FEM Fast Marching Methodによるボート航行距離の計算

西田徹志 [*]　杉原厚吉 [*]

概要

流れのある水面上を航行するボートの振舞いをモデル化することによって，ボート航行距離という概念を導入する．そして，この距離を計算する問題を偏微分方程式の境界値問題に定式化し，得られた境界値問題を解く数値計算法を構成する．その方法はeikonal方程式に対して提案されたfast marching法を改良したものである．数値実験により，提案した方法が効率性よく安定に動作することを示す．

FEM-like Fast Marching Method for the Computation of the Boat-Sail Distance

Tetsushi Nishida[*]　and　Kokichi Sugihara[*]

Abstract

A new concept called a boat-sail distance is introduced on the surface of water with flow. The problem of computing this distance is reduced to a boundary value problem of a partial differential equation, and a numerical method for solving this problem is constructed. The method is a modification of a so-called fast marching method originally proposed for the eikonal equation. Computational experiments show the efficiency and the stableness of the proposal method.

1 Introduction

Fast marching methods [4] are computational techniques that approximate the solutions to nonlinear eikonal equations of the form

$$F(x)|\nabla T(x)| = 1, \quad x \in \Omega, \quad F(x) > 0,$$
$$T(x) = g(x), \quad x \in \Gamma,$$

where Ω is a domain in $\mathbf{R}^2$ or $\mathbf{R}^3$, $F(x)$ is a known input function, and $g(x)$ is a known function representing the boundary condition. Here, $T(x)$ represents the arrival time at each point x.

In general, the solution of the eikonal equation is not unique, and in addition, is not differentiable, even if boundary data are smooth. The fast marching method is a numerical method which can handle this nondifferentiability efficiently and stably, and thus can construct physically correct nonsmooth solutions.

These methods are of use in a variety of applications, computing distances from complex curves and surfaces [5], shape-from-shading [5], photolithographic development [5], computing first arrivals in seismic travel times [6], construction of shortest geodesics on surfaces [1], optimal path planning around obstacles, and visibility and reflection calculations [4].

However, for a certain type of problems, the accuracy of the solutions computed by the fast marching methods is much less than the accuracy we want. The computation of a boat-sail distance, which we propose in this paper, is exactly one of such kind of problems.

Suppose that we want to travel on the surface of water with a boat. If there is no flow of water, the boat can move in any direction at the same maximum speed. If the water flows, on the other hand, the speed of the boat is anisotropic; the boat can move faster in the same direction as the flow, while it move only slowly in the direction opposite to the flow direction. Modeling this situation, we can introduce the boat-sail distance.

In order to construct a numerical method for computing the boat-sail distances, we reduce the problem to a boundary value problem of a partial differential equation. This idea is the same as the idea for reducing the problem of computing the Euclidean distance was reduced to a boundary value

[*]東京大学大学院情報理工学系研究科数理情報学専攻
Department of Mathematical Informatics, Graduate School of Information Sience and Technology, University of Tokyo
{nishida,sugihara}@mist.i.u-tokyo.ac.jp

problem of the eikonal equation [4]. Hence, our formulation can be considered a generalization of the eikonal equation. Therefore we applied the idea of the fast marching method to our equation, but we found that it did not work well.

In this paper, we consider the reasons why the original fast marching method does not work well, and extend the scheme of the fast marching method in such a way that it works very well for this kind of problems and show the efficiency and the stableness of our proposal method.

In Section 2, we introduce a mathematical model for the boat-sail distance. In Section 3, we derive a partial differential equation for representing the boat-sail distance. In Section 4, we construct a new method, which is a combination of the fast marching method and the finite-element method. In Section 5, we show some numerical examples. Finally, we give concluding remarks in Section 6.

2 Boat-Sail Distance

Let $\Omega \subset \mathbf{R^2}$ denote a two-dimensional domain with an (x, y) Cartesian coordinate system, and let $f(x, y) \in \mathbf{R^2}$ be a two-dimensional vector given at each point (x, y) in Ω. A physical interpretation is that Ω corresponds to the surface of water and $f(x, y)$ represents the velocity of the water flow. Hence, we call $f(x, y)$ the flow field. We assume that $f(x, y)$ is continuous in Ω.

Consider a boat that has the maximum speed F in any direction on the still water. Let Δt denote a short time interval. Suppose that the driver tries to move the boat at speed F in the direction v_F, where v_F is the unit vector, and hence the boat will move from the current point p to $p + \Delta t F v_F$ in time Δt if there is no water flow, as shown by the broken arrow in Fig 1. However, the flow of water also displaces the boat by $\Delta t f(x, y)$, and hence the actual movement Δu of the boat in time interval Δt is represented by $\Delta u = \Delta t F v_F + \Delta t f(x, y)$.

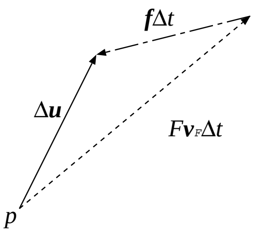

Fig. 1. Relations among the actual movement Δu of the boat, the water flow f and the velocity Fv_F of the boat.

Consequently, the effective speed of the boat in the water flow in given by

$$\left|\frac{\Delta u}{\Delta t}\right| = \left|F v_F + f(x, y)\right|. \tag{1}$$

We assume that F is large enough to satisfy the condition $F > \max_{(x,y)\in\Omega} |f(x, y)|$. Hence, the boat can move in any direction against the flow even if the direction of the boat sailing is opposite to the direction of the flow.

Let p and q be two points in Ω, and let $c(s) \in \Omega$ denote a curve from p to q with the arc-length parameter s $(0 \leq s \leq \overline{s})$ such that $c(0) = p$ and $c(\overline{s}) = q$. Then, the time, say $\delta(c, p, q)$, necessary for the boat to move from p to q along the curve $c(s)$ with the maximum speed is obtained by

$$\delta(c, p, q) \equiv \int_0^{\overline{s}} \frac{1}{\left|\frac{\Delta u}{\Delta t}\right|} \mathrm{d}s = \int_0^{\overline{s}} \frac{1}{|F v_F + f(x, y)|} \mathrm{d}s. \tag{2}$$

Let C be the set of all paths from p to q. We define $d(p, q)$ by

$$d(p, q) \equiv \inf_{c\in C} \delta(c, p, q). \tag{3}$$

That is, $d(p, q)$ represents the shortest time necessary for the boat to move from p to q. We can consider that $d(p, q)$ is proportional to the effective distance from p to q, and hence we abuse the term "distance" and call $d(p, q)$ the *boat-sail distance* from p to q. Note that $d(p, q)$ is not symmetric; the time necessary to move from p to q is not in general equal to the time necessary to move from q to p.

3 Reduction to a Boundary Value Problem

Suppose that we are given the flow field $f(x, y)$ and the point $p_0 = (x_0, y_0)$, called a boat harbor, in Ω. Let $T(x, y)$ be the shortest arrival time at which the boat departing p_0 at time 0 can reach the point $p = (x, y)$, that is, $T(x, y) \equiv d(p_0, p)$.

In this section, we derive the partial differential equation that should be satisfied by the unknown function $T(x, y)$.

Let C be an arbitrary positive constant. The equation $T(x, y) = C$ represents a curve, any point on which can be reached in time C by the boat departing p_0 at time 0. As shown in Fig. 2, assume that the boat moving along the shortest path passes through the point (x, y) at time C and reaches the point $(x + \Delta x, y + \Delta y)$ at time $C + \Delta t$, where Δt is positive and small. Hence, in particular, we get

$$T(x + \Delta x, y + \Delta y) - T(x, y) = \Delta t. \tag{4}$$

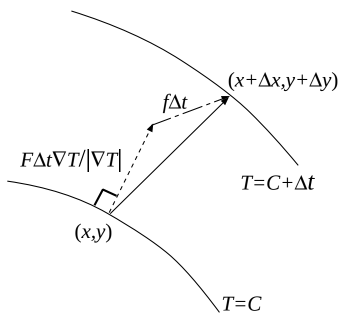

Fig. 2. Decomposition of the movement of a boat.

If there is no flow of water, the shortest path should be perpendicular to the curve $T = C$, and hence, the progress of the boat during time interval Δt is represented by $F\frac{\nabla T}{|\nabla T|}\Delta t$. On the other hand, the displacement of the boat caused by the flow of water is $f\Delta t$. Hence, the total motion of the boat is represented by

$$F\frac{\nabla T}{|\nabla T|}\Delta t + f\Delta t. \qquad (5)$$

Let us denote $T_x \equiv \frac{\partial T}{\partial x}$ and $T_y \equiv \frac{\partial T}{\partial y}$, respectively. Also let $g(x,y)$ and $h(x,y)$ denote the first and second components of $f(x,y)$. Then from the equation (5), we get

$$\Delta x = F\frac{T_x}{|\nabla T|}\Delta t + g\Delta t, \quad \Delta y = F\frac{T_y}{|\nabla T|}\Delta t + h\Delta t,$$

and linearly approximating $T(x+\Delta x, y+\Delta y)$, we get an approximate expression:

$$T(x+\Delta x, y+\Delta y) \approx T(x,y) + T_x(F\frac{T_x}{|\nabla T|}+g)\Delta t + T_y(F\frac{T_y}{|\nabla T|}+h)\Delta t. \quad (6)$$

Therefore, from this expression (6) and equation (4), we obtain

$$F|\nabla T| = 1 - \nabla T \cdot f. \qquad (7)$$

This is the partial differential equation that should be satisfied by the arrival time $T(x,y)$.

In the next section, we consider how to solve this partial differential equation numerically, together with the boundary condition

$$T(x_0, y_0) = 0. \qquad (8)$$

4 FEM-like Fast Marching Method and the Algorithm

Our partial differential equation is quadratic, but not linear. Hence, we cannot use the numerical method such as the finite difference method and so on. On the other hand, our equation has the property that the arrival time $T(x,y)$ is monotone increasing as we move along the shortest paths starting at p_0. A typical equation of this type is the eikonal equation [4]. This equation can be solved efficiently and stably by the fast marching method [4].

However, from numerical experiments [2, 3], we recognize that the fast marching method did not work for our equation. Hence, in order to fulfill our purposes, we propose a new scheme by modifying the fast marching method.

4.1 FEM-like Differences

In Ω, we place grid points $(x_i, y_j) = (i\Delta x, j\Delta y)$, $i, j = 0, \pm1, \pm2, \cdots$, where Δx and Δy are small constants and i and j are integers. For each grid point (x_i, y_j), we associate $T_{ij} = T(x_i, y_j)$. $T_{00} = T(x_0, y_0) = 0$ because of the boundary condition (8), while all the other T_{ij}'s are unknown variables.

Starting with the neighbors of (x_0, y_0), we want to compute T_{ij}'s grid by grid from smaller values to larger values. Hence, we use the modified upwind differences which we explain as follows.

In the finite element method, the domain $\Omega \subset \mathbf{R}^2$ is divided into many small regions called finite elements, which are triangles or rectangles, and the value of an interior point of a finite element is interpolated from the values on the vertices and on the edges of the finite element.

Consider a triangular element shown in Fig. 3(a), where the coordinates of nodes 1, 2 and 3 are (x_1, y_1), (x_2, y_2) and (x_3, y_3), respectively and nodes 4, 5 and 6 are the middle points of the edges. Let $T_1, T_2, \ldots, T_6$ be the values at the nodes $1, 2, \ldots, 6$, respectively. Then, the interpolation function T which represents the value at point (x,y) in the triangular element is represented by

$$T(x,y) = T_1\phi_1(2\phi_1-1)+T_2\phi_2(2\phi_2-1) + T_3\phi_3(2\phi_3-1)+4T_4\phi_2\phi_3+4T_5\phi_3\phi_1+4T_6\phi_1\phi_2, \qquad (9)$$

where $\phi_1 = \phi_1(x,y)$, $\phi_2 = \phi_2(x,y)$ and $\phi_3 = \phi_3(x,y)$ are the area coordinate functions:

$$\phi_i(x,y) = \frac{1}{D}(a_i + b_i x + c_i y), \qquad (10)$$

where

$$D = \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix},$$

and

$$\begin{aligned} a_1 &= x_2y_3 - x_3y_2, & b_1 &= y_2 - y_3, & c_1 &= x_3 - x_2, \\ a_2 &= x_3y_1 - x_1y_3, & b_2 &= y_3 - y_1, & c_2 &= x_1 - x_3, \\ a_3 &= x_1y_2 - x_2y_1, & b_3 &= y_1 - y_2, & c_3 &= x_2 - x_1. \end{aligned}$$

The functions $\phi_1\phi_2$, $\phi_2\phi_3$, $\phi_3\phi_1$ and $\phi_i(2\phi_i-1)$ for $i=1,2,3$ are called shape functions.

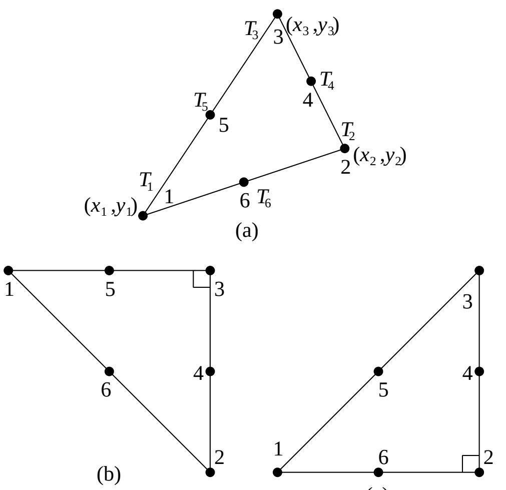

Fig. 3. Examples of triangular finite elements: the number $1,2,\ldots,6$ are nodes and $T_1,T_2,\ldots T_6$ are the values at the nodes.

Next, partially differentiating this interpolation function, we get the partial derivatives:

$$\frac{\partial T}{\partial x}(x,y) = \frac{(b_1T_1+b_2T_6+b_3T_5)\phi_1+(b_1T_6+b_2T_2+b_3T_4)\phi_2}{D} + \frac{(b_1T_5+b_2T_4+b_3T_3)\phi_3-(b_1T_1+b_2T_2+b_3T_3)}{D}, \tag{11}$$

$$\frac{\partial T}{\partial y}(x,y) = \frac{(c_1T_1+c_2T_6+c_3T_5)\phi_1+(c_1T_6+c_2T_2+c_3T_4)\phi_2}{D} + \frac{(c_1T_5+c_2T_4+c_3T_3)\phi_3-(c_1T_1+c_2T_2+c_3T_3)}{D}. \tag{12}$$

In order to know the values of the partial derivatives at the node 3, we substitute (x_3,y_3) into (11) and (12), and get

$$\frac{\partial T}{\partial x}(x_3,y_3) = \frac{3b_3T_3+4(b_2T_4+b_1T_5)-b_1T_1-b_2T_2}{D}, \tag{13}$$

$$\frac{\partial T}{\partial y}(x_3,y_3) = \frac{3c_3T_3+4(c_2T_4+c_1T_5)-c_1T_1-c_2T_2}{D}. \tag{14}$$

Assume that the angle at the node 3 is right as shown in Fig. 3(b), and that the length of the horizontal and vertical sides adjacent to the node 3 are $2\Delta x$ and $2\Delta y$, respectively. Then, the partial derivatives become

$$\begin{aligned}\frac{\partial T}{\partial x}(x_3,y_3) &= \frac{3T_3-4T_5+T_1}{2\Delta x},\\ \frac{\partial T}{\partial y}(x_3,y_3) &= \frac{3T_3-4T_4+T_2}{2\Delta y}.\end{aligned}$$

These derivatives coincide with the usual second-order upwind-like differences in the fast marching method. Thus, the equations (13) and (14) are generalizations of upwind-like differences. Hence, for instance, in the case where the node 2 forms a right angle as shown in Fig. 3 (c), the upwind-like differences at the node 3 are obtained by

$$\begin{aligned}\frac{\partial T}{\partial x}(x_3,y_3) &= \frac{4(T_4-T_5)-(T_2-T_1)}{2\Delta x},\\ \frac{\partial T}{\partial y}(x_3,y_3) &= \frac{3T_3-4T_4+T_2}{2\Delta y}.\end{aligned}$$

We call (13) and (14) the *second order FEM-like differences.*

However, if $T_1 < T_5$ or $T_2 < T_4$, the second order difference cannot be used [5], and so we have to prepare the first order difference. Consider a triangular element whose vertices are nodes 3, 4 and 5, as shown in Fig. 3. Then the interpolation function T^1 is represented by

$$T^1(x,y) = T_3\phi_3(x,y)+T_4\phi_4(x,y)+T_5\phi_5(x,y), \tag{15}$$

where ϕ_3, ϕ_4 and ϕ_5 are the area coordinate functions; we get these functions by replacing the nodes 1 and 2 with the nodes 5 and 4 in the equation (10), respectively. Considering the differences of the equation (15) in the same manner as the second order difference, we get

$$\frac{\partial T^1}{\partial x}(x_3,y_3) = \frac{b_3'T_3+b_4'T_4+b_5'T_5}{D'}, \tag{16}$$

$$\frac{\partial T^1}{\partial y}(x_3,y_3) = \frac{c_3'T_3+c_4'T_4+c_5'T_5}{D'}, \tag{17}$$

where

$$D' = \begin{vmatrix} 1 & x_3 & y_3 \\ 1 & x_4 & y_4 \\ 1 & x_5 & y_5 \end{vmatrix},$$

and

$$\begin{aligned} b_3' &= y_5-y_4, & c_3' &= x_4-x_5.\\ b_4' &= y_3-y_5, & c_4' &= x_5-x_3,\\ b_5' &= y_4-y_3, & c_5' &= x_3-x_4.\end{aligned}$$

We call the equations (16) and (17) the *first order FEM-like differences.*

Next, we combine the first and second differences and construct operators similar to upwind operators [3]. Let $\triangle_{123}$ be the triangle whose vertices are the nodes 1, 2 and 3, and $\triangle_{543}$ be the triangle whose vertices are the nodes 5, 4 and 3. Since the triangle $\triangle_{123}$ is similar to the triangle $\triangle_{543}$ and, in addition, the length of each edge of $\triangle_{123}$ is twice as that of the corresponding edge of $\triangle_{543}$, we get

$$D' = \frac{D}{4}, \quad \left\{ \begin{array}{ll} b_3' = b_3/2, & c_3' = c_3/2. \\ b_4' = b_2/2, & c_4' = c_2/2, \\ b_5' = b_1/2, & c_5' = c_1/2. \end{array} \right.$$

Hence, the first order FEM-like differences can be replaced by

$$\begin{aligned} \frac{\partial T^1}{\partial x}(x_3, y_3) &= \frac{2b_3T_3 + 2b_2T_4 + 2b_1T_5}{D} \equiv D_1^xT, \\ \frac{\partial T^1}{\partial x}(x_3, y_3) &= \frac{2c_3T_3 + 2c_2T_4 + 2c_1T_5}{D} \equiv D_1^yT. \end{aligned}$$

In addition, let us define $D_2^xT \equiv \frac{\partial T}{\partial x}(x_3, y_3) - D_1^xT$ and $D_2^yT \equiv \frac{\partial T}{\partial y}(x_3, y_3) - D_1^yT$. Then, the second order differences can be decomposed to

$$\begin{aligned} \frac{\partial T}{\partial x}(x_3, y_3) &= D_1^xT + D_2^xT, \\ \frac{\partial T}{\partial y}(x_3, y_3) &= D_1^yT + D_2^yT. \end{aligned}$$

Suppose that nodes 1 to 6 are on each grid point and let the node 3 be a grid point (x_i, y_j). Then, we get the *second order FEM-like operators*

$$D_{ij}^xT \equiv D_1^xT + \mathrm{sw}D_2^xT, \tag{18}$$

$$D_{ij}^yT \equiv D_1^yT + \mathrm{sw}D_2^yT, \tag{19}$$

where

$$\mathrm{sw} = \left\{ \begin{array}{ll} 1, & \text{if } T_1, T_2, T_4 \text{ and } T_5 \text{ are "known"}, \\ & T_2 \leq T_4 \text{ and } T_1 \leq T_5, \\ 0, & \text{otherwise.} \end{array} \right. \tag{20}$$

The word "known" means that the value T has already been computed on the grid point.

There is one thing we should note. We can use only "known" grid points in the operator (18) and (19). This means that D_2^xT and D_2^yT are valid only when T_1, T_2, T_4 and T_5 are "known", and D_1^xT and D_1^yT are valid only when T_4 and T_5 are "known". Therefore, if one of T_4 and T_5 is not "known", the operators (18) and (19) cannot be defined. In what follows, we call the triangle *available* if T_4 and T_5 are "known".

4.2 Choice of a Triangle

In the second-order operator of Sethian's fast marching method, the two vertical neighbors and the two horizontal neighbors are used to compute the value at the target point, as shown in Fig. 4(a), where the target point is represented by the double circles and the used neighbors are represented by dots. In this sence, Sethian's method uses a triangle with a horizontal edge of length $2\Delta x$ and a vertical edge of length $2\Delta y$ meet at the target point. Fig. 4(a) shows one of the four possible triangles; the other three are obtained when we rotate the triangle in Fig. 4(a) by $\pi/2$, π and $3\pi/2$ around the target point. Thus, in the second order operator of the fast marching method, we can choose one of the four possible triangles according to the direction of the shortest path to the target point.

In our new operator, on the other hand, we can use eight triangles. Fig. 4(b) shows an example, in which a vertical edge of length $2\Delta y$ and a slant edge meeting at the target point; the other seven triangles can be obtained by rotating this triangle by $\pi/2$, π and $3\pi/2$ around the target point, and by mirroring them with respect to the horizontal and the vertical lines passing through the target point. Therefore, we can use one of the eight possible triangles of the type in Fig. 4(b) instead of the four of the type in Fig. 4(a). Thus, we have larger freedom in the choice of a triangle. In what follows, let us call a triangle of the type in Fig. 4(a) a *standard triangle*, and a triangle of the type in Fig. 4(b) a *sharp triangle*.

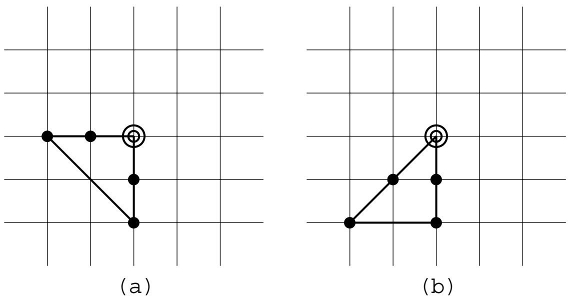

Fig. 4. Freedom in the choice of a triangle.

Now, we can use the eight sharp triangles. The next question is which triangle we should choose for the most precise computation.

The best triangle is what includes the shortest path to p_3. Consider the triangle p_1, p_2, p_3 shown in Fig. 5. Let n_1 and n_2 be outer normal vectors for the edges p_1p_3 and p_2p_3, respectively. Also, Let n_1' (n_2') be the vector directed from $p_1(p_2)$ to p_3. Then, the triangle include the shortest path to p_3 if and only if the direction of the shortest path is between n_1' and n_2'. Hence, from equation (5), this condition can be expressed by

$$\left(F\frac{\nabla T}{|\nabla T|}+f\right)\cdot n_1 \geq 0 \text{ and } \left(F\frac{\nabla T}{|\nabla T|}+f\right)\cdot n_2 \geq 0. \tag{21}$$

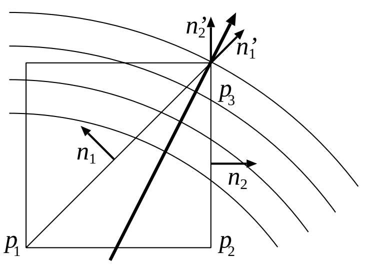

Fig. 5. Relation between the shortest path and the best triangle

Therefore, we have to find the sharp triangle that satisfies the condition (21). However, we have numerical errors in actual computation, and hence cannot expect that the condition (21) is always satisfied strictly. Hence, we choose the sharp triangle that maximizes

$$\min\left\{\left(F\frac{\nabla T}{|\nabla T|}+f\right)\cdot n_1, \left(F\frac{\nabla T}{|\nabla T|}+f\right)\cdot n_2\right\}. \tag{22}$$

Hence, for each target point p, we can construct the following strategy to select the best triangle to compute the value T at p.

Strategy for the Choice of the Best Sharp Triangle

1. Let A be the set of all available sharp triangles for the target point p.
2. For each triangle in A, compute the value T at p using the triangle, and adopt the value T that maximizes the value (22)

4.3 FEM-like Scheme

Let us define g_{ij} and h_{ij} by $g_{ij} = g(x_i, y_j)$ and $h_{ij} = h(x_i, y_j)$, respectively. We replace ∇T by $(D^x_{ij}T, D^y_{ij}T)$ and f by (g_{ij}, h_{ij}) in our equation (7). Then we obtain the finite difference version of the equation:

$$F^2\{(D^x_{ij}T)^2+(D^y_{ij}T)^2\}=(1-(D^x_{ij}T)g_{ij}+(D^y_{ij}T)h_{ij})^2. \tag{23}$$

We call this scheme the *FEM-like scheme.*

Note that this scheme can be applied only when the target point has one or more available sharp triangles. If there is no available triangle at the current target point, we postpone the computation of T until sharp triangles become available.

4.4 Algorithm

We solve our boundary value problem by the same strategy as the fast marching method by Sethian [4]. This method is similar to the Dijkstra method for computing all shortest paths from a start vertex in a graph. We consider the grid structure the graph in which the vertices are grid points and the edges connect the four neighbors of each grid point. We start with the boat harbor at which $T_{ij} = 0$, and compute T_{ij}'s one by one from the nearest grid point. The only difference from the Dijkstra method is that the quadratic equation (23) is solved to obtain the value of T_{ij}.

In the next algorithm, the grid points are classified into three groups: "known" points, "frontier" points and "far" points. The "known" points are points at which the values T_{ij} are known. The "frontier" points are points that are not yet known but are the neighbors of the "known" points. The "far" points are all the other points. In the algorithm, all the points other than the boat harbor are initially categorized as "far" points, and then changed to "frontier" points and eventually to "known" points.

Algorithm 1 (Boat-sail distance from a single harbor)

Input: flow function $f(x, y)$ in Ω and the boat harbor p_0.
Output: Arrival time T_{ij} at every grid point in Ω.
Procedure:

```
1. Set $T_{ij} \leftarrow 0$ for the grid point corresponding to the harbor, and $T_{ij} \leftarrow \infty$ for all the other points.
2. Name the grid point $p_0$ as "frontier", and all the other grid points as "far".
3. choose the "frontier" point $p = (x_i, y_i)$ with the smallest value of $T_{ij}$, and rename it as "known".
4. For all the neighbors of $p$ that are not "known", do 4.1, 4.2 and 4.3.
    4.1 If $p$ is "far", rename it as "frontier".
    4.2 Recompute the value of $T_{ij}$ by solving the equation (23).
    4.3 If the recomputed value $T_{ij}$ is smaller than the current value, update the value.
5. If all the grid points are "known", stop. Otherwise go to Step 3.
```

Let N be the number of the grid points in Ω. Step 1 and 2 of the above algorithm are done in

O(N) time. Just as the Dijkstra method, we use a heap data structure to store the "frontier" grid points with the key T_{ij}. Then, the addition of a new grid point to the heap and the deletion of the "frontier" grid point with the smallest T_{ij} can be done in O($\log N$) time. Hence, each processing of Steps 3 and 4 is done in O($\log N$) time. Since Steps 3, 4 and 5 are repeated N times, the total time complexity of Algorithm 1 is O($N \log N$).

5 Numerical Examples

In this section, we show the behavior of our method for computing the boat-sail distances in numerical examples. We consider shortest path problems in the flow field at this time.

Suppose that the flow field f and the boat harbor p is given, and that, for each query point q, we want to find the shortest path from p to q with respect to the boat-sail distance. To solve this problem, we first use Algorithm 1 to compute the arrival time $T(x, y)$ in Ω from the start point p. Next, from each query point p in Ω, we trace back the shortest path, until we reach the starting point p. For this purpose we solve the following ordinary differential equation.

Let $X(t) = (x(t), y(t))$ be the shortest path with parameter t such that

$$X(0) \equiv (x(0), y(0)) = q. \qquad (24)$$

We assume that, as t increases, the point $X(t)$ moves along the shortest path in the opposite way from q to p. The motion of the boat in time interval Δt is represented by the expression (5), and hence we get

$$X(t+\Delta t) - X(t) = -\left(F\frac{\nabla T}{|\nabla T|}\Delta t + f\Delta t\right), \quad (25)$$

and consequently we obtain the ordinary differential equation:

$$X_t = -\left(F\frac{\nabla T}{|\nabla T|} + f\right). \qquad (26)$$

Thus, the problem of constructing the shortest path is reduced to the ordinary differential equation (26) together with the initial condition (24). Since the arrival time $T(x, y)$ at the grid points has been obtained by algorithm 1, the gradient ∇T at any points can be computed by (11) and (12). Hence, we can construct the following algorithm to compute the shortest path.

Algorithm 2 (shortest path)

Input: arrival time $T(x, y)$ from p to all the grid points in Ω and the query point q.
Output: shortest path $X(t)$ from p to q.
Procedure:

```
1. $X(t) \leftarrow 0$, $t \leftarrow 0$, and fix a small positive real $\Delta t$.

2. Find a triangle of the type in Fig. 3(b) or (c) that includes the point $X(t)$, and compute the gradient $\nabla T$ at $X(t)$ using (11) and (12).

3. $X(t+\Delta t) \leftarrow X(t) - \left(F\frac{\nabla T}{|\nabla T|}\Delta t + f\Delta t\right)$.

4. $t \leftarrow t + \Delta t$.

5. If $X(t)$ is sufficiently close to $p$, stop. Otherwise go to Step 2.
```

Note that we cannot distinguish between the standard triangle and the sharp triangle in Step 2 of the above algorithm, because the point $X(t)$ is not a grid point; we can use any triangle that include $X(t)$.

We show three examples of the shortest paths in the flow field in Figs. 6(a) to 6(c). Here, we assumed that the speed F of a boat be 1. The arrows in the figures represent the directions and the relative speeds of the flow in the field. The lengths of the arrows in the same figure express the relative speeds of the flow; the longer is the arrow, the faster is the flow. The thin curves express the isoplethic curves of the first arrival time, and the thick curve expresses the shortest path.

Fig. 6(a) is for the circular flow $f = (-0.7\sin\theta, 0.7\cos\theta)$ in a doughnut region $\{(x, y) \mid 0.25 < x^2 + y^2 < 1\}$. Fig. 6(b) is for the flow field $f = (0.7(1 - y^2), 0.0)$ in a rectangular region $\{(x, y) \mid -1 < y < 1\}$. Finally, Fig. 6(c) shows the case for the flow $f = 0.35(1 - 0.25/z^2), z \in \mathbf{C}$ in the square region $\{(x, y) \mid -1 < |z| < 1\}$. This flow coincides with the theoretical flow pattern obtained when the cylinder of the radius 0.5 is placed at the center of region in the homogeneous, uncompressible and nonviscouse flow from left to right.

6 Concluding Remarks

We first defined the boat-sail distance, next derived the partial differential equation satisfied by the first arrival time, thirdly constructed a new scheme for computing the boat-sail distance, and finally showed computational experiments. The concept of the boat-sail distance is natural and intuitive, but the computation is not trivial. Actually the original definition of the boat-sail distance given by the equations (2) and (3) does not imply any explicit idea for computing this distance, because the shortest path is unknown. This seems the main reason why these concepts have not been studied from the computational point of view.

Our breakthrough toward efficient computation is that we succeeded in formulating the problem as the boundary value problem. The distance is defined according to the notion of the boat sailing, and hence a naive formulation will reach an initial value problem of a partial differential equation containing the time variable and its derivatives. In this paper, on the other hand, we concentrated on the first arrival time as the unknown function, and thus constructed an equation without time variable. Moreover, this partial differential equation is quadratic, which is not so simple as linear, but is still tractable. This formulation enables us to use the same idea as the fast marching method, which was originally proposed for the eikonal equation, and thus could construct efficient algorithms.

Acknowledgment

This work is supported by the 21st Century COE Program of the Information Science and Technology Strategic Core, and the Grant-in-aid for Scientific Research (S)15100001 of the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

[1] R. Kimmel and J. A. Sethian: Fast Marching Methods on Triangulated Domains, Proc. Nat. Acad. Sci., 95, 1998, pp. 8431–8435.

[2] T. Nishida and K. Sugihara: Voronoi diagram in the flow field. *Algorithms and Computation, 14th International Symposium, ISAAC 2003*, Kyoto, Springer, 2003, pp. 26–35.

[3] T. Nishida and K. Sugihara: FEM-like Fast Marching Method for the Computation of the Boat-Sail Distance and the Associated Voronoi Diagram. Technical Reports, METR 2003-45, Department of Mathematical Informatics, the University of Tokyo, 2003.

[4] J. A. Sethian: Fast marching method. *SIAM Review*, vol. 41 (1999), pp. 199–235.

[5] J. A. Sethian: *Level Set Methods and Fast Marching Methods, Second Edition.* Cambridge University Press, Cambridge, 1999.

[6] J. A. Sethisn and M. Popovici: Fast Marching Methods Applied to Computation of Seismic Travel Times, *Geophysics*, 64, 2, 1999.

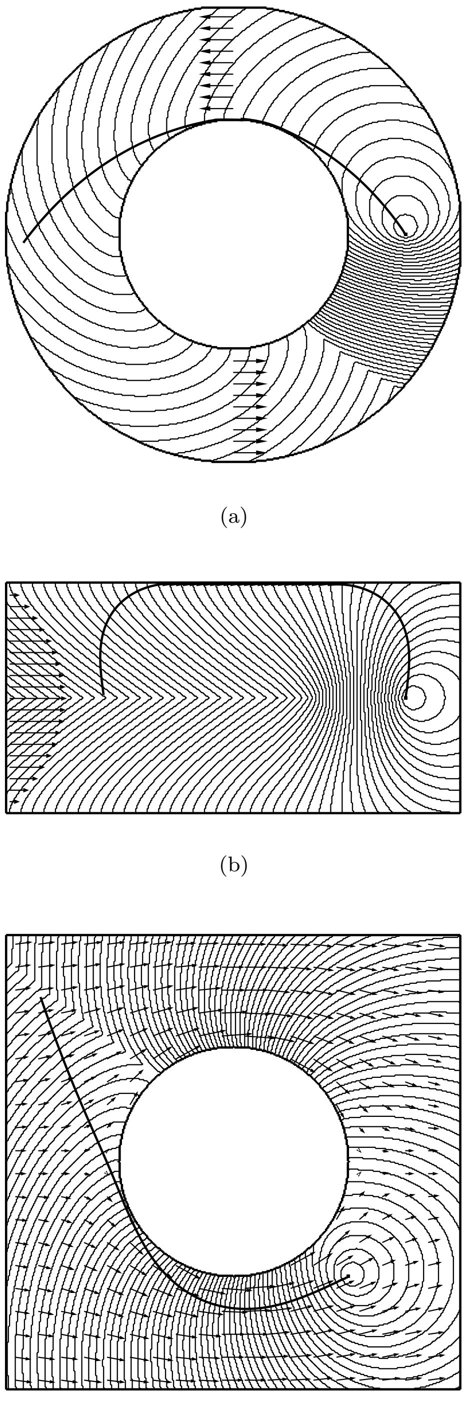

Fig. 6. Isoplethic curves of the first arrival time and the shortest paths to query points.