Neighbor-Gradient Single-Pass Method for Solving Anisotropic Eikonal Equation

Myong-Song Ho · Ji-Song Pak · Song-Mi Jo · Ju-Song Kim

Received: 12 April 2021 / Revised: 15 December 2021 / Accepted: 21 January 2022 / Published online: 21 March 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
We develop a single pass method for approximating the solution to an anisotropic eikonal equation related to the anisotropic min-time optimal trajectory problem. Ordered Upwind Method (OUM) solves this equation, which is a single-pass method with an asymptotic complexity. OUM uses the search along the accepted front (SAAF) to update the value at considered nodes. Our technique, which we refer to as “Neighbor-Gradient Single-Pass Method”, uses the minimizer of the Hamiltonian, in which the gradient is substituted with neighbor gradient information, to avoid SAAF. Our technique is considered in the context of control-theoretic problem. We begin by discussing SAAF of OUM. We then prove some properties of the value function and its gradient, which provide the key motivation for constructing our method. Based on these discussions, we present a new single-pass method, which is fast since it does not require SAAF. We test this method with several anisotropic eikonal equations to observe that it works well while significantly reducing the computational cost.

Keywords Anisotropic eikonal equation · Single-pass method · Optimal control · Viscosity solution · Ordered upwind method

1 Introduction
Hamilton–Jacobi equations are considered in several contexts, including classical mechanics, front propagation, control problems and differential games [1–5]. For optimal control problems, In particular, the value function can be characterized as the unique viscosity solution of a Hamilton–Jacobi–Bellman equation.

In this paper, we deal with generalized eikonal equations, which are associated to some min-time optimal control problems. These equations are used not only in optimal control problems but also in many fields, such as seismic travel time estimation, geodesics problems [6–8], etc.
Some types of generalized eikonal equations associated to some min-time optimal control problems are given as follows [9]:

\[
\sup_{a \in S_1} \{-a \cdot \nabla u(x)\} = 1 \quad (\text{homogeneous eikonal equation}) \tag{1}
\]

\[
\sup_{a \in S_1} \{-f_1(x)a \cdot \nabla u(x)\} = 1 \quad (\text{nonhomogeneous eikonal equation}) \tag{2}
\]

\[
\sup_{a \in S_1} \{-f_2(a)a \cdot \nabla u(x)\} = 1 \quad (\text{homogeneous anisotropic eikonal equation}) \tag{3}
\]

\[
\sup_{a \in S_1} \{-f_3(x, a)a \cdot \nabla u(x)\} = 1 \quad (\text{nonhomogeneous anisotropic eikonal equation}) \tag{4}
\]

where \(x \in \Omega \setminus \partial \Omega\), \(\Omega\) is a bounded convex open set in \(R^d\), \(\partial \Omega\) is a nonempty target set and \(f_1, f_2, f_3\) are given positive and Lipchitz continuous scalar functions, and \(S_1\) is the unit ball in \(R^d\), representing the set of the admissible controls. We will always consider the Dirichlet condition \(u = 0\) on \(\partial \Omega\). To simplify the notations, we restrict the discussion to the case \(d = 2\).

Various numerical methods based on Semi-Lagrangian schemes (which directly discretize Bellman’s dynamic programming principle) for solving Eqs. (1)–(4) have been proposed. [10–13]

Fast marching method (FMM) [2–4, 14] has been proposed to solve the isotropic eikonal equations such as (1) and (2). FMM is a local single-pass method, which is extremely fast.

An algorithm is said to be single-pass if each grid node is recomputed at most \(\gamma\) times where \(\gamma\) is a priori a (small) number which depends only on the equation and on the grid structure (not on the number of grid nodes), and said to be local if the computation at any grid node involves only the values of first neighboring nodes [9]. Single-pass methods divide the grid nodes in three groups: Accepted (\(\text{Acc}\)) region, Considered (\(\text{Cons}\)) region and Far (\(\text{Far}\)) region. \(\text{Acc}\) is the set of grid nodes whose values are definitively computed, \(\text{Cons}\) is the set of grid nodes whose values are computed but their values are not yet final, and \(\text{Far}\) is the set of grid nodes whose values are not yet computed. [9].

Unfortunately, FMM fails for anisotropic eikonal equations such as (3) and (4) because the neighboring simplex from which the characteristic approaches a node \(x\) may contain another node \(y\) such that causality does not hold: \(u(x) \leq u(y)\).

For the anisotropic eikonal equation, Ordered Upwind Method (OUM), which is a single-pass method, has been proposed [15–17]. OUM can solve the general anisotropic eikonal equation and it has an asymptotic complexity only a constant factor (related to the degree of anisotropy) worse than FMM. It is shown that the solutions produced by OUM in the boundary value formulation converge at a rate of at least the square root of the largest edge length in the mesh in terms of maximum error. [18]. OUM avoids the difficulty of FMM by searching along the accepted front (SAAF) to find a set of neighboring nodes (which may not be direct neighbors) whose values have been accepted, and then constructing a virtual simplex with these nodes from which to update. Although SAAF does not degrade the asymptotic complexity of OUM, it does significantly increase the computational cost in practice [19–21]. MAOUM (Monotone Acceptance OUM), which is a generalization of OUM, has been proposed. MAOUM computes the solution in a Fast Marching fashion, but employs large stencils that are pre-computed for each grid, and thus MAOUM is a two-pass method. [9].

Efficient methods such as Fast Sweeping Method (FSM) [22–27] and Fast Iterative Method (FIM) [28–30] have been proposed for solving anisotropic eikonal equations. These are not single-pass method.
FSM is an iterative method for solving anisotropic eikonal equations and it is similar to classical iterative method (ITM), but the grid is visited in a multiple-direction predefined order. Usually, a rectangular grid is iteratively swept along four directions: \(N \to S, E \to W, S \to N\) and \(W \to E\), where \(N, S, E, W\) stand for North, South, East and West respectively. FSM has been shown to be much faster than ITM, but as ITM, it is not single-pass. FIM shares some features with both iterative and single-pass methods. FIM updates only a few grid nodes in each iteration. These nodes are stored in a separated list, called active list. FIM is faster than classical iterative method, but as FSM, it is not single-pass. On the other hand, for solving the generalized eikonal equation in a moving medium, Characteristic Fast Marching Method (CFMM) \([31, 32]\), Operator Split-based Method \([33, 34]\) and Improved Characteristic Fast Marching Method (ICFMM) \([35]\) have been proposed. CFMM, ICFMM and Operator Split-based Method are not for anisotropic eikonal Eq. (3) but these are numerical methods for solving generalized eikonal equation in a moving medium.

\[
(\nabla T(x))^2 - \frac{(1 - \nabla(x) \cdot \nabla T(x))^2}{(F(x))^2} = 0, \quad x \in \Omega; \quad T(x) = q(x), \quad x \in \Gamma
\]

The CFMM and ICFMM are local single methods, which is very fast, but these methods fail when the Mach number \(\frac{\|v\|}{F}\) is comparable to 1. Operator Split-based Method solves generalized eikonal equation even when the Mach number is comparable to 1 but it is not single pass.

In this paper, we focus on OUM, which is a single-pass method for solving the anisotropic eikonal Eqs. (3) and (4).

We need to recall the procedures for updating the value at \(x \in Cons\) in OUM:

(a) Search along the accepted front to find the neighboring nodes (which may not be direct neighbors of \(x\)) that the distance from the line segment joining them to \(x\) is smaller than \(h \gamma\), where \(h\) is diameter of grid and \(\gamma\) is the anisotropic coefficient defined in \([16]\).

(b) For every found neighboring nodes, construct a virtual simplex from these nodes and calculate the approximated value from this simplex using the semi-Lagrangian scheme.

(c) Update the value at \(x\) with the minimum value among the approximated values for every found neighboring nodes.

Here, procedure (a) is SAAF, which increases the computational cost. There exists one simplex, which contains the characteristic for \(x\) among the virtual simplexes constructed by procedure (b). So procedures (a), (b) and (c) allow updating the value at \(x\) from the characteristic direction.

If the characteristic direction at \(x\) was a priori known, the computational cost decreases, since one would be able to update the value from the characteristic direction, without SAAF, i.e., update the value from the intersection point between the accepted front and the half line (initiated at the point \(x\) and pointing in the characteristic direction). Since the anisotropic eikonal Eqs. (3) and (4) are nonlinear, one cannot know the characteristic direction in advance.

We also need to recall the following well-known formula \([16]\):

\[
a_1 = \arg \min_{a \in S_1} \{ (\nabla u(x) \cdot a) f(x, a) \} \quad (5)
\]

where \(a_1\) is the optimal control value (the characteristic direction) at \(x\) (i.e., the minimizer of Hamiltonian). From this formula, if \(\nabla u(x)\) was a priori known, one would be able to obtain the optimal control value according to (1.5) and update the value at \(x\) without using SAAF. Unfortunately, \(\nabla u(x)\) is also unknown unless \(x\) is accepted.
We use a technique for updating the value at \( x \) from the minimizer of Hamiltonian, in which the gradient is substituted with the gradient at a neighboring accepted node.

The following questions may arise in using the above technique.

First question is that whether there exists at least one accepted neighboring grid node such that the gradient at this node is similar to \( \nabla u(x) \).

Second question is that whether the resulting error caused by using the minimizer of Hamiltonian (substituted with the gradient at the above neighboring accepted node) instead of the optimal control value may accumulate during the iteration of the algorithm and ultimately may fail to ensure convergence.

The answer to the first question is given in Lemma 2.1 of Sect. 2.

Before answering the second question, we pay attention to the following fact.

In the case of using OUM, one assumes that the approximated values at the accepted nodes are the same as the real values of value function, at a certain stage of the algorithm. Suppose that \( \tilde{x} \) is the node with smallest value among all the considered nodes, then the error between the approximated value and real value at \( \tilde{x} \) is \( O(h^2) \). This fact was demonstrated for validating OUM in [16]. The answer to the second question is similar as suggested above. Detailed proof is given in Sects. 2 and 4. We call the resulting method the Neighbor-Gradient Single-Pass Method.

In Sect. 2, we prove some properties related to the validity of our technique, including the answers to two above questions. Based on this, in Sect. 3, we present approximation scheme that updates the values at considered nodes by using the gradients at accepted neighboring nodes. In Sect. 4, we present a neighbor-gradient single-pass algorithm, implementation and computational complexity of our algorithm. In Sect. 5, we test the efficiency and convergence through numerical experiments. Section 6 discusses future directions and concludes the paper.

Some basic properties of the value function are also presented in Appendix.

## 2 Anisotropic Min-Time Problem and Properties of the Value Function

### 2.1 Statement of Problem

We consider an optimal trajectory problem for a vehicle moving with the speed depending upon the direction of motion and the current position of the vehicle inside the domain \( \Omega \).

The dynamics of the vehicle is defined by

\[
\begin{align*}
y'(t) & = f(y(t), \alpha(t))a(t) \\
y(0) & = x \in \Omega
\end{align*}
\]

where \( y(t) \) is the position of the vehicle at time \( t \), \( S_1 = \{ a \in \mathbb{R}^2 \mid \| a \| = 1 \} \) is the set of admissible control values, and \( A = \{ \alpha : \mathbb{R}^+ \rightarrow S_1 \mid \alpha(\cdot) \text{ is measurable} \} \) is the set of admissible controls.

We are interested in studying \( y(t) \) only while the vehicle remains inside \( \Omega \), i.e., until the exit time

\[
T(x, \alpha(\cdot)) = \inf\{ t \in \mathbb{R}^+ \mid y(t) \in \partial \Omega \}
\]

Then a min-time optimal trajectory problem is to find a control \( \alpha(\cdot) \) which minimizes the exit time \( T(x, \alpha(\cdot)) \). The value function of the min-time optimal trajectory problem is
defined by

\[ u(x) = \inf_{a(\cdot) \in A} T(x, \alpha), \quad x \in \Omega \tag{6} \]

Then this value function is the unique viscosity solution of the following Hamilton–Jacobi-Bellman’s equation. [16]

\[
\min_{a \in \mathcal{S}_1} \{ f(x, a) a \cdot \nabla u(x) \} = -1, \quad x \in \Omega \setminus \partial \Omega \tag{7}
\]

\[ u(x) = 0, \quad x \in \partial \Omega \]

It is clear that Eq. (6) coincides the anisotropic eikonal Eq. (4).

We will assume that there exists constants \( F_1, \ F_2 \) such that.

\[ F_1 \leq f(x, a) \leq F_2 \quad (F_1, F_2 > 0) \quad \text{for all} \quad x \in \Omega \quad \text{and} \quad a \in \mathcal{S}_1. \]

and that \( f(x, a) \) is Lipschitz continuous in the state variable, uniformly in the control value, that is.

\[ |f(x, a) - f(y, a)| < L \|x - y\| \quad \text{for all} \quad x, y \in \Omega \quad \text{and} \quad a \in \mathcal{S}_1. \]

Hamilton–Jacobi-Bellman PDE (2.1) can be rewritten in the form \( H(x, \nabla u(x)) = -1 \), where the Hamiltonian \( H(x, p) = -\min_{a \in \mathcal{S}_1} \{ p \cdot af(x, a) \} \). Moreover \( H \) is convex in the second argument. [16].

Assume that a regular rectangular grid \( X \) is defined on \( \Omega \), where the grid sizes in \( x \)- and \( y \)-coordinates are the same i.e., \( \Delta x = \Delta y \).

We define \( h := \sqrt{2} \Delta x \) and for each grid node \( x = (x_i, y_j) \in X \), we denote

\[ N(x) := \{(x_{i-1}, y_j), (x_{i+1}, y_j), (x_i, y_{j-1}), (x_i, y_{j+1}) \} \]

\[ ND(x) := \{(x_{i-1}, y_j), (x_{i+1}, y_j), (x_i, y_{j-1}), (x_i, y_{j+1}), (x_{i-1}, y_{j-1}), (x_{i+1}, y_{j+1}) \} \]

2.2 The Properties of the Value Function

We will prove some properties of the value function (7) and its gradient.

**Lemma 2.1** Consider two points \( x_1, x_2 \in \Omega \) and assume that \( u \) is differentiable at \( x_1 \). We define.

\[ V(x_1, x_2) := u(x_2) + \frac{\| x_2 - x_1 \|}{f(x_1, a_1)} \tag{8} \]

where \( a_1 = x_2 - x_1 / \| x_2 - x_1 \| \). Then.

\[ V(x_1, x_2) - u(x_1) = \| x_1 - x_2 \| \left( \nabla u(x_1) \cdot a_1 + \frac{1}{f(x_1, a_1)} \right) + O(\| x_1 - x_2 \|^2). \]

**Proof** By the definition of \( V(x_1, x_2) \) and differentiability of \( u \) at \( x_1 \), we obtain

\[ V(x_1, x_2) = u(x_1) + u(x_2) - u(x_1) + \frac{\| x_1 - x_2 \|}{f(x_1, a_1)} \]

\[ = u(x_1) + \nabla u(x_1) \cdot a_1 \| x_1 - x_2 \| + O(\| x_1 - x_2 \|^2) + \frac{\| x_1 - x_2 \|}{f(x_1, a_1)} \]

\[ = u(x_1) + \| x_1 - x_2 \| \left( \nabla u(x_1) \cdot a_1 + \frac{1}{f(x_1, a_1)} \right) + O(\| x_1 - x_2 \|^2). \]

\[ \square \]
Lemma 2.2  Consider a grid node $x \in \Omega$. Suppose that the characteristic for $x$ intersects the line segment $x_1 x_2$ at $x_0$, where $x_1, x_2 \in ND(x)$ and $x_1 \in N(x_2)$. Assume that $u$ is differentiable at $x, x_1$ and $x_2$. Then,

$$\|\nabla u(x_i) - \nabla u(x)\| = O(h), \quad i = 1, 2$$

Proof  By Lemma 3 in Appendix, we get

$$\|\nabla u(x_0) - \nabla u(x)\| = O(h)$$

Without loss of generality, let $x_2 \in N(x), \quad x_1 \in ND(x)(x_1 \notin N(x))$.

(i) Let us prove that $\|\nabla u(x_2) - \nabla u(x)\| = O(h)$.

By the fact that characteristics never emanate from the shocks-non differentiable point [16] and that $u$ is differentiable at $x_2$, we see that $u$ is differentiable at $x_0$. Therefore, we get

$$u(x_2) = u(x_0 + x_2 - x_0) = u(x_0) + u_x(x_0)\|x_2 - x_0\| + O\|x_2 - x_0\|^2$$

$$u(x_0) = u(x_2 + x_0 - x_2) = u(x_2) - u_x(x_2)\|x_2 - x_0\| + O\|x_2 - x_0\|^2$$

and thus

$$(u_x(x_0) - u_x(x_2))\|x_2 - x_0\| = O\|x_2 - x_0\|^2$$

$$u_x(x_0) - u_x(x_2) = O\|x_2 - x_0\| = O(h).$$

Using (9) and (10), it’s clear that

$$u_x(x) - u_x(x_2) = O(h)$$

Since $x$ and $x_2$ differ only in their second component, we get

$$u(x_2) = u(x + x_2 - x) = u(x) + u_y(x)\|x_2 - x\| + O\|x_2 - x\|^2$$

$$u(x) = u(x_2 + x - x_2) = u(x_2) - u_y(x_2)\|x_2 - x\| + O\|x_2 - x\|^2$$

and similar to (10), we get

$$u_y(x) - u_y(x_2) = O\|x_2 - x\| = O(h)$$

Therefore, from (11) and (12), we obtain $\|\nabla u(x) - \nabla u(x_2)\| = O(h)$.

(ii) Let us prove that $\|\nabla u(x_1) - \nabla u(x)\| = O(h)$.

Similar to the proof of (i), we can get $u_x(x_1) - u_x(x_0) = O(h)$.

Using (9), we obtain $u_x(x_0) - u_x(x) = O(h)$. Therefore,

$$u_x(x_1) - u_x(x) = O(h)$$

Considering the directional derivative of $u$ at $x_1$ in the direction $\vec{x_1}x$, we can get

$$u_{x_1 x}(x_1) = \frac{1}{\sqrt{2}}u_x(x_1) + \frac{1}{\sqrt{2}}u_y(x_1) = \frac{u(x) - u(x_1)}{\|x - x_1\|} + O(h)$$

$$= \frac{u(x_0) + \|x - x_0\|/ f(x_0, a) - u(x_1)}{\sqrt{2}h} + O(h)$$

and thus

$$\sqrt[3]{Springer}$$
\[ u_x(x_1) + u_y(x_1) = \frac{u(x_0) - u(x_1)}{h} + \frac{\|x - x_0\|}{f(x_0, a)h} + O(h) \]

\[ = \frac{u(x_0) - u(x_1)\|x_0 - x_1\|}{\|x_0 - x_1\|} + \frac{\|x - x_0\|}{f(x_0, a)h} + O(h) \]

\[ = u_x(x_1)\frac{\|x_0 - x_1\|}{h} + \frac{\|x - x_0\|}{f(x_0, a)h} + O(h). \]

Therefore,

\[ u_x(x_1)\frac{\|x_0 - x_2\|}{h} + u_y(x_1) = \frac{\|x - x_0\|}{f(x_0, a)h} + O(h) \quad (14) \]

Similarly, we can get

\[ u_x(x)\frac{\|x_0 - x_2\|}{\|x - x_0\|} + u_y(x)\frac{h}{\|x - x_0\|} = \frac{u(x) - u(x_0)}{\|x - x_0\|} = \frac{1}{f(x_0, a)} \]

\[ u_x(x)\frac{\|x_0 - x_2\|}{h} + u_y(x) = \frac{\|x - x_0\|}{f(x_0, a)h} \quad (15) \]

Then from (14) and (15), it’s clear that

\[ u_y(x_1) - u_y(x) = (u_x(x) - u_x(x_1))\frac{\|x_0 - x_2\|}{h} = \frac{\|x_0 - x_2\|}{h}O(h) = O(h) \quad (16) \]

Using (13) and (16), we obtain \( \|\nabla u(x) - \nabla u(x_1)\| = O(h) \)

\[ \square. \]

**Lemma 2.3** Consider a grid node \( x \in \Omega \). Suppose that the characteristic for \( x \) intersects the line segment \( x_1x_2 \), where \( x_1, x_2 \in ND(x) \) and \( x_1 \in N(x_2) \). Assume that \( u \) is differentiable at \( x, x_1 \) and \( x_2 \). Suppose that \( a_0 \) is the optimal control value at \( x \). For \( x' \in \{x_1, x_2\} \), we denote \( \hat{a} := \arg \min_{a \in S_1} \{\nabla u(x') \cdot a\cdot f(x, a)\} \). Then,

\[ \|\hat{a} - a_0\| = O(h) \]

\[ \nabla u(x') \cdot \hat{a} + \frac{1}{f(x, \hat{a})} = O(h). \]

**Proof** We define \( A(p) := \arg \min_{a \in S_1} \{p \cdot a \cdot f(x, a)\} \). In particular, we set \( A(\nabla u(x')) = \hat{a} \).

We denote \( \tilde{H}(p) := \nabla u(x) \cdot A(p) f(x, A(p)) \). Let us prove that \( \tilde{H}(p) \) is continuous at \( p = \nabla u(x) \).

Since \( a_0 \) is the optimal control value at \( x \), we have

\[ a_0 = \arg \min_{a \in S_1} \{\nabla u(x) \cdot a \cdot f(x, a)\} = A(\nabla u(x)), \quad \tilde{H}(\nabla u(x)) = \nabla u(x) \cdot a_0 f(x, a_0) = -1 \quad (17) \]

By the fact that \( A(\nabla u(x)) = a_0 \) from (17), we obtain

\[ p \cdot a_0 f(x, a_0) - \tilde{H}(\nabla u(x)) \leq \left| \tilde{H}(\nabla u(x)) - p \cdot a_0 f(x, a_0) \right| \]

\[ = |\nabla u(x) \cdot a_0 f(x, a_0) - p \cdot a_0 f(x, a_0)| \]

\[ = |(\nabla u(x) - p) \cdot a_0 f(x, a_0)| \leq \|\nabla u(x) - p\| \cdot f(x, a_0) \]

\[ = \|\nabla u(x) - p\| f_2. \]

From (17), it is clear that

\[ p \cdot a_0 f(x, a_0) \leq -1 + \|\nabla u(x) - p\| f_2. \]
Otherwise, since \( A(p) = \arg \min_{a \in S_1} \{ p \cdot af(x, a) \} \), we get

\[
p \cdot A(p)f(x, A(p)) < p \cdot a_0 f(x, a_0)
\]

(19)

In addition, we can get

\[
\tilde{H}(p) - p \cdot A(p)f(x, A(p)) \leq |\tilde{H}(p) - p \cdot A(p)f(x, A(p))| \\
= |\nabla u(x) \cdot A(p)f(x, A(p)) - p \cdot A(p)f(x, A(p))| \\
= |(\nabla u(x) - p) \cdot A(p)f(x, A(p))| \leq \|\nabla u(x) - p\|f(x, A(p)) \leq \|\nabla u(x) - p\|f_2
\]

and thus

\[
\tilde{H}(p) \leq p \cdot A(p)f(x, A(p)) + \|\nabla u(x) - p\|f_2
\]

(20)

From (17), we get

\[-1 = \nabla u(x) \cdot a_0 f(x, a_0) \leq \nabla u(x) \cdot A(p)f(x, A(p)) = \tilde{H}(p).
\]

(21)

By combining (18–21), we can get

\[-1 \leq \tilde{H}(p) \leq p \cdot A(p)f(x, A(p)) + \|\nabla u(x) - p\|f_2 \leq p \cdot a_0 f(x, a_0) + \|\nabla u(x) - p\|f_2 \leq -1 + 2\|\nabla u(x) - p\|f_2
\]

and thus

\[
|\tilde{H}(p) - (-1)| = |\tilde{H}(p) - \tilde{H}(\nabla u(x))| \leq 2\|\nabla u(x) - p\|f_2.
\]

(22)

Therefore, \( \tilde{H}(p) \) is continuous at \( p = \nabla u(x) \).

Since \( \|\nabla u(x) - \nabla u(x')\| = O(h) \) by Lemma 2.2, from (22), we get

\[
\nabla u(x) \cdot \hat{a} f(x, \hat{a}) + 1 = |\nabla u(x) \cdot \hat{a} f(x, \hat{a}) - (-1)| \\
= |\nabla u(x) \cdot A(\nabla u(x')) f(x, A(\nabla u(x'))) - (-1)| \\
= |\tilde{H}(\nabla u(x)) - \tilde{H}(\nabla u(x'))| \leq 2\|\nabla u(x) - \nabla u(x')\|f_2 \\
= 2f_2 O(h) = O(h).
\]

(23)

The equality \( \nabla u(x) \cdot \hat{a} f(x, \hat{a}) + 1 = |\nabla u(x) \cdot \hat{a} f(x, \hat{a}) - (-1)| \) since \( \min_{a \in S_1} \{\nabla u(x) \cdot af(x, a)\} = -1 \).

From (23) and bound of \( f \), we can get \( \nabla u(x) \cdot \hat{a} + \frac{1}{f(x,a)} = \frac{O(h)}{f(x,a)} = O(h) \).

Now, let us prove that \( \|\hat{a} - a_0\| = O(h) \).

Let us consider the sequence \( \{p_n\} \) such that \( p_n \rightarrow \nabla u(x) \).

Since \( A(p_n) \) is bounded sequence, it has convergent subsequence \( \{A(p_k)\}(A(p_k) \rightarrow a^*) \).

By the fact that \( p_k \rightarrow \nabla u(x) \) and that \( \tilde{H}(p) \) is continuous at \( p = \nabla u(x) \),

\( \tilde{H}(p_k) = \nabla u(x) \cdot A(p_k)f(x, A(p_k)) \rightarrow \tilde{H}(\nabla u(x)) = -1 \).

On the other hand, by the fact that \( \nabla u(x) \cdot af(x, a) \) is continuous in \( a \), we get

\[
\nabla u(x) \cdot A(p_k)f(x, A(p_k)) \rightarrow \nabla u(x) \cdot a^* f(x, a^*)
\]

and thus \( \nabla u(x) \cdot a^* f(x, a^*) = -1 \). Therefore \( a^* = a_0 \). Since \( A(p_n) \) is bounded sequence and the limit of all convergent subsequences are the same as \( a_0 \), we get \( A(p_n) \rightarrow a_0 \).

Therefore, \( A(p) \) is continuous at \( p_0 = \nabla u(x) \).
To complete the proof, we recall that \( A(\nabla u(x')) = \hat{a}, \quad A(\nabla u(x)) = a_0 \) and \( \| \nabla u(x) - \nabla u(x') \| = O(h) \).

Form the above lemmas (Lemma 2.1, 2.2 and 2.3), we can claim the following theorem which provides the key motivation for constructing our method.

**Theorem 2.1** Consider a simple closed curve \( \Gamma \subset \Omega \setminus \partial \Omega \) with the property that for any point \( x \) on \( \Gamma \), there exists a grid node \( \bar{x} \) inside \( \Gamma \) such that \( \| x - \bar{x} \| < h \). Suppose a grid node \( \bar{x} \) is such that \( u(\bar{x}) \leq u(x_i) \) for all the grid nodes \( x_i \in X \) inside \( \Gamma \). Then the following statements hold.

(i) **For the grid node \( \bar{x} \), Suppose that the characteristic for \( \bar{x} \) intersects the line segment \( x_1x_2 \) at \( x_0 \), where \( x_1, x_2 \in ND(\bar{x}) \) and \( x_1 \in N(x_2) \). Then at least one of \( x_1 \) and \( x_2 \) is on \( \Gamma \) or outside \( \Gamma \).

\[
\hat{a} := \arg \min_{a \in S_1} \| \nabla u(x_1) \cdot af(\bar{x}, a) \|, \quad V(\bar{x}, \hat{a}) := u(\hat{x}) + \frac{\| \bar{x} - \hat{x} \|}{f(\bar{x}, \hat{a})}
\]

(ii) **Suppose that \( x_1 \) is the grid node on \( \Gamma \) or outside \( \Gamma \) and we denote**

\[
\hat{a} := \arg \min_{a \in S_1} \| \nabla u(x_1) \cdot af(\bar{x}, a) \|, \quad V(\bar{x}, \hat{a}) := u(\hat{x}) + \frac{\| \bar{x} - \hat{x} \|}{f(\bar{x}, \hat{a})}
\]

Then

\[
V(\bar{x}, \hat{a}) - u(\bar{x}) = O(h^2),
\]

where \( \hat{x} \) is the intersection point between \( \Gamma \) and a half-line (initiated at the point \( \bar{x} \) and pointing in the direction of \( \hat{a} \)).

**Proof**

(i) **By Lemma 4 in Appendix, we get that** \( \min (u(x_1), u(x_2)) < u(\bar{x}) \).

Without loss of generality, assume that \( u(x_1) < u(\bar{x}) \).

Since \( u(\bar{x}) \leq u(x_i) \) for all the grid nodes \( x_i \in X \) inside \( \Gamma \), \( x_1 \) is not inside \( \Gamma \). Therefore \( x_1 \) is on \( \Gamma \) or outside \( \Gamma \).

(ii) **Suppose that \( a_0 \) is the optimal control value at \( \bar{x} \), i.e., \( a_0 = \arg \min_{a \in S_1} \| \nabla u(\bar{x}) \cdot af(\bar{x}, a) \| \).**

By [16, Lemma 3.1], we get

\[
\| \bar{x} - \hat{x} \| \leq h \gamma \quad (24)
\]

We define \( \tilde{a} := \frac{\bar{x} - \hat{x}}{\| \bar{x} - \hat{x} \|} \). By the Mean Value Theorem of Differentially, we get \( \| \tilde{a} \| = O(h) \).

Here, \( \tilde{a} \) indicates the angle between the vector \( \hat{a} \) and \( a_0 \).

On the other hand, By Lemma 2.3, we obtain \( \| \hat{a} - a_0 \| = O(h) \). Therefore, \( \hat{a} \) indicates the angle between the vector \( \hat{a} \) and \( \tilde{a} \).

Using the continuity of \( \Gamma \), we get

\[
\| \hat{x} - \bar{x} \| - \| \hat{x} - \bar{x} \| = O(h).
\]

Combining with (18), we get

\[
\| \hat{x} - \bar{x} \| = O(h).
\]

By Lemma 2.1, we obtain

\[
V(\bar{x}, \hat{a}) - u(\bar{x}) = \| \hat{x} - \bar{x} \| \left( \nabla u(\bar{x}) \cdot \hat{a} + \frac{1}{f(\bar{x}, a)} \right) + O(\| \hat{x} - \bar{x} \|)^2.
\]

By the fact that \( \| \hat{x} - \bar{x} \| = O(h) \) and Lemma 2.3, we get

\[
\nabla u(\bar{x}) \cdot \hat{a} + \frac{1}{f(\bar{x}, a)} = O(h).
\]

Therefore, \( V(\bar{x}, \hat{a}) - u(\bar{x}) = O(h^2) \). □
Remark 2.1 When we consider this theorem in the context of our algorithm, \( \Gamma' \) is accepted front and the fact that \( x' \) is on \( \Gamma' \) or outside \( \Gamma' \) means that \( x' \) is in \( \text{Acc} \), therefore it is possible to approximate the gradient of value function at \( x' \).

3 Approximation of the Value Function

As before, all the grid nodes are divided into 3 groups: \( \text{Acc} \) (accepted region), \( \text{Cons} \) (considered region), \( \text{Far} \) (far region). \( \text{AFF} \) is defined as a set of the accepted grid nodes \( x_i \) such that there exists a considered node \( \tilde{x} \in ND(x_i) \). Define the set \( \text{AFF} \) of the line segments \( x_jx_k \), where \( x_j, x_k \) are grid nodes in the \( \text{AFF} \) such that \( x_j \in N(x_k) \). For considered grid node \( x \), we denote \( \text{NF}(x) := \{ x_jx_k \in S_{\text{AFF}} | \exists \tilde{x} \in x_jx_k, ||\tilde{x} - x|| < h \gamma \} \).

Based on Theorem 2.1, we use a technique for approximating the value function by using a gradient at accepted neighboring grid nodes.

We define \( U \) to be the definitely computed value at accepted grid nodes.

For the grid node \( \tilde{x} = (x_j, y_j) \in \text{Acc} \), we obtain \( \nabla U(\tilde{x}) = (DX, DY) \) as follows.

Let \( x_L = (x_i-1, y_j), x_R = (x_{i+1}, y_j), x_D = (x_i, y_{j-1}), x_U = (x_i, y_{j+1}) \).

if \( x_L \in \text{Acc}, x_R \in \text{Acc} \) then \( DX = (U(x_R) - U(x_L))/(2\Delta x) \).

if \( x_L \in \text{Acc}, x_R \notin \text{Acc} \) then \( DX = (U(\tilde{x}) - U(x_L))/\Delta x \).

if \( x_R \in \text{Acc}, x_L \notin \text{Acc} \) then \( DX = 0 \).

if \( x_D \in \text{Acc}, x_U \in \text{Acc} \) then \( DY = (U(x_U) - U(x_D))/(2\Delta y) \).

if \( x_D \in \text{Acc}, x_U \notin \text{Acc} \) then \( DY = (U(\tilde{x}) - U(x_D))/\Delta y \).

if \( x_U \in \text{Acc}, x_D \notin \text{Acc} \) then \( DY = 0 \).

Now we propose a way to obtain the approximate value of \( U \) at \( x \in \text{Cons} \) using the gradients at the neighboring grid nodes belonging to \( \text{Acc} \). For \( x \in \text{Cons} \), we define

\[
G(x) := \bigcup_{\tilde{x} \in ND(x) \cap \text{Acc}} \nabla U(\tilde{x}), \quad A(x) := \left\{ \hat{a} = \arg\min_{a \in S_1} \tilde{g} \cdot af(x, a), \quad \tilde{g} \in G(x) \right\}
\]

Let \( l_{\hat{a}} \) be a half line initiated at the point \( x \) and pointing in the direction of \( \hat{a} \).

Suppose that the half line \( l_{\hat{a}} \) intersects the segment \( x_jx_{k_0} \in \text{NF}(x) \) at the point \( \bar{x} \). We set

\[
V_{\hat{a}}(\bar{x}) := V_{x_jx_{k_0}}(\bar{x}) = \frac{||\tilde{x} - x||}{f(x, \tilde{x} - x/||x - \tilde{x}||)} + \lambda U(x_j) + (1 - \lambda)U(x_{k_0}), \text{where} \quad \bar{x} = \lambda x_j + (1 - \lambda)x_{k_0} \text{and if none of every half line} \ l_{\hat{a}} \text{intersects} \ \text{NF}(x), \ \text{put} \ V_{\hat{a}}(\bar{x}) = \infty.
\]

For \( x' \in ND(x) \cap \text{Acc} \) we define \( V_{x'}(x) = \frac{||x' - x||}{f(x, x' - x/||x' - x||)} + U(x') \).

From the above discussion, we update the value at \( x \) as following.

\[
V(x) := \min_{\hat{a} \in A(x)} V_{\hat{a}}(x), \quad \min_{x' \in ND(x) \cap \text{Acc}} V_{x'}(x)
\] (25)

Here, we introduce the approximation scheme of OUM [16].

\[
V(x) := \min_{x_jx_k \in \text{NF}(x)} \min_{x' \in x_jx_k} V_{x_jx_kx'}(x)
\] (26)
4 Neighbor-Gradient Single Pass Algorithm

4.1 Algorithm

1. Move all grid nodes into Far and put \( U(x) = \infty \) for all \( x \in X \).
2. Move all grid nodes \( y(\partial \Omega \cap X) \) into Acc. (\( U(y) = 0 \)).
3. Update the value at the Far grid nodes neighboring to Acc by using (3.2) and move them into Acc, and update the gradient at all grid nodes in Acc. (See Sect. 3).
4. Move all the Far grid nodes neighboring to Acc into Acc and evaluate the tentative values \( V \) at these points by using (25).
5. Find the grid node \( \tilde{x} \) with the smallest value of \( V \) among all the grid nodes in Cons.
6. Move \( \tilde{x} \) into Acc, \( U(\tilde{x}) = V(\tilde{x}) \), and reevaluate the gradient at \( \tilde{x} \) and at all grid nodes in \( ND(\tilde{x}) \cap Acc \). (See Sect. 3).
7. Reevaluate the tentative value \( V \) for all the grid nodes in \( ND(\tilde{x}) \cap Acc \) and move the grid nodes in \( ND(\tilde{x}) \cap Far \) into Cons.
8. If Cons is not empty, then go to 5.

We present the pseudo code of the algorithm.

In the pseudo code, the function \texttt{updategradient}(x) reflects the gradient approximation scheme in Sect. 3 (Tables 1 and 2).

4.2 Implementation

In the first step of Algorithm, we set \( Acc = \partial \Omega \cap X \) and \( U(x) = 0(\ x \in Acc \). Therefore one cannot approximate the gradient at the nodes in Acc. So, we update the value at considered nodes by using OUM and put all the considered nodes into Acc. From [16, Lemma 3.1],

| Table 1 | Pseudo code of the algorithm |
|---------|-------------------------------|
|         | for each \( x \in X \) do \( V(x) \leftarrow \infty \) and add \( x \) to Far |
|         | for each \( x \in \partial \Omega \cap X \) do |
|         | \( u(x) \leftarrow 0 \) |
|         | add \( x \) to Acc |
|         | for each neighbor \( \tilde{x} \in Far \) of \( x \) do |
|         | \( U(\tilde{x}) \leftarrow \min(V(\tilde{x}), \|\tilde{x} - x\|/f(\tilde{x}, \|\tilde{x} - x\|)) \) |
|         | add \( \tilde{x} \) to Acc |
|         | end |
|         | end |
|         | for each \( x \in Acc \) do UpdateGradient(\( x \)) |
|         | for each \( x \in Acc \) do |
|         | for each neighbor \( \tilde{x} \in Far \) of \( x \) do UpdateNarrowBand(\( \tilde{x} \)) |
|         | end |
|         | While Cons \( \neq \emptyset \) do |
|         | \( x_{\text{min}} \leftarrow \arg\min_{x \in Cons} V(x) \) |
|         | add \( x_{\text{min}} \) to Acc |
|         | UpdateGradient(\( x_{\text{min}} \)) |
|         | for each neighbor \( \tilde{x}_{\text{min}} \in Acc \) of \( x_{\text{min}} \) do UpdateGradient(\( \tilde{x}_{\text{min}} \)) |
|         | for each neighbor \( \tilde{x}_{\text{min}} \in Acc \) of \( x_{\text{min}} \) do UpdateNarrowBand(\( \tilde{x}_{\text{min}} \)) |
|         | end |
approximated values at these nodes have second order accuracy. To increase the accuracy of the algorithm, one may combine OUM and our method. In detail, one initially uses OUM until 5% of all grid nodes are accepted. Once 5% of all grid nodes are accepted, one switches to our algorithm with Acc consisting of the accepted nodes.

In step 4 of our algorithm, we use (25) in Sect. 3. There, we have to find the intersection point of $NF(x)$ and $l_a$ without SAAF. Suppose that $l_a$ intersects the line segment $x_{j0}x_{k0} \in NF(x)$. Then, by the definition of $AF$ and $NF(x)$, $x_{j0}x_{k0}$ is the line segment with the shortest distance to $x$ among the line segments $x_jx_k (x_j, x_k \in Acc, x_j \in N(x_k))$ intersecting with $l_a$. Let $a = (a_1, a_2)$ and without loss of generality, suppose that $a_1, a_2 > 0$. The half-line $l_a$ intersects with one of the two types of line segments (joining two neighboring nodes), i.e., $(x_k, y_l)(x_k, y_{l+1})$ or $(x_k, y_l)(x_{k+1}, y).$ Suppose that $l_a$ intersects with line segment $(x_k, y_l)(x_k, y_{l+1})$ at $(x_k, y'_l)$, where $y_l < y'_l < y_{l+1}$. By the definition of $l_a$, we get \[ \frac{x_k - x_i}{a_1} = \frac{y'_l - y_l}{a_2}. \] Since $x_k - x_i = n \Delta x$, we obtain $x_k = x_i + n \Delta x, y'_l = y_l + \frac{a_2}{a_1} n \Delta x$. Therefore $dist(x, (x_k, y'_l)) = \sqrt{a_1^2 + a_2^2 \frac{n \Delta x}{a_1} + \frac{a_1 y'_l - x_i}{a_2}} \leq \gamma$. Similarly, we can get similar result in the case of that $l_a$ intersects with line segment $(x_k, y_l)(x_{k+1}, y).$ From the above results, we can find the line segment $x_{j0}x_{k0}$ as follows: While increasing $n$ from 1 to $\gamma$, stop increasing if $x_n, y_n \in Acc$, where $x_n = \binom{x_i + n \Delta x, y_j + \frac{a_2}{a_1} n \Delta x}{1}$, $y_n = \binom{x_i + n \Delta x, y_j + \frac{a_2}{a_1} n \Delta x}{1}$. Suppose that the above increasing stops at $k(1 \leq k \leq \gamma)$. Also, while increasing $n$ from 1 to $\gamma$, stop increasing if $x'_n, y'_n \in Acc$, where $x'_n = \binom{x_i + \frac{a_1}{a_2} \Delta x, y_j + n \Delta x}{1}$, $y'_n = \binom{x_i + \frac{a_1}{a_2} \Delta x, y_j + n \Delta x}{1}$. Suppose that the above increasing stops at $k'(1 \leq k' \leq \gamma)$. Then the line segment with the shorter distance to $x$ among $x'_{k'}, y'_{k'}$ and $x_ky_k$ is just $x_{j0}x_{k0}$.

### 4.3 Computational Complexity

Once the grid point $x \in \Omega$ belongs to Acc, no computation is needed at the point $x$. For every grid point $x \in \Omega$, the computation at $x$ is performed only when one of its neighboring) grid point is accepted as alive point. A computation at $x$ involves only finding the intersection of a given half line with $NF(x)$. ($O(2\gamma)$ complexity as discussed above, here $\gamma$ is the anisotropy coefficient.) On the other hand, for every grid point, there are at most eight neighboring grid points. Hence, the computational complexity of our method is $O(M \log M + \gamma M)$, where $M$...
is the number of grid points and the factor of log \( M \) reflects the necessity to maintain a sorted list of the values of \( U \) at the grid points belonging to Cons for step 5 of our algorithm.

5 Numerical Examples

In this section, we test our method with the following Equations [9, 15 and 16]: where \( m_\lambda, \mu(a) = (1 + (\lambda a_1 + \mu a_2)^2)^{-\frac{1}{2}} \) for \( \lambda, \mu \in R \) and for \( c_1, c_2, c_3, c_4 > 0 \),

| Equation  | Dynamics                                                                 |
|----------|---------------------------------------------------------------------------|
| HJB-1    | \( f(x, y, a) = m_\lambda, \mu(a)a \)                                   |
| HJB-2    | \( f(x, y, a) = 1/\sqrt{1 + (\nabla g(x, y) \cdot a)^2} \)                |
| HJB-3    | \( f(x, y, a) = F_2(x, y)m_{p1(x,y)q1(x,y)}(a)a \)                       |
| HJB-4    | \( f(x, y, a) = F_4(x, y)m_{p2(x,y)q2(x,y)}(a)a \)                       |

\[
C(x) = c_1 \sin \left( \frac{c_2 \pi x}{c_3} + c_4 \right), \quad c_1, \ c_2, \ c_3, \ c_4 > 0, \quad (F_1(x, y), \ F_2(x, y)) = \\
\begin{cases} 
(0.5, \ 1) & y \geq C(x) \\
(2, \ 3) & y < C(x)
\end{cases}
\]

\[
(F_3(x, y), \ F_4(x, y)) = \begin{cases} 
(0.2, \ 0.8) & y > C(x) + 0.25 \text{ or } y < C(x) - 0.25 \\
(1, \ 3) & C(x) < y \leq C(x) + 0.25 \\
(1, \ 1) & C(x) - 0.25 < y \leq C(x)
\end{cases}
\]

\[
M_1(x, y) = \sqrt{\frac{F_2^2(x, y)}{1 + C'^2(x)}}, \quad M_2(x, y) = \sqrt{\frac{F_4^2(x, y)}{1 + C'^2(x)}}, \quad q_1(x, y) = -M_1(x, y), \quad p_1(x, y) = M_1(x, y)C'(x)
\]

5.1 Convergence Test

Test 1 Here we solve equation HJB-1. For this problem, the speed function \( f(x, y, a) \) defined as

\[
f(x, y, a) = 1/\sqrt{1 + (\lambda a_1 + \mu a_2)^2}
\]

and corresponding HJB is

\[
\min_{a \in S_1} \left\{ \frac{\nabla u(x, y) \cdot a}{\sqrt{1 + (\lambda a_1 + \mu a_2)^2}} \right\} = -1.
\]

In this case, the exact solution of HJB-1 is given by.

\[
u(x, y) = \sqrt{1 + \lambda^2}x^2 + (1 + \mu^2)y^2 - 2\lambda \mu xy. \ [16].
\]
We set $\lambda = 5, \mu = -10$. In this case, the anisotropy efficient for this problem is $\gamma = \frac{F_2}{F_1} \approx 8.775097$, where $F_1 = 0.089310, \ F_2 = 0.783704$. We denote the exact solution of HJB-1 as $u_0$. We compare our numerical solution and OUM [16] in this example on various grid sizes. In Fig. 1, we show the contour lines of $u_0$ and of $u$ obtained using our method. In Table 3, we present the $L^1$, $L^2$ and $L^\infty$ norms of the error between the exact solution and the approximated solution obtained using our method and OUM [16]. We also present the CPU times. We also added the order of convergence with the $L^\infty$ norm.

In Table 3, $E_1 = \int_\Omega |u - u_0| \, dx$, $E_2 = \left( \int_\Omega |u - u_0|^2 \right)^{1/2}$, $E_\infty = \max_\Omega |u - u_0|$.

We have observed that our method captures viscosity solution of HJB-1 efficiently and accurately from Fig. 1 and Table 3.

From Table 3, we observed that our method is more efficient that OUM. The CPU time of our method is much faster than OUM. Although, the numerical error of our method is slightly bigger than OUM on the same grid size, the numerical error of our method on 200 × 200 grid size is smaller than the numerical error of OUM on 100 × 100 grid size. Furthermore, the CPU time of our method on 200 × 200 grid size is also much faster than CPU time of OUM on 100 × 100 grid size. In total, the computational cost for SAAF is much more than the computational cost for finding intersection points between lines $l_\hat{a}$ and $NF(x)$.

![Fig. 1](HJB-1,N_x = N_y = 400)

Contour plots of $u_0$(colored with red) and $u$(colored with blue) obtained using our method (Color figure online)

| $N_x$ = (N_y) | OUM [16] | Our method |
|---------------|-----------|-------------|
| 101           | 0.018380  | 0.001095    | 0.001563 | 2.736 | – |
| 201           | 0.01490   | 0.000490    | 0.001061 | 17.443 | 0.3028 |
| 401           | 0.010054  | 0.000348    | 0.000813 | 143.168 | 0.5675 |
| $N_x$ = (N_y) | OUM [16] | Our method |
| 101           | 0.026509  | 0.001085    | 0.002214 | 0.411 | – |
| 201           | 0.017757  | 0.000629    | 0.001380 | 1.757 | 0.57809 |
| 401           | 0.012297  | 0.000420    | 0.000976 | 6.847 | 0.53008 |
5.2 Efficiency Test

We also test our method with HJB-2 ~ HJB-4 to show the efficiency of our method. For these equations, solution computed by OUM on a $801 \times 801$ grid will be referred to as the “exact” solution and will be denoted by $u^{\text{exact}}$.

Test 2 Here we solve equation HJB-2. For this problem, the speed function $f(x, y, a)$ defined as

$$f(x, y, a) = \frac{1}{\sqrt{1 + (g_x a_1 + g_y a_2)^2}}$$

and corresponding HJB is

$$\min_{a \in S_1} \left\{ \frac{\nabla u(x, y) \cdot a}{\sqrt{1 + (g_x a_1 + g_y a_2)^2}} \right\} = -1.$$  

On a manifold $z = g(x, y)$, the manifold’s geodesic distance function $u$ is the viscosity solution of the above equation. In this case, we consider the manifold $g(x, y) = 0.9 \sin(2\pi x) \sin(2\pi y)$ as in [16]. The anisotropy efficient for this problem is $\gamma = \frac{F_2}{F_1} \approx 5.7$, where $F_1 = 0.174137$, $F_2 = 1$.

In Fig. 2 and Table 4, we show the numerical result of our method for this equation.

Test 3 Here we solve equation HJB-3. For this problem the speed function $f(x, y, a)$ defined as

$$f(x, y, a) = \frac{F_2(x, y)}{\sqrt{1 + (p(x, y) a_1 + q(x, y) a_2)^2}}.$$  

Fig. 2 (HJB-2, $N_x = N_y = 400$)
Contour plots of $u$(colored with blue) obtained using our method. (Color figure online)

Table 4 (HJB-2) The CPU time, error and the order of convergence between the “exact solution” $u^{\text{exact}}$ and approximate solution obtained using our method

| $N_x = (N_y)$ | $E_\infty$ | $E_1$ | $E_2$ | CPU time(s) | Order of convergence |
|---------------|------------|-------|-------|-------------|----------------------|
| 100           | 0.074309   | 0.013985 | 0.021332 | 0.575       | –                    |
| 200           | 0.016033   | 0.003654 | 0.004721 | 2.200       | 2.2125               |
| 400           | 0.003349   | 0.001302 | 0.001571 | 8.852       | 2.2592               |
where
\[
(F_1(x, y), F_2(x, y)) = \begin{cases} 
(0.5, 1) & y \geq C(x) \\
(2, 3) & y \leq C(x)
\end{cases}
\]
and
\[
M_1(x, y) = \sqrt{\frac{F_2^2(x, y)}{F_1^2(x, y) + C'(x)}},
q_1(x, y) = -M_1(x, y),
p_1(x, y) = M_1(x, y)C'(x).
\]

The corresponding HJB is
\[
\min_{a \in S_1} \left\{ \frac{F_2(x, y) \cdot a}{\sqrt{1 + (p_1(x, y)a_1 + q_1(x, y)a_2)}} \right\} = -1.
\]

The anisotropy efficient for this problem is \( \gamma = \frac{F_2}{F_1} = \frac{3}{0.5} = 6 \), where \( F_2 = 3 \), \( F_1 = 0.5 \).

In Fig. 3 and Table 5, we show the numerical result of our method for this equation.

In Fig. 3, we also draw the boundaries between each region on the figure, and to include the maximum and minimum velocity in each region.

**Test 4** Here we solve equation HJB-4. For this problem the speed function \( f(x, y, a) \) defined as
\[
f(x, y, a) = \frac{F_4(x, y)}{\sqrt{1 + (p(x, y)a_1 + q(x, y)a_2)}}.
\]

**Fig. 3 (HJB-4, \( N_x = N_y = 400 \))**
Contour plots of \( u \) (colored with blue) obtained using our method

(colors figure online)

**Table 5 (HJB-3)** The CPU time, error and the order of convergence between the “exact solution” \( u^{exact} \) and approximate solution obtained our method

| \( N_x = (N_y) \) | \( E_{\infty} \) | \( E_1 \) | \( E_2 \) | CPU time(s) | Order of convergence |
|------------------|--------|--------|--------|-------------|---------------------|
| 100              | 0.023268 | 0.004361 | 0.006848 | 0.509       | –                   |
| 200              | 0.011557 | 0.001968 | 0.003044 | 2.053       | 1.0095              |
| 400              | 0.004756 | 0.000614 | 0.000988 | 8.309       | 1.2809              |

(Springer)
In Fig. 4, we also draw the boundaries between each region on the figure, and to include the maximum and minimum velocity in each region.

5.3 A Three-Dimensional Example

In this subsection, we consider a three-dimensional example. The speed function \( f(x, a) \) defined as

\[
f(x, a) = v(x) \cdot a + \sqrt{F^2(x) - (v^2(x) - (v(x) \cdot a)^2)}
\]

where

\[
(F_3(x, y), F_4(x, y)) = \begin{cases} 
(0.2, 0.8) & y > C(x) + 0.25 \text{ or } y \leq C(x) - 0.25 \\
(1, 3) & C(x) < y \leq C(x) + 0.25 \\
(1, 1) & C(x) - 0.25 < y \leq C(x)
\end{cases}
\]

and \( M_2(x, y) = \sqrt{\frac{F_2^2(x, y)}{F_1^2(x, y) + C'(x)^2}}, \quad q_2(x, y) = -M_2(x, y), \quad p_2(x, y) = M_2(x, y)C'(x) \).

The corresponding HJB is

\[
\min_{a \in S_1} \left\{ \frac{F_4(x, y) \cdot a}{\sqrt{F_2^2(x, y)}} \right\} = -1.
\]

The anisotropy efficient for this problem is \( \gamma = \frac{F_2}{F_1} = \frac{3}{0.7} = 15 \), where \( F_2 = 3, \quad F_1 = 0.2 \).

In Fig. 4 and Table 6, we show the numerical result of our method for this equation.

In Fig. 4, we also draw the boundaries between each region on the figure, and to include the maximum and minimum velocity in each region.

### Table 6 (HJB-4)

The CPU time, error and the order of convergence between the “exact solution” \( u^{\text{exact}} \) and approximate solution obtained using our method.

| \( N_x = (N_y) \) | \( E_\infty \) | \( E_1 \) | \( E_2 \) | CPU time(s) | Order of convergence |
|------------------|-------|-------|-------|----------|---------------------|
| 100              | 0.051 | 0.018 | 0.016 | 0.525    | –                   |
| 200              | 0.026 | 0.005 | 0.007 | 2.085    | 0.9913              |
| 400              | 0.013 | 0.002 | 0.003 | 8.505    | 0.9814              |

In this table, we show the numerical result of our method for this problem.

\[
\text{Table 6 (HJB-4): The CPU time, error and the order of convergence between the “exact solution” } u^{\text{exact}} \text{ and approximate solution obtained using our method.}
\]
where \( x = (x_1, x_2, x_3) \), \( a = (a_1, a_2, a_3) \) and \( v(x) = (v_1(x), v_2(x), v_3(x)) \). The corresponding HJB is

\[
\min_{a \in S_1} \left\{ (v(x) \cdot a + \sqrt{F^2(x) - (v^2(x) - (v(x) \cdot a)^2)} \cdot a \right\} = -1.
\]

We solve this example by using our method in the case of

\[ F(x) = 2.5 - 0.4x_2 - 0.8 \exp[-20(x_1 - 0.25)^2 - 10(x_2 - 0.3)^2 - 20(x_3 - 0.3)^2] \]

and \( v(x) = (0.4, 0.8, 1.2) \) [33]. We set the target set \( \Gamma = \{(0.4, 0.6, 0.5), (0.6, 0.4, 0.5)\} \) and \( \Omega = [0, 1]^3 \).

The anisotropic coefficient of this problem is \( \gamma = \frac{f_2}{f_1} = 50.902797 \), where \( f_2 = 4 \), \( f_1 = 0.078516 \).

Figure 5 shows the solution to the anisotropic eikonal equation computed on a mesh of \( 151 \times 151 \times 151 \) using our Method. The contour lines of the solution on the cross section \( z = 0.5 \) are shown in Figure (b).

Throughout the above example, we confirmed that our method captures the viscosity solution accurately in the case of 3D problems. In the three-dimensional example, \( AF \) and \( NF(x) \) are not a set of line segments but a set of plane segments. In 2D examples, the line segment is made of two neighboring grid nodes.

However, in 3D examples, one must use a plane piece consisting of four neighboring grid nodes (e.g., \( x_i y_j z_k, x_{i-1} y_j z_k, x_i y_{j-1} z_k, x_{i-1} y_{j-1} z_k \)) (a rectangular plane piece) or a plane piece consisting of three diagonal neighboring grid nodes (e.g., \( x_i y_j z_k, x_{i-1} y_{j-1} z_k, x_i y_{j-1} z_{k+1} \)).

### 6 Conclusion

We have proposed a new single-pass method for approximating the solution to an anisotropic eikonal equation related to the anisotropic min-time optimal trajectory problem. We have tested the new method with several anisotropic eikonal equations and have observed that it works well while significantly reducing the computational cost. The method presented in this
paper is applicable for the anisotropic eikonal Eqs. (3) and (4). We are currently studying on extending our technique to a wider class of problems including the generalized eikonal equation in a moving medium. We believe that our technique will be applicable to more general Hamilton–Jacobi-Bellman equations, which are associated to min-time problems.

Acknowledgements The authors would like to thank the referees for their valuable comments and suggestions, which helped the authors to improve the article significantly.

Funding This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data availability Enquiries about data availability should be directed to the authors.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Appendix

Lemma 1 The viscosity solution $u(x)$ of (6) is semi-concave.

Proof Let us prove that $H(x, p)$ is Lipchitz continuous in $x$.

For arbitrary $x, y$, suppose that.

$$H(x, p) = -\min_{a \in S_1} \{ p \cdot af(x, a) \} - 1 = -p \cdot a_0 f(x, a_0) - 1,$$

$$H(y, p) = -\min_{a \in S_1} \{ p \cdot af(y, a) \} - 1 = -p \cdot a_1 f(y, a_1) - 1.$$ 

If $p \cdot a_0 f(x, a_0) \geq p \cdot a_1 f(y, a_1)$, then.

$$|H(x, p) - H(y, p)| = |p \cdot a_0 f(x, a_0) - p \cdot a_1 f(y, a_1)|$$

$$\leq |p \cdot a_0 f(y, a_0) - p \cdot a_0 f(x, a_0)| \leq p \|L\| \|x - y\|$$

If $p \cdot a_0 f(x, a_0) < p \cdot a_1 f(y, a_1)$, then.

$$|H(x, p) - H(y, p)| = |p \cdot a_0 f(x, a_0) - p \cdot a_1 f(y, a_1)|$$

$$\leq |p \cdot a_1 f(x, a_1) - p \cdot a_1 f(y, a_1)| \leq p \|L\| \|x - y\|$$

Therefore $H(x, p)$ is Lipchitz continuous in $x$. We also know that $H(x, p)$ is convex in the second argument and thus $u$ is semi-concave in $\Omega$. ([31]) \qed

Lemma 2 Assume that $u$ is differentiable at $x \in \Omega$ and suppose that $\alpha(\cdot)$ is an optimal control for $x$ and $y(\cdot)$ is the optimal trajectory for $x$. Then $\nabla u(y(\cdot))$ is continuous at $[0, +\infty)$.

Proof For arbitrary $t > 0$, $u$ is differentiable at $y(t)$ because characteristics never emanate from the shocks-non differentiable point. ([16]).

If $\nabla u$ is defined at $x \in \Omega \setminus \partial \Omega$, there exists constant $L$ such that $\|\nabla u\| \leq L$. [16].

We fix any $t \geq 0$. For arbitrary sequence $t_n \to t (t_n \geq 0)$, $y(t_n) \to y(t)$. The $\nabla u(y(t_n))$ is bounded. Therefore, it has convergent subsequence. Since $u$ is semi-concave, limits of all convergent subsequences are the same. i.e., $\nabla u(y(t_n))$. [36] Therefore $\nabla u(y(t_n)) \to \nabla u(y(t))$ and thus $\nabla u(y(\cdot))$ is continuous at $[0, +\infty)$. \qed
Lemma 3 Consider a grid node $x \in X$ and assume that $u$ is differentiable at $x$. Suppose that the characteristic for $x$ intersects the line segment $x_1x_2$ at $x_0$, where $x_1, x_2 \in ND(x), x_1 \in N(x_2)$. Then.

$$\|\nabla u(x_0) - \nabla u(x)\| = O(h).$$

Proof Suppose that $\alpha(\cdot)$ is an optimal control for $x$ and $y(\cdot)$ is the optimal trajectory for $x$. If $y(t_0) = x_0$, since $\nabla u(y(\cdot))$ is continuous at 0, we get.

$$\|\nabla u(y(t_0)) - \nabla u(y(0))\| = O(t_0).$$

By the Bellman’s optimality principle, we get $t_0 \leq \frac{x_0 - x_1}{F_2}$ and $\frac{x_0 - x_1}{F_1} \leq t_0$.

Therefore $O(t_0) = O(h)$. From the fact that $y(0) = x, y(t_0) = x_0$, we conclude that.

$$\|\nabla u(x_0) - \nabla u(x)\| = O(h).$$

□

Lemma 4 If $xx_1x_2$ is a sufficiently small simplex, which contains the characteristic for $x$, then.

$$\min(u(x_1), u(x_2)) < u(x).$$

Proof By the Bellman’s optimality principle, we get.

$$u(x) = u(x_1) + \frac{\|x_1 - x\|}{f(x, (x_1 - x)/\|x_1 - x\|)} + O(h^2) = \lambda u(x_1) + (1 - \lambda)u(x_2) + \frac{\|x_1 - x\|}{f(x, (x_1 - x)/\|x_1 - x\|)} + O(h^2).$$

Since $xx_1x_2$ is sufficiently small simplex, we obtain that $u(x) > \lambda u(x_1) + (1 - \lambda)u(x_2)$. Therefore, $\min(u(x_1), u(x_2)) < u(x)$. □

References

1. Osher, S., Fedkiw, R.: Level Set Methods and Dynamic Implicit Surfaces. Springer-Verlag, New York Inc (2003)
2. Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. Proc. Natl. Acad. Sci. 93, 1591–1595 (1996)
3. Sethian, J. A.: Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science, 2nd ed. In: Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press. (3) (1999)
4. Cristiani, E.: A fast marching method for Hamilton-Jacobi equation modeling monotone front propagations. J. Sci. Comput. 39(2), 189–205 (2009)
5. Chow, Y.T., Darbon, J., Osher, S., Yin, W.: Algorithm for overcoming the curse of dimensionality for time-dependent non-convex Hamilton-jacobi equations arising from optimal control and differential games problems. J. Sci. Comput. 73, 617–643 (2017)
6. Lelievre, P.G., Farquharson, C.G., Hurich, C.A.: Computing first arrival seismic travel times on unstructured 3-D tetrahedral grids using the fast marching method. Geophys. J. Int. 184(2), 885–896 (2011)
7. Tozza, S., Falcone, M.: Analysis and approximation of some shape-from-shading models for non-lambertian surfaces. J. Math. Imaging Vis. 55(2), 153–178 (2016)
8. Peyré, G., Péchaud, M., Keriven, R., Cohen, L.D: Geodesic methods in computer vision and graphics, foundations and trends in computer graphics and vision, 5(3–4), 197–397 (2010)
9. Cacace, S., Cristiani, E., Falcone, M.: Can local single-pass methods solve any stationary Hamilton–Jacobi–Bellman equation? SIAM J. Sci. Comput. 36, 570–587 (2014)
10. Falcone, M., Ferretti, R.: Semi-Lagrangian schemes for Hamilton-Jacobi equations, discrete representation formulae and Godunov methods. J. Comput. Phys. 175, 559–575 (2002)
11. Falcone, M., Ferretti, R.: Discrete time high-order schemes for viscosity solutions of Hamilton-Jacobi-Bellman equations. Numer. Math. 67, 315–344 (1994)
12. Cristiani, E., Falcone, M.: Fast semi-Lagrangian schemes for the Eikonal equation and applications. SIAM J. Numer. Anal. 45(5), 1979–2011 (2007)
13. Tsitsiklis, J.N.: Efficient algorithms for globally optimal trajectories. IEEE Trans. Automat. Control. 40, 1528–1538 (1995)
14. Sethian, J.A.: Fast marching methods. SIAM Rev. 41(2), 199–235 (1999)
15. Cacace, S., Cristiani, E., Falcone, M.: A local ordered upwind method for hamilton-jacobi and isaacs equations. In: Proceedings of 18th IFAC World Congress 2011, pp. 6800–6805
16. Sethian, J.A., Vladimirsky, A.: Ordered upwind methods for static Hamilton-Jacobi equations, theory and algorithms. SIAM J. Numer. Anal. 41(1), 323–363 (2003)
17. Sethian, J.A., Vladimirsky, A.: Ordered upwind methods for static Hamilton-Jacobi equations. Proc. Natl. Acad. Sci. 98(20), 11069–11074 (2001)
18. Shum, A., Morris, K., Khajepour, A.: Convergence Rate for the Ordered Upwind Method. J. Sci. Comput. 68(3), 889–913 (2016)
19. Alton, K.: Dijkstra-like Ordered Upwind Methods for Solving Static Hamilton-Jacobi Equations, Ph.D. thesis, University of British Columbia, Vancouver, 2010.
20. Alton, K., Mitchell, I.M.: Fast marching methods for stationary Hamilton-Jacobi equations with axis-aligned anisotropy. SIAM J. Numer. Anal. 43, 363–385 (2008)
21. Alton, K., Mitchell, I.M.: An ordered upwind method with pre-computed stencil and monotone node acceptance for solving static convex Hamilton-Jacobi equations. J. Sci. Comput. 51, 313–348 (2012)
22. Kao, C.Y., Osher, S., Tsai, Y.: Fast sweeping methods for static Hamilton-Jacobi equations. SIAM J. Numer. Anal. 42(2), 2004–2005 (2005)
23. Kao, C.Y., Osher, S., Qian, J.: Lax-Friedrichs sweeping scheme for static Hamilton-Jacobi equations. J. Comput. Phys. 196, 436–439 (2004)
24. Tsai, Y., Cheng, L., Osher, S., Zhao, H.: Fast sweeping algorithms for a class of Hamilton-Jacobi equations. SIAM J. Numer. Anal. 41, 673–694 (2004)
25. Qian, J., Zhang, Y.T., Zhao, H.: A fast sweeping method for static convex Hamilton-Jacobi equations. J. Sci. Comput. 31, 237–271 (2007)
26. Zhao, H.: A fast sweeping method for eikonal equations. Math. Comp. 74, 603–627 (2005)
27. Qian, J., Zhang, Y., Zhao, H.: Fast sweeping methods for Eikonal equations on triangulated meshes. SIAM J. Numer. Anal. 45, 83–107 (2007)
28. Jeong, Fu.Z., Pan, W.K., Kirby, Y., Whitaker, R.M.: A fast iterative method for solving the eikonal equation on triangulated surfaces. SIAM J. Sci. Comput. 33, A2468–A2488 (2011)
29. Jeong, W.K., Whitaker, R.T.: A fast iterative method for eikonal equations. SIAM J. Sci. Comput. 30, 2512–2534 (2008)
30. Jeong, W.K., Whitaker, R. T.: A fast iterative method for a class of Hamilton–Jacobi equations on parallel systems. University of Utah, Technical Report UUCS-07-010 (2007)
31. Dahiya, D., Baskar, S.: Characteristic fast marching method on triangular grids for the generalized eikonal equation in moving media. Wave Motion 59, 81–93 (2015)
32. Dahiya, D., Baskar, S., Coulouvrat, F.: Characteristic fast marching method for monotonically propagating fronts in a moving medium. SIAM J. Sci. Comput. 35(4), A1880–A1902 (2013)
33. Glowinski, R., Leung, S., Qian, J.: Operator-splitting based fast sweeping methods for isotropic wave propagation in a moving fluid. SIAM J. Sci. Comput. 38(2), A1195–A1223 (2016)
34. Glowinski, R., Liu, H., Leung, S., Qian, J.: A finite element/operator-splitting method for the numerical solution of the two dimensional elliptic Monge–Ampère equation, 79(1), 1–47 (2019)
35. Ho, M., Ri, J., Kim, S.: Improved characteristic fast marching method for the generalized eikonal equation in a moving medium. J. Sci. Comput. 81(3), 2484–2502 (2019)
36. Bardi, M., Capuzzo-Dolcetta, I.: Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations. Birkhauser, Boston (1997)
37. Sinestrari, C.: Semiconcavity of solutions of stationary Hamilton-Jacobi equations. Nonlinear. Anal. 24(9), 1321–1326 (1995)
38. Vladimirsky, A.: Fast Methods for Static Hamilton-Jacobi Partial Differential Equations, Ph.D. thesis, University of California, Berkeley, 2001.

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.