolds. As an example, in Fig. 3, we show offsets equidistant from the bounding box on a manifold that represents a complex machine part; the triangulation is obtained by mapping a regular triangular mesh in the xy plane onto the surface, thereby creating a large number of obtuse and near-degenerate triangles, including some with angles bigger than 160°.

Higher Order Versions. We now create higher accuracy Fast Marching Methods by using higher order difference approximations for the directional derivatives. It would seem that such approximations can be used only if the solution u is sufficiently
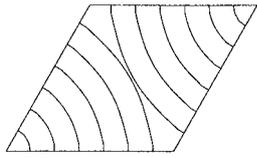
Fig. 4. First- and "second-order" computations of distance from two points. The shock line runs along the edges of simplexes.

Fig. 5. First- and "second-order" computations of distance from two points. The shock line is not aligned with the edges of simplexes.

smooth; nonetheless, the fact that at some points ∇u is undefined does not prevent us from using this approach: u is differentiable almost everywhere, and characteristics never emanate from the shocks, i.e., no information is created at the shock. However, the order of the difference approximation does not always correspond to the order of convergence of the method. Still such methods converge faster than the ones that use the first-order approximations (10). We further discuss the rate of convergence of the higher accuracy methods in *Numerical Tests*.

Cartesian higher order methods. A higher order Fast Marching Method on Cartesian grid was first presented by Sethian in ref. 10. Herein, we show that such methods can also be obtained from the directional derivative approximation perspective, as described in *Derivative approximations*.

For a Cartesian grid, the natural choice of the coordinate system will be aligned with the grid lines. Then, for any grid point x, a direction vector P_r is always equal (up to the sign) to one of the canonical basis vectors. Thus, for x inside the grid (i.e., away from the boundary), both points $x_{r,1} = x - hP_r$ and $x_{r,2} = x - 2hP_r$ are also present in the grid. Then we can use the well known second-order difference approximation for the directional derivative

$$v_r(x) \approx \frac{3u(x) - 4u(x_{r,1}) + u(x_{r,2})}{2h}. \qquad [5]$$

Using the notation introduced in *Derivative approximations*, we can write

$$a_r = \frac{3}{2h}; \quad b_r = \frac{-4u(x_{r,1}) + u(x_{r,2})}{2h}. \qquad [6]$$

Because this approximation is valid only inside the domain, we need to have the exact values of u for the grid points near the boundary to start the algorithm. If these values are not available, we can use the first-order Fast Marching Algorithm with much smaller mesh size to obtain the second-order accurate approximations of u at those points. Because this approximation is second-order only away from a singularity, we note that the exact (or second-order accurate) values of u are also needed for the grid points in the narrow band around any singularities at the boundary. Finally, we note that the same higher order difference approximations can be used even for non-Cartesian grids provided all of the grid points lie on the straight lines and are equidistant on those lines.

Higher order methods. Typically, we do not have orthogonal difference operators on an unstructured mesh. Fortunately, we

can still build higher order directional derivative approximations by using the gradient information at the grid points adjacent to x.

Consider a grid point x_r adjacent to x and the corresponding directional vector $P_r = x - x_r / \|x - x_r\|$. Supposing that both $u(x_r)$ and $\nabla u(x_r)$ are known, we can write a second-order approximation for the directional derivative

$$v_r(x) \approx 2 \frac{u(x) - u(x_r)}{\|x - x_r\|} - P_r \cdot \nabla u(x_r). \qquad [7]$$

Using the notation introduced in *Derivative approximations*, we can write

$$a_r = \frac{2}{\|x - x_r\|}; \quad b_r = \frac{-2u(x_r)}{\|x - x_r\|} - P_r \cdot \nabla u(x_r). \qquad [8]$$

We can also compute the second-order accurate approximation for the gradient at x, namely $\nabla u(x) = P^{-1}v \approx P^{-1}[u(x)a + b]$, provided $u(x)$ is known with at least the second-order accuracy as well. Thus, as the algorithm runs, we will store for each *Accepted* grid point the computed values of both u and ∇u to be used later when recomputing the *Considered* points.

Because this approach requires information about the gradient, we need to have the exact values of ∇u for the grid points on the boundary to start the algorithm. If these values are not available, we can use the first-order Fast Marching Algorithm with much denser mesh in the narrow band near the boundary to obtain the accurate approximations of ∇u at the points in that narrow band.

Finally, we note that an additional step of "gradient mapping" is required to use this higher order method on nonsmooth triangulated surfaces (J.A.S. and A.V., unpublished work).

Numerical tests. Next, we test the methods by finding the numerical solutions for the Eikonal equation $\|\nabla u(x)\| = 1$ in R^2 with different boundary conditions. The viscosity solution of this equation taken with zero boundary condition is the "distance from boundary" function. Two examples with shocks are considered: one where the shock line occurs along the grid lines and another where the shock line is not aligned with the grid. Both the L_2 and L_∞ errors are computed on the grid. Note that the rate of convergence in the L_2 norm generally corresponds to the order of the difference approximations. However, the rate of convergence in the L_∞ norm might be lower (depending on the location of shocks relative to the grid). This phenomenon is due to the fact that the higher order approximations are meaningful only where the solution is sufficiently smooth. Fortunately, viscosity solutions are differentiable almost everywhere, and no

Table 1. Errors for first- and "second order" schemes for example shown in Fig. 4.

Grid	L_2 error		L_∞ error	
	First-order	Second-order	First-order	Second-order
21^2	0.00818	0.00753	0.01500	0.02161
41^2	0.00432	0.00142	0.00785	0.00559
81^2	0.00213	0.00024	0.00386	0.00130
161^2	0.00108	0.00004	0.00195	0.00031

Table 2. Errors for first- and "second-order" schemes for example shown in Fig. 5

Grid	L_2 error		L_∞ error	
	First-order	Second-order	First-order	Second-order
21^2	0.00645	0.00533	0.01221	0.02034
41^2	0.00339	0.00117	0.00642	0.00644
81^2	0.00165	0.00022	0.00315	0.00203
161^2	0.00084	0.00005	0.00159	0.00071

Sethian and Vladimirsky
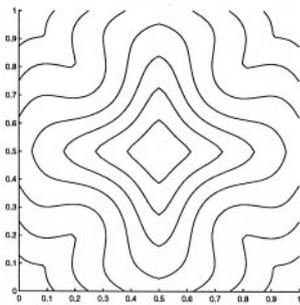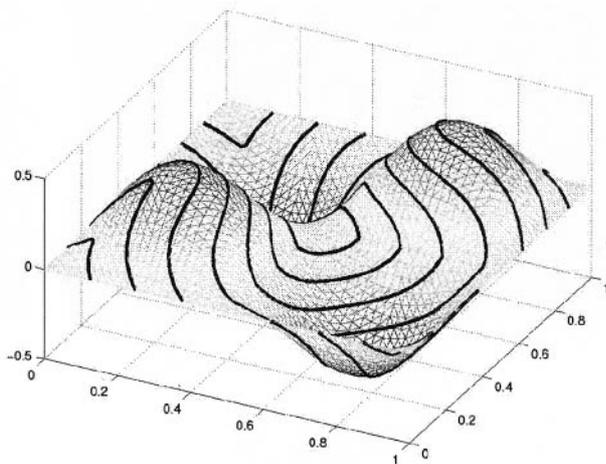
Fig. 6. Non-Eikonal static Hamilton–Jacobi equation.

information emanates from the shocks. The upwinding difference approximations used in our methods help the numerical solution to mimic this useful property of the true solution; thus, the L_2 norm convergence is not affected by the larger errors committed near the shocks.

The tests are performed on the parallelogram with vertices at $(0,0)$, $(1,0)$, $(0.5,\sqrt{3}/2)$, and $(1.5,\sqrt{3}/2)$. The exact values for the distance are used in the narrow band of radius 0.1 around the initial points to start the algorithm. We use the grid of equilateral triangles on this domain. The numerical results for a nonacute triangulation are very similar.

We first use the Fast Marching Method to compute the distance from two vertices: $(0,0)$ and $(1.5,\sqrt{3}/2)$. The shock line runs along the edges of simplexes (along the shorter diagonal of the parallelogram; Fig. 4). Table 1 shows the errors under mesh refinement.

We now compute the minimal distance from the other two vertices: $(1,0)$ and $(1/2,\sqrt{3}/2)$. In this case, the shock line is not aligned with the grid lines (edges of simplexes), and it runs along the longer diagonal (Fig. 5). Table 2 shows the errors under mesh refinement. The L_2 error is still second-order convergent for the second-order scheme, whereas its L_∞ error is lower order, although still much better than the L_∞ error of the first-order

scheme. This effect is to be expected; because of the grid alignment, the approximation is sometimes performed across the shock lines, which leads to the first-order errors there.

Eikonal Equations on Surfaces and More General Static Hamilton–Jacobi Equations. Suppose we are given a graph of a function $z = f(x,y)$ and attempt to solve the Eikonal equation $\|\nabla u\| = F(x,y)$ on that manifold. To be clear, $1/F(x,y)$ gives the speed in the direction normal to the level line $u = constant$ on the manifold $z = f(x,y)$. Projecting down onto the xy plane, this Eikonal equation translates into a particular static Hamilton–Jacobi equation on the plane. We now use this argument in reverse as follows. Consider any static Hamilton–Jacobi equation in the xy plane of the form

$$a(x,y)u_x^2 + b(x,y)u_xu_y + c(x,y)u_y^2 = g(x,y) \qquad [9]$$

together with boundary conditions for u. Now suppose we can find functions $p(x,y)$, $q(x,y)$, and $F(x,y)$ such that $p_y = q_x, F(x,y) > 0$, and

$$\begin{array}{ll} a(x,y) = (1 + q^2) & b(x,y) = -2pq \\ c(x,y) = (1 + p^2) & g(x,y) = (1 + p^2 + q^2)F^2(x,y). \end{array} \qquad [10]$$

It can be then shown (see ref. 10) that the solution of Eq. **9** can be obtained by solving the Eikonal equation $\|\nabla u\| = F(x,y)$ on the manifold $z = f(x,y)$ where $f_x = p$ and $f_y = q$. Thus, for any static Hamilton–Jacobi equation of the form given by the Eq. **9**, if we can find functions p and q satisfying the above, then we can *construct* the surface $z = f(x,y)$, approximate it with a triangulated mesh, and then solve the straightforward Eikonal problem on the manifold.

As an example, we consider the following equation.

$$\frac{\left[\begin{array}{l} [1 + \gamma \sin^2(2\pi x)\cos^2(2\pi y)]u_x^2 \\ - \dfrac{\gamma}{2} \sin(4\pi x)\sin(4\pi y)u_xu_y \\ + [1 + \gamma \cos^2(2\pi x)\sin^2(2\pi y)]u_y^2 \end{array}\right]}{[1 + \gamma \cos^2(2\pi x)\sin^2(2\pi y) + \gamma \sin^2(2\pi x)\cos^2(2\pi y)]} = 1, \qquad [11]$$

where $\gamma = (0.9\pi)^2$, $x,y \in [0,1]$, and the boundary condition is $u(0.5,0.5) = 0$. We can find functions $p = 0.9\pi\cos(2\pi x)\sin(2\pi y)$ and $q = 0.9\pi\sin(2\pi x)\cos(2\pi y)$, which satisfy our compatibility requirements, and then solve the Eikonal equation $\|\nabla u\| = 1$ on the surface $f(x,y) = 0.45\sin(2\pi x)\sin(2\pi y)$. Fig. 6 shows the evolving front on the surface and the solution to the original problem on the plane.

We have presented techniques for computing certain Hamilton–Jacobi equations on unstructured meshes. Further discussion may be found elsewhere (J.A.S. and A.V., unpublished work).

1. Sethian, J. A. (1996) *Proc. Natl. Acad. Sci. USA* **93,** 1591–1595.
2. Dijkstra, E. W. (1959) *Numer. Math.* **1,** 269–271.
3. Sethian, J. A. (1985) *Commun. Math. Phys.* **101,** 487–499.
4. Sethian, J. A. (1987) in *Variational Methods for Free Surface Interfaces*, eds. Concus, P. & Finn, R. (Springer, New York).
5. Osher, S. & Sethian, J. A. (1988) *J. Comp. Phys.* **79,** 12–49.
6. Adalsteinsson, D. & Sethian, J. A. (1995) *J. Comp. Phys.* **118,** 2, 269–277.
7. Rouy, E. & Tourin, A. (1992) *SIAM J. Numer. Anal.* **29,** 867–884.
8. Malladi, R. & Sethian, J. A. (1996) *Proc. Natl. Acad. Sci. USA* **93,** 9389–9392.
9. Tsitsiklis, J. N. (1995) *IEEE Trans. Autom. Control* **40,** 1528–1538.
10. Sethian, J. A. (1999) *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision and Materials Sciences*,(Cambridge Univ. Press, Cambridge, U.K.).
11. Kimmel, R. & Sethian, J. A. (1998) *Proc. Natl. Acad. Sci. USA* **95,** 8341–8435.
12. Sethian, J. A. (1996) in *Proceedings of the International Symposium on Microlithography* (SPIE, Santa Clara, CA), vol. 2726.
13. Helmsen, J., Puchett, E. G., Colella, P. & Dorr, M. (1996) in *Proceedings of the International Symposium on Microlithography* (SPIE, Santa Clara, CA), vol. 2726.

APPLIED MATHEMATICS