Fast Sweeping Methods for Factored Anisotropic Eikonal Equations: Multiplicative and Additive Factors

Songting Luo & Jianliang Qian

Journal of Scientific Computing

ISSN 0885-7474
Volume 52
Number 2

J Sci Comput (2012) 52:360-382
DOI 10.1007/s10915-011-9550-y

Springer

J Sci Comput (2012) 52:360–382
DOI 10.1007/s10915-011-9550-y

Fast Sweeping Methods for Factored Anisotropic Eikonal Equations: Multiplicative and Additive Factors

Songting Luo · Jianliang Qian

Received: 11 July 2011 / Revised: 26 August 2011 / Accepted: 17 October 2011 /
Published online: 29 October 2011

Abstract The viscosity solution of static Hamilton-Jacobi equations with a point-source condition has an upwind singularity at the source, which makes all formally high-order finite-difference scheme exhibit first-order convergence and relatively large errors. To obtain designed high-order accuracy, one needs to treat this source singularity during computation. In this paper, we apply the factorization idea to numerically compute viscosity solutions of anisotropic eikonal equations with a point-source condition. The idea is to factor the unknown traveltime function into two functions, either additively or multiplicatively. One of these two functions is specified to capture the source singularity so that the other function is differentiable in a neighborhood of the source. Then we design monotone fast sweeping schemes to solve the resulting factored anisotropic eikonal equation. Numerical examples show that the resulting monotone schemes indeed yield clean first-order convergence rather than polluted first-order convergence and both factorizations are able to treat the source singularity successfully.

1 Introduction

We consider the following anisotropic eikonal equation with a point-source condition,

$$\begin{cases} H(\mathbf{x}, \nabla T(\mathbf{x})) = \sqrt{\nabla T(\mathbf{x}) M(\mathbf{x}) \nabla T(\mathbf{x})^t} = 1, \quad \mathbf{x} \in \Omega \backslash \{\mathbf{x}_0\}, \\ T(\mathbf{x}_0) = 0, \end{cases} \tag{1}$$

where $\Omega \subset \mathbb{R}^N$ is a bounded open set, $\nabla T(\mathbf{x}) = \nabla T(x_1, \ldots, x_N) = (\frac{\partial T(\mathbf{X})}{\partial x_1}, \ldots, \frac{\partial T(\mathbf{X})}{\partial x_N})$, and $M(\cdot)$ is a symmetric positive definite matrix modeling the anisotropy. Such nonlinear first-order hyperbolic partial differential equations (PDE) arise in many applications such as classical mechanics, geosciences, geometrical optics, computer vision and optimal control. If

S. Luo · J. Qian (✉)
Department of Mathematics, Michigan State University, East Lansing, MI 48824, USA
e-mail: qian@math.msu.edu

S. Luo
e-mail: luos@math.msu.edu

$M(\mathbf{x}) = \frac{1}{S^2(\mathbf{x})} I$, then it reduces to the isotropic eikonal equation with a point-source condition,

$$\begin{cases} |\nabla T(\mathbf{x})| = S(\mathbf{x}), \quad \mathbf{x} \in \Omega \backslash \{\mathbf{x}_0\}, \\ T(\mathbf{x}_0) = 0, \end{cases} \tag{2}$$

where $S(\mathbf{x})$ is the slowness field.

Fast sweeping methods are a family of efficient methods for solving static Hamilton-Jacobi equations [3, 6–8, 10, 18, 19, 21, 24–26], and some essential ideas of these methods may trace back to [2, 20]. In [25] the fast sweeping method was systematically analyzed for eikonal equations. Since then the fast sweeping methods have undergone an intensive development for general static Hamilton-Jacobi equations in [3, 6–8, 10, 18, 19, 21, 26] and have found many different applications; see [9, 13] for examples.

It is well-known that the viscosity solution of (1) has an upwind source singularity. Formally, all finite-difference solvers, even high-order ones, can only exhibit at most polluted first-order convergence, since the initial error at the source can spread out to the whole space. Especially, first-order schemes such as the fast sweeping method [11, 18, 19, 25] and other methods [5, 22] can only have "polluted" accuracy of $O(|h \log h|)$ on an underlying mesh with grid size h. To validate clean first-order convergence rates, most of the published works [3, 6–8, 10, 15, 17–19, 21, 25, 26] have to rely on fixing a neighborhood of the source so that one can carry out mesh refinement in other regions to quantify error behaviors; this approach is ad hoc and is restricted in the sense that one has to assume that the related functions, such as slowness field, are constant in that fixed neighborhood. Another approach to treat this upwind singularity is adaptive gridding near the source as proposed and implemented in [16]. So the question is that: can we have a first-order scheme with clean first-order convergence without ad hoc special treatment near the source? This issue has been resolved for the isotropic eikonal equation (2) by using a factorization approach [4, 14, 23]. The unknown function is factored into two multiplicative factors. One of these two functions is specified to capture the source singularity so that the other function is differentiable in a neighborhood of the source. In [4] a fast sweeping scheme has been designed to compute the underlying function which satisfies a factored eikonal equation, and both convergence order and error behaviors have been improved. Since the underlying function is smooth in a neighborhood of the source, high-order schemes can be designed with ease to compute this function without ad hoc special treatment near the source; see [12] for such high-order sweeping schemes based on the Lax-Friedrichs Hamiltonian.

In this paper, we extend this factorization idea to compute the viscosity solution of the anisotropic eikonal equation (1). We present two approaches. One approach is to factor the unknown function into two multiplicative factors as in [4]; the other is to factor the unknown function into two additive factors. One of the two factors is specified analytically to capture the source singularity, so that the other function is differentiable in a neighborhood of the source. Then we design two first-order fast sweeping schemes to compute the underlying functions. Numerical examples show that both first-order schemes yield desired first-order convergence rate.

This paper is organized as follows. In Sect. 2, we derive factored anisotropic eikonal equations for both multiplicative and additive factorizations. In Sect. 3, we present fast sweeping methods for factored anisotropic eikonal equations. In Sect. 4, we present computational examples to illustrate our methods. Conclusions are given at the end.

2 Factored Anisotropic Eikonal Equation

In this section, we derive factored anisotropic eikonal equations for both multiplicative and additive factorizations. We present our derivations in 2-D only as the extension to higher dimension is straightforward. The anisotropic eikonal equation with a point-source condition in the 2-D case has the following form,

$$\begin{cases} \sqrt{a(\mathbf{x})T_x^2 - 2c(\mathbf{x})T_xT_y + b(\mathbf{x})T_y^2} = 1, \quad \mathbf{x} \in \Omega\backslash\{\mathbf{x}_0\}, \\ T(\mathbf{x}_0) = 0, \end{cases} \tag{3}$$

with $M(\mathbf{x}) = \begin{pmatrix} a(\mathbf{X}) & -c(\mathbf{X}) \\ -c(\mathbf{X}) & b(\mathbf{X}) \end{pmatrix}$, $\mathbf{x} = (x, y)$, and $\mathbf{x}_0 = (x_0, y_0)$.

2.1 Multiplicative Factors

We factorize the solution of the anisotropic eikonal equation (3) into multiplicative factors [4]. We assume that T can be written in the form of two multiplicative functions,

$$T = T_0\tau,$$

with T_0 a known function and τ the underlying function. Then we have

$$\nabla T = \nabla T_0\tau + T_0\nabla\tau.$$

By substituting the above into the anisotropic eikonal equation (3), we have the factored anisotropic equation,

$$\sqrt{\begin{aligned} &\tau^2\left(a{T_{0x}}^2 - 2cT_{0x}T_{0y} + b{T_{0y}}^2\right) + 2T_0\tau\left(aT_{0x}\tau_x - c(T_{0x}\tau_y + T_{0y}\tau_x)\right. \\ &\left. + bT_{0y}\tau_y\right) + T_0^2\left(a\tau_x^2 - 2c\tau_x\tau_y + b\tau_y^2\right) \end{aligned}} = 1. \tag{4}$$

By choosing an appropriate T_0 to capture the source singularity, the function τ is smooth near the source. Then τ can be computed accurately so that it can be used to recover accurate T, which will gain some improvement of the accuracy on T compared to the solutions obtained by solving (3) directly.

We choose T_0 as the viscosity solution of the following anisotropic eikonal equation,

$$\begin{cases} \sqrt{a_0{T_{0x}}^2 - 2c_0T_{0x}T_{0y} + b_0{T_{0y}}^2} = 1, \quad x \in \Omega\backslash\{(x_0, y_0)\}, \\ T_0(x_0, y_0) = 0, \end{cases} \tag{5}$$

with $a_0 = a(x_0, y_0)$, $b_0 = b(x_0, y_0)$ and $c_0 = c(x_0, y_0)$. Thus T_0 is chosen to be

$$T_0(x, y) = \sqrt{\frac{b_0(x - x_0)^2 + 2c_0(x - x_0)(y - y_0) + a_0(y - y_0)^2}{a_0b_0 - c_0^2}},$$

which captures the source singularity. Furthermore, we have

$$\lim_{(x,y)\to(x_0,y_0)} \frac{T(x, y)}{T_0(x, y)} = 1 \tag{6}$$

so that

$$\lim_{(x,y)\to(x_0,y_0)} \tau(x, y) = 1. \tag{7}$$

2.2 Additive Factors

We may also decompose T into two additive functions,

$$T = T_0 + \tau;$$

then we have

$$\nabla T = \nabla T_0 + \nabla \tau.$$

By substituting it into the anisotropic eikonal equation (3), we have the factored anisotropic eikonal equation,

$$\sqrt{\begin{array}{l} a\tau_x^2 - 2c\tau_x\tau_y + b\tau_y^2 + 2\tau_x(aT_{0x} - cT_{0y}) + 2\tau_y(bT_{0y} - cT_{0x}) \\ + a{T_{0x}}^2 - 2cT_{0x}T_{0y} + b{T_{0y}}^2 \end{array}} = 1. \tag{8}$$

We choose T_0 as the viscosity solution of (5) so that it captures the source singularity. Then the following holds:

$$\lim_{(x,y)\to(x_0,y_0)} \frac{\tau(x,y)}{T_0(x,y)} = 0. \tag{9}$$

2.3 Factorization: Multiplicative Versus Additive

Among the two factorizations, which one is preferred? To gain some insights, we carry out some preliminary analysis for the 1-D case. Without loss of generality, assume that the source $x_0 = 0$. Then

$$\begin{cases} |T'(x)| = S(x), & x \in \Omega\backslash\{0\}, \\ T(0) = 0; \end{cases} \quad \text{and} \quad \begin{cases} |T_0'(x)| = S_0 = S(0), & x \in \Omega\backslash\{0\}, \\ T_0(0) = 0, \end{cases} \tag{10}$$

where $S(x)$ is smooth.

The solutions T and T_0 have the following explicit forms,

$$T(x) = \begin{cases} \int_0^x S(t)dt, & x \geq 0, \\ -\int_0^x S(t)dt, & x < 0; \end{cases} \quad \text{and} \quad T_0(x) = S_0|x|. \tag{11}$$

In the case of multiplicative factorization, we have

$$\tau(x) = \begin{cases} \frac{\int_0^x S(t)dt}{S_0 x}, & x \neq 0, \\ 1, & x = 0. \end{cases} \tag{12}$$

By L'Hôpital's rule, it is easy to show that

$$\tau'(0) = \lim_{x\to 0} \tau'(x) = \lim_{x\to 0} \frac{S(x)x - \int_0^x S(t)dt}{S_0 x^2} = \lim_{x\to 0} \frac{S'(x)x}{2S_0 x} = \frac{S'(0)}{2S_0}.$$

Similarly, one can prove that $\tau^{(n)}(0) = \frac{S^{(n)}(0)}{(n+1)S_0}$ for $n \geq 1$. Therefore, if $S^{(n)}(0)$ exists, then $\tau^{(n)}(0)$ exists.

In the case of additive factorization, we have

$$\tau(x) = \begin{cases} \int_0^x S(t)dt - S_0 x, & x \geq 0, \\ -\int_0^x S(t)dt + S_0 x, & x < 0; \end{cases} \tag{13}$$

it follows that

$$\tau'(x) = \begin{cases} S(x) - S_0, & x > 0, \\ -S(x) + S_0, & x < 0. \end{cases} \tag{14}$$

Thus, $\tau'(0) = 0$. For $n > 1$, we have

$$\tau^{(n)}(x) = \begin{cases} S^{(n-1)}(x), & x > 0, \\ -S^{(n-1)}(x), & x < 0. \end{cases} \tag{15}$$

Therefore, for $n > 1$, only when $-\lim_{x\to 0^-} S^{(n-1)}(x) = \lim_{x\to 0^+} S^{(n-1)}(x)$, $\tau^{(n)}(0)$ exists.

Consequently, the above analysis indicates that the multiplicative factorization yields an underlying function τ which is as smooth as the right-hand side function S in the source neighborhood, while the additive factorization only yields an underlying function τ which will be smooth up to the first-order derivative without any condition and will be smooth up to higher-order derivatives with some conditions. Therefore, the additive factorization has some limitations in practice which may result in under-performance in comparison with the multiplicative factorization.

Moreover, the limit relations (7) and (9) indicate that both factorizations only impose some conditions on the underlying factors to control their local behaviors near the source point. Therefore, if T_0 is not a "good" approximation to T globally, then we may need more iterations to "correct" the underlying factors numerically.

Nevertheless, since T_0 captures the source singularity, both factorizations yield an underlying function τ which is differentiable near the source point, and this is our main motivation to carry out factorizations. As a result, we are able to design first-order fast sweeping methods to numerically compute τ with clean first-order convergence as the solution of the factored anisotropic eikonal equation (4) or (8), which are presented in the following section.

3 Numerical Schemes

We present our schemes on rectangular meshes only as the extension to triangular meshes can be done similarly.

3.1 Two-Dimensional Cases

We first derive the fast sweeping method for the factored anisotropic equation (4) [4, 18, 19, 25]. On a local mesh with grid size h, Fig. 1 shows an interior grid point C with four triangles that have C as a common vertex, and we discretize (4) on these four triangles. For example, on the triangle $\triangle CWS$ with $W = (x_W, y_W)$, $S = (x_S, y_S)$ and $C = (x_C, y_C)$, linear Taylor expansions yield

$$\begin{cases} \dfrac{\tau(C) - \tau(W)}{h} \approx \tau_x(C), \\ \dfrac{\tau(C) - \tau(S)}{h} \approx \tau_y(C). \end{cases} \tag{16}$$

Inserting these expansions into the factored anisotropic eikonal equation (4), we get a discretized equation,

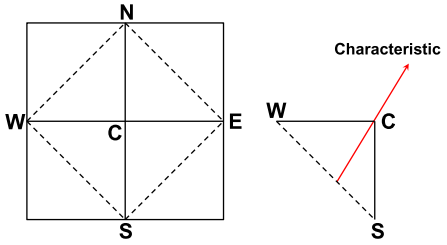

Fig. 1 2-D rectangular mesh, vertex C with four neighbors W, S, E and N

$$\sqrt{\begin{aligned}&\tau^2(C)\big(a{T_{0_x}}^2 - 2cT_{0_x}T_{0_y} + b{T_{0_y}}^2\big) + 2T_0\tau(C)\bigg(aT_{0_x}\frac{\tau(C)-\tau(W)}{h}\\&- c\bigg(T_{0_x}\frac{\tau(C)-\tau(S)}{h} + T_{0_y}\frac{\tau(C)-\tau(W)}{h}\bigg) + bT_{0_y}\frac{\tau(C)-\tau(S)}{h}\bigg)\\&+ T_0^2\bigg(a\bigg[\frac{\tau(C)-\tau(W)}{h}\bigg]^2 - 2c\frac{\tau(C)-\tau(W)}{h}\frac{\tau(C)-\tau(S)}{h} + b\bigg[\frac{\tau(C)-\tau(S)}{h}\bigg]^2\bigg)\end{aligned}} = 1. \tag{17}$$

Now given the values $\tau(W)$ and $\tau(S)$, we need to solve the equation (17) for $\tau(C)$. There are three scenarios:

1. There is only one solution for $\tau(C)$ from (17); i.e., the triangle supports one consistent candidate for $\tau(C)$.
2. There are two distinct solutions for $\tau(C)$ from (17); i.e., the triangle supports two consistent candidates for $\tau(C)$.
3. There is no solution for $\tau(C)$ from (17); i.e. the triangle does not support any consistent candidate for $\tau(C)$.

If Case 1 or 2 happens, we need to check whether a candidate value, denoted $\tau^h(C)$, for $\tau(C)$ that is consistent with the PDE satisfies the following **causality condition:** the characteristic for T passing through C, which is given by

$$\big(a(C)T_x^h(C) - c(C)T_y^h(C), b(C)T_y^h(C) - c(C)T_x^h(C)\big),$$

with $\nabla T^h(C) = \nabla\tau^h(C)T_0(C) + \tau^h(C)\nabla T_0(C)$, is in between the two sides $\overrightarrow{WC}$ and $\overrightarrow{SC}$ of the triangle as in Fig. 1.

If Case 3 happens, we will enforce the characteristic to be along the two sides $\overrightarrow{WC}$ and $\overrightarrow{SC}$ by using the characteristic equations,

$$\begin{cases}\dfrac{d(x,y)}{dt} = \left(\dfrac{\partial\mathcal{H}}{\partial p}, \dfrac{\partial\mathcal{H}}{\partial q}\right),\\[2ex] \dfrac{d(p,q)}{dt} = -\left(\dfrac{\partial\mathcal{H}}{\partial x}, \dfrac{\partial\mathcal{H}}{\partial y}\right),\\[2ex] \dfrac{d\tau}{dt} = (p,q)\cdot\dfrac{d(x,y)}{dt} = p\dfrac{\partial\mathcal{H}}{\partial p} + q\dfrac{\partial\mathcal{H}}{\partial q},\end{cases} \tag{18}$$

where $\mathcal{H}(x,y,p,q)$ is the Hamiltonian defined as the left-hand side of (4) with $p = \tau_x$ and $q = \tau_y$. In general, we will discretize the characteristic equations with linear approximations. In the current case, we can find a candidate value of $\tau^h(C)$ as follows. For example, if we enforce the characteristic to be along $\overrightarrow{WC}$, then $\frac{dy}{dt} = \frac{\partial\mathcal{H}}{\partial q} = 0$, yielding that $b(T_0\tau_y + T_{0_y}\tau) - c(T_0\tau_x + T_{0_x}\tau) = 0$; thus we have $(T_0\tau_y + T_{0_y}\tau) = \frac{c}{b}(T_0\tau_x + T_{0_x}\tau)$. By plugging

it into the factored eikonal equation (4), we have $\sqrt{\frac{ab-c^2}{b}}(T_0\tau_x + T_{0_x}\tau)^2 = 1$. Thus, by the finite difference discretization, a candidate value is given by

$$\tau^h(C) = \frac{T_0(C)\tau(W) + |WC|\sqrt{\frac{b(C)}{a(C)b(C)-c^2(C)}}}{T_0(C) + T_{0_x}(C)(x_C - x_W)}. \tag{19}$$

For each triangle, there may be multiple solutions for $\tau(C)$ satisfying the **causality condition**, and there may be even more on all four triangles. The fast sweeping method will pick up the minimum one according to recovered candidate values for T based on $T = \tau\, T_0$. Accordingly, we present the following local solver.

2-D Local Solver:

1. Initialization: $\tau(x_0, y_0) = 1$.
2. Gauss-Seidel iteration: sweeping the domain with four alternating orderings repeatedly:
   (1) $i = 1 : I, j = 1 : J$;
   (2) $i = 1 : I, j = J : 1$;
   (3) $i = I : 1, j = 1 : J$;
   (4) $i = I : 1, j = J : 1$.

   - At each gird point, discretize the factored eikonal equation on 4 triangles $\triangle CEN$, $\triangle CNW$, $\triangle CWS$ and $\triangle CSE$, and solve the discretized equations on each triangle. For example, on triangle $\triangle CWS$, solve equation (17) for two possible roots, denoted as $\tau_{C,1}$ and $\tau_{C,2}$.
     - If there are two real roots, $\tau_{C,1}$ and $\tau_{C,2}$, then
       * if both $\tau_{C,1}$, and $\tau_{C,2}$ satisfy the causality condition, denote $T_{WS} = \min\{\tau_{C,1}T_0(C), \tau_{C,2}T_0(C)\}$;
       * else if $\tau_{C,1}$ satisfies the causality condition, denote $T_{WS} = \tau_{C,1}T_0(C)$;
       * else if $\tau_{C,2}$ satisfies the causality condition, denote $T_{WS} = \tau_{C,2}T_0(C)$;
       * else if none of the two roots satisfies the causality condition, then use the method of characteristics on edges $\overrightarrow{WC}$ and $\overrightarrow{SC}$ to get two candidates values as in (19), denoted as τ_{WC} and τ_{SC} respectively.
         + if $\tau_{WC}T_0(C) \geq T(W)$ and $\tau_{SC}T_0(C) \geq T(S)$, denote $T_{WS} = \min\{\tau_{WC}T_0(C), \tau_{SC}T_0(C)\}$;
         + else if $\tau_{WC}T_0(C) \geq T(W)$, denote $T_{WS} = \tau_{WC}T_0(C)$;
         + else if $\tau_{SC}T_0(C) \geq T(S)$, denote $T_{WS} = \tau_{SC}T_0(C)$;
         + else, denote $T_{WS} = \infty$.
     - else, use the method of characteristics on edges $\overrightarrow{WC}$ and $\overrightarrow{SC}$ to get two candidates values as in (19), denoted as τ_{WC} and τ_{SC} respectively.
       * if $\tau_{WC}T_0(C) \geq T(W)$ and $\tau_{SC}T_0(C) \geq T(S)$, denote $T_{WS} = \min\{\tau_{WC}T_0(C), \tau_{SC}T_0(C)\}$;
       * else if $\tau_{WC}T_0(C) \geq T(W)$, denote $T_{WS} = \tau_{WC}T_0(C)$;
       * else if $\tau_{SC}T_0(C) \geq T(S)$, denote $T_{WS} = \tau_{SC}T_0(C)$;
       * else, denote $T_{WS} = \infty$.
   - Once we compute four values from the four triangles as above, denoted as T_{EN}, T_{NW}, T_{WS}, and T_{SE}, we choose the minimum from these four values, $T(C) = \min\{T_{EN}, T_{NW}, T_{WS}, T_{SE}\}$.
   - $\tau(C) = \frac{T(C)}{T_0(C)}$.

3. Convergence test: $|\tau^{new} - \tau^{old}|_\infty < tolerance$.

Next we consider the factored anisotropic equation (8). On a local mesh as in Fig. 1, we discretize the equation on the four triangles similarly as above. For example on $\triangle CWS$, we have the discretized equation,

$$\sqrt{\begin{aligned} &a\left[\frac{\tau(C)-\tau(W)}{h}\right]^2 - 2c\frac{\tau(C)-\tau(W)}{h}\frac{\tau(C)-\tau(S)}{h} + b\left[\frac{\tau(C)-\tau(S)}{h}\right]^2 \\ &+ 2\frac{\tau(C)-\tau(W)}{h}(aT_{0x}-cT_{0y}) + 2\frac{\tau(C)-\tau(S)}{h}(bT_{0y}-cT_{0x}) \\ &+ aT_{0x}{}^2 - 2cT_{0x}T_{0y} + bT_{0y}{}^2 \end{aligned}} = 1. \quad (20)$$

The local solver for the fast sweeping method is the same as above. The initialization is given as $\tau(x_0, y_0) = 0$. T is recovered by $\tau + T_0$. For the **causality condition**, in this case, the gradient of T is given by,

$$\nabla T^h(C) = \nabla T_0(C) + \nabla \tau^h(C).$$

When a characteristic is forced to propagate along edges, we use the method of characteristics (18) under linear approximations as in the case of multiplicative factors. For example, on edge $\overrightarrow{WC}$, a candidate value is given by,

$$\tau^h(C) = \tau(W) + \sqrt{\frac{b(C)}{a(C)b(C)-c^2(C)}}|WC| - T_{0x}(C)(x_C - x_W).$$

Proposition 3.1 *The discretization schemes* (17) *and* (20) *are consistent and monotone under the* **causality condition**.

Proof The consistency is obvious (e.g. [1, 11, 18, 19, 25]). We only sketch the proof for the monotonicity of (17). The monotonicity of (20) can be proved similarly.

Without loss of generality, we assume that the source $(x_0, y_0) = (0, 0)$; then

$$T_0(x, y) = \sqrt{\frac{b_0x^2 + 2c_0xy + a_0y^2}{a_0b_0 - c_0^2}}.$$

Denote the left hand side of (17) as $H^D(C, \tau(C), \tau(W), \tau(S))$. We want to show that

$$\frac{\partial H^D}{\partial \tau(C)} \geq 0, \qquad \frac{\partial H^D}{\partial \tau(W)} \leq 0, \quad \text{and} \quad \frac{\partial H^D}{\partial \tau(S)} \leq 0.$$

Taking derivatives, we have

$$\frac{\partial H^D}{\partial \tau(C)} = \left(T_{0x} + \frac{T_0}{h}\right)\left(aT_x^h - cT_y^h\right) + \left(T_{0y} + \frac{T_0}{h}\right)\left(bT_y^h - cT_x^h\right)|_{(x_C,y_C)}.$$

By the causality condition, we have

$$aT_x^h - cT_y^h|_{(x_C,y_C)} \geq 0 \quad \text{and} \quad \left(bT_y^h - cT_x^h\right)|_{(x_C,y_C)} \geq 0.$$

By the convexity of T_0 or by simple calculations, one can easily show that

$$T_{0x} + \frac{T_0}{h}|_{(x_C,y_C)} \geq 0 \quad \text{and} \quad T_{0y} + \frac{T_0}{h}|_{(x_C,y_C)} \geq 0.$$

Therefore, $\frac{\partial H^D}{\partial \tau(C)} \geq 0$.

Similarly, we have

$$\frac{\partial H^D}{\partial \tau(W)} = -\frac{T_0}{h}\big(aT_x^h - cT_y^h\big)|_{(x_C,y_C)} \le 0,$$
$$\frac{\partial H^D}{\partial \tau(S)} = -\frac{T_0}{h}\big(bT_y^h - cT_x^h\big)|_{(x_C,y_C)} \le 0.$$

Thus the monotonicity of (17) is proved. □

Remark 3.2 The consistency and monotonicity guarantee the convergence of the numerical solutions of the fast sweeping method to the right viscosity solutions [1, 11, 18, 19, 25].

Remark 3.3 Note that for the causality condition, we approximate ∇T with $\nabla T^h(C) = T_0(C)\nabla\tau^h(C) + \nabla T_0(C)\tau^h(C)$ and $\nabla T^h(C) = \nabla T_0(C) + \nabla\tau^h(C)$ for multiplicative and additive factorization, respectively, which is different from the approach used in [4], where $\nabla T^h(C) = (\frac{T^h(C)-T^h(W)}{h}, \frac{T^h(C)-T^h(S)}{h})$ on $\triangle CWS$. The former approximation guarantees that the numerical Hamiltonian is monotone, while the latter does not imply a monotone numerical Hamiltonian.

3.2 Three-Dimensional Cases

For 3-D cases, we assume that the anisotropy is modeled by

$$M(\mathbf{x}) = \begin{pmatrix} a(\mathbf{x}) & -d(\mathbf{x}) & -e(\mathbf{x}) \\ -d(\mathbf{x}) & b(\mathbf{x}) & -f(\mathbf{x}) \\ -e(\mathbf{x}) & -f(\mathbf{x}) & c(\mathbf{x}) \end{pmatrix},$$

with $\mathbf{x} = (x, y, z)$ and $\mathbf{x}_0 = (x_0, y_0, z_0)$.

The factored anisotropic eikonal equations with multiplicative and additive factors can be derived similarly as in 2-D cases, which are given by

$$\sqrt{\begin{aligned}&\tau^2\big(a{T_{0x}}^2 + b{T_{0y}}^2 + c{T_{0z}}^2 - 2dT_{0x}T_{0y} - 2eT_{0x}T_{0z} - 2fT_{0y}T_{0z}\big)\\ &+ 2\tau T_0\big(aT_{0x}\tau_x + bT_{0y}\tau_y + cT_{0z}\tau_z - d(\tau_x T_{0y} + T_{0x}\tau_y)\\ &- e(\tau_x T_{0z} + T_{0x}\tau_z) - f(\tau_y T_{0z} + T_{0y}\tau_z)\big)\\ &+ T_0^2\big(a\tau_x^2 + b\tau_y^2 + c\tau_z^2 - 2d\tau_x\tau_y - 2e\tau_x\tau_z - 2f\tau_y\tau_z\big)\end{aligned}} = 1, \tag{21}$$

and

$$\sqrt{\begin{aligned}&a\tau_x^2 + b\tau_y^2 + c\tau_z^2 - 2d\tau_x\tau_y - 2e\tau_x\tau_z - 2f\tau_y\tau_z + 2\big((aT_{0x} - dT_{0y} - eT_{0z})\tau_x\\ &+ (bT_{0y} - dT_{0x} - fT_{0z})\tau_y + (cT_{0z} - eT_{0x} - fT_{0y})\tau_z\big)\\ &+ a{T_{0x}}^2 + b{T_{0y}}^2 + c{T_{0z}}^2 - 2dT_{0x}T_{0y} - 2eT_{0x}T_{0z} - 2fT_{0y}T_{0z}\end{aligned}} = 1, \tag{22}$$

respectively.

The 3-D local solver for the fast sweeping method can be derived similarly as in 2-D cases. On a local mesh with grid size h, Fig. 2 shows an interior point C with eight tetrahedrons that have C as a common vertex, and we discretize (21) and (22) on these eight tetrahedrons.

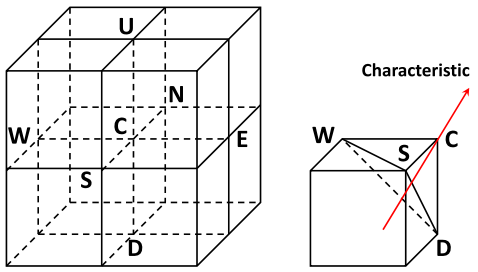

Fig. 2 3-D rectangular mesh, vertex C with six neighbors W, S, E, N, U and D

For example, on the tetrahedron $CWSD$ with $W = (x_W, y_W, z_W)$, $S = (x_S, y_S, z_S)$, $D = (x_D, y_D, z_D)$, and $C = (x_C, y_C, z_C)$, the discretizations of (21) and (22) are given by substituting (τ_x, τ_y, τ_z) with the linear Taylor expansions,

$$\begin{cases} \tau_x \approx \dfrac{\tau(C) - \tau(W)}{h}, \\ \tau_y \approx \dfrac{\tau(C) - \tau(S)}{h}, \\ \tau_z \approx \dfrac{\tau(C) - \tau(D)}{h}. \end{cases} \tag{23}$$

Similar to the 2-D local solver, given $\tau(W)$, $\tau(S)$ and $\tau(D)$, we need to solve discretized equations for $\tau(C)$. If there exist consistent candidates for $\tau(C)$, we need to check whether they satisfy the **causality condition**: the characteristic for T passing through C is inside the tetrahedron $CWSD$ as in Fig. 2. For example, if a consistent candidate is given by $\tau^h(C)$, then we require that the characteristic of T be inside the tetrahedron $CWSD$. Here the characteristic of T is approximated by,

$$\left(aT_x^h - dT_y^h - eT_z^h, bT_y^h - dT_x^h - fT_z^h, cT_z^h - eT_x^h - fT_y^h\right)|_C,$$

where $\nabla T^h(C) = \nabla\tau^h(C)T_0(C) + \tau^h(C)\nabla T_0(C)$ for multiplicative factors and $\nabla T^h(C) = \nabla\tau^h(C) + \nabla T_0(C)$ for additive factors, respectively, and $\nabla\tau^h(C)$ is approximated by (23).

We summarize the 3-D local solver as follows.

3-D Local Solver:

1. Initialization: $\tau(x_0, y_0, z_0) = 1$ for multiplicative factors and $= 0$ for additive factors.
2. Gauss-Seidel iteration: sweeping the domain with eight alternating orderings repeatedly:
   (1) $i = 1 : I$, $j = 1 : J$, $k = 1 : K$;
   (2) $i = 1 : I$, $j = J : 1$, $k = 1 : K$;
   (3) $i = I : 1$, $j = 1 : J$, $k = 1 : K$;
   (4) $i = I : 1$, $j = J : 1$, $k = 1 : K$;
   (5) $i = 1 : I$, $j = 1 : J$, $k = K : 1$;
   (6) $i = 1 : I$, $j = J : 1$, $k = K : 1$;
   (7) $i = I : 1$, $j = 1 : J$, $k = K : 1$;
   (8) $i = I : 1$, $j = J : 1$, $k = K : 1$.

   - At each gird point C as in Fig. 2, discretize the factored eikonal equation on eight tetrahedrons, and solve the discretized equations on each tetrahedron. For example, on the tetrahedron $CWSD$, solve the discretized equation for two possible roots, denoted $\tau_{C,1}$ and $\tau_{C,2}$.

  - if both two roots are consistent and satisfy the causality condition, we choose the minimum one according to the recovered candidate values of T.
  - else if only one of the two roots is consistent and satisfies the causality condition, we choose this one.
  - else if neither one satisfies the causality condition, then we apply the 2-D local solver on the three faces $\triangle CWS$, $\triangle CWD$ and $\triangle CSD$, and we choose the minimum one according to the recovered candidate values of T.
- Choose the minimum one according to the recovered candidate values of T among all eight tetrahedrons.

3. Convergence test: $|\tau^{new} - \tau^{old}|_\infty < tolerance$.

Remark 3.4 In the 3-D solver, for example on the tetrahedron $CWSD$, when there exist no roots that satisfy the causality condition, we need to apply the 2-D local solver on the three faces $\triangle CWS$, $\triangle CWD$ and $\triangle CSD$. On each face, the 3-D anisotropic eikonal equation reduces to the 2-D anisotropic eikonal equation

$$\sqrt{\alpha\tau_\mu^2 - 2\gamma\tau_\mu\tau_\nu + \beta\tau_\nu^2} = 1,$$

where,

- On $\triangle CWS$, $(\mu,\nu) = (x,y)$, $(\alpha,\beta,\gamma) = (a - e^2/c, b - f^2/c, d + fe/c)$;
- On $\triangle CWD$, $(\mu,\nu) = (x,z)$, $(\alpha,\beta,\gamma) = (a - d^2/b, c - f^2/b, e + fd/b)$;
- On $\triangle CSD$, $(\mu,\nu) = (y,z)$, $(\alpha,\beta,\gamma) = (b - d^2/a, c - e^2/a, f + ed/a)$.

Remark 3.5 In 3-D cases, without loss of generality, if $(x_0, y_0, x_0) = (0,0,0)$, then T_0 is given by

$$T_0 = \sqrt{\begin{array}{l}(b_0c_0 - f_0^2)x^2 + 2(c_0d_0 + e_0f_0)xy + 2(d_0f_0 + b_0e_0)xz \\ + \dfrac{(a_0c_0 - e_0^2)y^2 + 2(d_0e_0 + a_0f_0)yz + (a_0b_0 - d_0^2)z^2}{a_0b_0c_0 - 2d_0e_0f_0 - a_0f_0^2 - c_0d_0^2 - b_0e_0^2}.\end{array}}$$

Remark 3.6 For the 3-D cases, the consistency and monotonicity of the numerical scheme can be proved similarly as in 2-D cases. Thus, the convergence of the numerical solutions to the correct viscosity solutions can also be proved [1, 11, 18, 19, 25].

4 Numerical Examples

In this section we test our numerical algorithms on a variety of examples. We compare numerical solutions by three different formulations based on the multiplicatively factored eikonal equation, the additively factored eikonal equation, and the original eikonal equation, respectively. The numerical solution of the original eikonal equation is computed by the fast sweeping method as in [18, 19, 25]. We use ℓ^∞-norm to measure the error and choose $tolerance = 10^{-7}$.

4.1 2-D Examples

We first test both anisotropic and isotropic cases in 2-D. The first two isotropic cases have also been tested in [4], and we choose them because they have exact solutions.

Example 1 (Constant gradient of slowness squared) The slowness field is given by

$$S^2(\mathbf{x}) = S_0^2 + 2\mathbf{g}\cdot(\mathbf{x}-\mathbf{x}_0). \tag{24}$$

We test one setup with parameters:

- source point: $\mathbf{x}_0 = (x_0, y_0) = (0.25, 0.25)$.
- computational domain: $\mathbf{x} = (x, y) \in [0,\ 0.5]^2$.
- $S_0 = 2$.
- $\mathbf{g} = (g_1, g_2) = (0, -3)$.
- the exact solution $T(\mathbf{x}) = \bar{S}^2\sigma - |\mathbf{g}|^2\frac{\sigma^3}{6}$ with $\sigma = \frac{\sqrt{2(\bar{S}^2-\sqrt{\bar{S}^4-|\mathbf{g}|^2|\mathbf{x}-\mathbf{x}_0|^2})}}{|\mathbf{g}|}$, $\bar{S} = \sqrt{S_0^2 + \mathbf{g}\cdot(\mathbf{x}-\mathbf{x}_0)}$.
- $T_0(\mathbf{x}) = S_0|\mathbf{x}-\mathbf{x}_0|$.

Table 1 shows the comparison results in terms of mesh refinement for ℓ^∞-errors and number of iterations. As the mesh is refined, both schemes based on the factored eikonal equations yield clean first-order convergence while the scheme based on the original eikonal equation yields polluted first-order convergence. This clearly demonstrates that the proposed strategies to remove the source singularity work well.

In addition, we compare the CPU time for the factored approaches and the original method. The original method takes CPU time 116 seconds to reach maximum error of 0.0008043 after 5 iterations on a 3200×3200 mesh; the additive factorization takes CPU time 2.72 seconds to reach maximum error of 0.0003064 after 9 iterations on a 400×400 mesh; and the multiplicative factorization takes CPU time 0.06 seconds to reach maximum error of 0.0001409 after 13 iterations on a 50×50 mesh.

We also report the results based on the initialization near the source by the wrap-up method; namely, the initial condition is assigned on a disk with radius R centered at the source point, where $T = T_0$ on the disk. Table 1 shows the results.

Example 2 (Constant gradient of velocity) The slowness distribution has the form,

$$\frac{1}{S(\mathbf{x})} = \frac{1}{S_0} + \mathbf{g}\cdot(\mathbf{x}-\mathbf{x}_0). \tag{25}$$

We test one setup with parameters:

- source point: $\mathbf{x}_0 = (x_0, y_0) = (0.25, 0.25)$.
- computational domain: $\mathbf{x} = (x, y) \in [0,\ 0.5]^2$.
- $S_0 = 2$.
- $\mathbf{g} = (g_1, g_2) = (0, -1)$.
- the exact solution $T(\mathbf{x}) = \frac{1}{|\mathbf{g}|}\operatorname{arccosh}(1 + \frac{1}{2}S(\mathbf{x})S_0|\mathbf{g}|^2|\mathbf{x}-\mathbf{x}_0|^2)$, with arccosh being the inverse hyperbolic cosine function, i.e. $\operatorname{arccosh}(z) = \log(z + \sqrt{z^2-1})$.
- $T_0(\mathbf{x}) = S_0|\mathbf{x}-\mathbf{x}_0|$.

Table 2 shows the comparison results in terms of mesh refinement for ℓ^∞-errors and number of iterations. As the mesh is refined, both schemes based on the factored eikonal equations yield clean first-order convergence while the scheme based on the original eikonal equation yields polluted first-order convergence.

In addition, we compare the CPU time for the factored approaches and the original method. The original method takes CPU time 40.97 seconds to reach maximum error of 0.0016797 after 8 iterations on a 1600×1600 mesh; the additive factorization takes CPU

Table 1 Example 1: numerical error

Factored eikonal equation: multiplicatively			
Mesh	Error of T on $[0, 0.5]^2$	# iterations	Convergence order
50×50	0.0001409	13	–
100×100	0.0000705	13	0.9900
200×200	0.0000352	10	1.0020
400×400	0.0000176	10	1.0000
Factored eikonal equation: additively			
Mesh	Error of T on $[0, 0.5]^2$	# iterations	Convergence order
50×50	0.0024091	13	–
100×100	0.0012141	10	0.9886
200×200	0.0006105	9	0.9918
400×400	0.0003064	9	0.9946
Original eikonal equation			
Mesh	Error of T on $[0, 0.5]^2$	# iterations	Convergence order
50×50	0.0232938	5	–
100×100	0.0138471	5	0.7504
200×200	0.0080710	5	0.7788
400×400	0.0046246	5	0.8034
Original eikonal equation: wrap-up $R = 0.02$			
Mesh	Error of T on $[0, 0.5]^2$	# iterations	Convergence order
50×50	0.0205600	5	–
100×100	0.0104242	5	0.9799
200×200	0.0051901	5	1.0061
400×400	0.0025231	5	1.0406
800×800	0.0013361	5	0.9171
1600×1600	0.0007852	5	0.7669

time 3.53 seconds to reach maximum error of 0.0013361 after 10 iterations on a 400×400 mesh; and the multiplicative factorization takes CPU time 0.25 seconds to reach maximum error of 0.0015702 after 13 iterations on a 100×100 mesh.

We also report the results with the wrap-up method in Table 2. Initialization in the wrap-up method is done similarly as in Example 1.

Example 3 In this example, we test an anisotropic case with the following setup:

1. source point: $\mathbf{x}_0 = (x_0, y_0) = (0.5, 0.5)$.
2. computational domain: $\mathbf{x} = (x, y) \in [0,\ 1]^2$.
3. $a(x, y) = \dfrac{1.0}{e^{-2\sqrt{2(x-x_0)^2+2(x-x_0)(y-y_0)+(y-y_0)^2}}}$.
4. $b(x, y) = \dfrac{2.0}{e^{-2\sqrt{2(x-x_0)^2+2(x-x_0)(y-y_0)+(y-y_0)^2}}}$.

Table 2 Example 2: numerical error

Factored eikonal equation: multiplicatively			
Mesh	Error of T on $[0, 0.5]^2$	# iterations	Convergence order
50×50	0.0031484	13	–
100×100	0.0015702	13	1.0037
200×200	0.0007841	13	1.0018
400×400	0.0003918	13	1.0009
Factored eikonal equation: additively			
Mesh	Error of T on $[0, 0.5]^2$	# iterations	Convergence order
50×50	0.0106532	12	–
100×100	0.0053306	12	0.9989
200×200	0.0026689	12	0.9981
400×400	0.0013361	12	0.9982
Original eikonal equation			
Mesh	Error of T on $[0, 0.5]^2$	# iterations	Convergence order
50×50	0.0306498	8	–
100×100	0.0174656	8	0.8114
200×200	0.0098650	8	0.8241
400×400	0.0055181	8	0.8381
Original eikonal equation: wrap-up $R = 0.02$			
Mesh	Error of T on $[0,\ 0.5]^2$	# iterations	Convergence order
50×50	0.0278171	8	–
100×100	0.0138165	8	1.0096
200×200	0.0067192	8	1.0400
400×400	0.0031448	8	1.0953
800×800	0.0016133	8	0.9630
1600×1600	0.0011280	8	0.5162

5. $c(x, y) = \frac{1.0}{e^{-2\sqrt{2(x-x_0)^2+2(x-x_0)(y-y_0)+(y-y_0)^2}}}$.
6. the exact solution $T(x, y) = 1 - e^{-\sqrt{2(x-x_0)^2+2(x-x_0)(y-y_0)+(y-y_0)^2}}$.
7. $T_0(x, y) = \sqrt{2(x-x_0)^2 + 2(x-x_0)(y-y_0) + (y-y_0)^2}$.

Table 3 shows the comparison results in terms of mesh refinement for ℓ^∞-errors and number of iterations. As the mesh is refined, both schemes based on the factored eikonal equations yield clean first-order convergence while the scheme based on the original eikonal equation yields polluted first-order convergence, and this implies that the proposed strategies to remove the source singularity work well.

In addition, we compare the CPU time for the factored approaches and the original method. The original method takes CPU time 12.3 seconds to reach maximum error of 0.0048191 after 5 iterations on a 1280×1280 mesh; the additive factorization takes CPU

Table 3 Example 3: numerical error

Factored anisotropic eikonal equation: multiplicatively			
Mesh	Error of T on $[0,1]^2$	# iterations	Convergence order
40×40	0.0043553	14	–
80×80	0.0023646	14	0.8812
160×160	0.0012554	12	0.9134
320×320	0.0006547	9	0.9392
Factored anisotropic eikonal equation: additively			
Mesh	Error of T on $[0,1]^2$	# iterations	Convergence order
40×40	0.0158459	17	–
80×80	0.0079552	16	0.9941
160×160	0.0038892	16	1.0324
320×320	0.0019980	13	0.9609
Original anisotropic eikonal equation			
Mesh	Error of T on $[0,1]^2$	# iterations	Convergence order
40×40	0.0530904	5	–
80×80	0.0352187	5	0.5921
160×160	0.0224187	5	0.6516
320×320	0.0137871	5	0.7014
Original anisotropic eikonal equation: wrap-up $R = 0.02$			
Mesh	Error of T on $[0,1]^2$	# iterations	Convergence order
40×40	0.0530904	5	–
80×80	0.0325042	5	0.7078
160×160	0.0193081	5	0.7514
320×320	0.0109105	5	0.8235
640×640	0.0058674	5	0.8949
1280×1280	0.0031004	5	0.9203

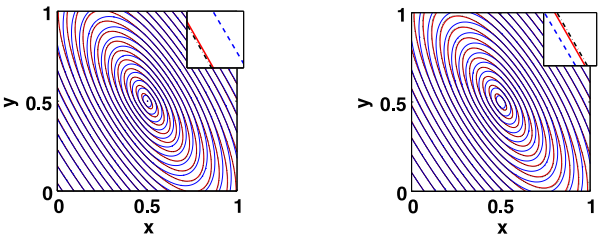

Fig. 3 (Color online) Example 3. Plots of T. *Left*: factored equation with multiplicative factors. *Right*: factored equation with additive factors. Zoom-in to the corner. *Red*: exact solution; *Blue-dashed*: original equation; *Black-dashed-dot*: factored equation

time 3.97 seconds to reach maximum error of 0.0019980 after 13 iterations on a 320×320 mesh; and the multiplicative factorization takes CPU time 0.67 seconds to reach maximum error of 0.0012554 after 12 iterations on a 160×160 mesh. Figure 3 shows the plots of numerical solutions on a 160×160 mesh.

We also report the results with the wrap-up method in Table 3.

Example 4 In this example, we test an anisotropic case that has been tested in [18] with the following setup:

1. source point: $\mathbf{x}_0 = (x_0, y_0) = (0.5, 0.5)$.
2. computational domain: $\mathbf{x} = (x, y) \in [0,\ 1]^2$.
3. $a(x, y) = 150.25(1 + \lambda \sin^2(\pi xy))$.
4. $b(x, y) = 50.75(1 + \delta \cos^2(\pi xy))$.
5. $c(x, y) = 86.16953(1 - \epsilon \sin^2(\pi xy))$.
6. $\lambda = 1, \delta = 1, \epsilon = 0.125$.
7. the exact solution: we compute the numerical solution with the factored equation (8) on a 5120×5120 mesh, and use it as an approximation of the exact solution.
8. $T_0(x, y) = \sqrt{\frac{50.75(1+0.5\delta)(x-x_0)^2+86.16963(2-\epsilon)(x-x_0)(y-y_0)+150.25(1+0.5\lambda)(y-y_0)^2}{150.25(1+0.5\lambda)50.75(1+0.5\delta)-(86.16953(1-0.5\epsilon))^2}}$.

Table 4 shows the comparison results in terms of mesh refinement for ℓ^∞-errors and number of iterations. As the mesh is refined, both schemes based on the factored eikonal equations yield well-behaved first-order convergence while the scheme based on original eikonal equation yields polluted first-order convergence.

In addition, we compare the CPU time for the factored approaches and the original method. The original method takes CPU time 5.76 seconds to reach maximum error of 0.000630 after 8 iterations on a 640×640 mesh; the additive factorization takes CPU time 0.04 seconds to reach maximum error of 0.000459 after 13 iterations on a 40×40 mesh; and the multiplicative factorization takes CPU time 0.05 seconds to reach maximum error of 0.000376 after 17 iterations on a 40×40 mesh. Figure 4 shows the plots of numerical solutions on a 160×160 mesh.

We also report the results with the wrap-up method in Table 4.

4.2 3-D Examples

We test both anisotropic and isotropic cases in 3-D.

Example 5 (Constant gradient of slowness squared) In this example, we test a 3-D isotropic case (Appendix in [4]) as in Example 1 with the slowness field

$$S^2(\mathbf{x}) = S_0^2 + 2\mathbf{g} \cdot (\mathbf{x} - \mathbf{x}_0). \tag{26}$$

We test one setup with parameters:

- source point: $\mathbf{x}_0 = (x_0, y_0, z_0) = (0.25, 0.25, 0.25)$.
- computational domain: $\mathbf{x} = (x, y, z) \in [0,\ 0.5]^3$.
- $S_0 = 2$.
- $\mathbf{g} = (g_1, g_2, g_3) = (0, -3, 0)$.
- the exact solution $T(\mathbf{x})$ and $T_0(\mathbf{x})$ are similar as in Example 1.

Table 5 shows the comparison results in terms of mesh refinement for ℓ^∞-errors and number of iterations. As the mesh is refined, both schemes based on the factored eikonal equations yield clean first-order convergence while the scheme based on the original eikonal equation yields polluted first-order convergence. This clearly demonstrates that the proposed strategies to remove the source singularity work well. Figures 5 and 6 show the plots of slices of the numerical solutions at $z = z_0$, $y = y_0$, and $x = x_0$ for multiplicative factors and additive factors, respectively, on an 80×80 mesh.

Table 4 Example 4: numerical error

Factored anisotropic eikonal equation: multiplicatively			
Mesh	Error of T on $[0.3, 0.7]^2$	# iterations	Convergence order
40×40	0.000376	17	–
80×80	0.000215	17	0.8064
160×160	0.000120	14	0.8413
320×320	0.000065	13	0.8845
640×640	0.000033	13	0.9780
Factored anisotropic eikonal equation: additively			
Mesh	Error of T on $[0.3, 0.7]^2$	# iterations	Convergence order
40×40	0.000459	13	–
80×80	0.000237	13	0.9536
160×160	0.000124	10	0.9345
320×320	0.000063	10	0.9769
640×640	0.000031	8	1.0231
Original anisotropic eikonal equation			
Mesh	Error of T on $[0.3, 0.7]^2$	# iterations	Convergence order
40×40	0.004564	8	–
80×80	0.002843	8	0.6773
160×160	0.001754	8	0.6968
320×320	0.001063	8	0.7225
640×640	0.000630	8	0.7547
Original anisotropic eikonal equation: wrap-up $R = 0.02$			
Mesh	Error of T on $[0.3, 0.7]^2$	# iterations	Convergence order
40×40	0.004564	8	–
80×80	0.002563	8	0.8325
160×160	0.001432	8	0.8398
320×320	0.000783	8	0.8709
640×640	0.000409	8	0.9369

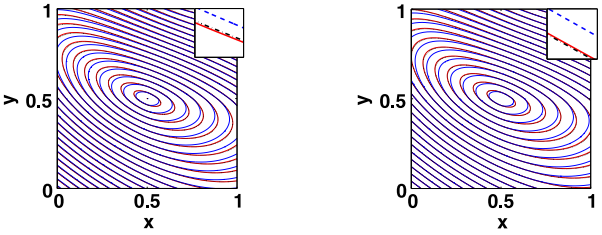

Fig. 4 (Color online) Example 4. Plots of T. *Left*: factored equation with multiplicative factors. *Right*: factored equation with additive factors. Zoom-in to the corner. *Red*: exact solution; *Blue-dashed*: original equation; *Black-dashed-dot*: factored equation

Example 6 (Constant gradient of velocity) In this example, we test a 3-D isotropic case (Appendix in [4]) as in Example 2 with the slowness field, the form,

Table 5 Example 5: numerical error

Factored eikonal equation: multiplicatively			
Mesh	Error of T on $[0,0.5]^3$	# iterations	Convergence order
$40 \times 40 \times 40$	0.0002560	17	–
$80 \times 80 \times 80$	0.0001280	17	1.0000
$160 \times 160 \times 160$	0.0000640	17	1.0000
Factored eikonal equation: additively			
Mesh	Error of T on $[0,0.5]^3$	# iterations	Convergence order
$40 \times 40 \times 40$	0.0034139	18	–
$80 \times 80 \times 80$	0.0017216	17	0.9877
$160 \times 160 \times 160$	0.0008666	17	0.9903
Original eikonal equation			
Mesh	Error of T on $[0,0.5]^3$	# iterations	Convergence order
$40 \times 40 \times 40$	0.0462817	9	–
$80 \times 80 \times 80$	0.0275778	9	0.7469
$160 \times 160 \times 160$	0.0161005	9	0.7764

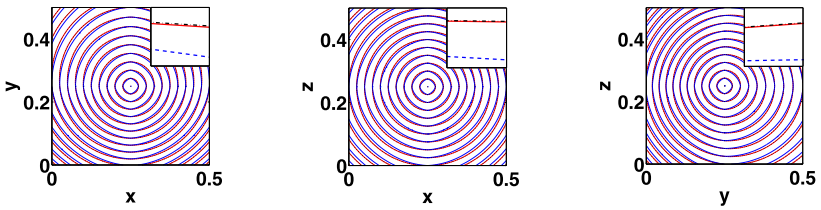

Fig. 5 (Color online) Example 5. Plots of T with multiplicative factors. *From left to right*: slices at $z = z_0$, $y = y_0$ and $x = x_0$ respectively. Zoom-in to the corner. *Red*: exact solution; *Blue-dashed*: original equation; *Black-dashed-dot*: factored equation

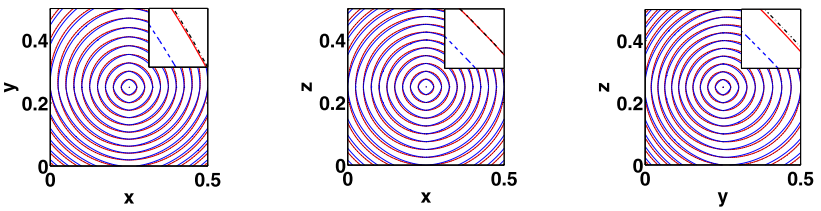

Fig. 6 (Color online) Example 5. Plots of T with additive factors. *From left to right*: slices at $z = z_0$, $y = y_0$ and $x = x_0$ respectively. Zoom-in to the corner. *Red*: exact solution; *Blue-dashed*: original equation; *Black–dashed-dot*: factored equation

$$\frac{1}{S(\mathbf{x})} = \frac{1}{S_0} + \mathbf{g} \cdot (\mathbf{x} - \mathbf{x}_0). \tag{27}$$

We test one setup with parameters:

Table 6 Example 6: numerical error

Factored eikonal equation: multiplicatively			
Mesh	Error of T on $[0, 0.5]^3$	# iterations	Convergence order
$40 \times 40 \times 40$	0.0046817	22	–
$80 \times 80 \times 80$	0.0023305	24	1.0064
$160 \times 160 \times 160$	0.0011628	24	1.0030
Factored eikonal equation: additively			
Mesh	Error of T on $[0, 0.5]^3$	# iterations	Convergence order
$40 \times 40 \times 40$	0.0138180	22	–
$80 \times 80 \times 80$	0.0069401	24	0.9935
$160 \times 160 \times 160$	0.0034843	24	0.9941
Original eikonal equation			
Mesh	Error of T on $[0, 0.5]^3$	# iterations	Convergence order
$40 \times 40 \times 40$	0.0539078	16	–
$80 \times 80 \times 80$	0.0312133	16	0.7883
$160 \times 160 \times 160$	0.0178780	16	0.8040

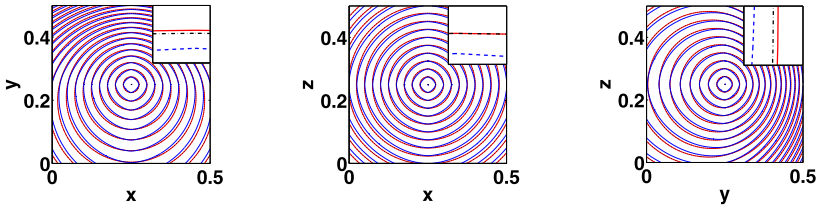

Fig. 7 (Color online) Example 6. Plots of T with multiplicative factors. *From left to right*: slices at $z = z_0$, $y = y_0$ and $x = x_0$ respectively. Zoom-in to the corner. *Red*: exact solution; *Blue-dashed*: original equation; *Black-dashed-dot*: factored equation

- source point: $\mathbf{x}_0 = (x_0, y_0, z_0) = (0.25, 0.25, 0.25)$.
- computational domain $\mathbf{x} = (x, y, z) \in [0,\ 0.5]^3$.
- $S_0 = 2$.
- $\mathbf{g} = (g_1, g_2, g_3) = (0, -1, 0)$.
- the exact solution $T(\mathbf{x})$ and $T_0(\mathbf{x})$ are similar as in Example 2.

Table 6 shows the comparison results in terms of mesh refinement for ℓ^∞-errors and number of iterations. As the mesh is refined, both schemes based on the factored eikonal equations yield clean first-order convergence while the scheme based on the original eikonal equation yields polluted first-order convergence. This clearly demonstrates that the proposed strategies to remove the source singularity work well. Figures 7 and 8 show the plots of slices of the numerical solutions at $z = z_0$, $y = y_0$, and $x = x_0$ for multiplicative factors and additive factors, respectively, on an 80×80 mesh.

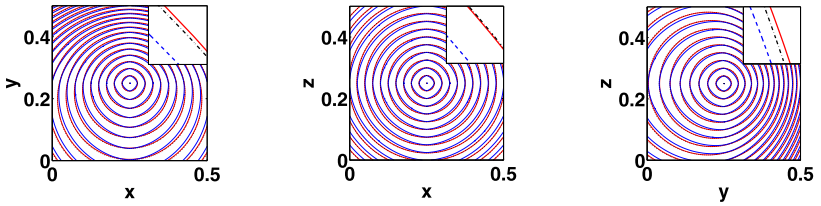

Fig. 8 (Color online) Example 6. Plots of T with additive factors. *From left to right*: slices at $z = z_0$, $y = y_0$ and $x = x_0$ respectively. Zoom-in to the corner. *Red*: exact solution; *Blue-dashed*: original equation; *Black–dashed-dot*: factored equation

Example 7 In this example, we test a 3-D anisotropic case with the following setup:

1. source point: $\mathbf{x}_0 = (x_0, y_0, z_0) = (0.5, 0.5, 0.5)$.
2. computational domain: $\mathbf{x} = (x, y, z) \in [0,\ 1]^3$.
3. $a(x, y, z) = \frac{1.0}{e^{-2W(x,y,z)}}$.
4. $b(x, y, z) = \frac{3.0}{e^{-2W(x,y,z)}}$.
5. $c(x, y, z) = \frac{2.0}{e^{-2W(x,y,z)}}$.
6. $d(x, y, z) = \frac{0.5}{e^{-2W(x,y,z)}}$.
7. $e(x, y, z) = \frac{0.3}{e^{-2W(x,y,z)}}$.
8. $f(x, y, z) = \frac{0.1}{e^{-2W(x,y,z)}}$.
9. the exact solution: $T(x, y, z) = 1 - e^{-W(x,y,z)}$.
10. $W(x, y, z) = \frac{\sqrt{\begin{array}{l}5.99(x-x_0)^2+2.06(x-x_0)(y-y_0)+1.9(x-x_0)(z-z_0)\\ +1.91(y-y_0)^2+0.5(y-y_0)(z-z_0)+2.75(z-z_0)^2\end{array}}}{\sqrt{5.19}}$.
11. $T_0(x, y, z) = W(x, y, z)$.

Table 7 shows the comparison results in terms of mesh refinement for ℓ^∞-errors and number of iterations. As the mesh is refined, both schemes based on the factored eikonal equations yield clean first-order convergence while the scheme based on the original eikonal equation yields polluted first-order convergence. This clearly demonstrates that the proposed strategies work well to remove the source singularity. Figures 9 and 10 show the plots of slices of the numerical solutions on an 80×80 mesh at $z = z_0$, $y = y_0$, and $x = x_0$ for multiplicative factors and additive factors, respectively.

Remark 4.1 Note that for the factored equations, the fast sweeping scheme with the **causality condition** and the approximation of ∇T given above may need more iterations compared to that for the original equation. However, the number of iterations does not increase when the mesh is refined. If we approximate ∇T with linear Taylor expansions in the **causality condition** as in [4] (or Remark 3.3), the improvement of accuracy can also be obtained without increasing the number of iterations as in [4]; that is, the number of iterations is equal to that of the fast sweeping method for the original equation when the scheme converges according to a certain tolerance; however, the resulting scheme is not monotone.

For the factored approaches, it is clear that the factored equations are more complicated than the original anisotropic eikonal equations in terms of the number of basic operations at each grid point, which results in more CPU time for the factored approaches than that of the original fast sweeping method if the computation is performed on the same mesh. However,

Table 7 Example 7: numerical error

Factored anisotropic eikonal equation: multiplicatively			
Mesh	Error of T on $[0,1]^3$	# iterations	Convergence order
$40 \times 40 \times 40$	0.0031412	19	–
$80 \times 80 \times 80$	0.0016459	18	0.9324
$160 \times 160 \times 160$	0.0008482	17	0.9564
Factored anisotropic eikonal equation: additively			
Mesh	Error of T on $[0,1]^3$	# iterations	Convergence order
$40 \times 40 \times 40$	0.0106848	24	–
$80 \times 80 \times 80$	0.0053540	21	0.9969
$160 \times 160 \times 160$	0.0026813	18	0.9977
Original anisotropic eikonal equation			
Mesh	Error of T on $[0,1]^3$	# iterations	Convergence order
$40 \times 40 \times 40$	0.0351380	9	–
$80 \times 80 \times 80$	0.0224838	9	0.6441
$160 \times 160 \times 160$	0.0138448	9	0.6995

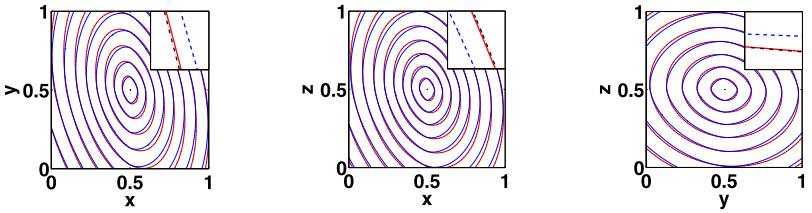

Fig. 9 (Color online) Example 7. Plots of T with multiplicative factors. *From left to right*: slices at $z = z_0, y = y_0$ and $x = x_0$ respectively. Zoom-in to the corner. *Red*: exact solution; *Blue-dashed*: original equation; *Black-dashed-dot*: factored equation

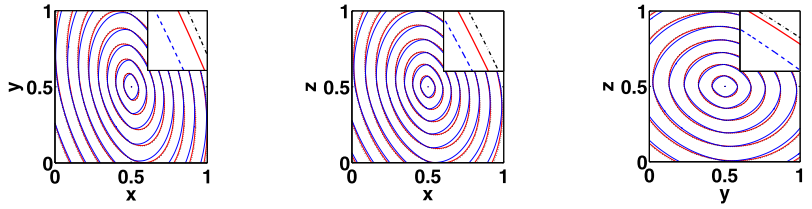

Fig. 10 (Color online) Example 7. Plots of T with additive factors. *From left to right*: slices at $z = z_0$, $y = y_0$ and $x = x_0$ respectively. Zoom-in to the corner. *Red*: exact solution; *Blue-dashed*: original equation; *Black-dashed-dot*: factored equation

in order to obtain the same accuracy, the factored approaches in general take less CPU time than the original fast sweeping method.

Remark 4.2 For the wrap-up method, the size of the disk near the source, the parameter R, is ad hoc. Although there are cases in which the wrap-up method can obtain good results when R is chosen appropriately, how to find this parameter R is itself a difficult problem when the medium is inhomogeneous; see [16] for analysis in a slightly different problem to estimate such a parameter. For the factorization approaches proposed here, we do not need to worry about such ad hoc parameters.

5 Conclusion

We extend the factorization idea to the anisotropic eikonal equation with a point-source condition. Besides factoring the unknown function into two multiplicative factors, one can also factor the unknown function into two additive factors. One of the two factors is specified analytically to capture the source singularity, which makes the other factor differentiable near the source. Fast sweeping schemes are designed to numerically solve the factored equations, and the schemes are monotone. Numerical examples show that monotone schemes indeed yield clean first-order convergence rather than polluted first-order convergence; moreover, in comparison to the case without treating source singularity, numerical errors with such factorization-based singularity treatment are in most cases decreased significantly.

Acknowledgements Qian is partially supported by NSF 0810104, NSF 0830161 and NSF 1115363.

References

1. Barles, G., Souganidis, P.E.: Convergence of approximation schemes for fully nonlinear second order equations. Asymptot. Anal. **4**, 271–283 (1991)
2. Boué, M., Dupuis, P.: Markov chain approximations for deterministic control problems with affine dynamics and quadratic cost in the control. SIAM J. Numer. Anal. **36**(3), 667–695 (1999)
3. Cecil, T., Osher, S.J., Qian, J.: Simplex free adaptive tree fast sweeping and evolution methods for solving level set equations in arbitrary dimension. J. Comput. Phys. **213**, 458–473 (2006)
4. Fomel, S., Luo, S., Zhao, H.-K.: Fast sweeping method for the factored eikonal equations. J. Comput. Phys. **228**(17), 6440–6445 (2009)
5. Helmsen, J., Puckett, E., Colella, P., Dorr, M.: Two new methods for simulating photolithography development in 3d. Proc. SPIE **2726**, 253–261 (1996)
6. Kao, C., Osher, S., Qian, J.: Lax-Friedrichs sweeping schemes for static Hamilton-Jacobi equations. J. Comput. Phys. **196**, 367–391 (2004)
7. Kao, C., Osher, S., Tsai, Y.: Fast sweeping method for static Hamilton-Jacobi equations. SIAM J. Numer. Anal. **42**, 2612–2632 (2005)
8. Kao, C., Osher, S., Qian, J.: Legendre transform based fast sweeping methods for static Hamilton-Jacobi equations on triangulated meshes. J. Comput. Phys. **227**, 10209–10225 (2008)
9. Leung, S., Qian, J.: An adjoint state method for three-dimensional transmission traveltime tomography using first-arrivals. Commun. Math. Sci. **4**, 249–266 (2006)
10. Li, F., Shu, C.-W., Zhang, Y.-T., Zhao, H.-K.: A second-order discontinuous Galerkin fast sweeping method for eikonal equations. J. Comput. Phys. **227**, 8191–8208 (2008)
11. Luo, S.: Numerical methods for static Hamilton-Jacobi equations. Ph.D Thesis, University of California, Irvine (2009)
12. Luo, S., Qian, J.: Factored singularities and high-order Lax-Friedrichs sweeping schemes for point-source traveltimes and amplitudes. J. Comput. Phys. **230**, 4742–4755 (2011)
13. Luo, S., Leung, S., Qian, J.: An adjoint state method for numerical approximation of continuous traffic congestion equilibria. Commun. Comput. Phys. **10**, 1113–1131 (2011)
14. Pica, A.: Fast and accurate finite-difference solutions of the 3D eikonal equation parametrized in celerity. In: 67th Ann. Internat. Mtg, Soc. of Expl. Geophys, pp. 1774–1777 (1997)
15. Qian, J., Symes, W.W.: Paraxial eikonal solvers for anisotropic quasi-P traveltimes. J. Comput. Phys. **173**, 1–23 (2001)

16. Qian, J., Symes, W.W.: Adaptive finite difference method for traveltime and amplitude. Geophysics **67**, 167–176 (2002)
17. Qian, J., Symes, W.W.: Finite-difference quasi-P traveltimes for anisotropic media. Geophysics **67**, 147–155 (2002)
18. Qian, J., Zhang, Y.-T., Zhao, H.-K.: A fast sweeping methods for static convex Hamilton-Jacobi equations. J. Sci. Comput. **31**(1/2), 237–271 (2007)
19. Qian, J., Zhang, Y.-T., Zhao, H.-K.: Fast sweeping methods for eiknonal equations on triangulated meshes. SIAM J. Numer. Anal. **45**, 83–107 (2007)
20. Rouy, E., Tourin, A.: A viscosity solutions approach to shape-from-shading. SIAM J. Numer. Anal. **29**, 867–884 (1992)
21. Tsai, Y., Cheng, L.-T., Osher, S., Zhao, H.-K.: Fast sweeping algorithms for a class of Hamilton-Jacobi equations. SIAM J. Numer. Anal. **41**, 673–694 (2003)
22. Tsitsiklis, J.N.: Efficient algorithms for globally optimal trajectories. IEEE Trans. Autom. Control **40**, 1528–1538 (1995)
23. Zhang, L., Rector, J.W., Hoversten, G.M.: Eikonal solver in the celerity domain. Geophys. J. Int. **162**, 1–8 (2005)
24. Zhang, Y.-T., Zhao, H.-K., Qian, J.: High order fast sweeping methods for static Hamilton-Jacobi equations. J. Sci. Comput. **29**, 25–56 (2006)
25. Zhao, H.-K.: A fast sweeping method for eikonal equations. Math. Comput. **74**, 603–627 (2005)
26. Zhao, H.-K.: Parallel implementations of the fast sweeping method. J. Comput. Math. **25**, 421–429 (2007)