SIAM J. NUMER. ANAL.
Vol. 45, No. 1, pp. 83–107

FAST SWEEPING METHODS FOR EIKONAL EQUATIONS ON TRIANGULAR MESHES*

JIANLIANG QIAN†, YONG-TAO ZHANG‡, AND HONG-KAI ZHAO‡

Abstract. The original fast sweeping method, which is an efficient iterative method for stationary Hamilton–Jacobi equations, relies on natural ordering provided by a rectangular mesh. We propose novel ordering strategies so that the fast sweeping method can be extended efficiently and easily to any unstructured mesh. To that end we introduce multiple reference points and order all the nodes according to their l^p-metrics to those reference points. We show that these orderings satisfy the two most important properties underlying the fast sweeping method: (1) these orderings can cover all directions of information propagating efficiently; (2) any characteristic can be decomposed into a finite number of pieces and each piece can be covered by one of the orderings. We prove the convergence of the new algorithm. The computational complexity of the algorithm is nearly optimal in the sense that the total computational cost consists of $O(M)$ flops for iteration steps and $O(M\log M)$ flops for sorting at the predetermined initialization step which can be efficiently optimized by adopting a linear time sorting method, where M is the total number of mesh points. Extensive numerical examples demonstrate that the new algorithm converges in a finite number of iterations independent of mesh size.

Key words. eikonal equations, fast sweeping, Hamilton–Jacobi, viscosity solution

AMS subject classifications. Primary, 54C40, 14E20; Secondary, 46E25, 20C20

DOI. 10.1137/050627083

1. Introduction.

The eikonal equation in its simplest form says that the magnitude of the gradient of the eikonal is constant: $|\nabla T| = 1$, where T is the so-called eikonal. Because it appears in a variety of applications, it is essential to develop fast and efficient numerical methods to solve such an equation. In this work, we design a class of fast sweeping methods on triangulated domains for an eikonal equation of the following form:

$$\begin{cases} |\nabla T(\mathbf{x})| = f(\mathbf{x}), & \mathbf{x} \in \Omega \setminus \Gamma, \\ T(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Gamma \subset \Omega, \end{cases} \tag{1.1}$$

where $f(\mathbf{x})$ is a nonnegative function, Ω is an open, bounded polygonal domain in R^d, and Γ is a subset of Ω.

Two key points in designing an efficient numerical algorithm for solving such a nonlinear boundary value problem of hyperbolic type are (1) a numerical discretization that is both consistent with the causality of the PDE and able to deal with singularities in the solution gradient, and (2) a fast algorithm to solve the resulting large nonlinear system of equations. There are usually two types of methods for solving the nonlinear system: time marching methods and direct methods. Time marching methods add to the equation a pseudo–time variable which transforms the problem into

*Received by the editors March 17, 2005; accepted for publication (in revised form) July 6, 2006; published electronically January 12, 2007.

http://www.siam.org/journals/sinum/45-1/62708.html

†Department of Mathematics and Statistics, Wichita State University, Wichita, KS 67260-0033 (qian@math.wichita.edu, qian@math.ucla.edu). The research of this author was supported by NSF grant DMS-0542174.

‡Department of Mathematics, University of California, Irvine, CA 92697-3875 (zyt@math.uci.edu, zhao@math.uci.edu). The research of the third author was partially supported by ONR grant N00014-02-1-0090, DARPA grant N00014-02-1-0603, and the Sloan Foundation Fellowship.

a time dependent one and evolve the solution to the steady state. Due to the finite speed of propagation and the Courant–Friedrichs–Lewy (CFL) condition for stability, many iterations are needed to reach the steady state solution. The last two decades have witnessed much effort towards solving the eikonal equation directly: upwinding schemes [32, 31], dynamic programming sweeping methods [27], Jacobi iterations [26], semi-Lagrangian schemes [8], fast marching-type methods [30, 10, 28, 13], down-n-out approaches [7], wavefront expanding methods [23], adaptive upwinding methods [19, 21], fast sweeping methods [2, 37, 29, 35, 12, 34, 11, 36, 33]; see also the references therein. Accuracy of numerical solutions is determined by the discretization scheme. For example, if a first-order monotone scheme is used, in general only the $h^{1/2}$ convergence rate can be shown [6] and the $h \log h$ convergence rate is optimal for the eikonal equation [35].

Among all these methods, both the fast marching method and the fast sweeping method are designed to solve the nonlinear discretized system directly and efficiently by exploiting causality of the underlying PDE. In terms of complexity, the fast marching method [30, 10, 28, 13] has the complexity of $O(M \log M)$, where M is the total number of mesh points and the $\log M$ factor comes from the heapsort algorithm needed for sorting out the causality order at each step, while the fast sweeping method has the complexity of $O(M)$, where the constant in O depends on the equation, and this was proved in [35] for eikonal equations on rectangular grids. For a particular problem on a fixed grid, one method could be faster than the other. When the grid is more refined the fast sweeping method will be faster eventually. In [9], concrete and detailed comparisons are presented for various numerical examples. In terms of accuracy there is no difference since they are two different ways of solving the same nonlinear discretized equation. The main difference between these two methods lies in the use of causality. The fast marching method enforces the causality sequentially and on the fly during each update step; that is why a heapsort algorithm is needed to order all possible candidates and pick up the correct one by the causality at each step; once a point is accepted it cannot be revisited and its value cannot be changed afterwards. On the other hand the fast sweeping method is an iterative method of Gauss–Seidel type which is extremely simple to implement; such a simple iterative method for a nonlinear problem is able to achieve an optimal complexity because it can capture the causality of the PDE in a parallel way, as shown in [35]. Since it is an iterative method by nature the fast sweeping method is applicable to other situations such as higher order schemes with ease [34, 33], nonconvex Hamiltonians [12], and parallel implementation [36].

On the other hand, most of these methods are based on rectangular meshes. However, it is important to design fast methods on triangulated meshes as well. For example, in seismics a subsurface velocity model usually consists of several irregular interfaces, and in robotic path planning an obstacle may have an irregular boundary. Thus, for applications involving irregular boundaries or interfaces, it is much desired to triangulate a computational domain into irregular meshes to fit with boundaries or interfaces. Kimmel and Sethian [13] extended the fast marching method to triangulated domains to compute geodesics on manifolds.

In this work, we extend the fast sweeping method to triangulated domains by introducing novel ordering processes into the sweeping strategy. The resulting method is proved to be convergent, and numerical examples demonstrate that the method converges in a finite number of iterations independent of mesh size. The computational complexity of the new algorithm is nearly optimal in the sense that the total

computational cost consists of $O(M)$ flops for iteration steps and $O(M\log M)$ flops for sorting at the predetermined initialization step, which can be efficiently optimized by adopting a linear time sorting method.

An essential property of the eikonal equation is that it is hyperbolic, and a stable scheme must look for information by following characteristics in an upwind fashion, which is equivalent to the simple causality for the eikonal equation in that its solution is always increasing (or decreasing) along a characteristic. To satisfy such a property, it is crucial for a scheme of computing viscosity solutions to be based on a monotone numerical Hamiltonian [1, 17]. Once we have in place such a discretization for eikonal equations, the problem reduces to one of solving the resulting nonlinear system efficiently; the fast sweeping method is designed to do exactly that. The original fast sweeping method was inspired by the work in [2]. The fast sweeping method uses Gauss–Seidel iterations with alternate sweeping orderings to solve the nonlinear system. The fact that the iterative algorithm for a nonlinear system can converge in a finite number of iterations independent of mesh size is quite remarkable; even for a linear system, such as the discretized system for the Laplace equation, this is not true.

The crucial idea underlying the fast sweeping method is the following [35]: all directions of characteristics can be divided into a finite number of groups; any characteristic can be decomposed into a finite number of pieces that belong to one of the above groups; there are systematic orderings that can follow the causality of each group of directions simultaneously.

On a rectangular grid there are natural orderings of all grid points. For example, in the two-dimensional (2-D) case, all directions of the characteristics can be partitioned into four groups, up-right, up-left, down-right, and down-left, and it is very natural to order all the nodes according to their indexes in ascent or descent orders [2, 37, 29, 11, 35, 12, 34], which yields four possible orderings to cover all those four directions of characteristics.

However, on an unstructured mesh, only local connection of the nodes is available and natural ordering no longer exists. To overcome these difficulties we propose general ordering strategies by introducing multiple reference points and ordering all the nodes according to their l^p-distances to those reference points. For example, information is propagated as plane waves in different directions when the l^1-metric is used or as spherical waves with different centers when the l^2-metric is used. We show that these orderings satisfy the key properties essential for the fast sweeping method to converge and numerically demonstrate that the fast sweeping method converges in a finite number of iterations independent of mesh size. Although it may still cost $O(M\log M)$ by a comparison-based sorting method, the ordering step in our algorithm may be made to be $O(M)$ by a linear time sorting method since we know the distribution of nodes at the initial step. For example, the radix sorting method [4] may be used for such a purpose. Moreover this initial ordering is done for a fixed mesh once and for all. This is different from other methods based on heap sorting to maintain a dynamic data structure. Therefore the methods proposed here are very efficient and extremely easy to write in any number of dimensions.

The rest of the paper is organized as follows. In section 2, we construct local solvers at each node on a triangulated mesh, propose novel ordering strategies, and detail fast sweeping algorithms. In section 3, we analyze the new algorithm and prove convergence results. In section 4, we present various numerical examples to illustrate the efficiency and the accuracy of the new method. We conclude the paper in section 5.

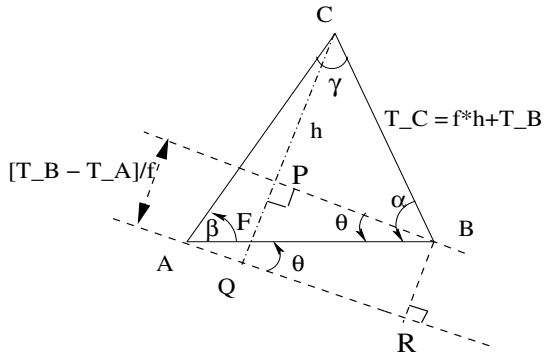


FIG. 2.1. *Update the value at C in a triangle when causality is satisfied.*

2. Fast sweeping methods on unstructured meshes.

2.1. 2-D local solvers. Take $d = 2$ in (1.1):

$$\begin{cases} \sqrt{T_x^2 + T_y^2} = f(x,y), & (x,y) \in \Omega \subset R^2, \\ T(x,y) = g(x,y), & (x,y) \in \Gamma \subset \Omega, \end{cases} \tag{2.1}$$

where $f(\mathbf{x})$ is a nonnegative function, Ω is an open, bounded polygonal domain in R^d, and Γ is a subset of Ω.

We consider a triangulation $\mathcal{T}_h$ of Ω which consists of nonoverlapping, nonempty, and closed triangles $\mathcal{T}$, with diameter $h_{\mathcal{T}}$, such that $\bar{\Omega} = \cup_{\mathcal{T} \in \mathcal{T}_h} \mathcal{T}$. We assume that $\mathcal{T}_h$ satisfies the following conditions:

- No more than μ triangles have a common vertex; $h = \sup_{\mathcal{T} \in \mathcal{T}_h} h_{\mathcal{T}} < 1$.
- $\mathcal{T}_h$ is regular; there exists a constant ω_0 independent of h such that if $\rho_{\mathcal{T}}$ is the diameter of the largest ball $B \subset \mathcal{T}$, then for all $\mathcal{T} \in \mathcal{T}_h$, $h_{\mathcal{T}} \le \omega_0 \rho_{\mathcal{T}}$.

For a given triangle $\triangle ABC$, we denote $\angle A = \beta$, $\angle B = \alpha$, and $\angle C = \gamma$; $\overline{AB} = c$, $\overline{AC} = b$, and $\overline{BC} = a$ are the lengths of the edges AB, AC, and BC, respectively.

During the solution process we need a local solver at vertex C for each triangle; see Figure 2.1. Given the values T_A and T_B at A and B of triangle $\triangle ABC$, we want to calculate the value T_C at C.

To make the description specific, we introduce the definition of causality.

DEFINITION 2.1. *Under the above regular triangulation we consider a local scheme based on piecewise linear reconstructions. By the causality condition of isotropic wave propagation for updating the travel-time at the node C from travel-times T_A and T_B, we mean that the ray which is orthogonal to the wavefront and passes through C must fall inside the triangle $\triangle ABC$.*

We notice that in isotropic wave propagation the ray direction is the same as the gradient direction of the travel-time field and thus it is the same as the outward normal of the wavefront.

First we assume that $\triangle ABC$ is acute. To construct a first-order scheme we determine a planar wavefront from the known values T_A and T_B. Suppose that the angle is θ between the incoming wavefront and the edge AB.

Without loss of generality, we further assume that $T_B \ge T_A$. If T_C is determined by both T_A and T_B, then by the Huygens principle the wavefront must first pass through the vertex A, then B, and finally C. To guarantee this, the following conditions must be satisfied:

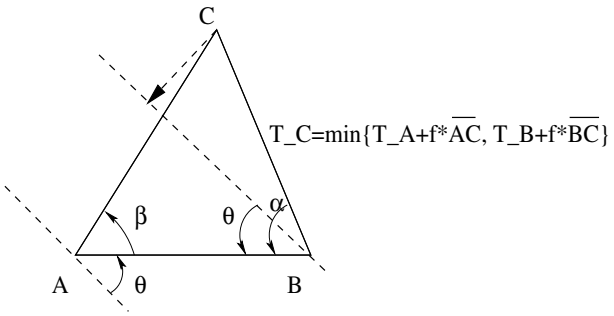


FIG. 2.2. *Update the value at vertex C in a triangle when causality is not satisfied.*

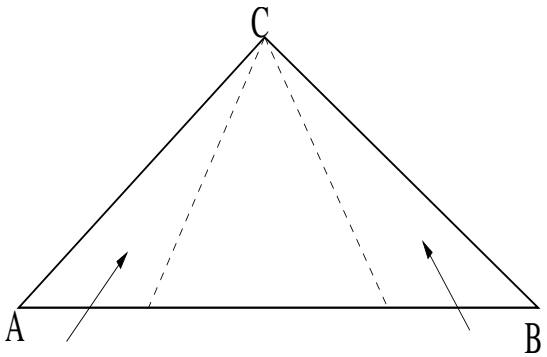


FIG. 2.3. *C and its obtuse triangle.*

- $[T_B - T_A]/f_C \le \overline{AB} = c$; i.e., it is possible for the wavefront to propagate from A to B with the given speed, where f_C is the value of $f(C)$, the inverse of the speed at C.
- $\theta \le \alpha$ so that the wavefront passes through B first rather than C.
- $\theta + \beta < \frac{\pi}{2}$; otherwise the causality is violated since the vertical line from C to the wavefront does *not* fall inside the triangle; see Figure 2.2.

If all n triangles $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$ around C are acute, the wavefront can be captured well in one of these triangles, no matter which direction the wave comes from. However, if one of the triangles is obtuse and the wavefront comes in just from this obtuse angle, then the situation is different; there are two possible cases: (i) if the normal of the wavefront is contained between those two dotted lines in Figure 2.3, then the value at C can be updated using values at A and B even though the accuracy will be degraded; (ii) otherwise, the value at C cannot be updated by A and B correctly [25]. These will be shown in numerical examples in section 4.

In order to treat obtuse triangles, we adopt the strategy used in [25]. As illustrated in Figure 2.4, if $\angle C$ is obtuse, then we connect C to a vertex D of a neighboring triangle to cut the obtuse angle into two smaller angles. If these two angles are both acute, then we are done, as shown in Figure 2.4(a); otherwise, if one of the smaller angles is still obtuse, then we keep connecting C to the vertexes of the neighboring triangles of the next level until all new angles at C are acute, as shown in Figure 2.4(b). All these added edges are "virtual"; i.e., they exist only when the value at C is updated. Because such a treatment depends on a given mesh, we only need to do that once before the iteration in the algorithm begins; the resulting algorithm is

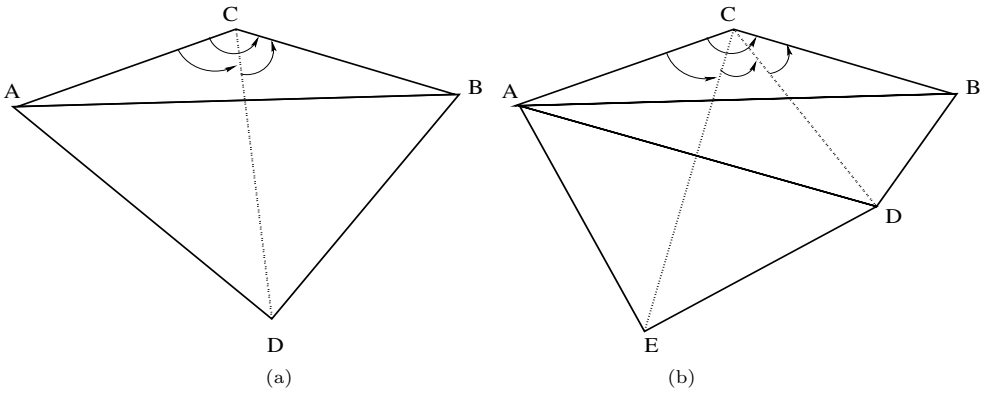


FIG. 2.4. *A strategy to treat obtuse angles.*

simple with almost no extra computational cost, as shown by numerical examples in section 4. This construction is different from the one used in [13].

We first give a geometric version of our local solvers.

A 2-D LOCAL SOLVER (Version 1: given $T_A \le T_B$, determine $T_C = T_C(T_A, T_B)$).

1. If $[T_B - T_A] \le c\, f_C$, then

$$\theta = \arcsin\left(\frac{[T_B - T_A]}{c\, f_C}\right);$$

   (a) if $\max(0, \alpha - \frac{\pi}{2}) \le \theta \le \frac{\pi}{2} - \beta$, then

$$h = \overline{CP} = a\sin(\alpha - \theta); T_C = \min\{T_C, h\, f_C + T_B\};$$

   (b) else

$$T_C = \min\{T_C, T_A + b\, f_C, T_B + a\, f_C\};$$

2. else

$$T_C = \min\{T_C, T_A + b\, f_C, T_B + a\, f_C\}.$$

The angle condition,

$$\max\left(0, \alpha - \frac{\pi}{2}\right) \le \theta \le \frac{\pi}{2} - \beta,$$

can be obtained in the following way:

1. If $\beta > \frac{\pi}{2}$, then the causality condition is not valid.
2. If $\beta < \frac{\pi}{2}$, then we must have $\theta \le \frac{\pi}{2} - \beta$; otherwise, the causality is violated since the vertical line from C to the wavefront does *not* fall inside the triangle. Furthermore,
   (a) from this condition we can directly deduce that $\alpha \ge \theta$, since $\angle C = \gamma < \frac{\pi}{2}$ by construction;
   (b) if $\alpha \ge \frac{\pi}{2}$, then we must have $\alpha - \theta \le \frac{\pi}{2}$ so that the ray from C reaching the wavefront is located inside the triangle.

The following algorithm unifies all the cases into one.

A 2-D LOCAL SOLVER (Version 2: given T_A and T_B, determine $T_C = T_C(T_A, T_B)$).

1. If $|T_B - T_A| \le c\, f_C$, then

$$\theta = \arcsin\left(\frac{[T_B - T_A]}{c\, f_C}\right);$$

(a) if $\max(0, \alpha - \frac{\pi}{2}) \le \theta \le \frac{\pi}{2} - \beta$ or $\alpha - \frac{\pi}{2} \le \theta \le \min(0, \frac{\pi}{2} - \beta)$, then

$$h = \overline{CP} = a\, \sin(\alpha - \theta); H = \overline{CQ} = b\, \sin(\beta + \theta);$$

$$T_C = \min\{T_C, 0.5\, (h\, f_C + T_B) + 0.5\, (H\, f_C + T_A)\};$$

(b) else

$$T_C = \min\{T_C, T_A + b\, f_C, T_B + a\, f_C\};$$

2. else

$$T_C = \min\{T_C, T_A + b\, f_C, T_B + a\, f_C\}.$$

In the special case that a given mesh is rectangular and $\alpha = \beta = \frac{\pi}{4}$, it is straightforward to verify that the above local solver reduces to the one given in [35]. Therefore, the local solver is consistent with the one on rectangular meshes.

If a triangle is acute, then the angle conditions in Version 2 reduce to one condition:

$$\alpha - \frac{\pi}{2} \le \theta \le \frac{\pi}{2} - \beta;$$

otherwise, the two angle conditions cannot be combined into one, since there are gaps corresponding to one of the angles α or β being obtuse. See Figures 2.1 and 2.2 for illustrations.

We emphasize that both updating algorithms require that $\angle C = \gamma < \frac{\pi}{2}$, but one of the other two angles may be obtuse.

2.2. A 3-D local solver. A local solver in three dimensions can be derived similarly. Take $d = 3$ in (1.1):

$$\begin{cases} \sqrt{T_x^2 + T_y^2 + T_z^2} = f(x, y, z), & (x, y, z) \in \Omega \subset R^3, \\ T(x, y, z) = g(x, y, z), & (x, y, z) \in \Gamma \subset \Omega. \end{cases} \tag{2.2}$$

Equation (2.2) is solved in the domain Ω, which has a triangulation $\mathcal{T}_h$ consisting of tetrahedrons. We consider every vertex and all tetrahedrons which are associated to this vertex. Again the question reduces to one of calculating the numerical solution at the current central vertex for each tetrahedron; see Figure 2.5.

Given the values T_A, T_B, and T_C at A, B, and C of the tetrahedron $ABCD$, we need to calculate the value T_D at the current central vertex D. The key is to determine the normal direction $\vec{\mathbf{n}}$ of the wavefront and determine whether the causality condition is satisfied or not. Analogous to Definition 2.1, the ray which has direction $\vec{\mathbf{n}}$ and passes through D must fall inside the tetrahedron $ABCD$ so as to satisfy the causality condition. To check the causality condition numerically, we first compute the coordinates of the point E at which the ray passing through D with direction $\vec{\mathbf{n}}$ intersects the plane spanned by A, B, and C; afterwards, we check to see whether E is inside $\triangle ABC$ or not.

Without loss of generality, we assume that $T_A = \min\{T_A, T_B, T_C\}$.

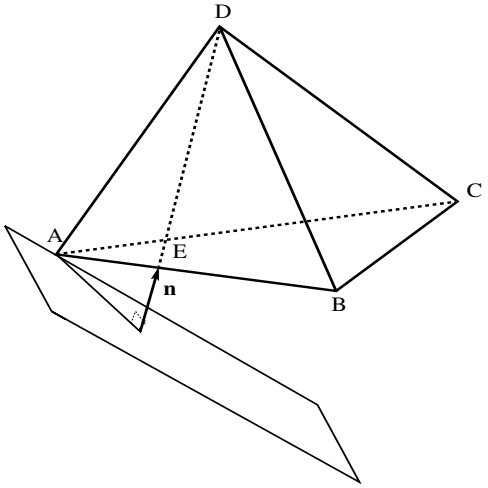


FIG. 2.5. *A 3-D local solver.*

A 3-D LOCAL SOLVER (given T_A, T_B, and T_C, determine $T_D = T_D(T_A, T_B, T_C)$).

1. If $[T_B - T_A] \le \overline{AB} \cdot f_D$ and $[T_C - T_A] \le \overline{AC} \cdot f_D$, then we solve the quadratic equation for the normal direction $\vec{\mathbf{n}}$ of the wavefront:

$$\begin{cases} \overrightarrow{AB} \cdot \vec{\mathbf{n}} = [T_B - T_A]/f_D, \\ \overrightarrow{AC} \cdot \vec{\mathbf{n}} = [T_C - T_A]/f_D, \\ \quad |\vec{\mathbf{n}}| = 1; \end{cases} \tag{2.3}$$

   (a) if there exist solutions $\vec{\mathbf{n}}^{(i)}, i = 1, 2$, for the quadratic equation (2.3) and the area $|\triangle EAB| + |\triangle EAC| + |\triangle EBC| = |\triangle ABC|$ for an $\vec{\mathbf{n}}^{(i)}$, then

$$T_D = \min\{T_D, T_A + (|\overrightarrow{AD} \cdot \vec{\mathbf{n}}^{(i)}|) \cdot f_D\};$$

   (b) else, apply the 2-D local solver on surfaces $\triangle ABD, \triangle ACD$, and $\triangle BCD$ and take the minimal one;

2. else, apply the 2-D local solver on surfaces $\triangle ABD, \triangle ACD$, and $\triangle BCD$ and take the minimal one.

2.3. Sweeping orders and a complete algorithm. An essential ingredient for making the fast sweeping method [35] successful is a systematic ordering that covers all directions of characteristics efficiently. With a causality preserving discretization in place, information along characteristics of certain directions is captured simultaneously in each sweeping ordering. Moreover, once the solution at a node gets its correct value, i.e., the smallest possible value, it will not change in later iterations. There are natural orderings on rectangular meshes. For example, in 2-D cases [35], all directions can be divided into four groups, up-right, up-left, down-left, and down-right, which can be covered by the orderings $i = 1 : I, j = 1 : J$; $i = 1 : I, j = J : 1$; $i = I : 1, j = 1 : J$; $i = I : 1, j = J : 1$, respectively, where i and j are the running indexes in the x- and y-directions, respectively. However, such natural orderings no longer exist on an unstructured mesh.

To devise efficient fast sweeping methods on unstructured meshes, we propose systematic orderings by introducing multiple reference points and sorting all the nodes according to their l^p-distances to each individual reference point. In this paper we

focus on $p = 1$ and 2 and give explicit geometric interpretation. The argument works for all other p's.

The l^p-metric for a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in R^n$ is defined as $|\mathbf{x}|_p = (\sum_{j=1}^{n} |x_j|^p)^{1/p}$. Without abuse of notation we also use $|\mathbf{x}|$ to denote the 2-norm of a vector $\mathbf{x}$. For example, in two dimensions, we first fix a reference point $\mathbf{x}_{\text{ref}}$; if we sweep through all nodes according to $|\mathbf{x} - \mathbf{x}_{\text{ref}}|_1$ in the ascent (or descent) order, then the sweeping wavefront is an outgoing (or incoming) plane wave since the unit ball of the l^1-metric is a tilted square. If we use $|\mathbf{x} - \mathbf{x}_{\text{ref}}|_2$ to order all nodes, then the sweeping wavefront is an outgoing (or incoming) spherical wave.

Next we address the following questions:

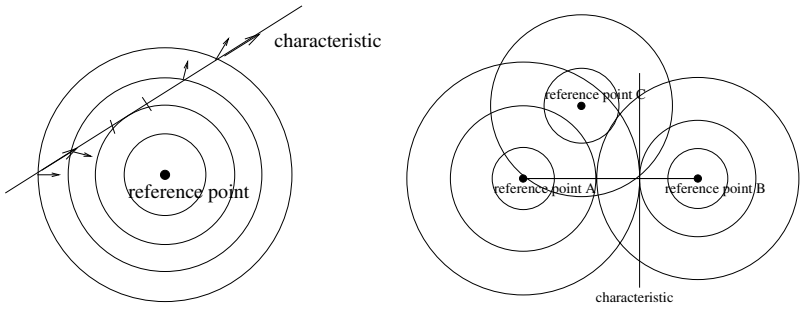
1. How many references points are needed in a systematic ordering that can cover all directions of information propagating?
2. How many iterations are needed for the algorithm to converge?

To address the first question, we have to understand the directional relation between a sweeping wavefront and a characteristic. In the continuous case the following is a basic fact: if the propagating direction of the sweeping wavefront forms an acute angle with the direction of the characteristic, then the causality along this characteristic can be captured in this ordering. As illustrated in Figure 2.6, if we use the l^2-metric, i.e., with a spherical sweeping wavefront, a straight characteristic in any direction can be partitioned into two pieces by the tangent point to a particular spherical sweeping wavefront, and each piece forms an acute angle to the outgoing or incoming sweeping wavefront. If all characteristics are straight lines, which is the case when the right-hand side of the eikonal equation is constant, we cover almost all characteristics by sweeping all nodes according to the l^2-distance to a single reference point in both ascent and descent orders alternately. However, for all characteristics at the tangent point, the normal of the sweeping wavefront is orthogonal to the direction of characteristics. So information will not propagate across the tangent point from one piece to other pieces effectively. To remedy this problem we introduce another reference point. Now all directions of characteristics can be covered effectively by the four orderings except one direction, which is orthogonal to the line connecting these two reference points, as shown in Figure 2.6. Therefore we need at least three noncollinear reference points and we sweep through all the nodes according to their l^2-distances to these reference points in ascent and descent orderings; a total of six orderings cover all directions of information propagating along characteristics. It can be easily seen that four noncoplanar reference points are needed in three dimensions.

If we use the l^1-metric, the sweeping wavefront is a tilted square. For each reference point, as shown in Figure 2.7, the whole plane can be divided into four quadrants, and each quadrant can be covered by one planar sweeping wavefront. If we choose two reference points such that the computational domain lies in different quadrants of these two reference points, all directions of characteristics can be covered by the four orderings corresponding to the ascent and descent sorting according to the l^1-metric; see Figure 2.7.

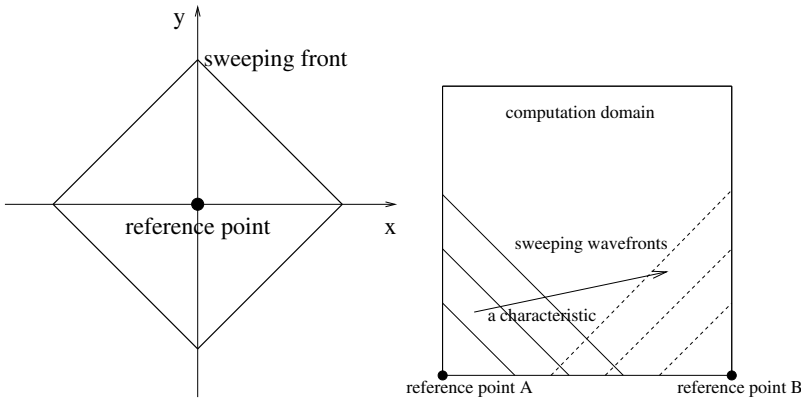
When characteristics are not straight lines, any characteristic can be divided into a finite number of pieces so that each piece can be covered effectively by one of the orderings, as shown in [35]. The total number of sweepings is increased due to curved characteristics, but it is still finite. The number of iterations will be estimated in section 3.

In terms of numerical implementation on a particular mesh some remarks are in order.



(a) one reference point (b) three reference points

FIG. 2.6. *Reference points and sweeping wavefronts for the l^2-metric.*



(a) one reference point (b) two reference points

FIG. 2.7. *Reference points and sweeping wavefronts for the l^1-metric.*

The domain of dependence for a node in the discrete case is a region rather than only the characteristic that passes through the node in the continuous case. On a triangular mesh, the propagating direction of a sweeping wavefront has to fall into the triangle which satisfies the causality criterion in Definition 2.1 so that the two neighbors that determine the current vertex have already been updated in the current sweeping. Numerically this means that the normal of the sweeping wavefront has to make an acute angle with the characteristic that passes through this vertex.

The criterion for an optimal choice of reference points and their locations on a triangular mesh is that all directions of characteristics should be covered with minimal redundancy. In practice, it is better if these reference points are evenly spaced both spatially and angularly with respect to the data set or boundary where the solution is prescribed. In our numerical tests we use the corners as reference points if the computational domain is rectangular. Other points, such as the center point of the domain or middle points of each edge, can be used as well. The number of iterations needed for convergence may be different for different choices of reference points but it will be finite.

If we have only a point source as the boundary condition on a rectangular mesh and we use that point as the single reference point, then the square wave sweeping accesses nodes in the ascent order in the same way that the down-n-out model does

[32, 7], and the spherical wave sweeping shares some similarities with the expanding wavefront model proposed in [31, 23]. However, we are not aware of any work accessing the nodes in the way similar to the plane wave sweeping proposed here.

The above isotropic metrics are suitable for ordering nodes in solving isotropic eikonal equations. For general anisotropic eikonal equations considered in [24, 18, 20], we may introduce anisotropic Riemannian metrics [5] to sort all the nodes, derive a local solver to update solutions at each node by using phase velocity and group velocity, as illustrated in [24, 18], and design fast sweeping methods accordingly; see [22] for a recent work along this direction.

Now we summarize local solvers and sweeping orderings into a complete algorithm.

THE FAST SWEEPING ALGORITHM ON A TRIANGULAR MESH.

1. Initialization:
   (a) Triangulate the computational domain Ω. Add virtual edges to cut obtuse angles if there are any.
   (b) Choose multiple reference points: $\mathbf{x}_{\text{ref}}^i$, $i = 1, \ldots, R$.
   (c) Sort all nodes according to their l^p-distances to the reference points in ascent and descent orders, and put them into arrays:

$$\begin{aligned} &S_i^+ : \text{ascent order}, i = 1, 2, \ldots, R; \\ &S_i^- : \text{descent order}, i = 1, 2, \ldots, R. \end{aligned} \tag{2.4}$$

   (d) Assign exact values or interpolated values $T^{(0)}$ at vertexes on or near the given boundary Γ, and keep these values fixed during the iterations. At all other vertexes, assign large positive values N to $T^{(0)}$, where N should be larger than the maximum of the true solution, and these values will be updated in later iterations.
2. Gauss–Seidel iteration for $k = 0, 1, \ldots$:
   (a) For $i = 1, \ldots, R$:
      i. For $j = +, -$:
         A. To every vertex $C \in S_i^j$ and every triangle associated with C, f_C=$f(C)$, apply the local solver;
         B. Convergence test: $\|T^{(k+1)} - T^{(k)}\| \le \epsilon$ for $\epsilon > 0$ given, where $\|\cdot\|$ is some specified norm.

We remark that during the Gauss–Seidel iteration the numerical solution at C is calculated using the current values of its neighbors in every triangle. The smallest one will be taken as the possible new value. If this smallest new value is smaller than the current value at C, then the numerical solution at C is updated to be the smallest new value.

In passing we point out that the sorting procedure in the above algorithm can cost $O(M\log M)$ flops if a comparison-based sorting method is used; however, to achieve an optimal $O(M)$ complexity for the algorithm, we may use a radix sorting method [4] in that we know the distribution of nodes. Radix sorting runs an $O(M)$ counting sort on each digit of the key, starting with the least significant and working for bounded integers. For general distances computed in the above algorithm, we argue that a fixed number of digits is sufficient because in some sense the order of updates does not matter too much for two nodes sufficiently close to each other. Moreover, this initial ordering is done for a fixed mesh once and for all.

3. Convergence results. In this section we prove convergence of the fast sweeping algorithm on triangular meshes. In the following analysis, we consider a regular

triangulation $\mathcal{T}_h$ of Ω with the property that all the inner angles of the triangles in $\mathcal{T}_h$ satisfy $\leq \frac{\pi}{2}$.

Considering a triangle $\triangle ABC$ in which T_A and T_B are given, we update the travel-time T_C at the vertex C. Denoting

$$p_1 = \frac{T_C - T_A}{b}, \;\; p_2 = \frac{T_C - T_B}{a}, \;\; p_3 = \frac{T_B - T_A}{c},$$

we adopt the framework in [3] to show consistency and monotonicity of the Godunov numerical Hamiltonian resulting from the local solver introduced in section 2.

LEMMA 3.1 (Godunov numerical Hamiltonian). *Assuming that the causality condition holds, the updating formula for the local solver is one of the solutions for the following equations:*

$$\text{(3.1)} \qquad \begin{cases} \frac{(T_C-T_A)^2}{b^2} - 2\frac{(T_C-T_A)(T_C-T_B)}{a\,b}\cos\gamma + \frac{(T_C-T_B)^2}{a^2} = f_C^2\,\sin^2\gamma \\ \qquad\qquad \textit{if } |p_3| \leq f_C \textit{ and } \alpha - \frac{\pi}{2} \leq \arcsin(\frac{p_3}{f_C}) \leq \frac{\pi}{2} - \beta; \\ \max\left(\frac{T_C-T_A}{b}, \frac{T_C-T_B}{a}\right) = f_C \quad \textit{otherwise.} \end{cases}$$

Here $\angle C = \gamma$, $\angle A = \beta$, $\angle B = \alpha$, and $f_C = f(C)$. This discretization for the eikonal equation is based on the Godunov numerical Hamiltonian:

$$\text{(3.2)} \qquad \hat{H}_C\left(\frac{T_C - T_A}{b}, \frac{T_C - T_B}{a}\right) = f_C,$$

where

$$\text{(3.3)} \quad \hat{H}_C(p_1, p_2) = \begin{cases} \frac{1}{\sin\gamma}\sqrt{p_1^2 - 2p_1\,p_2\cos\gamma + p_2^2} \\ \qquad\qquad \textit{if } |p_3| \leq f_C \textit{ and } \alpha - \frac{\pi}{2} \leq \arcsin(\frac{p_3}{f_C}) \leq \frac{\pi}{2} - \beta; \\ \max(p_1, p_2) \quad \textit{otherwise.} \end{cases}$$

Proof. By Version 2 of the local solver, we have

$$\text{(3.4)} \quad T_C = \begin{cases} \frac{1}{2}(T_A + T_B) + \frac{\sin(\alpha-\beta)}{2\sin\gamma}(T_B - T_A) + \frac{\sin\alpha\,\sin\beta}{\sin\gamma}\sqrt{c^2 f_C^2 - (T_B - T_A)^2} \\ \qquad \text{if } |p_3| \leq f_C \text{ and } \alpha - \frac{\pi}{2} \leq \arcsin(\frac{p_3}{f_C}) \leq \frac{\pi}{2} - \beta; \\ \min(T_A + bf_C, T_B + af_C) \quad \text{otherwise.} \end{cases}$$

By solving (3.1), we have

$$\text{(3.5)} \quad T_C = \begin{cases} \frac{1}{2}(T_A + T_B) + \frac{b^2-a^2}{2c^2}(T_B - T_A) \pm \frac{a\,b\sin\gamma}{c^2}\sqrt{c^2 f_C^2 - (T_B - T_A)^2} \\ \qquad \text{if } |p_3| \leq f_C \text{ and } \alpha - \frac{\pi}{2} \leq \arcsin(\frac{p_3}{f_C}) \leq \frac{\pi}{2} - \beta; \\ \min(T_A + bf_C, T_B + af_C) \quad \text{otherwise;} \end{cases}$$

one of the two roots corresponds to (3.4).

Next we derive the numerical Hamiltonian. Denote $A : (x_A, y_A)$, $B : (x_B, y_B)$, and $C : (x_C, y_C)$. Since the causality condition holds, we have

$$\text{(3.6)} \qquad \frac{T_C - T_A}{b} = \nabla T(C) \cdot \left(\frac{x_C - x_A}{b}, \frac{y_C - y_A}{b}\right)^t + o(h^2),$$

$$\text{(3.7)} \qquad \frac{T_C - T_B}{a} = \nabla T(C) \cdot \left(\frac{x_C - x_B}{a}, \frac{y_C - y_B}{a}\right)^t + o(h^2),$$

where t denotes the transpose of vectors. Furthermore we have

$$\begin{pmatrix} \frac{T_C - T_A}{b} \\ \frac{T_C - T_B}{a} \end{pmatrix} = \mathbf{P}\nabla T(C) + o(h^2), \tag{3.8}$$

where

$$\mathbf{P} = \begin{pmatrix} \frac{x_C - x_A}{b} & \frac{y_C - y_A}{b} \\ \frac{x_C - x_B}{a} & \frac{y_C - y_B}{a} \end{pmatrix}.$$

Ignoring higher-order terms and solving for ∇T_C, we have

$$|\nabla T(C)| \approx \begin{cases} \frac{1}{\sin\gamma}\sqrt{\frac{(T_C - T_A)^2}{b^2} - 2\frac{(T_C - T_A)(T_C - T_B)}{a\,b}\cos\gamma + \frac{(T_C - T_B)^2}{a^2}} \\ \qquad \text{if } |p_3| \le f_C \text{ and } \alpha - \frac{\pi}{2} \le \arcsin(\frac{p_3}{f_C}) \le \frac{\pi}{2} - \beta; \\ \max\left(\frac{T_C - T_A}{b}, \frac{T_C - T_B}{a}\right) \quad \text{otherwise;} \end{cases} \tag{3.9}$$

this is the Godunov numerical Hamiltonian for the eikonal equation. ∎

LEMMA 3.2 (consistency and causality). *The Godunov numerical Hamiltonian*

$$\hat{H}_C(p_1, p_2) = \begin{cases} \frac{1}{\sin\gamma}\sqrt{p_1^2 - 2p_1\,p_2\cos\gamma + p_2^2} \\ \qquad \textit{if } |p_3| \le f_C \textit{ and } \alpha - \frac{\pi}{2} \le \arcsin(\frac{p_3}{f_C}) \le \frac{\pi}{2} - \beta; \\ \max(p_1, p_2) \quad \textit{otherwise} \end{cases} \tag{3.10}$$

is consistent; namely,

$$\hat{H}_C\left(\frac{T_C - T_A}{b}, \frac{T_C - T_B}{a}\right) = |\mathbf{p}| \tag{3.11}$$

if $\nabla T_h = \mathbf{p} \in \mathcal{R}^2$. *It is monotone if the causality condition holds:* $0 \le \gamma_1 \le \gamma$, *where* γ_1 *is the angle from the edge CA to the ray (i.e., the vertical line to the wavefront) CQ counterclockwise; see Figure* 2.1.

Proof. By $\nabla T_h = \mathbf{p} \in \mathcal{R}^2$, we have

$$\begin{pmatrix} \frac{T_C - T_A}{b} \\ \frac{T_C - T_B}{a} \end{pmatrix} = \mathbf{P}\mathbf{p}. \tag{3.12}$$

Inserting this into the numerical Hamiltonian, we have (3.11).

Differentiating $\hat{H}_C(p_1, p_2)$ with respect to p_1 and p_2, the monotonicity of the Hamiltonian requires

$$\frac{\partial \hat{H}_C}{\partial p_1} \ge 0, \qquad \frac{\partial \hat{H}_C}{\partial p_2} \ge 0; \tag{3.13}$$

these can be satisfied if and only if $\cos\gamma \le \frac{p_2}{p_1} \le \frac{1}{\cos\gamma}$. By

$$p_1 = \frac{T_C - T_A}{b} = f_C \sin(\beta + \theta), \tag{3.14}$$

$$p_2 = \frac{T_C - T_B}{a} = f_C \sin(\alpha - \theta), \tag{3.15}$$

where $\theta = \arcsin(\frac{p_3}{f_C})$, we have

$$\cos\gamma \le \frac{\sin(\beta + \theta)}{\sin(\alpha - \theta)} \le \frac{1}{\cos\gamma}, \tag{3.16}$$

which is equivalent to the causality condition $0 \le \gamma_1 \le \gamma$, since $\gamma_1 = \frac{\pi}{2} - (\beta + \theta)$ and $\gamma_1 = (\gamma + \alpha - \theta) - \frac{\pi}{2}$. ∎

LEMMA 3.3 (monotonicity). *The fast sweeping algorithm is monotone and Lipschitz continuous, i.e.,*

$$1 \ge \frac{\partial T_C}{\partial T_B} \ge 0, \quad 1 \ge \frac{\partial T_C}{\partial T_A} \ge 0, \tag{3.17}$$

and

$$\frac{\partial T_C}{\partial T_B} + \frac{\partial T_C}{\partial T_A} = 1. \tag{3.18}$$

Proof. Consider the case that $T_A \le T_B$. We need only verify that the above inequalities hold when T_C is updated by

$$T_C = h\, f_C + T_B, \tag{3.19}$$

which is the case that the causality condition is satisfied. From Version 1 of the local solver we have

$$\frac{\partial T_C}{\partial T_B} = 1 + a f_C \cos(\alpha - \theta) \left(-\frac{\partial \theta}{\partial T_B} \right) \tag{3.20}$$

$$= 1 - \frac{a \cos(\alpha - \theta)}{c \cos\theta}; \tag{3.21}$$

$$\frac{\partial T_C}{\partial T_A} = a f_C \cos(\alpha - \theta) \left(-\frac{\partial \theta}{\partial T_A} \right) \tag{3.22}$$

$$= \frac{a \cos(\alpha - \theta)}{c \cos\theta}. \tag{3.23}$$

From Figure 2.1, we have $a\cos(\alpha - \theta) = \overline{PB}$, $c\cos(\theta) = \overline{AR}$, and $\overline{PB} \le \overline{AR}$; therefore, $1 \ge \frac{\partial T_C}{\partial T_B} \ge 0$, $1 \ge \frac{\partial T_C}{\partial T_A} \ge 0$, and $\frac{\partial T_C}{\partial T_B} + \frac{\partial T_C}{\partial T_A} = 1$. ∎

LEMMA 3.4 (maximum change principle). *In the Gauss–Seidel iteration for the fast sweeping algorithm, the maximum change of T_h at any vertex is less than or equal to the maximum change of T_h at its neighboring points.*

Proof. This follows from the above monotonicity property proved in Lemma 3.3. ∎

LEMMA 3.5 (order preserving). *The fast sweeping algorithm is monotone in the initial data.*

Proof. By the monotonicity property of the solution, if $T_h(C) \le R_h(C)$ at all vertexes initially, then $T_h(C) \le R_h(C)$ at all vertexes after any number of Gauss–Seidel iterations. ∎

LEMMA 3.6 (nonincreasing). *The solution of the fast sweeping algorithm is nonincreasing with each Gauss–Seidel iteration.*

Proof. This is evident from the updating formula, which updates the current value only if it is larger than the newly computed value during the Gauss–Seidel iteration. ∎

LEMMA 3.7 (l^∞-contraction). *Let $T^{(k)}$ and $R^{(k)}$ be two numerical solutions at the kth iteration of the fast sweeping algorithm. Let $\|\cdot\|_\infty$ be the maximum norm. Then*

$$\|T^{(k)} - R^{(k)}\|_\infty \le \|T^{(k-1)} - R^{(k-1)}\|_\infty; \tag{3.24}$$

$$0 \le \max_C \left\{ T_C^{(k)} - T_C^{(k+1)} \right\} \le \max_C \left\{ T_C^{(k-1)} - T_C^{(k)} \right\}. \tag{3.25}$$

Proof. Assume that the first update at the kth iteration is at C,

$$T_C^{(k)} = \min\{T_C^{(k-1)}, \bar{T}\},$$

where $\bar{T}$ is the solution computed from its neighbors $T_A^{(k-1)}$ and $T_B^{(k-1)}$. The same is true for $R_C^{(k)}$. By the maximum change principle, we have

$$|T_C^{(k)} - R_C^{(k)}| \le \|T^{(k-1)} - R^{(k-1)}\|_\infty. \tag{3.26}$$

For an update at any other node later in the iteration, the neighboring values used for the update are either from the previous iteration or from an earlier update in the current iteration, both of which satisfy the above bound. By induction, we have l^∞-contraction (3.24). By the monotonicity of the fast sweeping algorithm and (3.24), setting $R^{(k)} = T^{(k-1)}$ we conclude (3.25). ∎

THEOREM 3.8 (convergence). *The solution of the fast sweeping algorithm converges monotonically to the solution of the discretized system.*

Proof. Denote the numerical solution after the kth iteration by $T_C^{(k)}$. Since $T_C^{(k)}$ is bounded below by 0 and is nonincreasing with Gauss–Seidel iterations, $T_C^{(k)}$ is convergent for all C. After each sweep for each C at each triangle, we have by the monotonicity of the numerical Hamiltonian

$$\frac{(T_C^{(k)} - T_A^{(k)})^2}{b^2 \sin^2\gamma} - 2\frac{(T_C^{(k)} - T_A^{(k)})(T_C^{(k)} - T_B^{(k)})}{a\, b \sin^2\gamma}\cos\gamma + \frac{(T_C^{(k)} - T_B^{(k)})^2}{a^2 \sin^2\gamma} \ge f_C^2 \tag{3.27}$$

because any later update of neighbors of $T_C^{(k)}$ in the same iteration is nonincreasing. Moreover, it is easy to see that after $T_C^{(k)}$ is updated, the function

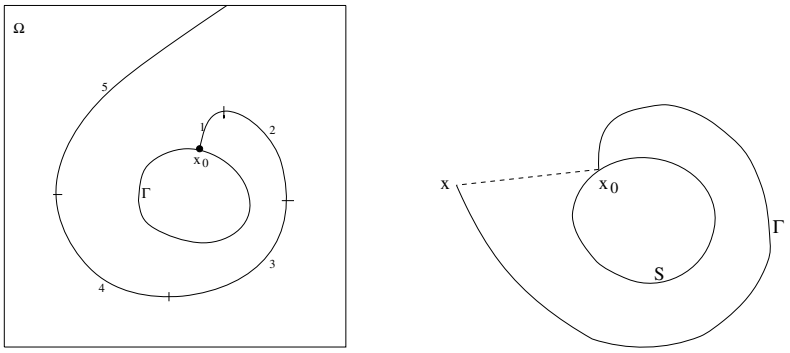
$$\begin{aligned} F(T_A^{(k)}, T_B^{(k)}) = {} & \frac{(T_C^{(k)} - T_A^{(k)})^2}{b^2 \sin^2\gamma} - 2\frac{(T_C^{(k)} - T_A^{(k)})(T_C^{(k)} - T_B^{(k)})}{a\, b \sin^2\gamma}\cos\gamma \\ & + \frac{(T_C^{(k)} - T_B^{(k)})^2}{a^2 \sin^2\gamma} - f_C^2 \end{aligned} \tag{3.28}$$

is Lipschitz continuous in $T_A^{(k)}$ and $T_B^{(k)}$, and the Lipschitz constant is bounded by

$$2\max\left\{ \frac{|T_C^{(k)} - T_A^{(k)}|}{b^2 \sin^2\gamma} + \frac{|T_C^{(k)} - T_B^{(k)}|}{a\, b \sin^2\gamma}\cos\gamma, \frac{|T_C^{(k)} - T_B^{(k)}|}{a^2 \sin^2\gamma} + \frac{|T_C^{(k)} - T_A^{(k)}|}{a\, b \sin^2\gamma}\cos\gamma \right\}.$$

Since $T_C^{(k)}$ is monotonically convergent for all C, we can have an upper bound $Z > 0$ for the Lipschitz constant. Let $\delta^{(k)} = \max\{T_C^{(k-1)} - T_C^{(k)}\}$ be the maximum change at all grid points during the kth iteration. By the l^∞-contraction property and the convergence property of $T_C^{(k)}$, $\delta^{(k)}$ converges monotonically to zero. After the kth iteration, we have

$$\begin{aligned} 0 \le {} & \frac{(T_C^{(k)} - T_A^{(k)})^2}{b^2 \sin^2\gamma} - 2\frac{(T_C^{(k)} - T_A^{(k)})(T_C^{(k)} - T_B^{(k)})}{a\, b \sin^2\gamma}\cos\gamma + \frac{(T_C^{(k)} - T_B^{(k)})^2}{a^2 \sin^2\gamma} - f_C^2 \\ & \le Z\delta^{(k)}. \end{aligned} \tag{3.29}$$



(a) (b)

FIG. 3.1. *Partitioning of a characteristic.*

Thus $T^{(k)}$ converges to the solution to (3.1). □

Note that the monotone convergence is very important during iterations. Once the solution at a node reaches the minimal value that it can get, it is the correct value at that node and will not change in later iterations.

Next we show the estimate for the total number of iterations needed for convergence. As pointed out above, given a systematic ordering, any characteristic can be partitioned into a finite number of pieces and each piece will be covered correctly by one of the sweeping orderings, as shown in Figure 3.1(a). Since these pieces have to be captured sequentially the total number of iterations needed is proportional to the number of pieces. Finally the number of pieces needed to partition a characteristic is related to the directional change of the characteristic. We now give an estimate on the total variation of the tangent direction of any characteristic in a fixed domain Ω.

Denote $H(\mathbf{p}, \mathbf{x}) = |\mathbf{p}| - f(\mathbf{x})$, where $\mathbf{p} = \nabla T$. The characteristic equation for the eikonal equation is

$$\begin{cases} \dot{\mathbf{x}} = \nabla_{\mathbf{p}} H = \frac{\nabla T}{f(\mathbf{x})}, \\ \dot{\mathbf{p}} = -\nabla_{\mathbf{x}} H = \nabla f(\mathbf{x}), \\ \dot{T} = \nabla T \cdot \dot{\mathbf{x}} = f(\mathbf{x}), \end{cases}$$

where $\dot{}$ denotes the derivative along characteristics parametrized by the arc length s.

Since $|\dot{\mathbf{x}}| = 1$, it was shown in [35] that the curvature bound along a characteristic is

$$|\ddot{\mathbf{x}}| \le \left| \frac{\nabla f(\mathbf{x})}{f(\mathbf{x})} \right|. \tag{3.30}$$

LEMMA 3.9. *Assuming that $f(\mathbf{x})$ is strictly positive and C^1 in Ω, the total variation of the tangent direction of the shortest characteristic L from an initial point $\mathbf{x}_0 \in \Gamma$ to a point $\mathbf{x} \in \Omega$ is bounded by*

$$\int_L |\ddot{\mathbf{x}}| ds \le \frac{DKf_M}{f_m}, \tag{3.31}$$

where s is the arc-length along the characteristic L, D is the diameter of domain Ω, and

$$K = \sup_{\mathbf{x}\in\Omega} \left| \frac{\nabla f(\mathbf{x})}{f(\mathbf{x})} \right|, \qquad f_M = \sup_{\mathbf{x}\in\Omega} f(\mathbf{x}), \qquad f_m = \inf_{\mathbf{x}\in\Omega} f(\mathbf{x}).$$

Proof. The existence of the shortest path L, yielding the first-arrival travel-time from an initial point $\mathbf{x}_0 \in \Gamma$ to a point $\mathbf{x} \in \Omega$, is guaranteed by the results in [14, 16]. If L is a single characteristic curve, then from (3.30) we have

$$\int_L |\ddot{\mathbf{x}}| ds \leq \int_L \frac{|\nabla f(\mathbf{x})|}{f(\mathbf{x})} ds \leq K \int_L ds, \tag{3.32}$$

where s is the arc-length; see Figure 3.1(b). The travel-time at $\mathbf{x}$ is $T(\mathbf{x}) = \int_L f(s)ds$. This travel-time, which is the first arrival time at $\mathbf{x}$, is smaller than the travel-time along the direct path from $\mathbf{x}_0$ to $\mathbf{x}$. So we have

$$f_m \int_L ds \leq \int_L f(s)ds = T(\mathbf{x}) \leq \int_{\mathbf{x}_0}^{\mathbf{x}} f(s)ds \leq f_M |\mathbf{x} - \mathbf{x}_0|. \tag{3.33}$$

Hence

$$\text{length}(L) = \int_L ds \leq \frac{D f_M}{f_m}. \tag{3.34}$$

Together with (3.32) we finish the proof. In general L may be composed of several pieces of characteristic curves. The above integral may be broken into several parts accordingly, but the same proof goes through. □

According to the above lemma the maximal number of sweeping needed to cover all characteristics can be bounded by $C \times \frac{DKf_M}{f_m}$, where the constant C may depend on the number of reference points and orderings.

Here is a discrete version of the above argument [36]. For an appropriate upwind scheme the corresponding discretized nonlinear system of equations has a solution (see Theorem 3.8). We can classify all nodes into a few groups according to the solution. All nodes in each group have a dependence pattern similar to their neighbors. For example, on a rectangular grid in two dimensions, almost all grid points can be divided into simply connected regions. In each region the value at a grid point depends on two of its neighbors in the following ways: (1) left and down neighbors; (2) left and up neighbors; (3) right and down neighbors; (4) right and up neighbors. By the Gauss–Seidel iteration each connected region can be covered by one of the orderings simultaneously when the ordering is in the upwind direction of the dependence pattern. The number of connected regions is proportional to the number of directional changes of characteristics which is bounded above. This relates the number of iterations for the fast sweeping method to the above bound. On a triangular mesh, because an arbitrary unstructured mesh may accommodate much more information flowing directions than a rectangular mesh, the situation is more complicated. However, given a triangulation and a choice of the reference points, all nodes can be partitioned into a finite number of connected regions. In each region the nodal dependence follows one of the orderings according to the increase/decrease of the distance to the reference points. For example, all those connected nodes, whose values depend on neighboring nodes that are closer to one of the reference points, belong to one region. The number of regions is proportional to the bound above. Although the triangulation and the choice of the reference points may affect the number of iterations, it is finite for a fixed setup.

4. Numerical examples. Now we show numerical examples in both two and three dimensions to illustrate the efficiency and the accuracy of our algorithm. In all the examples we have used the quick-sort method to order the nodes, though a radix sorting method may be implemented as well.

Our computational experience indicates that for an acute triangulation, using four corners in 2-D rectangular domains or eight corners in 3-D rectangular domains as the reference points is sufficient for the algorithm to converge in a finite number of iterations. For a triangulation with some obtuse triangles, more reference points may be needed. However, if the virtual splitting of obtuse angles as described in section 2.1 is used, then no extra reference point is needed; the results in convergence and accuracy are similar to those with all triangles being acute.

In all the presented examples the number of iterations is independent of the mesh size. The convergence of iteration is measured as full convergence in terms of the l^∞-norm; i.e., the iteration stops when the successive error reaches machine zero. On the other hand, the convergence order of the method is measured in the l^1-norm, as advocated by Lin and Tadmor [15].

We note that in our implementation, the convergence test is checked for every sweeping; here one sweeping is defined as passing through each node once according to a given ordering of nodes. So the iteration numbers reported in numerical examples are, in fact, the sweeping numbers needed for the algorithm to converge.

4.1. 2-D acute triangulation. We first triangulate the computational domain into acute triangles, then we refine the mesh uniformly by cutting each triangle in the coarse mesh into four smaller similar ones. We have chosen the four corners as the reference points in Examples 1, 2, and 3, with both the l^1- and l^2-metric–based sortings.

Example 1 (two-circle problem). The eikonal equation (2.1) with $f(x, y) = 1$. The computational domain is $\Omega = [-2, 2] \times [-2, 2]$; Γ consists of two circles of equal radius 0.5 with centers located at $(-1, 0)$ and $(\sqrt{1.5}, 0)$, respectively. The exact solution is the distance function to Γ. An acute triangulation is used in the computation. The solution is shown in Figure 4.1(a).

Example 2 (shape-from-shading). This example is taken from [26], in which

$$f(x, y) = 2\pi\sqrt{[\cos(2\pi x)\sin(2\pi y)]^2 + [\sin(2\pi x)\cos(2\pi y)]^2}. \tag{4.1}$$

$\Gamma = \{(\frac{1}{4}, \frac{1}{4}),\ (\frac{3}{4}, \frac{3}{4}),\ (\frac{1}{4}, \frac{3}{4}),\ (\frac{3}{4}, \frac{1}{4}),\ (\frac{1}{2}, \frac{1}{2})\}$, consisting of five isolated points. The computational domain is $\Omega = [0, 1] \times [0, 1]$. $T(x, y) = 0$ is prescribed on the boundary of the unit square. The solution to this problem is the shape function, which has the brightness $I(x, y) = 1/\sqrt{1 + f(x, y)^2}$ under vertical lighting. We have used acute triangulations for the following two cases.

Case a.

$$g\left(\frac{1}{4},\ \frac{1}{4}\right) = g\left(\frac{3}{4},\ \frac{3}{4}\right) = 1, \quad g\left(\frac{1}{4},\ \frac{3}{4}\right) = g\left(\frac{3}{4},\ \frac{1}{4}\right) = -1, \quad g\left(\frac{1}{2},\ \frac{1}{2}\right) = 0.$$

The exact solution for this case is smooth,

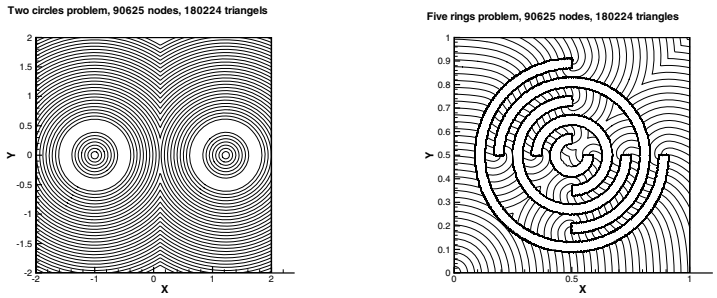
$$T(x, y) = \sin(2\pi x)\sin(2\pi y).$$

Case b.

$$g\left(\frac{1}{4},\ \frac{1}{4}\right) = g\left(\frac{3}{4},\ \frac{3}{4}\right) = g\left(\frac{1}{4},\ \frac{3}{4}\right) = g\left(\frac{3}{4},\ \frac{1}{4}\right) = 1, \quad g\left(\frac{1}{2},\ \frac{1}{2}\right) = 2.$$

The exact solution for this case is nonsmooth,

$$T(x, y) = \begin{cases} \max(|\sin(2\pi x)\sin(2\pi y)|,\ 1 + \cos(2\pi x)\cos(2\pi y)) \\ \qquad\qquad \text{if } |x + y - 1| < \frac{1}{2} \text{ and } |x - y| < \frac{1}{2}; \\ |\sin(2\pi x)\sin(2\pi y)| \qquad \text{otherwise.} \end{cases}$$



(a) (b)

FIG. 4.1. (a) *Example* 1: *two-circle problem.* (b) *Example* 3: *five-ring problem.*

TABLE 4.1
Accuracy tests for Examples 1 *and* 2. *Acute triangulation. Four corners as the reference points.*

		Two-circle		Shape (Case a)		Shape (Case b)	
Nodes	Elements	L^1 error	Order	L^1 error	Order	L^1 error	Order
1473	2816	7.71E-3	–	4.54E-2	–	2.83E-2	–
5716	11264	4.21E-3	0.87	2.54E-2	0.84	1.62E-2	0.81
22785	45056	2.18E-3	0.95	1.34E-2	0.92	8.76E-3	0.89
90625	180224	1.11E-3	0.97	6.90E-3	0.96	4.60E-3	0.93

TABLE 4.2
Iteration numbers for Examples $1, 2,$ *and* 3. *Acute triangulation. Spherical wave sweeping based on the* l^2*-metric ordering. Four corners as the reference points.*

Nodes	Elements	Two-circle	Shape (Case a)	Shape (Case b)	Five-ring
1473	2816	6	9	9	19
5716	11264	6	13	13	20
22785	45056	8	11	13	21
90625	180224	8	11	13	21

Example 3 (five-ring). The computational domain is $\Omega = [0,1] \times [0,1]$, Γ is a point source at $(0,0)$, and a five-ring obstacle is placed in the computational domain. This example is borrowed from [9]. Here we also use an acute triangulation. The solution is shown in Figure 4.1(b).

From Table 4.1, we can see that the accuracy of the algorithm for Examples 1 and 2 is first order. Although the same discretized nonlinear system is solved, no matter which ordering metric is used, different ordering strategies may result in different numbers of iterations, as illustrated in Tables 4.2 and 4.3, where we have applied orderings based on l^1- and l^2-metrics, respectively. Certainly, the two tables also indicate that the iteration number does not depend on the mesh size as the mesh is refined.

Table 4.4 shows the number of iterations needed using the l^1-metric with only two reference points. The two reference points are two corners that are not diagonal to each other.

On the other hand, Table 4.5 shows that a simple extension of the ordering strategy used for rectangular meshes, i.e., sorting all vertexes according to the ascent and descent orders of their x- and y-coordinates, may result in more iterations.

TABLE 4.3
Iteration numbers for Examples 1, 2, *and* 3. *Acute triangulation. Planar wave sweeping based on the* l^1*-metric ordering. Four corners as the reference points.*

Nodes	Elements	Two-circle	Shape (Case a)	Shape (Case b)	Five-ring
1473	2816	7	12	9	26
5716	11264	7	12	9	27
22785	45056	7	16	9	27
90625	180224	7	15	9	27

TABLE 4.4
Iteration numbers for Examples 1, 2, *and* 3. *Acute triangulation. Planar wave sweeping based on the* l^1*-metric ordering using only two reference points.*

Nodes	Elements	Two-circle	Shape (Case a)	Shape (Case b)	Five-ring
1473	2816	6	12	8	16
5716	11264	6	12	9	25
22785	45056	7	17	9	29
90625	180224	7	14	10	29

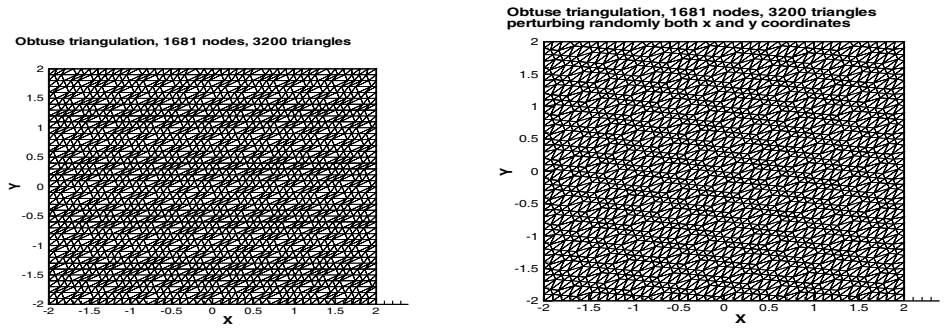
TABLE 4.5
Iteration numbers for Examples 1, 2, *and* 3. *Acute triangulation. Nodes are sorted by* x*- and* y*-coordinates. Four corners as the reference points.*

Nodes	Elements	Two-circle	Shape (Case a)	Shape (Case b)	Five-ring
1473	2816	9	9	9	22
5716	11264	9	10	14	26
22785	45056	13	18	15	33
90625	180224	13	13	15	33

4.2. 2-D obtuse triangulation. We test our strategy for treating a triangulation which has obtuse angles. The obtuse triangulations are constructed by perturbing randomly the x-coordinates of vertexes (Figure 4.2(a)) or perturbing randomly both the x-coordinates and the y-coordinates of vertexes (Figure 4.2(b)) in a uniform triangulation. The uniform triangulation, in turn, is obtained by connecting the diagonal line in every rectangle of a rectangular mesh and cutting every rectangle into two equivalent isosceles triangles. The perturbation range is $[-0.5h, 0.5h]$, where h is the length of an isosceles triangle. We use Example 1 in section 4.1 as a test example and apply spherical-wave sweepings.

As a first test, we use the obtuse triangulation as in Figure 4.2(a), choose four corners of the computational domain as the reference points, and sweep through all the nodes according to both ascent and descent sortings. The accuracy and the number of iterations for the algorithm without and with the obtuse-angle treatment are listed in Table 4.6.

As a second test, we use eight reference points which include both the four corners and four middle points of the four edges of the computational domain, and we use only ascent orders. The accuracy and the number of iterations for the algorithm without and with the obtuse-angle treatment are listed in Table 4.7 for the obtuse triangulation as in Figure 4.2(a) and in Table 4.8 for the obtuse triangulation as in Figure 4.2(b). Comparing Tables 4.6, 4.7, and 4.8, we can see that more reference points may help us reduce the number of sweepings needed in the algorithm. Roughly speaking, for different meshes the errors from the algorithm with the obtuse-angle treatment are decreased $2 \sim 4$ times in comparison to the errors from the algorithm without such a treatment, as shown in both Table 4.7 and Table 4.8. The first-order



(a) (b)

FIG. 4.2. *Obtuse triangulations.* (a) *Perturbing randomly the x-coordinate of vertexes in a uniform triangulation;* (b) *perturbing randomly x- and y-coordinates of vertexes.*

TABLE 4.6

Two-circle problem. Obtuse triangulation (Figure 4.2(a)*). Spherical wave sweepings:* 4 *reference points (*4 *corners of computational domain). Both ascent and descent orderings.*

			Before treatment			After treatment		
Elements	Obtuse elements	Max obtu	L^1 error	Order	Iter.	L^1 error	Order	Iter.
200	78	120°	6.70E-2	–	6	4.26E-2	–	5
800	528	115°	2.49E-2	1.43	8	1.71E-2	1.32	6
3200	958	125°	2.90E-2	−0.22	15	9.71E-3	0.81	12
12800	5890	118°	1.98E-2	0.55	34	4.60E-3	1.08	18
51200	40558	116°	4.71E-3	2.07	44	2.31E-3	0.99	24

TABLE 4.7

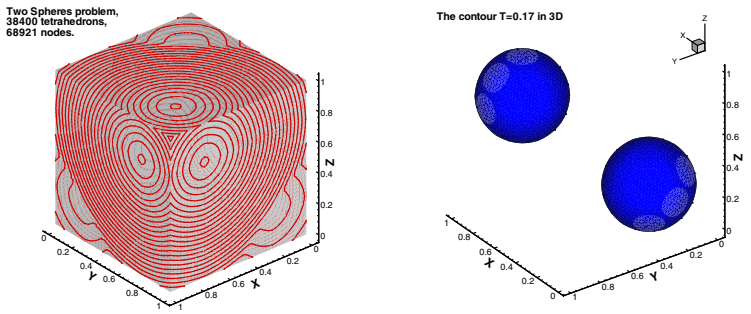
Two-circle problem. Obtuse triangulation (Figure 4.2(a)*). Spherical wave sweepings:* 8 *reference points (*4 *corners and* 4 *middle points of the* 4 *edges). Only ascent ordering.*

			Before treatment			After treatment		
Elements	Obtuse elements	Max obtu	L^1 error	Order	Iter.	L^1 error	Order	Iter.
200	78	120°	6.70E-2	–	4	4.26E-2	–	4
800	528	115°	2.49E-2	1.43	8	1.71E-2	1.32	6
3200	958	125°	2.91E-2	−0.22	8	9.71E-3	0.81	8
12800	5890	118°	1.98E-2	0.55	8	4.60E-3	1.08	9
51200	40558	116°	4.72E-3	2.07	13	2.31E-3	0.99	11

TABLE 4.8

Two-circle problem. Obtuse triangulation (Figure 4.2(b)*). Spherical wave sweepings:* 8 *reference points (*4 *corners and* 4 *middle points of the* 4 *edges). Only ascent ordering.*

			Before treatment			After treatment		
Elements	Obtuse elements	Max obtu	L^1 error	Order	Iter.	L^1 error	Order	Iter.
200	81	106°	3.55E-2	–	4	3.08E-2	–	4
800	727	108°	2.30E-2	0.63	8	1.70E-2	0.86	4
3200	1344	106°	1.32E-2	0.80	8	8.04E-3	1.08	6
12800	5909	106°	7.73E-3	0.77	11	4.66E-3	0.79	10
51200	50560	108°	3.88E-3	0.99	14	1.89E-3	1.31	14



(a) (b)

FIG. 4.3. *Two-sphere problem. Use a tetrahedral mesh.* (a) *The surface contour,* 30 *equally spaced contour lines from* $T = 0$ *to* $T = 0.742402$ *(produced automatically by the plotting software);* (b) *the contour plot of* $T = 0.17$ *in the* 3*-D case.*

TABLE 4.9
Two-sphere problem. Comparison between tetrahedral meshes and rectangular meshes. Spherical wave sweepings: 8 *corners as reference points. Both ascent and descent orderings.*

		Unstructured mesh			Structured mesh		
Nodes	Elements	L^1 error	Order	Iter.	L^1 error	Order	Iter.
9261	48000	1.25E-2	–	12	1.77E-2	–	15
68921	384000	7.17E-3	0.81	12	1.02E-2	0.80	15
531441	3072000	3.79E-3	0.92	12	5.41E-3	0.91	16

accuracy with the obtuse-angle treatment is more regular than that without the treatment. Moreover, comparing the errors in Table 4.6 with those in Table 4.7, we can observe that without the obtuse-angle treatment different sweeping ordering strategies yield slightly different numerical solutions, and with the obtuse-angle treatment different sweeping ordering strategies yield the same solutions up to machine zero. This indicates that the causality of PDEs may *not* be captured accurately if obtuse angles are *not* treated.

4.3. A 3-D example. We test our 3-D fast sweeping methods on tetrahedral meshes. We use a two-sphere problem as an example: the eikonal equation (2.3) with $f(x, y, z) = 1$.

The computational domain is $\Omega = [0, 1] \times [0, 1] \times [0, 1]$; Γ consists of two spheres of equal radius 0.1 with centers located at $(0.25, 0.25, 0.25)$ and $(0.75, 0.75, 0.75)$, respectively. The exact solution is the distance function to Γ.

We first partition the computational domain into identical rectangular cubes. Then a tetrahedral mesh is obtained by cutting each cube into six tetrahedrons.

The results in Figure 4.3 are obtained by using a tetrahedral mesh which consists of $40 \times 40 \times 40 \times 6 = 384000$ tetrahedrons. We choose the eight corners of the computational domain as the reference points. Both ascent and descent orderings are used, and the ordering strategy is based on the l^2-metric. Figure 4.3(a) shows contours of the solution on the surface of the domain, and Figure 4.3(b) shows 3-D plots of the contour $T = 0.17$.

In Table 4.9, we present the accuracy and numbers of iterations when the tetrahedral mesh is refined. To calibrate the result, we apply the same sweeping ordering to the rectangular mesh from which the tetrahedral mesh is obtained. For the rectangular mesh we use the local solver for rectangular grids as in [35]. Although the

TABLE 4.10

Two-circle problem. Comparison between triangular meshes and rectangular meshes. Spherical wave sweepings: 4 corners as reference points.

		Unstructured mesh			Structured mesh		
Nodes	Elements	L^1 error	Order	Iter.	L^1 error	Order	Iter.
1681	3200	9.85E-3	–	5	1.46E-2	–	5
6561	12800	5.30E-3	0.89	5	7.91E-3	0.88	5
25921	51200	2.74E-3	0.95	5	4.13E-3	0.94	5
103041	204800	1.39E-3	0.98	5	2.10E-3	0.97	5

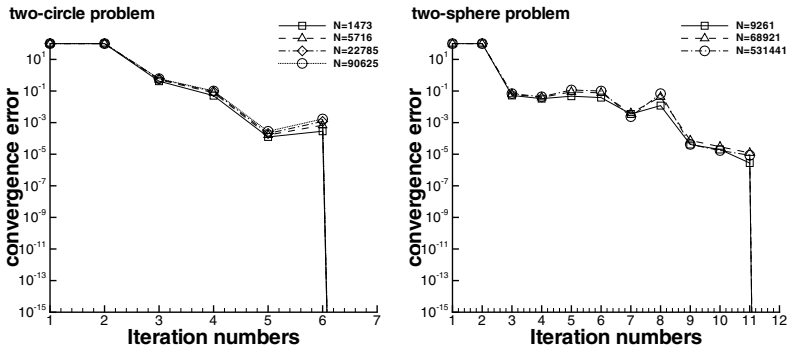


FIG. 4.4. log *plot of convergence error for* 2-*D and* 3-*D examples.*

nodes are the same, the local solvers at each node are different so that the discretized nonlinear systems of the equation are different. The comparison results are also shown in Table 4.9. It is obvious from the table that the local solver on unstructured meshes can achieve higher accuracy than that on structured meshes since the former uses more neighboring points at each node and captures directions of characteristics more accurately than the latter.

Also we can see from Table 4.9 that if the l^2-metric is used for ordering, the number of iterations on an unstructured mesh can be less than that on a structured one. However, the local solver at each node for an unstructured mesh is more expensive than that for a rectangular mesh. Most importantly, we see that both iteration numbers do not change as the mesh is refined. So our ordering strategy works for both cases.

A similar comparison for a 2-D case, Example 1 of section 4.1, is presented in Table 4.10; again the local solver on unstructured meshes achieves higher accuracy than that on structured meshes.

4.4. Typical convergence behavior. Figure 4.4 shows the typical behavior of convergence error of the fast sweeping method in terms of the difference between two consecutive iterations in maximum norm. It demonstrates that the exact solution (up to machine error) to the discretized system is achieved in a finite number of iterations independent of mesh size.

5. Conclusion. We propose novel ordering strategies to extend the fast sweeping method to unstructured meshes. To that end we introduce multiple reference points and order all the nodes according to their l^p-metrics to those reference points. Information propagating along all characteristics can be covered efficiently by the

systematic orderings. We prove that the new algorithm converges and numerical examples demonstrate that the algorithm converges in a finite number of iterations independent of mesh size. The computational complexity of the new algorithm is nearly optimal in the sense that the total computational cost consists of $O(M)$ flops for iteration steps and $O(M\log M)$ flops for sorting at the predetermined initialization step, which can be efficiently optimized by adopting a linear time sorting method, where M is the total number of mesh points. Extensive numerical examples demonstrate the accuracy and the efficiency of the new fast sweeping method.

Acknowledgment. Qian thanks Profs. Stan Osher, William W. Symes, and Eitan Tadmor for encouragement in this project. Qian also thanks Prof. Ian Mitchell for comments related to sorting algorithms. The authors would like to thank the anonymous referees for constructive comments on the paper.

REFERENCES

[1] M. Bardi and S. Osher, *The nonconvex multi-dimensional Riemann problem for Hamilton-Jacobi equations*, SIAM J. Math. Anal., 22 (1991), pp. 344–351.
[2] M. Boué and P. Dupuis, *Markov chain approximations for deterministic control problems with affine dynamics and quadratic costs in the control*, SIAM J. Numer. Anal., 36 (1999), pp. 667–695.
[3] B. Cockburn and J. Qian, *Continuous dependence results for Hamilton–Jacobi equations*, in Collected Lectures on the Preservation of Stability under Discretization, D. Estep and S. Tavener, eds., SIAM, Philadelphia, 2002, pp. 67–90.
[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.
[5] R. Courant and D. Hilbert, *Methods of Mathematical Physics*, Vol. II, John Wiley and Sons, New York, 1962.
[6] M. G. Crandall and P. L. Lions, *Two approximations of solutions of Hamilton–Jacobi equations*, Math. Comp., 43 (1984), pp. 1–19.
[7] J. Dellinger and W. W. Symes, *Anisotropic finite-difference traveltimes using a Hamilton–Jacobi solver*, in Proceedings of the 67th Annual International Meeting, Soc. Expl. Geophys., Expanded Abstracts, Soc. Expl. Geophys., Tulsa, OK, 1997, pp. 1786–1789.
[8] M. Falcone and R. Ferretti, *Discrete time high-order schemes for viscosity solutions of Hamilton-Jacobi-Bellman equations*, Numer. Math., 67 (1994), pp. 315–344.
[9] P. A. Gremaud and C. M. Kuster, *Computational study of fast methods for the eikonal equations*, SIAM J. Sci. Comput., 27 (2006), pp. 1803–1816.
[10] J. Helmsen, E. Puckett, P. Colella, and M. Dorr, *Two new methods for simulating photolithography development in 3-D*, Proc. SPIE, 2726 (1996), pp. 253–261.
[11] C.-Y. Kao, S. Osher, and Y.-H. Tsai, *Fast sweeping methods for static Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 42 (2005), pp. 2612–2632.
[12] C. Y. Kao, S. J. Osher, and J. Qian, *Lax–Friedrichs sweeping schemes for static Hamilton–Jacobi equations*, J. Comput. Phys., 196 (2004), pp. 367–391.
[13] R. Kimmel and J. A. Sethian, *Computing geodesic paths on manifolds*, in Proc. Natl. Acad. Sci., 95 (1998), pp. 8431–8435.
[14] S. N. Kruzkov, *Generalized solutions of nonlinear first order equations with several independent variables.* II, Math. USSR-Sb., 1 (1967), pp. 93–116.
[15] C.-T. Lin and E. Tadmor, *High-resolution nonoscillatory central schemes for Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 21 (2000), pp. 2163–2186.
[16] P. L. Lions, *Generalized Solutions of Hamilton-Jacobi equations*, Pitman, Boston, MA, 1982.
[17] S. Osher and C.-W. Shu, *High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 28 (1991), pp. 907–922.
[18] J. Qian and W. W. Symes, *Paraxial eikonal solvers for anisotropic quasi-P traveltimes*, J. Comput. Phys., 173 (2001), pp. 1–23.
[19] J. Qian and W. W. Symes, *Adaptive finite difference method for traveltime and amplitude*, Geophys., 67 (2002), pp. 167–176.
[20] J. Qian and W. W. Symes, *Finite-difference quasi-P traveltimes for anisotropic media*, Geophys., 67 (2002), pp. 147–155.
[21] J. Qian and W. W. Symes, *Paraxial geometrical optics for quasi-P waves: Theories and*

numerical methods, Wave Motion, 35 (2002), pp. 205–221.
[22] J. Qian, Y.-T. Zhang, and H. K. Zhao, *Fast sweeping methods for static Hamilton–Jacobi equations on triangular meshes*, J. Sci. Comput., submitted.
[23] F. Qin, Y. Luo, K. B. Olsen, W. Cai, and G. T. Schuster, *Finite difference solution of the eikonal equation along expanding wavefronts*, Geophys., 57 (1992), pp. 478–487.
[24] F. Qin and G. T. Schuster, *First-arrival traveltime calculation for anisotropic media*, Geophys., 58 (1993), pp. 1349–1358.
[25] R. Rawlinson and M. Sambridge, *Wave front evolution in strongly heterogeneous layered media using the fast marching method*, Geophys. J. Internat., 156 (2004), pp. 631–647.
[26] E. Rouy and A. Tourin, *A viscosity solutions approach to shape-from-shading*, SIAM J. Numer. Anal., 29 (1992), pp. 867–884.
[27] W. A. Schneider, Jr., K. Ranzinger, A. Balch, and C. Kruse, *A dynamic programming approach to first arrival traveltime computation in media with arbitrarily distributed velocities*, Geophys., 57 (1992), pp. 39–50.
[28] J. A. Sethian, *Level Set Methods*, Cambridge University Press, Cambridge, UK, 1996.
[29] Y.-H. R. Tsai, L.-T. Cheng, S. Osher, and H.-K. Zhao, *Fast sweeping algorithms for a class of Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 41 (2003), pp. 673–694.
[30] J. N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Tran. Automat. Control, 40 (1995), pp. 1528–1538.
[31] J. van Trier and W. W. Symes, *Upwind finite-difference calculation of traveltimes*, Geophys., 56 (1991), pp. 812–821.
[32] J. Vidale, *Finite-difference calculation of travel times*, Bull. Seismol. Soc. Amer. 78 (1988), 2062–2076.
[33] Y.-T. Zhang, H. K. Zhao, and S. Chen, *Fixed-point iterative sweeping methods for static Hamilton-Jacobi equations*, Methods Appl. Anal., to appear.
[34] Y.-T. Zhang, H. K. Zhao, and J. Qian, *High order fast sweeping methods for static Hamilton-Jacobi equations*, J. Sci. Comput., 29 (2006), pp. 25–56.
[35] H. K. Zhao, *Fast sweeping method for eikonal equations*, Math. Comp., 74 (2005), pp. 603–627.
[36] H. K. Zhao, *Parallel Implementations of the Fast Sweeping Method*, research report, UCLA CAM 06-13, University of California, Los Angeles, 2006.
[37] H. K. Zhao, S. Osher, B. Merriman, and M. Kang, *Implicit and nonparametric shape reconstruction from unorganized points using variational level set method*, Comput. Vision and Image Understanding, 80 (2000), pp. 295–319.