Computing 80, 121–136 (2007)
DOI 10.1007/s00607-007-0223-5
Printed in The Netherlands

Computing

An efficient algorithm for characteristic tracking on two-dimensional triangular meshes

J. Liu, Fort Collins, and **H. Chen, R. Ewing, G. Qin**, College Station

Received June 21, 2006; revised December 8, 2006
Published online: April 30, 2007

Abstract

Tracking characteristics on unstructured meshes is an important part of many numerical methods in computational fluid mechanics. In this paper, we propose an efficient algorithm for characteristic tracking on two-dimensional unstructured triangular meshes. Numerical experiments, including an example for applying this algorithm with the Eulerian-Lagrangian localized adjoint method (ELLAM) to solve a convection-dominated convection-diffusion problem, are presented to demonstrate the efficiency of this algorithm.

AMS Subject Classifications: 65M25, 65Y20, 68W01.

Keywords: Auxiliary mesh, characteristic, convection-diffusion, triangular mesh, unstructured mesh, velocity field.

1. Introduction

Characteristic methods for fluid flow problems became viable when Douglas and Russell described their *modified method of characteristics* (MMOC) in 1982 [4]. Other characteristic methods include ELLAM [3], [10], [13], Galerkin-Lagrangian methods [9], and the semi-Lagrangian method [5], [14]. However, it is challenging to implement characteristic methods on unstructured meshes. As pointed out by the review paper [10]: *The tracking is a major part of the overall computational effort, especially when time steps are large and a point is tracked across several cells in a time step.* This considerable difficulty of implementation was also commented in [1]. Probably, this is a main reason why numerical experiments on two or three-dimensional unstructured meshes are rarely reported in the literature on characteristic methods.

Mathematically, characteristic tracking is expressed as solving an initial value problem of an ordinary differential equation (ODE):

$$\begin{cases} \dfrac{d\mathbf{y}}{ds} = \mathbf{v}(\mathbf{y}, s), \\ \mathbf{y}(s; \mathbf{x}, t)|_{s=t} = \mathbf{x}, \end{cases} \tag{1}$$

where $\mathbf{y}(s; \mathbf{x}, t)$ is the characteristic passing through point $\mathbf{x}$ at time t and $\mathbf{v}$ is a given velocity field. An implementation of characteristic tracking involves many aspects:

e.g., a mesh on the domain Ω, a mesh on the space-time boundary $\partial\Omega \times [t_{n-1}, t_n]$; a (continuous or discrete) definition of the velocity field $\mathbf{v}$ on the prism $\Sigma_n = \Omega \times [t_{n-1}, t_n]$; and a proposed numerical method for solving the ODE, e.g., the Euler or Runge-Kutta methods.

When tracking a characteristic, we need to determine in which element the foot, head, or an intermediate point of the characteristic lies. This searching task is simple for one dimensional or rectangular or structured meshes, but complicated for unstructured meshes. In [5], the QuadTree algorithm was proposed for general quadrilateral meshes. The algorithm establishes a tree data structure to organize the neighborhood information about nodes and elements. As pointed out by the author, this could be inefficient for highly distorted grids and have adverse affects on the efficiency of numerical schemes. Therefore, a special algorithm for the much more regular icosahedral mesh was also discussed. Another searching algorithm was designed in [14] for the semi-Lagrangian method on unstructured grids. This algorithm also relies on the neighborhood information in meshes, which could be very complicated. In this paper, we propose setting up an auxiliary rectangular mesh to assist characteristic tracking. The auxiliary mesh covers the given unstructured mesh and has roughly the same mesh size as the given mesh. The connectivity information about the triangles in the real mesh and the rectangles in the auxiliary mesh is formed as a look-up table and used for locating points on characteristics.

The rest of this paper is structured as follows. Section 2 presents a full description of the algorithm. Section 3 briefly reviews the ELLAM formulation. Section 4 reports our numerical experiments on two typical examples demonstrating the efficiency of the tracking algorithm and one example using ELLAM to solve a convection-diffusion problem on a two-dimensional unstructured triangular mesh. Section 5 concludes with some discussions.

2. Description of the algorithm

2.1. Outline of the algorithm

Let Ω be a planar polygonal domain equipped with a quasi–uniform triangular mesh $\mathcal{T}_h$ (*real mesh*).

(1) Find the minimal rectangular domain $\mathcal{R}$ that covers Ω;
(2) Set up a rectangular mesh (*auxiliary mesh*) on $\mathcal{R}$ that has approximately the same mesh size;
(3) Sort out the information about the connection between the triangles in the real mesh and the rectangles in the auxiliary mesh;
(4) At each time step, for either forward tracking or backward tracking, utilize the connectivity information obtained in Step (3), apply the criterion for a point being inside a triangle, and then locate the head or foot of a characteristic;
(5) Special treatment is needed when a characteristic intersects with the inflow/outflow boundaries.

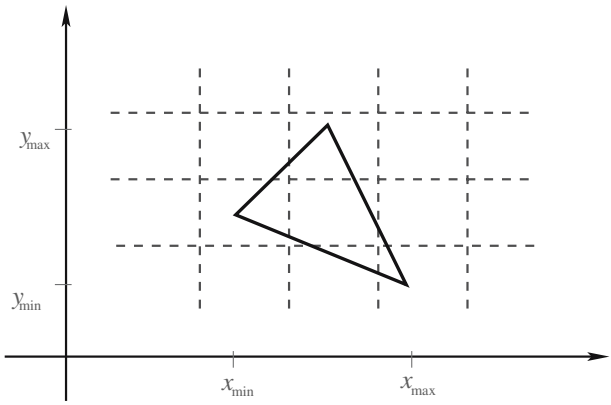

Fig. 1. A triangle intersects with seven rectangles

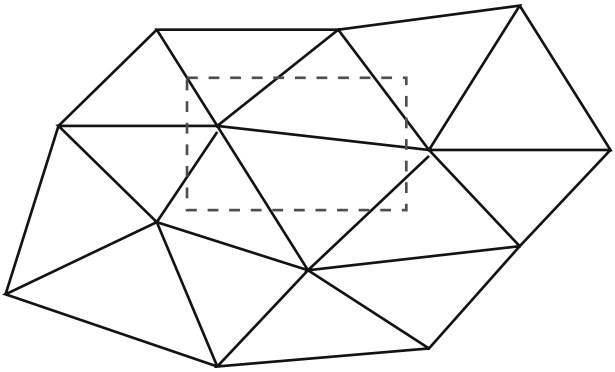

Fig. 2. Eight triangles sharing a rectangular box

2.2. Auxiliary mesh and its connection with real mesh

The minimal rectangular domain $\mathcal{R}$ that covers Ω can be taken as

$$\mathcal{R} := [X_{\min}, X_{\max}] \times [Y_{\min}, Y_{\max}],$$

where $X_{\min}, X_{\max}, Y_{\min}, Y_{\max}$ are the minimum and maximum of X,Y-coordinates of the nodes in the triangular mesh $\mathcal{T}_h$. Then we set up a rectangular mesh about size h on the rectangular domain $\mathcal{R}$. For each triangle in the real mesh, we find out how many rectangles and which rectangles in the auxiliary mesh intersect with the triangle. (see Fig. 1). Then we invert the above connectivity information to generate a look-up table of rectangles versus the triangles intersecting with them (see Fig. 2). This table is a rugged array and will be stored for later use.

2.3. Locating the head/foot of a characteristic

In applications of characteristic methods, one carries out forward tracking or backward tracking of characteristics. Usually characteristics start from quadrature points

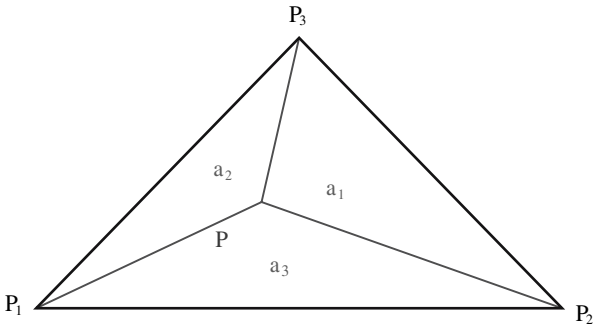

Fig. 3. Barycentric coordinates of a point inside a triangle

in finite elements or some important points describing fluid fronts. For each characteristic, we first locate the head (forward tracking) or the foot (backward tracking) in the auxiliary rectangular mesh. That is, we find out which rectangular box contains the head or foot. For this rectangular box, we retrieve from the look-up table constructed in Sect. 2.2 the list of triangles that intersect with this rectangle. Sweeping through this list of triangles, and applying the criterion for a point being inside a triangle, we find the very triangle in which the head or foot resides.

Criterion of a point being inside a triangle: Let T be a triangle with three vertices P_1, P_2, P_3 and P be an arbitrary point on the plane. Let $A = \Delta P_1 P_2 P_3$ be the area of triangle T and $A_i (i = 1, 2, 3)$ be the area of the triangle when P_i is replaced by P. (see Fig. 3.) Let $a_i := A_i / A$. Then, $P \in \Delta P_1 P_2 P_3$ if and only if

$$a_i \geq 0 \ (i = 1, 2, 3), \quad a_1 + a_2 + a_3 = 1. \tag{2}$$

Here (a_1, a_2, a_3) are called barycentric coordinates of the point inside the triangle $\Delta P_1 P_2 P_3$.

The barycentric coordinates are also very useful for finite element approximations. If the Lagrangian $\mathcal{P}_1$ element is used on a triangle and the nodal values of the shape function at the three vertices are known, say, $u_i = u(P_i)$, $i = 1, 2, 3$, then the shape function value at any point P inside the triangle is

$$u(P) = a_1 \, u(P_1) + a_2 \, u(P_2) + a_3 \, u(P_3).$$

For characteristic methods like ELLAM or MMOC, backward tracking is usually employed. The foot of a characteristic could be anywhere on a triangular mesh. Once the foot of the characteristic is located in a triangle element, the above formula can be used for interpolation.

In [14], a different criterion is proposed for determining whether a point is inside a given triangle. It is based on the inner products of the vectors starting from the vertices of the triangle to the given point and the inward normals of the corresponding edges. It requires comparable amount of computations as the criterion in Eq. (2),

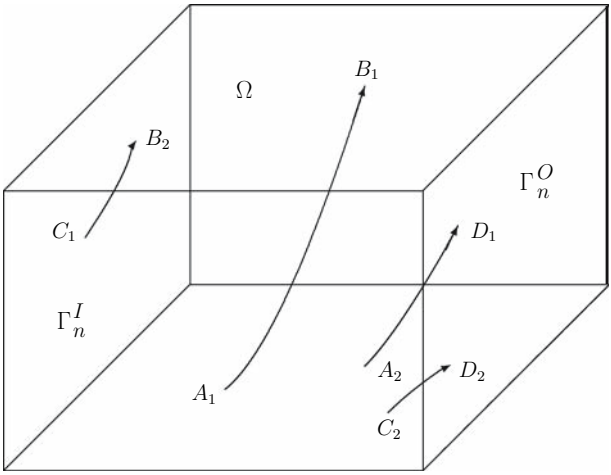

Fig. 4. Characteristics and flows

and can also be used with our search algorithm. But the criterion in Eq. (2) is preferred, since the barycentric coordinates can be immediately used for interpolation on the numerical solution. However, the criterion proposed in [14] needs another round of computations for interpolation parameters.

2.4. Characteristics intersecting with boundaries

Let $\partial\Omega$ be the boundary of the spatial domain Ω and $\Gamma_n := \partial\Omega \times [t_{n-1}, t_n]$. The inflow, outflow, and no flow space-time boundaries during time period $[t_{n-1}, t_n]$ are denoted by Γ_n^I, Γ_n^O, Γ_n^N, respectively, and defined as

$$\begin{aligned}
\Gamma_n^I &:= \{(\mathbf{x}, t) \mid \mathbf{x} \in \partial\Omega, t \in [t_{n-1}, t_n], \mathbf{v}(\mathbf{x}, t) \cdot \mathbf{n}(\mathbf{x}) < 0\},\\
\Gamma_n^O &:= \{(\mathbf{x}, t) \mid \mathbf{x} \in \partial\Omega, t \in [t_{n-1}, t_n], \mathbf{v}(\mathbf{x}, t) \cdot \mathbf{n}(\mathbf{x}) > 0\},\\
\Gamma_n^N &:= \{(\mathbf{x}, t) \mid \mathbf{x} \in \partial\Omega, t \in [t_{n-1}, t_n], \mathbf{v}(\mathbf{x}, t) \cdot \mathbf{n}(\mathbf{x}) = 0\},
\end{aligned} \tag{3}$$

where $\mathbf{n}(\mathbf{x})$ is the outward unit normal vector on $\partial\Omega$.

The head or foot (or both) of a characteristic could hit the boundaries at any time $t \in (t_{n-1}, t_n)$. Figure 4 provides an illustration of all four possibilities in characteristic tracking, assuming that the spatial domain Ω is a two-dimensional rectangle. The left and front faces of the space-time domain $\Omega \times [t_{n-1}, t_n]$ are assumed to be inflow boundaries, and the right and back faces are outflow boundaries. In the figure,

- A_1B_1 goes from the domain to the domain;
- A_2D_1 goes from the domain to the outflow boundary;
- C_1B_2 goes from the inflow boundary to the domain; and
- C_2D_2 goes from the inflow boundary to the outflow boundary.

Suppose Ω is a two-dimensional polygonal domain. Let $J_n := [t_{n-1}, t_n]$ be a typical time step and $d_n := \|\mathbf{v}\|_\infty \Delta t_n$. We make a polygonal domain Ω_n that is inside

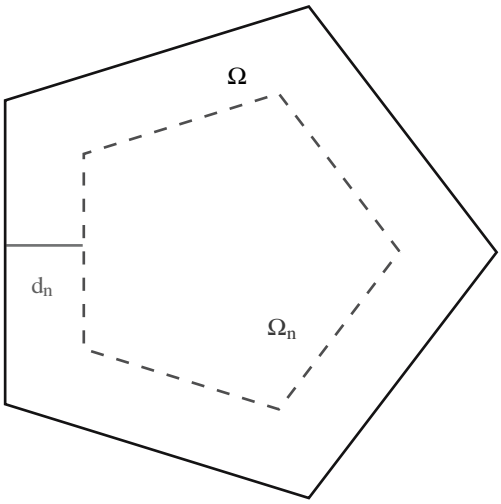

Fig. 5. The inner domain and its margins to the boundaries

Ω yet has the same shape as Ω. To be precise, the sides of Ω_n are parallel to the corresponding sides of Ω with distance d_n (see Fig. 5)

A characteristic emanating from any point inside the inner domain Ω_n will stay inside the real domain Ω during the time period $[t_{n-1}, t_n]$. We can apply the algorithm discussed in the previous subsection directly to locate the foot of the characteristic. But for a characteristic starting from a point in the strip $\Omega \setminus \Omega_n$, it might hit the boundary at any time $t \in (t_{n-1}, t_n)$. For such a characteristic, we have to check the intermediate points:

- If an intermediate point runs out of the original domain Ω, then we have to modify the foot/head of the characteristic and tracking stops;
- If an intermediate point is still in the strip $\Omega \setminus \Omega_n$, then we have to check if the next intermediate point is inside the real domain Ω;
- If an intermediate point falls inside the inner domain Ω_n, then we no longer need any check for the remaining intermediate points.

Numerical methods based on characteristic tracking, e.g., ELLAM, are usually not subject to the Courant-Friedrichs-Lewy (CFL) condition. In order to get better numerical performances, we use micro time steps satisfying the CFL condition for characteristic tracking. Then the time period $[t_{n-1}, t_n]$ is partitioned into $t_{n,i} = t_{n-1} + i\delta t_n$ $(0 \le i \le m)$, where m is an integer, $\delta t_n := (t_n - t_{n-1})/m$, and δt_n satisfies the CFL condition (see Fig. 6)

Correction to characteristics intersecting with boundaries: Let us take a backward characteristic as an example. At time $t \in (t_{n,i-1}, t_{n,i})$, the approximate characteristic goes across the inflow boundary $Q_{j-1}Q_j$. On this approximate characteristic, we have $P_i \in \Omega$ but point $P_{i-1} \notin \Omega$. We need to replace P_{i-1} by the intersection P

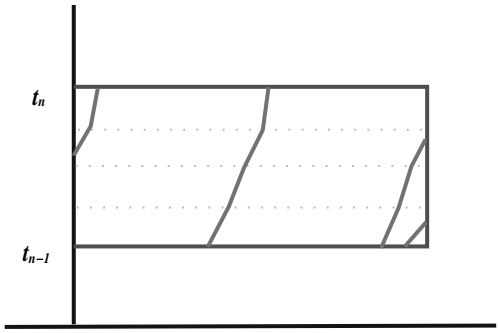

Fig. 6. Micro time steps on characteristics

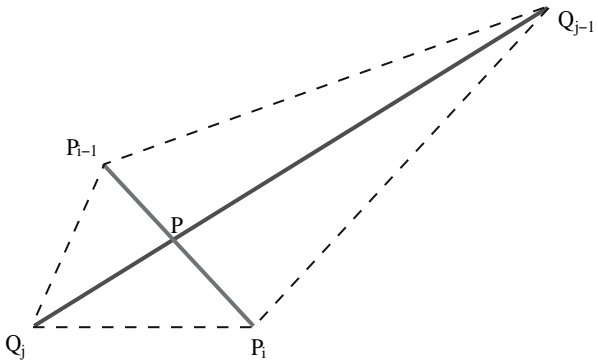

Fig. 7. An approximate characteristic goes across the domain boundary

of line segment $P_{i-1}P_i$ with the inflow boundary. Assume Ω is a convex polygon. Notice that P is on line segment $P_{i-1}P_i$ and can be written as $P = (1-\alpha)P_{i-1}+\alpha P_i$ for some $0 < \alpha < 1$. Actually, it can be proved that α is the ratio of the area of triangle $\Delta Q_j Q_{j-1} P_{i-1}$ to the area of quadrilateral $Q_j P_i Q_{j-1} P_{i-1}$. Accordingly, we replace $t_{n,i-1}$ by $t^* = t_{n,i} - \alpha(t_{n,i} - t_{n,i-1})$ (see Fig. 7).

3. ELLAM formulation

An excellent overview about the state of research on ELLAM up to 2002 was given in [10]. In this section, we outline and apply the ELLAM methodology to study the following unsteady-state linear convection-diffusion equation

$$u_t + \nabla \cdot (\mathbf{v}u - \mathbf{D}\nabla u) = f(\mathbf{x}, t), \qquad \mathbf{x} \in \Omega, \quad t \in [0, T]. \tag{4}$$

Here, $\Omega \subset \mathbb{R}^d (d = 1, 2, 3)$ with boundary $\Gamma := \partial\Omega$; $u(\mathbf{x}, t)$ is the unknown function; $\mathbf{v}(\mathbf{x}, t)$ is a velocity field; $\mathbf{D}(\mathbf{x}, t)$ is a diffusion-dispersion tensor; and $f(\mathbf{x}, t)$ is a source/sink term.

Let Γ^I, Γ^O, Γ^N be the inflow, outflow, and no flow space-time boundaries identified by

$$\begin{aligned}\Gamma^I &:= \{\mathbf{x} \mid \mathbf{x} \in \partial\Omega, \mathbf{v} \cdot \mathbf{n}(\mathbf{x}) < 0\},\\ \Gamma^O &:= \{\mathbf{x} \mid \mathbf{x} \in \partial\Omega, \mathbf{v} \cdot \mathbf{n}(\mathbf{x}) > 0\},\\ \Gamma^N &:= \{\mathbf{x} \mid \mathbf{x} \in \partial\Omega, \mathbf{v} \cdot \mathbf{n}(\mathbf{x}) = 0\}.\end{aligned} \tag{5}$$

The ELLAM framework can treat any boundary conditions [13], but we restrict ourselves to the following boundary and initial conditions that are typical in applications:

$$\begin{aligned} u(\mathbf{x}, t) &= g^O(\mathbf{x}, t), && \mathbf{x} \in \Gamma^O, && t \in [0, T],\\ (\mathbf{v}u - \mathbf{D}\nabla u)(\mathbf{x}, t) \cdot \mathbf{n} &= g^I(\mathbf{x}, t), && \mathbf{x} \in \Gamma^I, && t \in [0, T],\\ -\mathbf{D}\nabla u(\mathbf{x}, t) \cdot \mathbf{n} &= 0, && \mathbf{x} \in \Gamma^N, && t \in [0, T],\\ u(\mathbf{x}, 0) &= u_0(\mathbf{x}), && \mathbf{x} \in \Omega. \end{aligned} \tag{6}$$

Let $0 = t_0 < t_1 < \ldots < t_{n-1} < t_n < \ldots < t_N = T$ be a partition of $[0, T]$ with $\Delta t_n := t_n - t_{n-1}$. We multiply Eq. (4) by test functions $w(\mathbf{x}, t)$ that vanish outside the space-time strip $\Omega \times (t_{n-1}, t_n]$ and are discontinuous in time at time t_{n-1}. Then integration by parts leads us to the following weak form:

$$\begin{aligned} &\int_\Omega u(\mathbf{x}, t_n) w(\mathbf{x}, t_n) d\mathbf{x} + \int_{t_{n-1}}^{t_n} \int_\Omega (\mathbf{D}\nabla u) \cdot \nabla w d\mathbf{x} dt\\ &\quad + \int_{t_{n-1}}^{t_n} \int_{\partial\Omega} (\mathbf{v}u - \mathbf{D}\nabla u) \cdot \mathbf{n} w dS - \int_{t_{n-1}}^{t_n} \int_\Omega u(w_t + \mathbf{v} \cdot \nabla w) d\mathbf{x} dt\\ &= \int_\Omega u(\mathbf{x}, t_{n-1}) w(\mathbf{x}, t_{n-1}^+) d\mathbf{x} + \int_{t_{n-1}}^{t_n} \int_\Omega (fw)(\mathbf{x}, t) d\mathbf{x} dt, \end{aligned} \tag{7}$$

where dS is the differential element on $\partial\Omega$ and $w(\mathbf{x}, t_{n-1}^+) := \lim_{t \to t_{n-1}^+} w(\mathbf{x}, t)$ takes into account the fact that $w(\mathbf{x}, t)$ are discontinuous in time at time t_{n-1}.

ELLAM takes advantage of the hyperbolic nature of convection-diffusion equations to require the test functions to satisfy the adjoint equation

$$w_t + \mathbf{v} \cdot \nabla w = 0. \tag{8}$$

This cancels the last term on the left-hand side of the weak form, and implies that test functions are constants along the characteristics defined by the initial value problem of the ordinary differential equation shown in Eq. (1).

For any $\mathbf{x} \in \Omega$, if $(\mathbf{x}, t_n)$ backtracks along a characteristic to $(\mathbf{x}^*, t^*) \in \Gamma_n^I$ or Ω at time t_{n-1}, we define $\Delta t^I(\mathbf{x}, t_n) := t_n - t^*$ or $t_n - t_{n-1}$, respectively. Similarly,

for any $(\mathbf{y}, t) \in \Gamma_n^O$, if it backtracks to $(\mathbf{y}^*, t^*) \in \Gamma_n^I$ or Ω at time t_{n-1}, we define $\Delta t^O(\mathbf{y}, t) := t - t^*$ or $t - t_{n-1}$, respectively. See Fig. 4 for more details, where A_1 and A_2 are in Ω at time t_{n-1}, while B_1 and B_2 are in Ω at time t_n, C_1 and C_2 are on the inflow boundary Γ_n^I, while D_1 and D_2 are on the outflow boundary Γ_n^O.

By enforcing the backward Euler quadrature on Ω at time t_n and Γ_n^O, we obtain

$$\begin{aligned}
&\int_{t_{n-1}}^{t_n} \int_{\Omega} f(\mathbf{x}, t) w(\mathbf{x}, t) d\mathbf{x} dt \\
&\quad = \int_{\Omega} \Delta t^I(\mathbf{x}, t_n) f(\mathbf{x}, t_n) w(\mathbf{x}, t_n) d\mathbf{x} \\
&\qquad + \int_{\Gamma_n^O} \Delta t^O(\mathbf{y}, t) f(\mathbf{y}, t) w(\mathbf{y}, t)\ (\mathbf{v} \cdot \mathbf{n}) dS\ + E(f, w).
\end{aligned}$$

Similarly, the diffusion term can be evaluated as

$$\begin{aligned}
&\int_{t_{n-1}}^{t_n} \int_{\Omega} ((\mathbf{D}\nabla u) \cdot \nabla w)(\mathbf{x}, t) d\mathbf{x} dt \\
&\quad = \int_{\Omega} \Delta t^I(\mathbf{x}, t_n)((\mathbf{D}\nabla u) \cdot \nabla w)(\mathbf{x}, t_n) d\mathbf{x} \\
&\qquad + \int_{\Gamma_n^O} \Delta t^O(\mathbf{y}, t)((\mathbf{D}\nabla u) \cdot \nabla w)(\mathbf{y}, t)\ (\mathbf{v} \cdot \mathbf{n}) dS + E(D, u, w).
\end{aligned}$$

Dropping the error terms in the source and diffusion terms and breaking up the boundary term, we obtain the following reference equation: Find $u(\mathbf{x}, t) \in H^1(\Omega \times (t_{n-1}, t_n])$ such that for any $w(\mathbf{x}, t) \in H^1(\Omega \times (t_{n-1}, t_n])$ satisfying the adjoint equation (8), the following holds:

$$\begin{aligned}
&\int_{\Omega} u(\mathbf{x}, t_n) w(\mathbf{x}, t_n) d\mathbf{x} + \int_{\Omega} \Delta t^I(\mathbf{x}, t_n)(\mathbf{D}\nabla u \cdot \nabla w)(\mathbf{x}, t_n) d\mathbf{x} \\
&\qquad + \int_{\Gamma_n^O} \Delta t^O(\mathbf{y}, t)(\mathbf{D}\nabla u) \cdot \nabla w)(\mathbf{y}, t)\ (\mathbf{v} \cdot \mathbf{n}) dS \\
&\qquad + \int_{\Gamma_n^O} (\mathbf{v}u - \mathbf{D}\nabla u) \cdot \mathbf{n}\ w(\mathbf{y}, t) dS + \int_{\Gamma_n^I} (\mathbf{v}u - \mathbf{D}\nabla u) \cdot \mathbf{n}\ w(\mathbf{y}, t) dS \\
&\quad = \int_{\Omega} u(\mathbf{x}, t_{n-1}) w(\mathbf{x}, t_{n-1}^+) d\mathbf{x} + \int_{\Omega} \Delta t^I(\mathbf{x}, t_n) f(\mathbf{x}, t_n) w(\mathbf{x}, t_n) d\mathbf{x} \\
&\qquad + \int_{\Gamma_n^O} \Delta t^O(\mathbf{y}, t) f(\mathbf{y}, t) w(\mathbf{y}, t)\ (\mathbf{v} \cdot \mathbf{n}) dS. \qquad (9)
\end{aligned}$$

Table 1. Statistics of experiments on example 1

	Mesh 1	Mesh 2	Mesh 3	Mesh 4
Number of elements/nodes	324/183	1296/689	5184/2673	82944/41793
Size of auxiliary mesh	20x20	40x40	80x80	318x318
Minimal number of triangles intersecting with a rectangle	2	2	2	2
Maximal number of triangles intersecting with a rectangle	8	9	10	12
Average number of triangles intersecting with a rectangle	5.2	5.4	5.5	5.6
Percentage of characteristics intersecting with boundaries	2.5%	4.0%	4.2%	5.5%

For the first term on the right-hand side of the above equation, replacing the dummy variable $\mathbf{x}$ by $\mathbf{x}^*$, we rewrite the term as

$$\begin{aligned} &\int_{\Omega} u(\mathbf{x}^*, t_{n-1}) w(\mathbf{x}^*, t_{n-1}^+) d\mathbf{x}^* \\ &\quad = \int_{\Omega} u(\mathbf{x}^*, t_{n-1}) w(\mathbf{x}, t_n) \mathbf{J}(\mathbf{x}^*, \mathbf{x}) d\mathbf{x}, \end{aligned} \tag{10}$$

where $\mathbf{x}^* = \mathbf{y}(t_{n-1}; \mathbf{x}, t_n)$ is obtained by backtracking $(\mathbf{x}, t_n)$ along a characteristic to time t_{n-1} and $\mathbf{J}$ is the Jacobian. Therefore, we have to track characteristics accurately and efficiently.

4. Numerical experiments

In this section, we test our algorithm on several different triangular meshes to demonstrate its efficiency. We also use this tracking algorithm and ELLAM to solve a convection-diffusion problem on a two-dimensional unstructured triangular mesh.

Example 1: This example deals with a rotating velocity field $\mathbf{v} = (-y, x)$ defined on a square $\Omega = [-1, 1] \times [-1, 1]$. Four triangular meshes are generated by the PDE toolbox in Matlab based on the Delaunay triangulation. Figure 8 is a sketch of the first triangulation. As revealed by Fig. 9, for quasi–uniform triangular meshes, most rectangles in the auxiliary meshes intersect with about 6 triangles, while few rectangles intersect with more than 8 triangles. Table 1, Row 6 lists the average number of triangles intersecting with a rectangle for each triangular mesh. These numbers are also the average search steps for the triangle containing a point of interest.

We take $T = 2 * \pi$ and $\Delta t = T/64$. For each time step, $\delta t = \Delta t/16$ is used as the micro time step for characteristic tracking. We randomly choose a time step $[14\Delta t, 15\Delta t]$ and carry out backward tracking from the barycenter of each triangle, using the Euler method. The last row in Table 1 lists the percentage of characteristics intersecting with the domain boundaries among all characteristics emanating from the centers of triangles. The reason why a higher percentage of characteristics hit

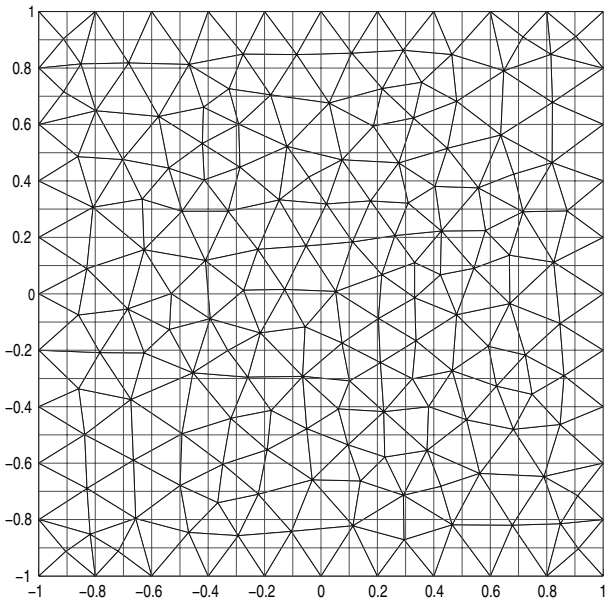

Fig. 8. The real and auxiliary meshes for example 1, mesh 1

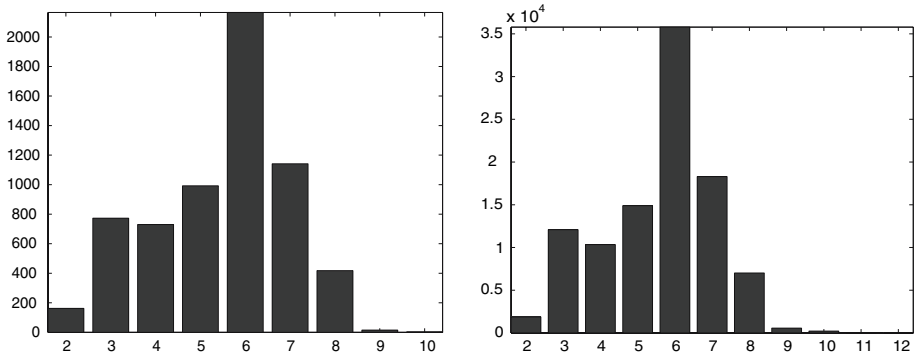

Fig. 9. Example 1: distribution of rectangles intersecting with different numbers of triangles

the domain boundary is that we decrease mesh sizes from Mesh 1 to Mesh 4, while we are using the same time step size.

Example 2: *A swirling velocity field:* In this interesting example proposed by R. LeVeque in [7], the domain Ω is the unit square $[0, 1] \times [0, 1]$ and the velocity field $\mathbf{v}(x, y) = (v_1, v_2)$ is given by

$$v_1(x, y) = \sin^2(\pi x)\sin(2\pi y), \quad v_2(x, y) = -\sin(2\pi x)\sin^2(\pi y).$$

This is an incompressible velocity field (div $\mathbf{v} = 0$). The velocity vanishes ($\mathbf{v} = \mathbf{0}$) on the four sides and at the center of the square, i.e., these points are the equilibriums

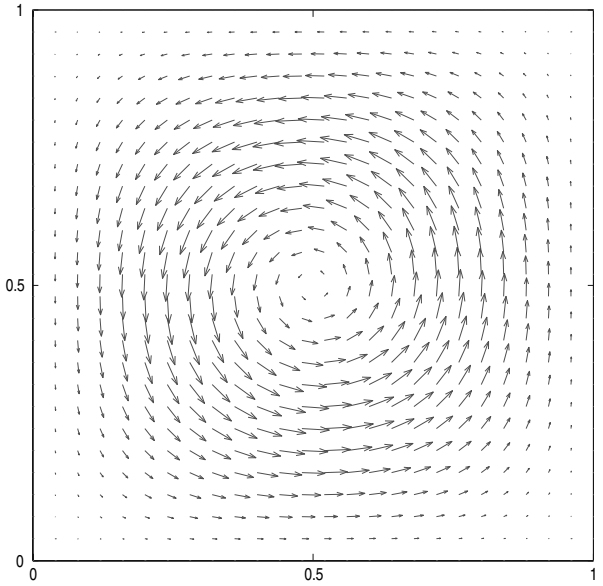

Fig. 10. The swirling velocity field in example 2

of the velocity field. We have noflow boundary types on the entire boundary of the domain. (see Fig. 10.) Any characteristic starting from any point inside the domain will stay strictly inside the domain, even though it approaches the boundary asymptotically.

The ODEs for characteristics

$$\frac{dx}{dt} = \sin^2(\pi x)\sin(2\pi y), \quad \frac{dy}{dt} = -\sin(2\pi x)\sin^2(\pi y)$$

are nonlinear. Numerical methods for solving ODEs have to be employed, since a closed form of the exact solution is not available. However, it can be deduced that the primary function

$$F(x, y) = \sin(\pi x)\sin(\pi y) \tag{11}$$

is a constant on each characteristic. So we calculate $|F(\text{head}) - F(\text{foot})|$ to check accuracy of characteristic tracking.

Table 2 lists numerical results for this example. A triangular mesh with 1312 elements and 697 nodes is generated by applying the mesh generator built in the PDE Toolbox in Matlab. The time period is [0, 1] with a uniform partition $\Delta t = 0.01$. We randomly choose a time step $[14\Delta t, 15\Delta t]$ and carry out backward tracking from the barycenter of each triangle. Both the Euler method and the second-order Runge-Kutta method (RK2) are used in combination with different micro time steps δt. To check accuracy of characteristic tracking, the maximum of $|F(\text{head}) - F(\text{foot})|$ is measured with the head running over the centers of all triangles. Clearly, the second-order Runge-Kutta method performs much better than the Euler method.

Table 2. Numerical results for example 2

Micro time step	$\max\{\|F(\text{head}) - F(\text{foot})\|\}$	
$\delta t/\Delta t$	Euler	RK2
1/4	1.387E-4	8.821E-8
1/16	2.774E-5	3.380E-9
1/64	6.605E-6	1.901E-10
1/256	1.632E-6	1.158E-11
1/1024	4.068E-7	7.213E-13

The results in Table 2 reflect the first order and second order accuracy of the Euler method and RK2, respectively. It is also observed in the numerical experiments that no characteristic intersects the boundary, as expected.

Example 3: ELLAM on an unstructured mesh: The domain is $\Omega = [-0.5, 0.5] \times [-0.5, 0.5]$, and the velocity $\mathbf{v} = (2y, -2x)$ is incompressible. The characteristic with head (x, y, t) and foot $(x^*, y^*, 0)$ satisfies

$$\begin{bmatrix} x^* \\ y^* \end{bmatrix} = \begin{bmatrix} \cos(2t) & -\sin(2t) \\ \sin(2t) & \cos(2t) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{12}$$

We consider the time period $[0, T] = [0, \pi]$, which is needed for one complete rotation. The initial condition

$$u_0(x, y) = \exp\Big(- \frac{(x - x_c)^2 + (y - y_c)^2}{2\sigma^2} \Big) \tag{13}$$

is a Gaussian hill centered at $(x_c, y_c) = (-0.25, 0)$ with a standard deviation $\sigma = 0.0447$, which yields $2\sigma^2 = 0.0040$. The exact solution for the convection-diffusion equation with a constant diffusion D and $f = 0$ is

$$u(x, y, t) = \frac{2\sigma^2}{2\sigma^2 + 4Dt} \exp\Big(- \frac{(x^* - x_c)^2 + (y^* - y_c)^2}{2\sigma^2 + 4Dt} \Big), \tag{14}$$

where x^*, y^* are expressed in Eq. (12). In our numerical experiment, we take $D = 10^{-4}$.

In [13], a uniform rectangular mesh with 64×64 elements and a time step $\Delta t = \pi/40$ are used to obtain a numerical solution with $U_{\max} = 0.8302$. Here we use an unstructured triangular mesh with local refinement, (shown in Fig. 11), with only 2662 triangle elements and 1350 nodes. The largest side of triangle elements is $H = 0.12$, while the smallest side is $h = 0.0096$. The Lagrangian $\mathcal{P}_1$ element and the second-order Gaussian quadrature are used on all triangles. The global time step is also $\Delta t = \pi/40$ and the backward Euler tracking uses 40 micro steps. A numerical solution with $U_{\max} = 0.8645$ is obtained. It is known that the exact solution has a maximum of 0.8642. Compared with the rectangular mesh used in [13], we use less than one-third elements and solve for less than one-third unknowns, $(2663/(64^2 * 2) = 0.325, 1350/65^2 = 0.320)$, but obtain an even better numerical solution (see Fig. 11). Therefore, unstructured meshes with local refinement are more

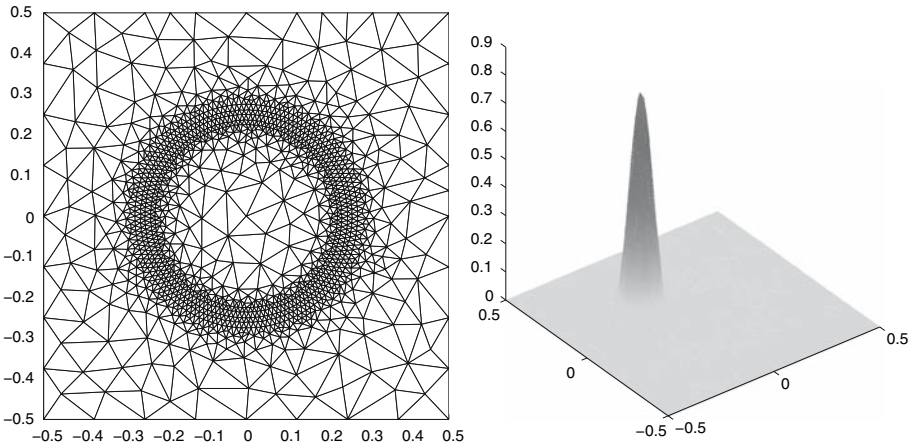

Fig. 11. Example 3: the unstructured mesh and numerical solution

efficient than uniform rectangular meshes, but rely on efficient characteristic tracking. While revising our paper, we noticed the numerical examples presented in [15]. However, the paper does not mention any tracking algorithms. We also notice that the unstructured meshes actually do not align with the steep fronts or other features of the numerical solutions. The cost on numerical integration (24 quadrature points per triangle) seems too high. But we would like to test our tracking algorithm and ELLAM implementation on those examples, when the mesh data become available to us.

5. Discussions

Characteristic tracking can also be implemented based on the neighborhood information in an unstructured mesh. However, large time steps are used in characteristic methods, the foot of a characteristic is not necessarily in the immediate neighborhood of the head. This implementation might end up with looking at neighbors of neighbors, and becomes inefficient.

For our tracking algorithm, the quality of a triangular mesh should not have a big impact on the performance of this algorithm, although it is usually preferred that the *minimum angle requirement* is met for the triangular mesh. For a quasi–uniform triangular mesh, the auxiliary rectangular mesh can be chosen as uniform. For meshes with local refinement, hierarchical uniform auxiliary meshes can be considered to further improve the efficiency.

The minimal side length of triangles in a real mesh could be used as a parameter to guide the generation of the auxiliary mesh. As revealed by our numerical experiments, the smaller the mesh size of the auxiliary mesh, the fewer the triangles intersecting with a rectangular box. Hence, there are less search steps for the triangle containing a point of interest. Since a rectangular mesh is a tensor product of two one-dimensional meshes, only minimal storage is needed for the auxiliary

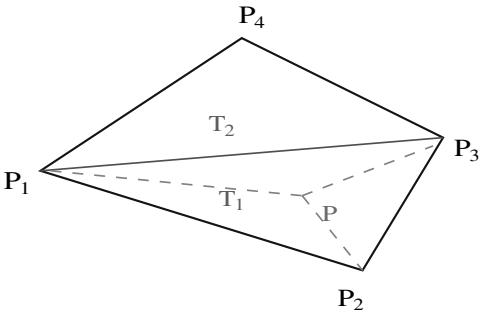

Fig. 12. A quadrilateral is split into 2 triangles

information: the partitions on x-, y-axes and the table of rectangles versus triangles discussed in Sect. 2.2.

In summary, the algorithm presented in this paper is efficient, since it has a lower overhead in setting up the auxiliary mesh and the look-up table and also a minimal number of searching steps in locating points on characteristics. Specifically, we have the following observations regarding the spatial and temporal complexities [6] of our tracking algorithm, assuming a given two-dimensional unstructured mesh has n triangular elements.

- The spatial complexity of the auxiliary mesh is $\mathcal{O}(2\sqrt{n})$. Furthermore, if the auxiliary mesh is constructed as a uniform mesh, then the spatial complexity could be $\mathcal{O}(1)$.
- The auxiliary mesh size is approximately the same as the real mesh size; a triangular intersects with about nine rectangles so, setting up the look-up table requires only $\mathcal{O}(9n)$ operations.
- For a quasi–uniform triangular mesh, the look-up table has spatial complexity $\mathcal{O}(6n)$.
- For a quasi–uniform triangular mesh, the average number of searching steps in the look-up table is 6, or $\mathcal{O}(1)$.

The algorithm can also be extended to the case when quadrilateral elements are included in the real mesh. We only need to set up a criterion for a point to be inside a quadrilateral. This could be done by splitting a quadrilateral into two triangles, as illustrated by Fig. 12. Clearly, this algorithm can also be extended to three-dimensional unstructured meshes.

The algorithm presented in this paper has been implemented in C++, following the object-oriented programming paradigm [2], [11], [12]. Currently we are investigating strategies for dimension-independent implementation for our two and three dimensional algorithms and integration of our code with other software packages based on characteristic tracking, as suggested in [8]. This will be reported in our future work.

Acknowledgement

The authors would like to express their thanks to the anonymous reviewers, whose constructive suggestions have improved the quality of this paper. We thank Ms. Christine Yang and Ms. Rachelle Barr also for their kind help.

References

[1] Arbogast, T., Huang, C. S.: A fully mass and volume conserving implementation of a characteristic method for transport problems. SIAM J. Sci. Comput. *28*, 2001–2022 (2006).
[2] Bangerth, W.: Using modern features of C++ for adaptive finite element methods: dimension-independent programming in deal. II. Proc. IMACS 2000 World Congress, Lausanne, Switzerland, August 21–25, 2000.
[3] Celia, M. A., Russell, T. F., Herrera, I., Ewing, R. E.: An Eulerian-Lagrangian localized adjoint method for the advection-diffusion equation. Adv. Water Res. *13*, 187–206 (1990).
[4] Douglas, J. Jr., Russell, T. F.: Numerical methods for convection-dominated diffusion problems based on combining the method of characteristics with finite element or finite difference procedures. SIAM J. Numer. Anal. *19*, 871–885 (1982).
[5] Giraldo, F. X.: The Lagrange-Galerkin spectral element method on unstructured quadrilateral grids. J. Comput. Phys. *147*, 114–146 (1982).
[6] Knuth, D. E.: The art of computer programming, vol. 1: Fundamental algorithms, 3rd ed. Reading: Addison-Wesley 1997.
[7] LeVeque, R. J.: High-resolution conservative algorithms for advection in incompressible flow. SIAM J. Numer. Anal. *33*, 627–665 (1996).
[8] Liu, J., Ewing, R. E.: An operator splitting method for nonlinear reactive transport equations and its implementation based on DLL and COM. Current trends in high performance computing and its applications. Lecture Notes in Computational Science and Engineering. New York: Springer 2005, pp. 93–102.
[9] Pironneau, O: On the transport-diffusion algorithm and its applications to the Navier-Stokes equations. Numer. Math. *38*, 309–332 (1981/82).
[10] Russell, T. F., Celia, M. A.: An overview of research on Eulerian-Lagrangian localized adjoint methods (ELLAM). Adv. Water Res. *25*, 1215–1231 (2002).
[11] Stroustrup, B.: The C++ programming language, 3rd ed. Reading: Addison-Wesley 1997.
[12] Sun, T., Ewing, R. E., Chen, H., Lyons, S. L., Qin, G.: Objected-oriented programming for general mixed finite element methods. SIAM Workshop in Object Oriented Methods for Intel-operable Scientific and Engineering Computing (also Technical report, Institute for Scientific Computation, Texas A&M University), 1998.
[13] Wang, H., Dahle, H. K., Ewing, R. E., Espedal, M. S., Sharpley, R. C., Man, S.: An ELLAM scheme for advection-diffusion equations in two dimensions. SIAM J. Sci. Comput. *20*, 2160–2194 (1999).
[14] Xiu, X., Karniadakis, G. E.: A semi-Lagrangian high-order method for Navier-Stokes equations. J. Comput. Phys. *172*, 658–684 (2001).
[15] Younes, A., Ackerer, P.: Solving the advection-diffusion equation with the Eulerian-Lagrangian localized adjoint method on unstructured meshes and nonuniform time stepping. J. Comput. Phys. *208*, 384–402 (2005).

J. Liu
Colorado State University
Department of Mathematics
Fort Collins, CO 80523-1874
USA
e-mail: liu@math.colostate.edu

H. Chen, R. E. Ewing and G. Qin
Institute for Scientific Computation
Texas A&M University
College Station, TX 77843-3404
USA
e-mail: hchen@isc.tamu.edu
e-mail: richard-ewing@tamu.edu
e-mail: guan.qin@tamu.edu