Fast sweeping methods for eikonal equations on triangular meshes

Jianliang Qian[1], Yong-Tao Zhang[2], and Hong-Kai Zhao[3]

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 distances 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 propagation efficiently; (2) any characteristics can be decomposed into a finite number of pieces and each piece can be covered by one of the orderings. We prove that the new 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 $O(MlogM)$ 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. We show extensive numerical examples to demonstrate the accuracy and efficiency of the new algorithm.

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 such an equation appears in a variety of applications, it is essential to develop fast, efficient numerical methods to solve such an equation. In this work, we design a class of fast sweeping methods on triangulated domains for the 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 positive function, Ω is a bounded computational domain in R^d and Γ is a subset of Ω.

[1]Department of Mathematics, University of California, Los Angeles, CA 90095-1555, USA. E-mail: qian@math.ucla.edu. Supported by ONR Grant #N00014-02-1-0720.

[2]Department of Mathematics, University of California, Irvine, CA 92697-3875, USA. E-mail: zyt@math.uci.edu

[3]Department of Mathematics, University of California, Irvine, CA 92697-3875, USA. E-mail: zhao@math.uci.edu. The research is partially supported by ONR Grant #N00014-02-1-0090, DARPA Grant #N00014-02-1-0603 and Sloan Fellowship Foundation.

The 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 is able to deal with singularities, (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 a pseudo-time variable which transforms the problem into a time dependent one and evolve the solution to steady state. Due to the finite speed of propagation and the CFL condition for stability, many iterations are needed to reach the steady state solution. The last two decades have witnessed a lot of efforts towards solving the eikonal equation directly: starting from upwinding schemes [27, 26], dynamic programming sweeping methods [22], Jacobi iterations [21], semi-Lagrangian schemes [8], fast marching type methods [25, 10, 23, 13], down-n-out approaches [7, 12], wavefront expanding methods [18], adaptive upwinding methods [16], fast sweeping methods [2, 30, 24, 29, 11, 28] and the references therein. The fast marching method and the fast sweeping method are both efficient methods designed to solve the nonlinear system directly using causality of the PDE. In terms of complexity, the fast marching method [25, 10, 23, 13] has the complexity of $O(MlogM)$, where M is the total number of mesh points. The $\log M$ factor comes from the heapsort algorithm for sorting out the causality order at each step and the constant in O does not depend on M or the equation. The fast sweeping method has the complexity of $O(M)$ where the constant in O depends on the equation and this was proved in [29] for rectangular grids. The accuracy is determined by the discretization scheme. If the first order monotone scheme is used, in general only $h^{1/2}$ convergence can be shown [6] and $h \log h$ convergence is the optimal [29]. On the other hand, most of these methods are based on rectangular meshes. However, it is also very important to design fast methods on triangulated meshes in practice as well. For examples, in seismics a subsurface velocity model usually consists of several irregular interfaces; 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. We show that the resulting methods converge 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 $O(MlogM)$ 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 that its solution is always increasing (or decreasing) along a characteristic. To satisfy such a property, it is crucial for a scheme for computing viscosity solutions to be based on a monotone numerical Hamiltonian [1, 14]. Once we have such a discretization for the eikonal equation in place, the problem boils down to how to solve the resulting nonlinear system efficiently; the fast sweeping method was exactly designed to do that. The original fast sweeping method was inspired by the work [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 behind the fast sweeping method is the following [29]: 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 2-dimensional 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 indices in ascent or descent orders [2, 30, 24, 29, 11, 28], which yields four possible orderings to cover all those four directions of characteristics. However, on an unstructured mesh, there is only local connection information of the nodes available and there is no natural global ordering any more. So far no fast sweeping method is proposed for unstructured meshes yet. In this work 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 examples, information is propagated as plane waves in different directions when using l^1 distance or as spherical waves with different centers when using l^2 distance. We show that these orderings satisfy the key properties essential for the fast sweeping method to converge and the fast sweeping method converges in a finite number of iterations independent of mesh size. Although it may still cost $O(MlogM)$ 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 implement in any number of dimensions.

The rest of the paper is organized as follows. In Section 2, we give implementation details of our algorithm, which includes local solvers at each node on a triangulated mesh for the eikonal equation and the ordering strategies. 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 accuracy of the new method. We conclude the paper in Section 5.

2 Fast sweeping methods on unstructured meshes

2.1 Local solvers

Due to the hyperbolic nature of the eikonal equation, we need to design a numerical Hamiltonian that follows the causality of the PDE. It is relatively easy to achieve this on rectangular grids; it is not so straightforward on triangulated meshes. For the sake of clarity we consider the two dimensional case first.

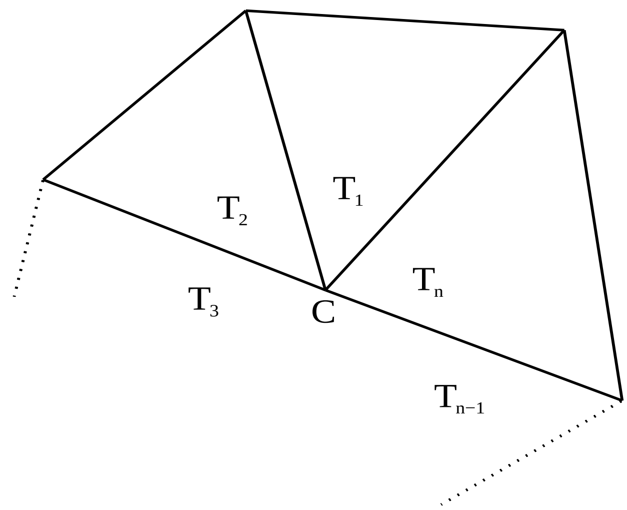


Figure 2.1: Vertex C and all its triangles.

Take $d = 2$ in equation (1.1):

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

We consider triangulation Γ_h of Ω into non-overlapping, nonempty, open triangles $\mathcal{T}$, with diameter $h_{\mathcal{T}}$, such that $\Omega = \cup_{\mathcal{T}\in\Gamma_h}\mathcal{T}$. We assume that Γ_h satisfy the following conditions:

- Intersecting triangles have either a common vertex or a common edge;
- No more than μ triangles have a common vertex;
- $h = \sup_{\mathcal{T}\in\Gamma_h} h_{\mathcal{T}} < 1$;
- Γ_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 $\mathcal{T} \in \Gamma_h$, $h_{\mathcal{T}} \leq \omega_0 \rho_{\mathcal{T}}$.

Therefore, equation (2.1) is solved in the domain Ω, which has a triangulation Γ_h consisting of triangles. We consider every vertex and all triangles which are associated with this vertex. See Figure 2.1 for a vertex C and its n triangles $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n$. During the Gauss-Seidel iteration the numerical solution at vertex 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 old value at C, then the numerical solution at C is updated to be the smallest new value.

Now the task reduces to calculating the value at C at each triangle; see Figure 2.2. Given the values T_A and T_B at vertices 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 the isotropic wave propagation for updating the travel-time at the vertex C from travel-times T_A and T_B, we mean that the ray which is orthogonal to the wavefront and passing through the vertex C must fall inside the triangle $\triangle ABC$.*

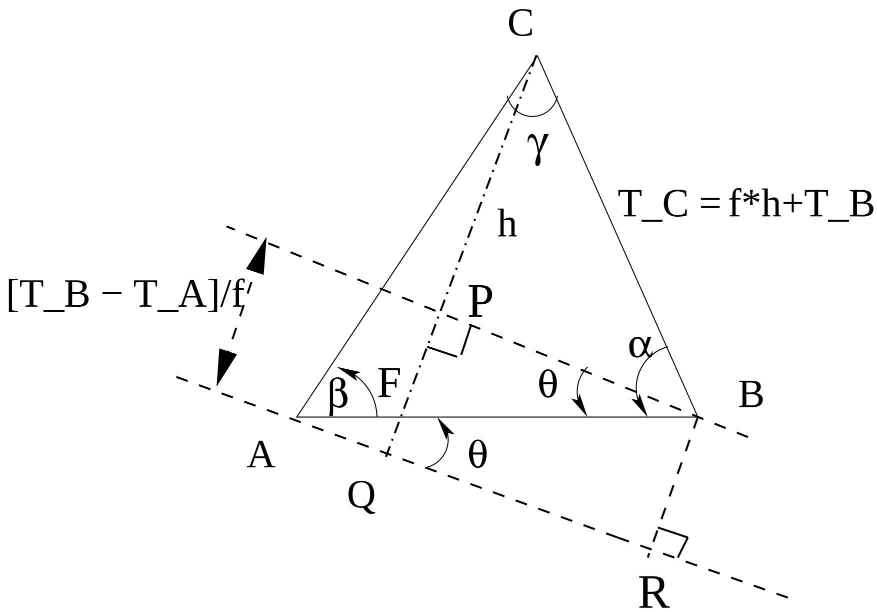


Figure 2.2: Update the value at vertex C in a triangle when causality is satisfied.

We notice that in the 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 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 between the coming wavefront and the edge AB is θ. 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.

Without loss of generality, we may assume that $T_B > T_A$. 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:

- $[T_B - T_A]/f_C \leq \overline{AB} = c$, 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 $f(C)$, which is the inverse of the speed at C;
- $\theta \leq \alpha$ so that the wavefront passes through the vertex B first rather than the vertex C;
- $\theta + \beta < \frac{\pi}{2}$; otherwise the causality is violated since the vertical line from the vertex C to the wavefront does *not* fall inside the triangle; see Figure 2.3.

If all n triangles $\mathcal{T}_1$, $\mathcal{T}_2$, $\cdots$, $\mathcal{T}_n$ around the vertex 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 and there are two possible cases: (i) if the normal of the wavefront is contained between those two dotted lines in Figure 2.4, then the value at vertex C can be updated using values at A and B even though the accuracy will be degraded; (ii) otherwise if the normal of the wavefront comes in

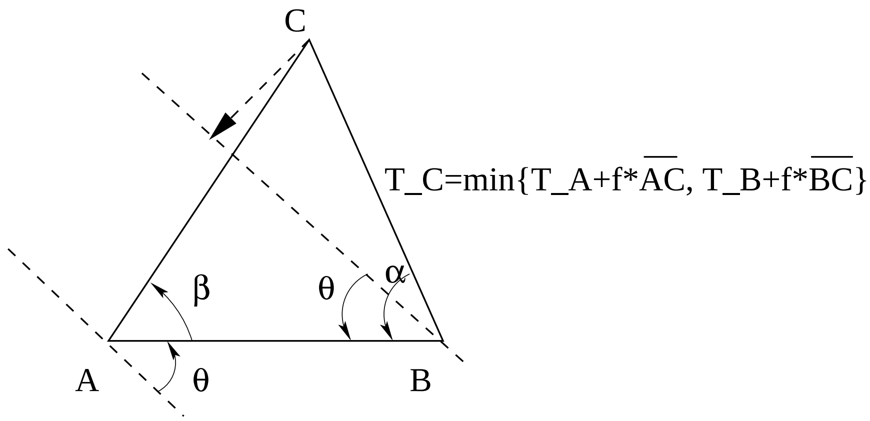


Figure 2.3: Update the value at vertex C in a triangle where causality is not satisfied.

as shown in Figure 2.4, then the value at vertex C can not be updated by A and B correctly [20]. These will be shown in numerical examples in Section 4.

In order to treat obtuse triangles, we adopted the strategy used in [20]. As illustrated in Figure 2.5: if $\angle C$ is obtuse, we connect C to the 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 2.5(a); otherwise if one of the smaller angles is still obtuse, then we keep connecting C to the vertices of the neighboring triangles of the next level, until all new angles at C are acute as shown in Figure 2.5(b). But all these added edges are "virtual"; i.e. they only exist when the value at C is updated. Because such a treatment depends on the mesh only, we only need to do that once before the iteration in the algorithm begins; the resulting algorithm is 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. Without loss of generality, we assume $T_A \le T_B$.

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,

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

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 the vertex 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 in one.

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

$$h = \overline{CP} = \overline{BC}\ \sin(\alpha - \theta) = a\ \sin(\alpha - \theta),$$

$$H = \overline{CQ} = \overline{AC}\ \sin(\beta + \theta) = b\ \sin(\beta + \theta);$$

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

   (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 the 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 [29]. 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 can not be combined into one, since there are gaps corresponding to one of the angles α or β being obtuse. See Figures 2.2 and 2.3 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.

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

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

The equation (2.2) is solved in the domain Ω, which has a triangulation Γ_h consisting of tetrahedra. We consider every vertex and all tetrahedra which are associated with this

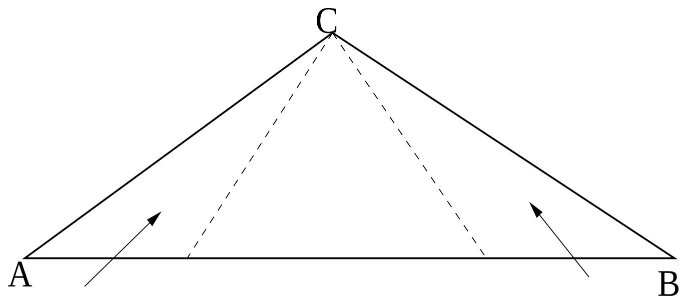


Figure 2.4: Vertex C and its obtuse triangle.

vertex. Similar to the two dimensional case, the numerical solution at every vertex is calculated using the current values of its neighbors in every tetrahedron. The smallest one will be taken as the possible new value. If this smallest new value is smaller than the old value, then the numerical solution at this vertex is updated to be the smallest new value. Again the question reduces to how to calculate the numerical solution at the current central vertex at each tetrahedron; see Figure 2.6.

Given the values T_A, T_B and T_C at vertices 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 the vertex D must fall inside the tetrahedron $ABCD$ so as to satisfy the causality condition. To check such a 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, then we check to see whether E is inside $\triangle ABC$ or not.

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

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

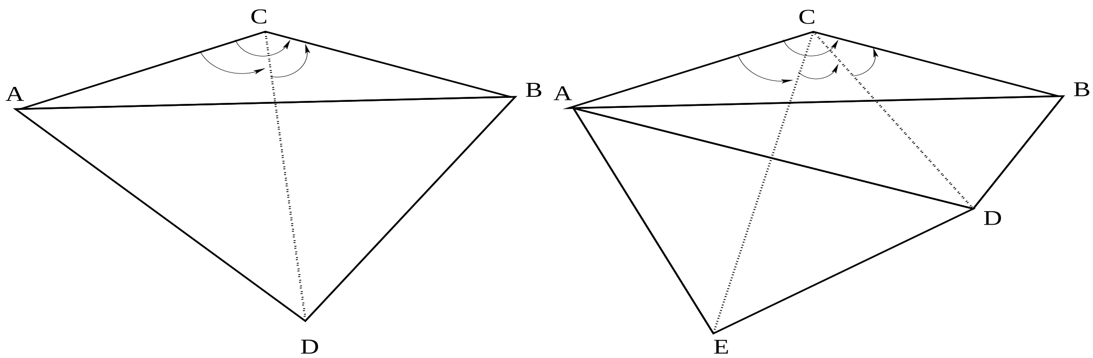
$$\begin{cases} \overrightarrow{AB} \cdot \vec{\mathbf{n}} & = [T_B - T_A]/f_D, \\ \overrightarrow{AC} \cdot \vec{\mathbf{n}} & = [T_C - T_A]/f_D, \\ |\vec{\mathbf{n}}| & = 1; \end{cases} \tag{2.3}$$

   (a) If there exist solutions $\vec{\mathbf{n}}^{(i)}, i = 1, 2$ for the quadratic equations (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 2D local solver on surfaces $\triangle ABD, \triangle ACD$ and $\triangle BCD$ and take the minimal one.

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



(a) (b)

Figure 2.5: A strategy to treat obtuse angles.

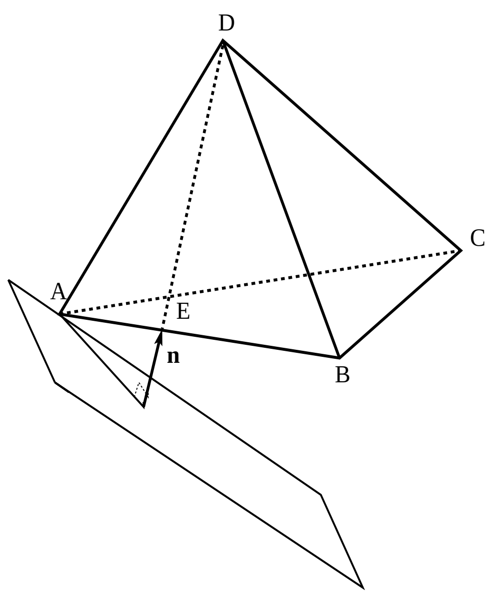


Figure 2.6: A 3-D local solver.

2.2 Sweeping orders and a complete algorithm

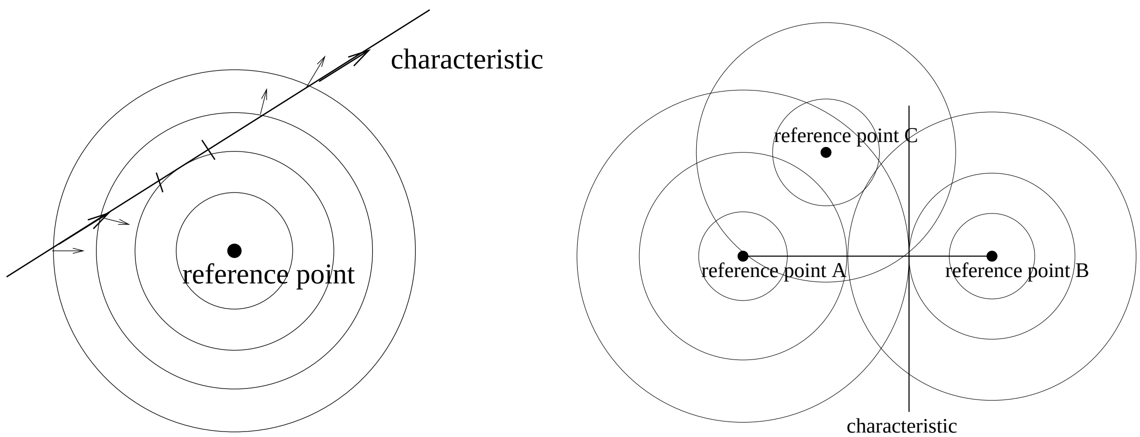
An essential ingredient for making the fast sweeping method [29] successful is a systematic ordering that covers all directions of characteristics efficiently. With a causality preserving discretization in place, information along characteristics of certain directions is captured simultaneously in each 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 two dimensional case [29], 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 indices in 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 will focus on $p = 1, 2$, give explicit geometric interpretation and prove convergence. 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 $\|\mathbf{x}\|_p = (\sum_{j=1}^{n} |x_j|^p)^{1/p}$. For example, in two dimensions, we first fix a reference point $\mathbf{x}_{ref}$; if we sweep all nodes according to $\|\mathbf{x} - \mathbf{x}_{ref}\|_1$ in the ascent (or descent) order, then the sweeping wavefront is an outgoing (or incoming) plane wave since the unit ball of l^1 metric is an tilted square. If we use $\|\mathbf{x} - \mathbf{x}_{ref}\|_2$ to order all nodes, then the sweeping wavefront is an outgoing (or incoming) spherical wave. We will now address the following questions:

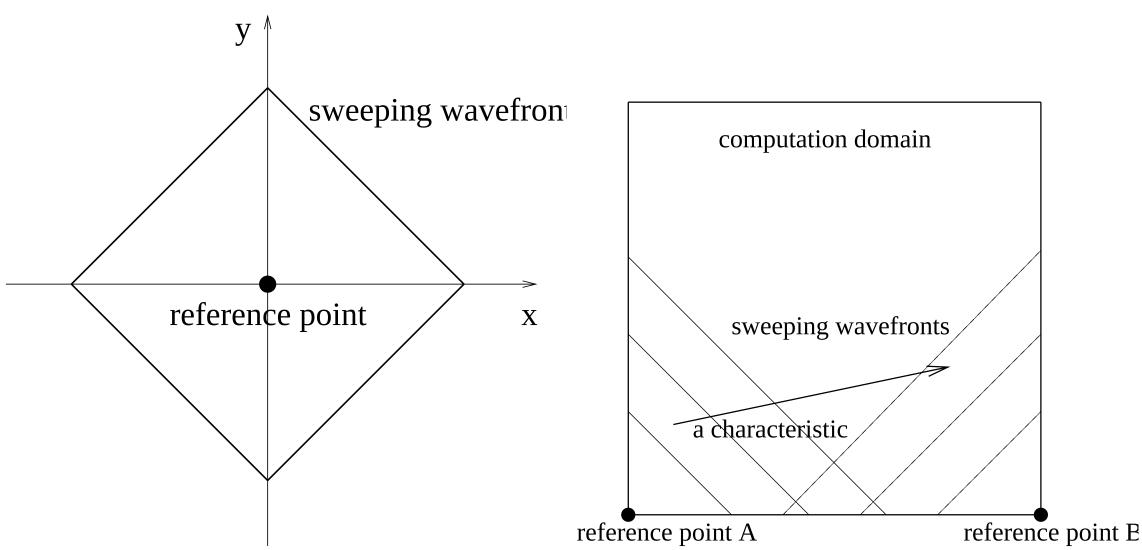
1. How many references points are needed in a systematic ordering that can cover all directions of information propagation?
2. How many iterations are needed for the algorithm to converge?

To address the first question, we have to understand the directional relation between a sweeping wavefront and a characteristic. In the continuous case a basic fact is: if the propagation 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 is illustrated in Figure 2.7, 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 righthand side of the eikonal equation is constant, we almost cover 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 is shown in Figure 2.7. Therefore we need at least three non-collinear reference points and we sweep according to their l^2 distances to these reference points in ascent and descent orderings; total six orderings cover all directions of information propagation along characteristics. It can be easily seen that four non-coplanar 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 is shown in Figure 2.8, 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 2.8.



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

Figure 2.7: Reference points and sweeping wavefronts for the l^2- metric.



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

Figure 2.8: Reference points and sweeping wavefronts for the l^1- metric.

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 is shown in [29]. The total number of sweepings is increased due to curved characteristics, but it is still finite and independent of mesh size. The number of iterations will be estimated in Section 3.

In terms of numerical implementation on a particular mesh we have some complications. For example, the domain of dependence for a node in the discrete case is a region instead of only the characteristic that passes through the node in the continuous case. On a triangular mesh, the propagation 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. When a node has many associated edges on a triangular mesh, more reference points may be needed to cover all directions efficiently; this is different from the case for a rectangular mesh. On a two-dimensional rectangular mesh, each node has four associated edges with fixed directions so that simple orderings can cover all directions efficiently [29]. The criterion for an optimal choice of reference points and their locations on a triangular mesh is: all directions of characteristics should be covered with minimal redundancy. In practice it is best that 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 corner points as reference points if a 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.

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 as the down-n-out model does [27, 7, 12], and the spherical wave sweeping shares some similarities with the expanding wavefront model proposed in [26, 18]. 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 the isotropic eikonal equation. For general anisotropic eikonal equations considered in [19, 15, 17], we may introduce anisotropic Riemannian metrics [5] to sort all the nodes, and we may design fast sweeping methods accordingly; this constitutes an ongoing work.

Now we may summarize local solvers and sweeping orderings into a full 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: $\mathbf{x}^i_{ref}$, i=1,$\cdots$, M.
   - (c) Sort all nodes according to their l^p distances to the reference points in ascent and descent orders, and put them into arrays:

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

   - (d) Assign exact values or interpolated values $T^{(0)}$ at vertices on or near the given boundary Γ, and these values are fixed during the iterations. At all other vertices, 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, $\cdots$:
   (a) For i=1, $\cdots$, M :
      i. To every vertex $C \in S_i^+$ and every triangle associated with C, f_C=$f(C)$, apply the local solver.
      ii. To every vertex $C \in S_i^-$ 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.

In passing we point out that the sorting procedure in the above algorithm can cost $O(MlogM)$ flops if a comparison based sorting method is used; however, to achieve an optimal $O(M)$ complexity for the algorithm, we may use the radix sorting method [4] in that we know the distribution of nodes. Radix sorting runs an O(n) 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 way the order of updates does not matter for two nodes with distances sufficiently close together.

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 Γ_h of Ω with the property that all the inner angles of the triangles in Γ_h satisfy $\le \frac{\pi}{2}$.

Considering a triangle $\triangle ABC$ in which T_A and T_B are given, we update the traveltime T_C at the vertex C. Denote

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

In the following theorem, we will adopt the framework given in [3] to show the consistency and the monotonicity of the Godunov numerical Hamiltonian.

Theorem 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*

$$\begin{cases} \frac{(T_C-T_A)^2}{b^2} - 2\frac{(T_C-T_A)(T_C-T_B)}{a\,b}\cos\gamma + \frac{(T_C-T_B)^2}{a^2} = f_C^2\ \sin^2\gamma \\ \qquad\qquad \textit{if } |p_3| \le f_C \textit{ and } \alpha - \frac{\pi}{2} \le \arcsin(\frac{p_3}{f_C}) \le \frac{\pi}{2} - \beta; \\ \max\left(\frac{T_C-T_A}{b}, \frac{T_C-T_B}{a}\right) = f_C, \quad \textit{otherwise.} \end{cases} \tag{3.5}$$

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

$$\hat{H}_C\left(\frac{T_C - T_A}{b}, \frac{T_C - T_B}{a}\right) \quad = \quad f_C, \tag{3.6}$$

where

$$\hat{H}_C(p_1, p_2) = \begin{cases} \frac{1}{\sin\gamma}\sqrt{p_1^2 - 2p_1\, p_2\cos\gamma + p_2^2} \\ \qquad\qquad \textit{if } |p_3| \le f_C \textit{ and } \alpha - \frac{\pi}{2} \le \arcsin(\frac{p_3}{f_C}) \le \frac{\pi}{2} - \beta; \\ \max\left(p_1, p_2\right), \quad \textit{otherwise.} \end{cases} \tag{3.7}$$

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

$$T_C = \begin{cases} \frac{1}{2}(T_A+T_B)+\frac{\sin(\alpha-\beta)}{2\sin\gamma}(T_B-T_A)+\frac{\sin\alpha\ \sin\beta}{\sin\gamma}\sqrt{c^2 f_C^2-(T_B-T_A)^2} \\ \qquad \text{if } |p_3|\le f_C \text{ and } \alpha-\frac{\pi}{2}\le \arcsin(\frac{p_3}{f_C})\le\frac{\pi}{2}-\beta; \\ \min\left(T_A+bf_C, T_B+af_C\right), \quad \text{otherwise.} \end{cases} \tag{3.8}$$

By solving equation (3.5), we have

$$T_C = \begin{cases} \frac{1}{2}(T_A+T_B)+\frac{b^2-a^2}{2c^2}(T_B-T_A)\pm\frac{a\,b\sin\gamma}{c^2}\sqrt{c^2 f_C^2-(T_B-T_A)^2} \\ \qquad \text{if } |p_3|\le f_C \text{ and } \alpha-\frac{\pi}{2}\le \arcsin(\frac{p_3}{f_C})\le\frac{\pi}{2}-\beta; \\ \min\left(T_A+bf_C, T_B+af_C\right), \quad \text{otherwise;} \end{cases} \tag{3.9}$$

one of the two roots corresponds to equation (3.8).

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

$$\frac{T_C-T_A}{b} = \nabla T(C)\cdot\left(\frac{x_C-x_A}{b},\frac{y_C-y_A}{b}\right)+o(h^2), \tag{3.10}$$

$$\frac{T_C-T_B}{a} = \nabla T(C)\cdot\left(\frac{x_C-x_B}{a},\frac{y_C-y_B}{a}\right)+o(h^2); \tag{3.11}$$

then we have

$$\begin{pmatrix} \frac{T_C-T_A}{b} \\ \frac{T_C-T_B}{a} \end{pmatrix} = \mathbf{P}\cdot\nabla T(C)+o(h^2), \tag{3.12}$$

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

$$|\nabla T(C)| \approx \begin{cases} \frac{1}{\sin\gamma}\sqrt{\frac{(T_C-T_A)^2}{b^2}-2\frac{(T_C-T_A)(T_C-T_B)}{a\,b}\cos\gamma+\frac{(T_C-T_B)^2}{a^2}} \\ \qquad \text{if } |p_3|\le f_C \text{ and } \alpha-\frac{\pi}{2}\le \arcsin(\frac{p_3}{f_C})\le\frac{\pi}{2}-\beta; \\ \max\left(\frac{T_C-T_A}{b},\frac{T_C-T_B}{a}\right), \quad \text{otherwise;} \end{cases} \tag{3.13}$$

this is the Godunov numerical Hamiltonian for the eikonal equation. $\square$

Theorem 3.2 (Consistency and Causality) *The above Godunov numerical Hamiltonian*

$$\hat{H}_C(p_1,p_2) = \begin{cases} \frac{1}{\sin\gamma}\sqrt{p_1^2-2p_1\,p_2\cos\gamma+p_2^2} \\ \qquad \textit{if } |p_3|\le f_C \textit{ and } \alpha-\frac{\pi}{2}\le \arcsin(\frac{p_3}{f_C})\le\frac{\pi}{2}-\beta; \\ \max\left(p_1,p_2\right), \quad \textit{otherwise.} \end{cases} \tag{3.14}$$

is consistent; namely,

$$\hat{H}_C\left(\frac{T_C-T_A}{b},\frac{T_C-T_B}{a}\right) = |\mathbf{p}| \tag{3.15}$$

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 2.2.

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

$$\begin{pmatrix} \frac{T_C - T_A}{b} \\ \frac{T_C - T_B}{a} \end{pmatrix} = \mathbf{P}\mathbf{p}. \tag{3.16}$$

Inserting this into the numerical Hamiltonian, we have equation (3.15).

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

$$\frac{\partial \hat{H}_C}{\partial p_1} \geq 0, \qquad \frac{\partial \hat{H}_C}{\partial p_2} \geq 0; \tag{3.17}$$

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

$$p_1 = \frac{T_C - T_A}{b} = f_C \sin(\beta + \theta), \tag{3.18}$$

$$p_2 = \frac{T_C - T_B}{a} = f_C \sin(\alpha - \theta), \tag{3.19}$$

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

$$\cos\gamma \leq \frac{\sin(\beta+\theta)}{\sin(\alpha-\theta)} \leq \frac{1}{\cos\gamma}, \tag{3.20}$$

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

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

$$1 \geq \frac{\partial T_C}{\partial T_B} \geq 0, \qquad 1 \geq \frac{\partial T_C}{\partial T_A} \geq 0, \tag{3.21}$$

and

$$\frac{\partial T_C}{\partial T_B} + \frac{\partial T_C}{\partial T_A} \quad = \quad 1. \tag{3.22}$$

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

$$T_C \quad = \quad h\, f_C + T_B, \tag{3.23}$$

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

$$\frac{\partial T_C}{\partial T_B} \quad = \quad 1 + a f_C \cos(\alpha - \theta)\left(-\frac{\partial \theta}{\partial T_B}\right) \tag{3.24}$$

$$\quad = \quad 1 - \frac{a\cos(\alpha-\theta)}{c\cos\theta}; \tag{3.25}$$

$$\frac{\partial T_C}{\partial T_A} = af_C \cos(\alpha - \theta) \left(-\frac{\partial \theta}{\partial T_A} \right) \tag{3.26}$$

$$= \frac{a \cos(\alpha - \theta)}{c \cos \theta}. \tag{3.27}$$

From Figure 2.2, we have $a\cos(\alpha - \theta) = \overline{PB}$, $c\cos(\theta) = \overline{AR}$ and $\overline{PB} \leq \overline{AR}$; therefore, $1 \geq \frac{\partial T_C}{\partial T_B} \geq 0$, $1 \geq \frac{\partial T_C}{\partial T_A} \geq 0$ and $\frac{\partial T_C}{\partial T_B} + \frac{\partial T_C}{\partial T_A} = 1$. $\Box$

Theorem 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 Theorem 3.3. $\Box$

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

Proof. From the monotonicity property of the solution, if $T_h(C) \leq R_h(C)$ at all vertices initially, then $T_h(C) \leq R_h(C)$ at all vertices after any number of Gauss-Seidel iterations. $\Box$

Theorem 3.6 (Non-increasing) *The solution of the fast sweeping algorithm is non-increasing with each Gauss-Seidel iteration.*

Proof. This is evident from the updating formula which only updates the current value if it is larger than newly computed value during the Gauss-Seidel iteration. $\Box$

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

$$\|T^{(k)} - R^{(k)}\|_\infty \leq \|T^{(k-1)} - R^{(k-1)}\|_\infty; \tag{3.28}$$

$$0 \leq \max_C \left\{ T_C^{(k)} - T_C^{(k+1)} \right\} \leq \max_C \left\{ T_C^{(k-1)} - T_C^{(k)} \right\}. \tag{3.29}$$

Proof. Assume that the first update at the k-th iteration is at point 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 $R_C^{(k)}$. From the maximum change principle, we have

$$|T_C^{(k)} - R_C^{(k)}| \leq \|T^{(k-1)} - R^{(k-1)}\|_\infty. \tag{3.30}$$

For an update at any other vertex 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 (3.28). By the monotonicity of the fast sweeping algorithm and (3.28), setting $R^{(k)} = T^{(k-1)}$ we conclude (3.29). $\Box$

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 k-th iteration by $T_C^{(k)}$. Since $T_C^{(k)}$ is bounded below by 0 and is non-increasing 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,

$$\frac{(T_C^{(k)}-T_A^{(k)})^2}{b^2\,\sin^2\gamma}-2\frac{(T_C^{(k)}-T_A^{(k)})(T_C^{(k)}-T_B^{(k)})}{a\,b\,\sin^2\gamma}\cos\gamma+\frac{(T_C^{(k)}-T_B^{(k)})^2}{a^2\,\sin^2\gamma}\geq f_C^2, \tag{3.31}$$

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

$$\begin{aligned} F(T_A^{(k)},T_B^{(k)}) \;=\; & \frac{(T_C^{(k)}-T_A^{(k)})^2}{b^2\,\sin^2\gamma}-2\frac{(T_C^{(k)}-T_A^{(k)})(T_C^{(k)}-T_B^{(k)})}{a\,b\,\sin^2\gamma}\cos\gamma \\ & +\frac{(T_C^{(k)}-T_B^{(k)})^2}{a^2\,\sin^2\gamma}-f_C^2 \end{aligned} \tag{3.32}$$

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

$$2\max\left\{\frac{|T_C^{(k)}-T_A^{(k)}|}{b^2\,\sin^2\gamma}+\frac{|T_C^{(k)}-T_B^{(k)}|}{a\,b\,\sin^2\gamma}\cos\gamma,\frac{|T_C^{(k)}-T_B^{(k)}|}{a^2\,\sin^2\gamma}+\frac{|T_C^{(k)}-T_A^{(k)}|}{a\,b\,\sin^2\gamma}\cos\gamma\right\}. \tag{3.33}$$

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 k-th iteration. By the l^∞-contraction property and the convergence property of $T_C^{(k)}$, $\delta^{(k)}$ converges monotonically to zero. After the k-th iteration, we have

$$\begin{aligned} 0 \;\leq\; & \frac{(T_C^{(k)}-T_A^{(k)})^2}{b^2\,\sin^2\gamma}-2\frac{(T_C^{(k)}-T_A^{(k)})(T_C^{(k)}-T_B^{(k)})}{a\,b\,\sin^2\gamma}\cos\gamma+\frac{(T_C^{(k)}-T_B^{(k)})^2}{a^2\,\sin^2\gamma}-f_C^2 \\ \leq\; & Z\delta^{(k)}. \end{aligned} \tag{3.34}$$

Thus $T^{(k)}$ converges to the solution to equation (3.5). $\Box$

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 that value will not change in later iterations.

Now we show the estimate for the total number of iterations that is needed for convergence. As pointed out before, 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 3.9(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 characteristics is related to 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:

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

Since $|\dot{\mathbf{x}}| = 1$, it was shown in [29] the curvature bound along a characteristic is:

$$|\ddot{\mathbf{x}}| \leq \left| \frac{\nabla f(\mathbf{x})}{f(\mathbf{x})} \right|. \tag{3.35}$$

Theorem 3.9 *Assuming that $f(\mathbf{x})$ is strictly positive and C^1 in Ω, the total variation of the tangent direction of a characteristic in Ω is bounded by:*

$$\int_\gamma |\ddot{\mathbf{x}}| ds \leq \frac{DKf_M}{f_m}, \tag{3.36}$$

where s is the arclength, 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. Let γ be any characteristic, from (3.35) we have

$$\int_\gamma |\ddot{\mathbf{x}}| ds \leq \int_\gamma \frac{|\nabla f(\mathbf{x})|}{f(\mathbf{x})} ds \leq K \int_\gamma ds \tag{3.37}$$

where s is the arc length. Let the characteristic γ join a point $\mathbf{x}_0 \in \Gamma$ from the initial front to a point $\mathbf{x} \in \Omega$ in the domain; see Figure 3.9(b). The travel-time at $\mathbf{x}$ is $T(\mathbf{x}) = \int_\gamma f(s)ds$. 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

$$f_m \int_\gamma ds \leq \int_\gamma f(s)ds = T(\mathbf{x}) \leq \int_{\mathbf{x}_0}^{\mathbf{x}} f(s)ds \leq f_M \|\mathbf{x} - \mathbf{x}_0\|. \tag{3.38}$$

Hence

$$\text{length}(\gamma) = \int_\gamma ds \leq \frac{Df_M}{f_m}. \tag{3.39}$$

Together with (3.37) we finish the proof. □

Hence the maximal number of sweeping needed to cover all characteristics can be bounded by $C \times \dfrac{DKf_M}{f_m}$, where the constant C may depend on the number of reference points and orderings, but it is independent of the mesh size.

4 Numerical Examples

Now we show numerical examples in both two and three dimensions to illustrate the efficiency and accuracy of our algorithm. In all the examples we have used the quicksort 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 domains or eight corners in 3-D domains as the reference points are

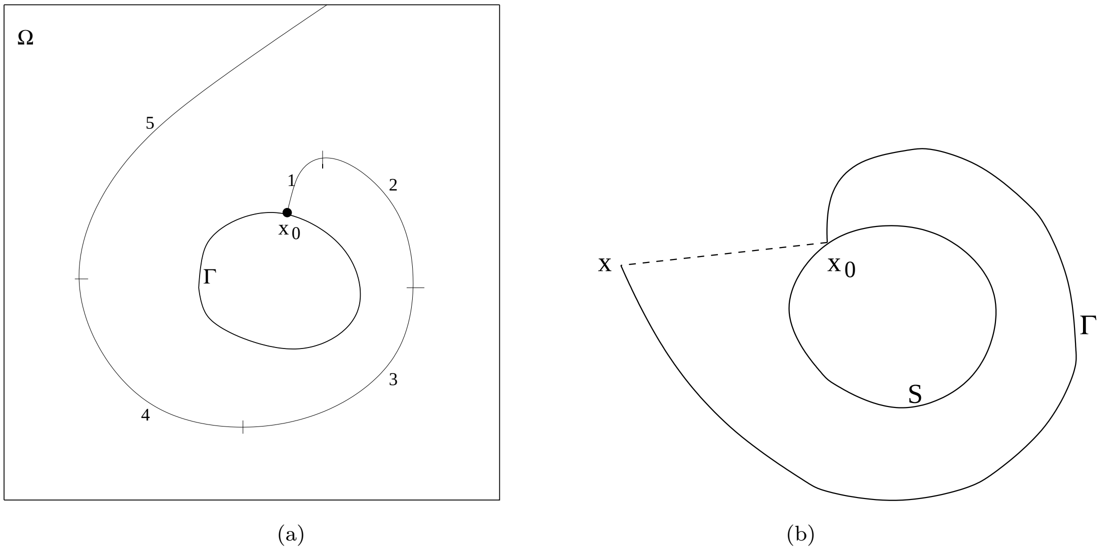


Figure 3.9: Partitioning of a characteristic.

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. The number of iterations is independent of mesh size for all the examples shown here. If we have an irregular computational domain, we may also add more reference points to fit with the irregular geometry; however, we will deal with this issue in the future work.

In all the examples, the convergence of iteration is measured as full convergence, i.e., the iteration stops when the successive error reaches machine zero. Figure 4.1 shows the typical behavior of convergence error, the difference between two consecutive iterations in maximum norm, for the fast sweeping method. It shows that we get the exact solution (up to machine error) to the discrete system in a finite number of iterations independent of mesh size. It is not like the convergence behavior with certain contraction rate.

4.1 2-D acute triangulation

We first triangulate the computational domain. A typical acute triangulation is shown in Figure 4.2, and the refinement of the mesh is uniform, i.e., 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 l^1 and l^2 based sortings. We have also used two reference points in the case of l^1- metric based sorting. First we describe the examples.

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 4.3.

Example 2 (shape-from-shading). This example is taken from [21]. The eikonal equation (2.1) with

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

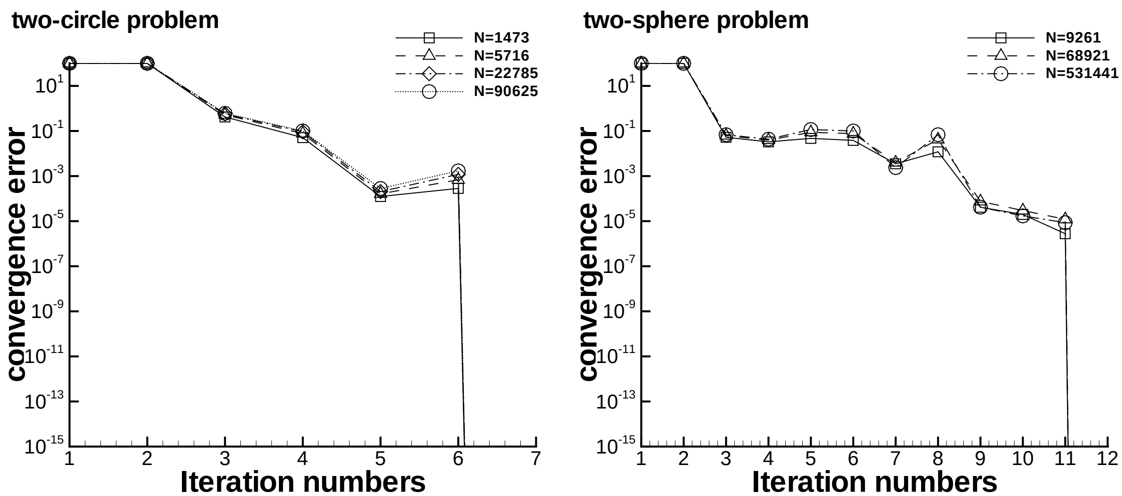


Figure 4.1: log plot of convergence error

$\Gamma = \{(\frac{1}{4},\frac{1}{4}),(\frac{3}{4},\frac{3}{4}),(\frac{1}{4},\frac{3}{4}),(\frac{3}{4},\frac{1}{4}),(\frac{1}{2},\frac{1}{2})\}$, consisting of five isolated points. The computational domain $\Omega = [0,1]\times[0,1]$. $T(x,y) = 0$ is prescribed at 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.

Case a.

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

The exact solution for this case is

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

a smooth function.

Case b.

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

The exact solution for this case is

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

this solution is *not* smooth.

We have used acute triangulations for both cases. The solutions are shown in Figure 4.4.

Example 3 (five-ring problem) The eikonal equation (2.1). The computational domain is $\Omega = [0,1]\times[0,1]$; Γ is the point source $(0,0)$, and five ring obstacles are placed in the computational domain. This is an example borrowed from [9]. Here we also use an acute triangulation. The solution is shown in Figure 4.5.

From Table 4.1, we can see that the accuracy of the algorithm for Examples 1 and 2 is first order. Although no matter which ordering metric is used, the same discretized

Table 4.1: Accuracy tests for Examples 1 and 2. Acute triangulation.

		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 sweeping wavefront based on l^2 metric ordering.

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

nonlinear system is solved. However, different ordering strategies may result in different numbers of iterations, as shown in Table 4.2 and Table 4.3. 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 vertices 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 sweeping wavefront based on l^1 metric ordering.

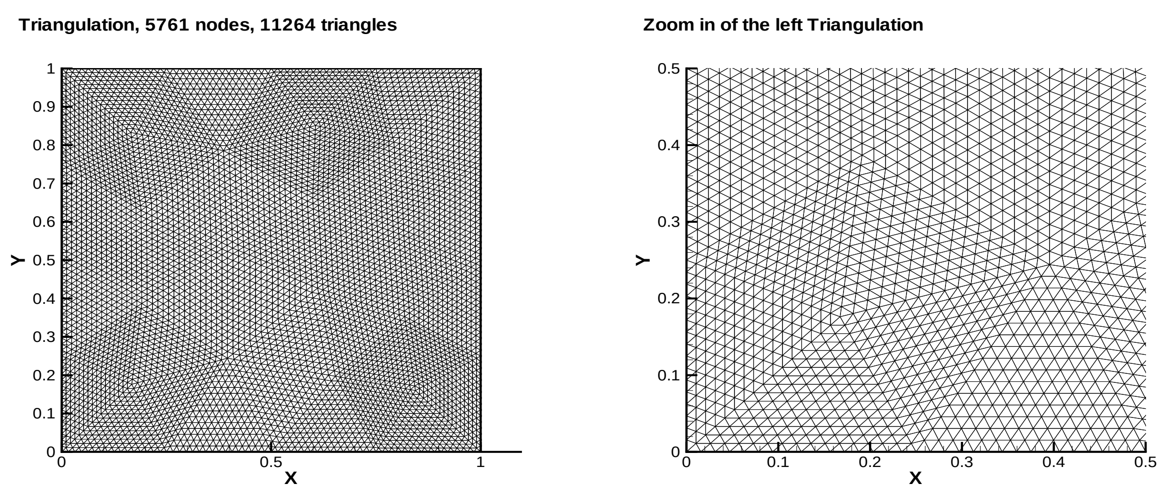
Nodes	Elements	two-circle	shape (case a)	shape (case b)	five-ring
1473	2816	7	12	9	26
5716	11264	7	12	9	27
22785	45056	7	16	9	27
90625	180224	7	15	9	27

Table 4.4: Iteration numbers for Examples 1, 2 and 3. Acute triangulation. Planar sweeping wavefront based on 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.

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



(a) (b)

Figure 4.2: An acute triangulation. (a): the whole mesh; (b): zoom in.

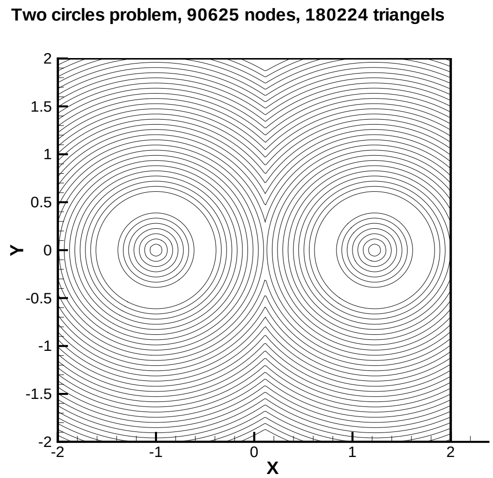


Figure 4.3: Example 1: two-circle problem. Acute triangulation. 30 equally spaced contour lines from $T = 0.111755$ to $T = 1.6762$.

4.2 2-D obtuse triangulation

In this section, we test our strategy for treating a triangulation which has obtuse angles. The obtuse triangulation is constructed by perturbing randomly the x coordinates of vertices in a uniform triangulation. This 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 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 iteration numbers for the algorithm without and with the 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 iteration numbers for the algorithm without and with the treatment are listed in Table 4.7. Comparing to Table 4.6, 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. The first order accuracy with the treatment is more regular than that without the treatment. Moreover, comparing the errors in Table 4.6 with those in Table 4.7, without the obtuse-angel treatment different sweeping ordering strategies yield slightly different numerical solutions; but with the obtuse-angel 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.

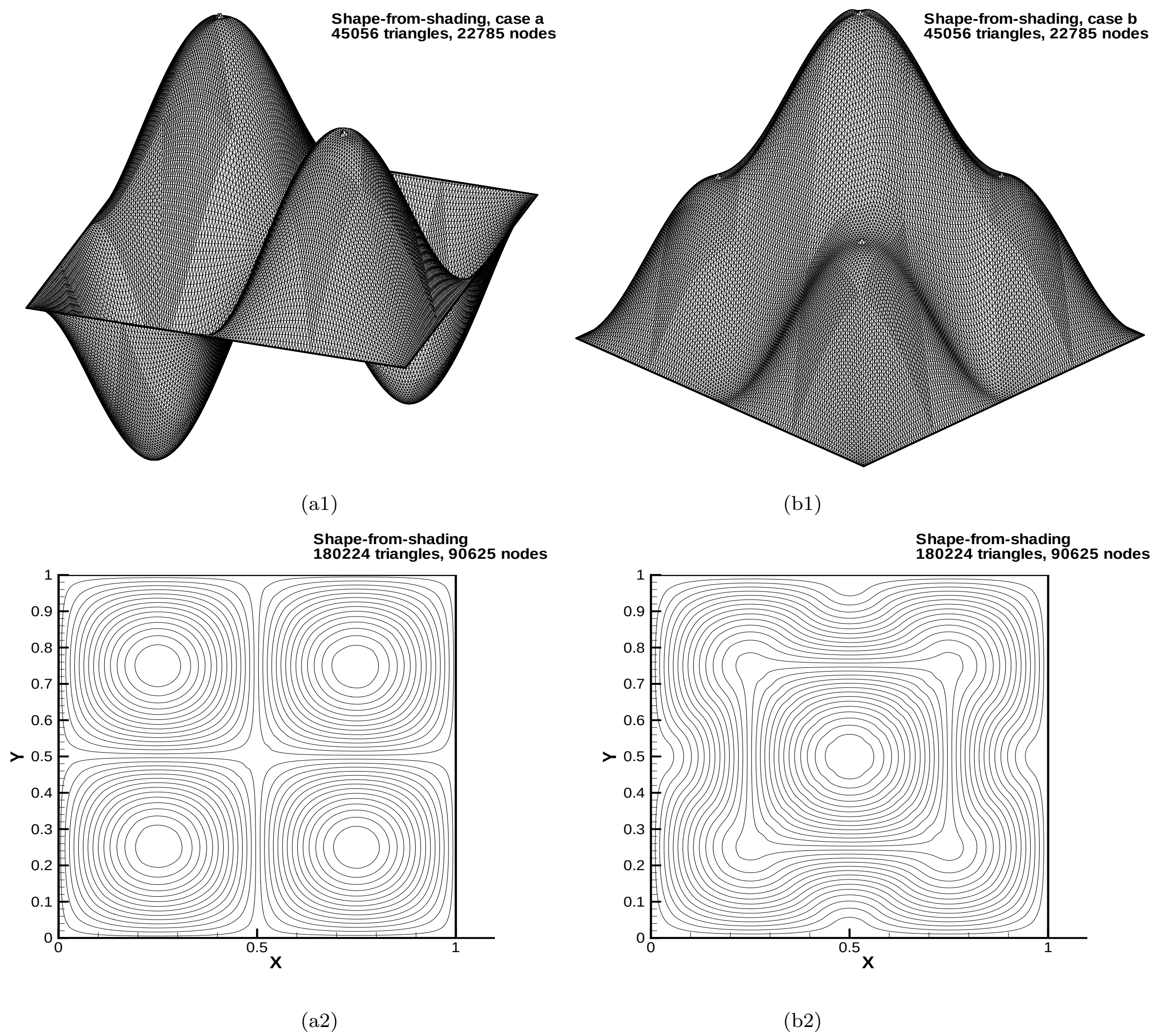


Figure 4.4: Example 2: shape-from-shading. Acute triangulation. Left: case (a); right: case (b); top: three-dimensional view; bottom: contour lines, 30 equally spaced contour lines from $T = -1$ to $T = 1$ for case (a), and from $T = 0$ to $T = 2$ for case (b).

Table 4.6: Two-circle problem. Obtuse triangulation. Spherical wave sweepings: 4 reference points (4 corners of computational domain). Both ascent and descent orderings.

			before treatment			after treatment		
Elements	Obtuse ele	max obtu	L^1 error	order	ite	L^1 error	order	ite
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

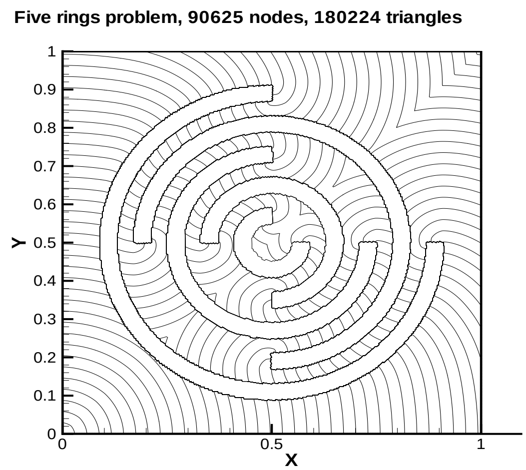
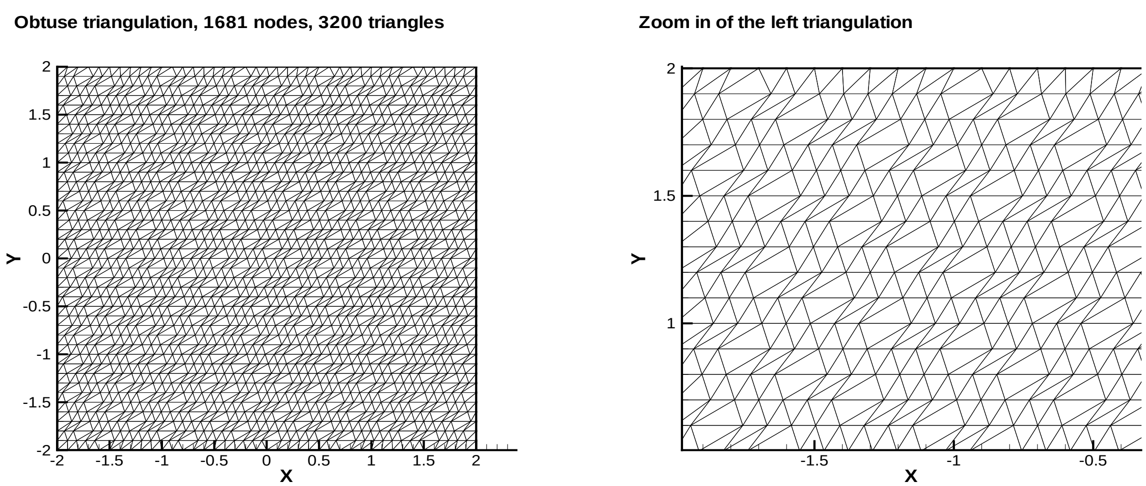


Figure 4.5: Example 3: five-ring problem. Acute triangulation. 100 equally spaced contour lines from $T = 0$ to $T = 2.89$.



(a)

(b)

Figure 4.6: An obtuse triangulation. (a): the whole mesh; (b): zoom in.

Table 4.7: Two-circle problem. Obtuse triangulation. Spherical wave sweepings: 8 reference points (4 corners and 4 middle points of the 4 sides of computational domain). Only ascent ordering.

			before treatment			after treatment		
Elements	Obtuse ele	max obtu	L^1 error	order	ite	L^1 error	order	ite
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

4.3 A 3-D example

In this section 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 $f(x, y, z) = 1$.

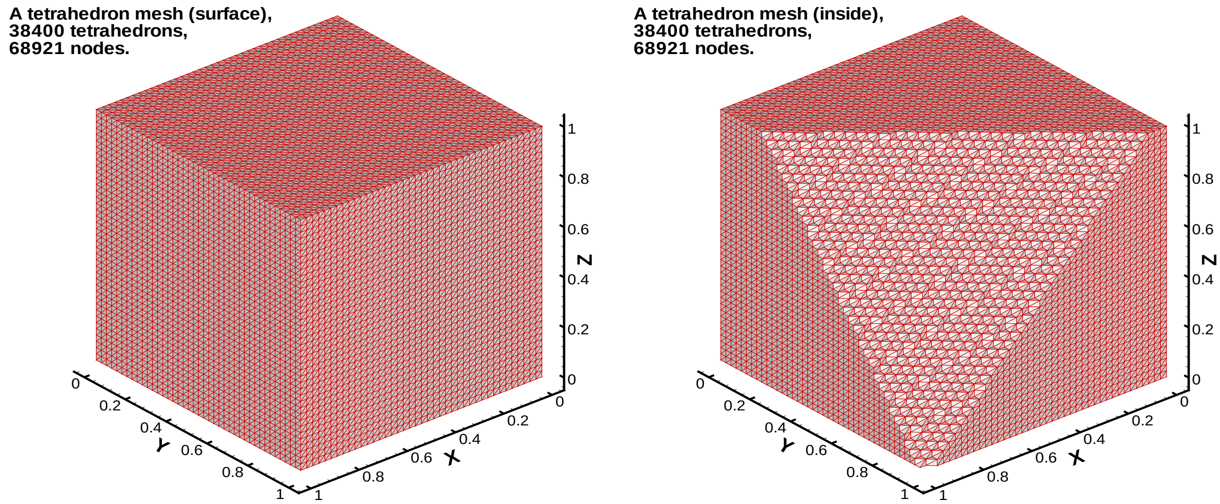
The computational domain is $\Omega = [0, 1] \times [0, 1] \times [0, 1]$; Γ consists of two spheres of equal radius 0.1 with centers located at $(0.25, 0.25, 0.25)$ and $(0.75, 0.75, 0.75)$, respectively. The exact solution is the distance function to Γ.

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

Figure 4.7 shows a tetrahedral mesh obtained in this way from a $40 \times 40 \times 40$ rectangular mesh. So the total tetrahedra in the mesh is $40 \times 40 \times 40 \times 6 = 384000$. Figure 4.7(a) shows the surface of the mesh, and Figure 4.7(b) shows a part of the interior of the tetrahedral mesh. 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.

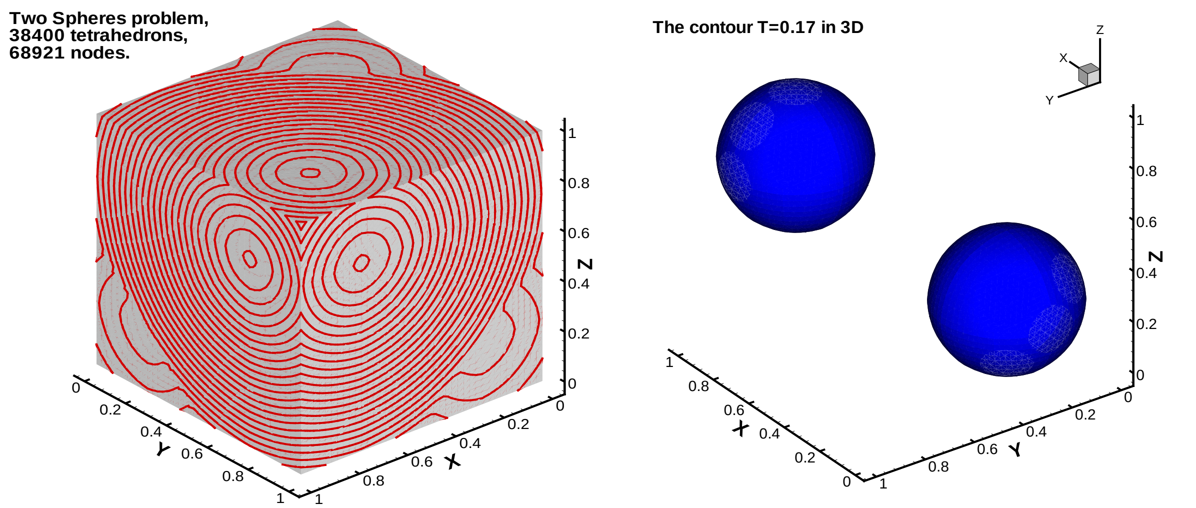
The results in Figure 4.7 are obtained by using the mesh in Figure 4.6. Figure 4.8(a) shows the contour plot of the solution on the surface of the domain, and Figure 4.8(b) shows 3-D plots of the contour $T = 0.17$.

In Table 4.8, 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 grid as in [29]. Although the nodes are the same, the local solver at each node is different and hence the discretized nonlinear systems of equations are different. The comparison results are also shown in Table 4.8. It is obvious from the table that the local solver on unstructured meshes can achieve better 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 4.8 that if the l^2 distance is used for ordering, the iteration number on an unstructured mesh can be less than that on a structured one. However the local solver at each node for the unstructured mesh is more expensive than for the 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) (b)

Figure 4.7: A tetrahedral mesh. (a): surfaces of the mesh; (b): a part of the interior of the mesh.



(a) (b)

Figure 4.8: Two-sphere problem. Using the tetrahedral mesh shown in Figure 4.7. (a): the surface contour, 30 equally spaced contour lines from $T = 0$ to $T = 0.742402$; (b): the contour plot of $T = 0.17$ in the 3-D case.

Table 4.8: 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	ite	L^1 error	order	ite
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

5 Conclusion

We proposed novel ordering strategies to extend the fast sweeping method to unstructured meshes. To that end we introduced multiple reference points and ordered all the nodes according to their l^p distances to those reference points. Information propagation along all characteristics can be covered efficiently by the systematic orderings. The proved convergence results established that the new 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 $O(MlogM)$ 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 demonstrated accuracy and efficiency of the new fast sweeping method.

References

[1] M. Bardi and S. J. Osher. The nonconvex multi-dimensional Riemann problem for Hamilton-Jacobi equations. *SIAM J. Math. Anal.*, 22:344–351, 1991.

[2] M. Boue and P. Dupuis. Markov chain approximations for deterministic control problems with affine dynamics and quadratic costs in the control. *SIAM J. Numer. Anal.*, 36:667–695, 1999.

[3] B. Cockburn and J. Qian. Continuous dependence results for Hamilton-Jacobi equations. Collected lectures on the preservation of stability under discretization. Edited by D. Estep and S. Tavener. SIAM, Philadelphia, PA, 2002, pp.67-90., 2002.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms.* The MIT Press, Cambridge, MA., 1990.

[5] R. Courant and D. Hilbert. *Methods of mathematical physics*, volume II. John Wiley-Sons, 1962.

[6] M. G. Crandall and P. L. Lions. Two approximations of solutions of Hamilton-Jacobi equations. *Math. Comp.*, 43:1–19, 1984.

[7] J. Dellinger and W. W. Symes. Anisotropic finite-difference traveltimes using a Hamilton-Jacobi solver. In *67th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts*, pages 1786–1789. Soc. Expl. Geophys., 1997.

[8] M. Falcone and R. Ferretti. Discrete time high-order schemes for viscosity solutions of Hamilton-Jacobi-Bellman equations. *Numer. Math.*, 67:315–344, 1994.

[9] P. A. Gremaud and C. M. Kuster. Computational study of fast methods for the eikonal equations. Submitted to SIAM J. Sci. Comp., 2004.

[10] J. Helmsen, E. Puckett, P. Colella, and M. Dorr. Two new methods for simulating photolithography development in 3-D. *Proc. SPIE*, 2726:253–261, 1996.

[11] C.Y. Kao, S.J. Osher, and J. Qian. Lax-Friedrichs sweeping schemes for static Hamilton-Jacobi equations. *J. Comp. Phys.*, 196:367–391, 2004.

[12] S. Kim and R. Cook. 3-D traveltime computation using second-order ENO scheme. *Geophysics*, 64:1867–1876, 1999.

[13] R. Kimmel and J. A. Sethian. Computing geodesic paths on manifolds. *Proc. Natl. Acad. Sci. USA*, 95:8431–8435, 1998.

[14] S. J. Osher and C. W. Shu. High-order Essentially NonOscillatory schemes for Hamilton-Jacobi equations. *SIAM J. Num. Anal.*, 28:907–922, 1991.

[15] J. Qian and W. W. Symes. Paraxial eikonal solvers for anisotropic quasi-P traveltimes. *J. Comput. Phys.*, 173:1–23, 2001.

[16] J. Qian and W. W. Symes. Adaptive finite difference method for traveltime and amplitude. *Geophysics*, 67:167–176, 2002.

[17] J. Qian and W. W. Symes. Finite-difference quasi-P traveltimes for anisotropic media. *Geophysics*, 67:147–155, 2002.

[18] F. Qin, Y. Luo, K. B. Olsen, W. Cai, and G. T. Schuster. Finite difference solution of the eikonal equation along expanding wavefronts. *Geophysics*, 57:478–487, 1992.

[19] F. Qin and G. T. Schuster. First-arrival traveltime calculation for anisotropic media. *Geophysics*, 58:1349–1358, 1993.

[20] R. Rawlinson and M. Sambridge. Wave front evolution in strongly heterogeneous layered media using the fast marching method. *Geophys. J. Internat.*, 156:631–647, 2004.

[21] E. Rouy and A. Tourin. A viscosity solutions approach to shape-from-shading. *SIAM J. Num. Anal.*, 29:867–884, 1992.

[22] W. A. Jr. Schneider, K. Ranzinger, A. Balch, and C. Kruse. A dynamic programming approach to first arrival traveltime computation in media with arbitrarily distributed velocities. *Geophysics*, 57:39–50, 1992.

[23] J. A. Sethian. *Level set methods*. Cambridge Univ. Press, 1996.

[24] R. Tsai, L.-T. Cheng, S. J. Osher, and H. K. Zhao. Fast sweeping method for a class of Hamilton-Jacobi equations. *SIAM J. Numer. Anal.*, 41:673–694, 2003.

[25] J.N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Tran. Automatic Control*, 40:1528–1538, 1995.

[26] J. van Trier and W. W. Symes. Upwind finite-difference calculation of traveltimes. *Geophysics*, 56:812–821, 1991.

[27] J. Vidale. Finite-difference calculation of travel times. *Bull. Seis. Soc. Am.*, 78:2062–2076, 1988.

[28] Y. T. Zhang, H. K. Zhao, and J. Qian. High order fast sweeping methods for static Hamilton-Jacobi equations. UCLA CAM 04-37; J. Sci. Comp. (under revision), 2004.

[29] H. K. Zhao. Fast sweeping method for eikonal equations. *Math. Comp.*, 74:603–627, 2005.

[30] H.K. Zhao, S. Osher, B. Merriman, and M. Kang. Implicit and non-parametric shape reconstruction from unorganized points using variational level set method. *Computer Vision and Image Understanding*, 80:295–319, 2000.