

Uniformly accurate discontinuous Galerkin fast sweeping methods for Eikonal equations

Yong-Tao Zhang¹, Shanqin Chen², Fengyan Li³, Hongkai Zhao⁴, Chi-Wang Shu⁵

ABSTRACT

In [F. Li, C.-W. Shu, Y.-T. Zhang, H. Zhao, Journal of Computational Physics 227 (2008) 8191-8208], we developed a fast sweeping method based on a hybrid local solver which is a combination of a discontinuous Galerkin (DG) finite element solver and a first order finite difference solver for Eikonal equations. The method has second order accuracy in the L^1 norm and a very fast convergence speed, but only first order accuracy in the L^∞ norm for the general cases. This is an obstacle to the design of higher order DG fast sweeping methods. In this paper, we overcome this problem by developing uniformly accurate DG fast sweeping methods for solving Eikonal equations. In order to achieve both high order accuracy and fast convergence rate (linear computational complexity), the central question is how to enforce the causality property of Eikonal equations in the compact DG local solver. We design novel causality indicators which guide the information flow directions for the DG local solver. The values of these indicators are initially provided by the first order finite difference fast sweeping method, and they are updated during iterations along with the solution. The use of causality indicators (1) allows us to compute the solution more efficiently, i.e., to only compute the solution at cells whose current causality information is consistent with the current sweeping directions, (2) is more robust than using the solution itself near singularities, such as shocks, (3) can guide the DG local solver to provide a solution for all elements of the computational mesh without switching back to the first order finite difference solver as in our previous work. We observe both a uniform second order accuracy in the L^∞ norm (in smooth regions) and the fast convergence speed (linear computational complexity) in the numerical examples.

¹Department of Mathematics, University of Notre Dame, Notre Dame, IN 46556-4618, USA. E-mail: yzhang10@nd.edu. Research partially supported by NSF grant DMS-0810413 and Oak Ridge Associated Universities (ORAU) Ralph E. Powe Junior Faculty Enhancement Award.

²Department of Mathematical Sciences, Indiana University South Bend, South Bend, IN 46634, USA. E-mail: chen39@iusb.edu

³Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY, 12180, USA. E-mail: lif@rpi.edu. Research partially supported by NSF grant DMS-0652481, NSF CAREER award DMS-0847241 and an Alfred P. Sloan Research Fellowship.

⁴Department of Mathematics, University of California, Irvine, CA 92697-3875, USA. E-mail: zhao@math.uci.edu

⁵Division of Applied Mathematics, Brown University, Providence, RI 02912, USA. E-mail: shu@dam.brown.edu. Research partially supported by DOE grant DE-FG02-08ER25863 and NSF grant DMS-0809086.

Key Words: fast sweeping methods, discontinuous Galerkin methods, uniform accuracy, static Hamilton-Jacobi equations, Eikonal equations

1 Introduction

In this paper, we consider numerical solutions of the Eikonal equations

$$\begin{cases} |\nabla\phi(\mathbf{x})| = f(\mathbf{x}), & \mathbf{x} \in \Omega \setminus \Gamma, \\ \phi(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Gamma \subset \Omega, \end{cases} \quad (1.1)$$

where $f(\mathbf{x})$ is a positive function and $f(\mathbf{x})$ and $g(\mathbf{x})$ are Lipschitz continuous, Ω is a computational domain in R^d and Γ is a subset of Ω . The Eikonal equations (1.1) form a very important class of the static Hamilton-Jacobi equations

$$\begin{cases} H(\nabla\phi(\mathbf{x}), \mathbf{x}) = 0, & \mathbf{x} \in \Omega \setminus \Gamma, \\ \phi(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Gamma \subset \Omega, \end{cases} \quad (1.2)$$

where the Hamiltonian H is Lipschitz continuous and is often nonlinear. The concept of viscosity solutions for Hamilton-Jacobi (H-J) equations was introduced in [4]. The numerical calculations of static Hamilton-Jacobi equations appear in many applications, such as optimal control, differential games, image processing and computer vision, geometric optics, seismic waves, crystal growth, robotic navigation, level set methods, etc.

A class of numerical methods for static H-J equations is to treat the problem as a stationary boundary value problem: discretize the problem into a system of nonlinear equations and design an efficient numerical algorithm to solve the system. Among such methods are the fast marching method and the fast sweeping method. The fast marching method [22, 18, 6, 19, 20] is based on the Dijkstra's algorithm [5]. The solution is updated by following the causality in a sequential way; i.e., the solution is updated pointwise in the order that the solution is strictly increasing (decreasing). Two essential ingredients are needed in the fast marching algorithm: an upwind difference scheme and a heap-sort algorithm. The resulting complexity of the fast marching method is of order $O(N \log N)$ for N grid points, where the $\log N$ factor comes from the heap-sort algorithm. Recently, an $O(N)$ implementation of the fast marching algorithm for solving Eikonal equations is developed in [24]. The improvement is achieved by introducing the untidy priority queue, obtained via a quantization of the priorities in the marching computation. However, the numerical solution obtained by this algorithm is not an exact solution to the discrete system due to quantization. The extra error introduced must be controlled to be at the same order as the numerical error of the discretization scheme. It is shown in [15] that the complexity of this algorithm is $O(f_{\max}/f_{\min}N)$ in order to achieve an accuracy that is independent of the variation of $f(\mathbf{x})$. In the fast sweeping method [1, 28, 21, 9, 27, 11, 26, 25, 13, 14, 10], Gauss-Seidel iterations with alternating orderings are

combined with upwind finite differences. In contrast to the fast marching method, the fast sweeping method follows the causality along characteristics in a parallel way; i.e., all characteristics are divided into a finite number of groups according to their directions and each Gauss-Seidel iteration with a specific sweeping ordering covers a group of characteristics simultaneously; no heap-sort is needed. The fast sweeping method is optimal in the sense that the number of iterations for the convergence is independent of the total number of grid points N [27], so that the complexity of the algorithm is $O(N)$, although the constant in the complexity depends on the equation. The algorithm is extremely simple to implement. Moreover, the iterative framework is more flexible for general equations and high order methods.

The high order finite difference type fast sweeping method developed in [26] is based on high order WENO approximations. It provides a quite general framework, and is easy to incorporate any order of accuracy and any type of numerical Hamiltonian into the framework. For example, the fifth order version was developed recently in [23, 17]. Much faster convergence speed than that of the time-marching approach can be achieved. Due to the wide stencil of the high order finite difference approximation to the derivatives, some downwind information is used and the computational complexity of high order finite difference type fast sweeping methods is slightly more than linear.

Discontinuous Galerkin (DG) methods, on the other hand, can achieve high order accuracy by using very compact stencil. The DG method is a class of finite element methods, using discontinuous piecewise polynomials as approximations for the solution and test functions [3]. The first DG fast sweeping method was developed in [12] for solving the Eikonal equations. The local solver is based on the P^1 (piecewise-linear) version of a DG method developed in [2] for directly solving the time-dependent H-J equations. The causality property of the Eikonal equations is incorporated into the flux of the DG solver according to a similar procedure as the first order finite difference fast sweeping method [27], by identifying the cell averages in the DG solutions as the point values in the finite difference scheme. The causality condition enforced this way leads to a fast convergence of this DG sweeping method, however the DG local solver can *not* provide a solution for all cells. In [12], a hybrid DG local solver is proposed to resolve this issue, i.e., in those cells where the second order DG local solver can not provide a solution, the first order finite difference type Godunov scheme [27] is used. As a result, the method in [12] has second order accuracy in the L^1 norm and a very fast convergence speed, but in general the scheme only has first order accuracy in the L^∞ norm. This is an obstacle to the design of higher order DG fast sweeping methods.

In this paper, we overcome this difficulty and develop uniformly accurate DG fast sweeping methods on general cartesian meshes. In order to achieve both high order accuracy and fast convergence rate (i.e. linear computational complexity) in the DG fast sweeping methods, the central question is how to enforce the causality property of Eikonal equations in the compact DG local solver. We design novel causality indicators which guide the information flow directions for the DG local solver. The values of these indicators are initially provided by the first order finite difference fast sweeping method, and they are updated during iterations along with the solution. The use of causality indicators allows us to compute the solution more efficiently, i.e., to only compute the solution at cells whose current causality information is consistent with the current sweeping directions, and it is more robust than using the solution itself near singularities, such as shocks. The resulting algorithm can provide a solution of the DG local solver for all cells of the computational mesh without switching back to the first order finite difference solver as in [12]. Hence the numerical values on all cells after the iterations converge are the solution of the DG scheme. Both a uniform second order accuracy in the L^∞ norm (in smooth regions) and the linear computational complexity are obtained.

The rest of the paper is organized as follows. The detailed algorithm is described in Section 2. In Section 3 we provide numerical examples to show the uniform accuracy and linear computational complexity of the proposed algorithm. Concluding remarks are given in Section 4.

2 Uniformly accurate DG fast sweeping methods

In this section, we design uniformly accurate DG fast sweeping methods for the Eikonal equations (1.1). For simplicity we consider the two dimensional problems. The extension to higher dimensions is straightforward.

We first construct a cartesian mesh $\Omega_h = \cup_{1 \leq i \leq N, 1 \leq j \leq M} I_{ij}$ covering the computational domain Ω , where $I_{ij} = I_i \times J_j$ and $I_i = [x_{i-1/2}, x_{i+1/2}]$, $J_j = [y_{j-1/2}, y_{j+1/2}]$. The centers of I_i , J_j are denoted by $x_i = \frac{1}{2}(x_{i-1/2} + x_{i+1/2})$ and $y_j = \frac{1}{2}(y_{j-1/2} + y_{j+1/2})$, and the sizes are denoted by $h_i = x_{i+1/2} - x_{i-1/2}$, $l_j = y_{j+1/2} - y_{j-1/2}$. The centers of the cells I_{ij} form a grid $\Theta_h = \{(x_i, y_j), 1 \leq i \leq N, 1 \leq j \leq M\}$. The grid Θ_h is called a dual mesh of Ω_h .

We present the algorithm on a general cartesian mesh. The important components of the proposed algorithm are described separately below.

2.1 Initial causality determination

To achieve the fast convergence in the fast sweeping methods, a key step is to reliably determine the causality for the nonlinear Eikonal equation (1.1). We propose to determine the causality initially by the first order finite difference fast sweeping method [27]. The algorithm will be formulated on a general cartesian mesh.

We identify the cell I_{ij} of Ω_h by its center (x_i, y_j) , which is a grid point of Θ_h . ϕ_{ij} is used to denote the numerical solution of the first order finite difference fast sweeping method for (1.1) at (x_i, y_j) , and $f_{ij} \triangleq f(x_i, y_j)$. We assign two integer flags to each cell I_{ij} of Ω_h to indicate the information flow directions, denoted by caux_{ij} and cauy_{ij} , which are called the **causality indicators** of the cell I_{ij} . $\text{caux}_{ij}=0$ indicates that in the x -direction, the information is propagating from the left neighboring cell $I_{i-1,j}$ to the cell I_{ij} , while $\text{caux}_{ij}=1$ indicates that the information is propagating from the right neighboring cell $I_{i+1,j}$ to the cell I_{ij} . Similarly, $\text{cauy}_{ij}=0$ indicates that in the y -direction, the information is propagating from the bottom neighboring cell $I_{i,j-1}$ to the cell I_{ij} , while $\text{cauy}_{ij}=1$ indicates that the information is propagating from the top neighboring cell $I_{i,j+1}$ to the cell I_{ij} . If there is no information flowing into I_{ij} from the x or y -direction, then we set the flag of that direction to be 10, i.e., $\text{caux}_{ij}=10$ or $\text{cauy}_{ij}=10$. We perform the first order finite difference (FD) fast sweeping method on the dual mesh Θ_h to obtain the information flow pattern and record it in the arrays $\text{flagx}(i, j)$ and $\text{flagy}(i, j)$, for $1 \leq i \leq N, 1 \leq j \leq M$.

On the grid Θ_h , the PDE (1.1) is discretized as

$$\left[\left(\frac{\phi_{ij} - a}{r_1} \right)^+ \right]^2 + \left[\left(\frac{\phi_{ij} - b}{r_2} \right)^+ \right]^2 = 1, \quad (2.1)$$

where a, b, r_1 and r_2 are determined by the causality. The implementation is as follows. At interior grid points $(x_i, y_j) : i = 2, 3, \dots, N-1, j = 2, 3, \dots, M-1$, denote the grid sizes around the grid point (x_i, y_j) by $d_l \triangleq x_i - x_{i-1}$, $d_r \triangleq x_{i+1} - x_i$, $d_b \triangleq y_j - y_{j-1}$, $d_t \triangleq y_{j+1} - y_j$.

If $\phi_{i-1,j} + d_l \cdot f_{ij} < \phi_{i+1,j} + d_r \cdot f_{ij}$, then

$$a = \phi_{i-1,j}, \quad r_1 = d_l \cdot f_{ij};$$

$$\text{caux}_{ij} = 0;$$

else

$$a = \phi_{i+1,j}, \quad r_1 = d_r \cdot f_{ij};$$

$$\text{caux}_{ij} = 1.$$

Similarly,

If $\phi_{i,j-1} + d_b \cdot f_{ij} < \phi_{i,j+1} + d_t \cdot f_{ij}$, then

$$b = \phi_{i,j-1}, \quad r_2 = d_b \cdot f_{ij};$$

$$\text{cauy}_{ij} = 0;$$

else

$$b = \phi_{i,j+1}, \quad r_2 = d_t \cdot f_{ij};$$

$$\text{cauy}_{ij} = 1.$$

At the boundary of the computational domain, one sided difference is used. Namely, at the left boundary $i = 1$, we take $a = \phi_{2,j}$, $r_1 = d_r \cdot f_{1j}$, $\text{caux}_{1j} = 1$; at the right boundary $i = N$, we take $a = \phi_{N-1,j}$, $r_1 = d_l \cdot f_{Nj}$, $\text{caux}_{Nj} = 0$; at the top boundary $j = M$, we take $b = \phi_{i,M-1}$, $r_2 = d_b \cdot f_{iM}$, $\text{cauy}_{iM} = 0$; at the bottom boundary $j = 1$, we take $b = \phi_{i,2}$, $r_2 = d_t \cdot f_{i1}$, $\text{cauy}_{i1} = 1$. Denote $s_1 \triangleq r_1/f_{ij}$ and $s_2 \triangleq r_2/f_{ij}$, then the unique solution for the quadratic equation (2.1) is

$$\phi_{ij} = \begin{cases} \frac{as_2^2 + bs_1^2 + s_1s_2\sqrt{r_1^2 + r_2^2 - (a-b)^2}}{s_1^2 + s_2^2}, & \text{if } -r_2 < b-a < r_1, \\ b + r_2, & \text{if } b-a \leq -r_2, \\ a + r_1, & \text{if } b-a \geq r_1. \end{cases} \quad (2.2)$$

The detailed procedure of using the first order FD fast sweeping method to determine the initial causality for the DG solver is given below.

Procedure I: Determination of the initial causality for the DG solver.

1. Initialization:

(a) According to the boundary condition $\phi(\mathbf{x}) = g(\mathbf{x})$, $\mathbf{x} \in \Gamma$, assign exact values or interpolated values at grid points in or near Γ . These values are fixed during iterations. Large positive values are used as initial guess at all other grid points, and these values should be larger than the maximum of the true solution, and they will be updated in later iterations.

(b) Initialize the causality arrays: $\text{flag}_x(i, j) = 10$, $\text{flag}_y(i, j) = 10$, for $1 \leq i \leq N, 1 \leq j \leq M$.

2. Iterations: solve the discretized nonlinear system (2.1) by Gauss-Seidel iterations with four alternating direction sweepings:

$$(1) \ i = 1 : N, \ j = 1 : M;$$

$$(2) \ i = N : 1, \ j = 1 : M;$$

$$(3) \ i = N : 1, \ j = M : 1;$$

$$(4) \ i = 1 : N, \ j = M : 1.$$

Equation (2.2) is used to solve (2.1), and the current values of the neighbors of the grid point (i, j) are used due to the Gauss-Seidel philosophy. If the solution ϕ_{ij} of (2.1) is smaller than the current value at the grid point (i, j) , then we update its value $\phi_{ij}^{\text{new}} = \phi_{ij}$ and update the causality arrays:

$$\begin{aligned} &\text{If } -r_2 < b - a < r_1, \text{ then} \\ &\quad \text{flagx}(i,j) = \text{caux}_{ij}, \text{ flagy}(i,j) = \text{cauy}_{ij}; \\ &\text{If } b - a \leq -r_2, \text{ then} \\ &\quad \text{flagx}(i,j) = 10, \text{ flagy}(i,j) = \text{cauy}_{ij}; \\ &\text{If } b - a \geq r_1, \text{ then} \\ &\quad \text{flagx}(i,j) = \text{caux}_{ij}, \text{ flagy}(i,j) = 10. \end{aligned}$$

3. Convergence: if

$$\|\phi^{\text{new}} - \phi^{\text{old}}\|_{L^\infty} \leq \delta,$$

where δ is a given convergence threshold value and $\|\cdot\|_{L^\infty}$ denotes the L^∞ norm, the iteration converges and stops. We take $\delta = 10^{-11}$ in all numerical experiments of this paper. ■

2.2 DG local solver

In this subsection, we describe a piecewise linear DG local solver for the Eikonal equations (1.1) on a general cartesian mesh. This local solver is based on a DG method developed recently for directly solving the time-dependent Hamilton-Jacobi equations [2]. The local solver has a similar form as the one in our previous work [12]. We will emphasize their differences in the following.

On the cartesian mesh Ω_h , we define the piecewise linear finite element space as

$$V_h^1 = \{v : v|_{I_{ij}} \in P^1(I_{ij}), \ i = 1, \dots, N, \ j = 1, \dots, M\} \quad (2.3)$$

where $P^1(I_{ij})$ denotes all linear polynomials on I_{ij} . As in [2, 12], the DG scheme for the Eikonal

equations (1.1) is defined as: find $\phi_h(x, y) \in V_h^1$, such that

$$\begin{aligned}
\int_{I_{ij}} |\nabla \phi_h(x, y)| v_h(x, y) dx dy &+ \alpha_{l,ij} \int_{y_{j-1/2}}^{y_{j+1/2}} [\phi_h](x_{i-\frac{1}{2}}, y) v_h(x_{i-\frac{1}{2}}^+, y) dy \\
&+ \alpha_{b,ij} \int_{x_{i-1/2}}^{x_{i+1/2}} [\phi_h](x, y_{j-\frac{1}{2}}) v_h(x, y_{j-\frac{1}{2}}^+) dx \\
&+ \alpha_{r,ij} \int_{y_{j-1/2}}^{y_{j+1/2}} [\phi_h](x_{i+\frac{1}{2}}, y) v_h(x_{i+\frac{1}{2}}^-, y) dy \\
&+ \alpha_{t,ij} \int_{x_{i-1/2}}^{x_{i+1/2}} [\phi_h](x, y_{j+\frac{1}{2}}) v_h(x, y_{j+\frac{1}{2}}^-) dx \\
&= \int_{I_{ij}} f(x, y) v_h(x, y) dx dy, \quad i = 1, \dots, N, \quad j = 1, \dots, M \quad (2.4)
\end{aligned}$$

holds for any $v_h(x, y) \in V_h^1$. Here $[\phi_h]$ denotes the jump of ϕ_h across the cell interface. $\alpha_{l,ij}, \alpha_{b,ij}, \alpha_{r,ij}, \alpha_{t,ij}$ are local constants which depend on the numerical solutions in the neighboring cells of I_{ij} and the causality of the Eikonal equation. They are called **local causality constants** and will be discussed in detail in Section 2.2.1.

Remark: The way to calculate local causality constants in this local solver (2.4) is different from that in [12], which will be described in the following subsection 2.2.1.

2.2.1 Calculations of local causality constants

The linear polynomial $\phi_h(x, y)$ on I_{ij} can be represented by $\phi_h|_{I_{ij}} = \phi_{ij} + u_{ij}\xi_i + v_{ij}\eta_j$, where $\xi_i = \frac{x-x_i}{h_i}$ and $\eta_j = \frac{y-y_j}{l_j}$. Let us denote $H_1 \triangleq \frac{\partial H}{\partial \phi_x}$ and $H_2 \triangleq \frac{\partial H}{\partial \phi_y}$. The constants $\alpha_{l,ij}, \alpha_{b,ij}, \alpha_{r,ij}, \alpha_{t,ij}$ are approximations of $H_1(\nabla \phi_h)$ and $H_2(\nabla \phi_h)$ in the four neighboring cells of I_{ij} , furthermore, the causality indicators are used to define them.

$$\alpha_{l,ij} = \begin{cases} \max(0, H_1(\nabla \phi_h)|_{I_{i-1,j}}) = \max\left(0, \frac{u_{i-1,j}l_j}{\sqrt{(u_{i-1,j}l_j)^2 + (v_{i-1,j}h_{i-1})^2}}\right), & \text{If flagx(i,j)=0 and } (u_{i-1,j}l_j)^2 + (v_{i-1,j}h_{i-1})^2 \neq 0; \\ \text{skip current cell,} & \text{If flagx(i,j)=0 and } u_{i-1,j} = v_{i-1,j} = 0; \\ 0, & \text{If flagx(i,j)=1 or flagx(i,j)=10.} \end{cases} \quad (2.5)$$

For the first case, if we have $\max(0, H_1(\nabla \phi_h)|_{I_{i-1,j}}) = 0$, then we need to correct the current causality indicator $\text{flagx}(i,j)$ to be $\text{flagx}(i,j)=10$. By doing this, we shut down this information flow direction of this cell in the current iteration. This could happen when the initial causality determined by the first order fast sweeping method contradicts with the causality obtained by the second order DG local solver. This often happens near shocks. For example, if a cell is at a shock location, the directions of characteristics are from left to right on the left of the shock and the

directions of characteristics are from right to left on the right of the shock. In addition, skipping current cell due to lack of proper updated information may save computational cost. Likewise,

$$\alpha_{r,ij} = \begin{cases} \min(0, H_1(\nabla\phi_h)|_{I_{i+1,j}}) = \min\left(0, \frac{u_{i+1,j}l_j}{\sqrt{(u_{i+1,j}l_j)^2 + (v_{i+1,j}h_{i+1})^2}}\right), & \text{If flagx(i,j)=1 and } (u_{i+1,j}l_j)^2 + (v_{i+1,j}h_{i+1})^2 \neq 0; \\ \text{skip current cell,} & \text{If flagx(i,j)=1 and } u_{i+1,j} = v_{i+1,j} = 0; \\ 0, & \text{If flagx(i,j)=0 or flagx(i,j)=10.} \end{cases} \quad (2.6)$$

For the first case, if we have $\min(0, H_1(\nabla\phi_h)|_{I_{i+1,j}}) = 0$, then we need to correct the current causality indicator $\text{flagx}(i,j)$ to be $\text{flagx}(i,j)=10$. Similarly,

$$\alpha_{b,ij} = \begin{cases} \max(0, H_2(\nabla\phi_h)|_{I_{i,j-1}}) = \max\left(0, \frac{v_{i,j-1}h_i}{\sqrt{(u_{i,j-1}l_{j-1})^2 + (v_{i,j-1}h_i)^2}}\right), & \text{If flagy(i,j)=0 and } (u_{i,j-1}l_{j-1})^2 + (v_{i,j-1}h_i)^2 \neq 0; \\ \text{skip current cell,} & \text{If flagy(i,j)=0 and } u_{i,j-1} = v_{i,j-1} = 0; \\ 0, & \text{If flagy(i,j)=1 or flagy(i,j)=10.} \end{cases} \quad (2.7)$$

For the first case, if we have $\max(0, H_2(\nabla\phi_h)|_{I_{i,j-1}}) = 0$, then we need to correct the current causality indicator $\text{flagy}(i,j)$ to be $\text{flagy}(i,j)=10$. Finally,

$$\alpha_{t,ij} = \begin{cases} \min(0, H_2(\nabla\phi_h)|_{I_{i,j+1}}) = \min\left(0, \frac{v_{i,j+1}h_i}{\sqrt{(u_{i,j+1}l_{j+1})^2 + (v_{i,j+1}h_i)^2}}\right), & \text{If flagy(i,j)=1 and } (u_{i,j+1}l_{j+1})^2 + (v_{i,j+1}h_i)^2 \neq 0; \\ \text{skip current cell,} & \text{If flagy(i,j)=1 and } u_{i,j+1} = v_{i,j+1} = 0; \\ 0, & \text{If flagy(i,j)=0 or flagy(i,j)=10.} \end{cases} \quad (2.8)$$

For the first case, if we have $\min(0, H_2(\nabla\phi_h)|_{I_{i,j+1}}) = 0$, then we need to correct the current causality indicator $\text{flagy}(i,j)$ to be $\text{flagy}(i,j)=10$. If both $\text{flagx}(i,j)=10$ and $\text{flagy}(i,j)=10$, we will skip the current cell in the current iteration.

Remark: We use causality indicators as guiding conditions to define local causality constants in the DG local solver (2.4). The local causality constants provide important ‘‘upwind’’ information in the flux of the DG local solver. Hence the DG local solver defined in this paper has different flux from that in [12]. Moreover, the local causality constants defined in this paper have the property ≤ 1 , which is consistent with the property of $H_1(\nabla\phi_h)$ and $H_2(\nabla\phi_h)$.

2.2.2 The quadratic system

On any given element I_{ij} , by taking $v_h = 1$, ξ_i , η_j , the DG formulation (2.4) is converted from the integral form to a quadratic system:

$$\sqrt{u_{ij}^2 + v_{ij}^2 r_{ij}^2} + e_{ij} \phi_{ij} + \beta_{ij} u_{ij} + \lambda_{ij} v_{ij} = R_{1,ij} \quad (2.9)$$

$$12\beta_{ij} \phi_{ij} + d_{ij} u_{ij} = R_{2,ij} \quad (2.10)$$

$$12\lambda_{ij} \phi_{ij} + g_{ij} v_{ij} = R_{3,ij} \quad (2.11)$$

where

$$\begin{aligned} r_{ij} &= \frac{h_i}{l_j} \\ \beta_{ij} &= -\frac{1}{2}(\alpha_{l,ij} + \alpha_{r,ij}), \quad \lambda_{ij} = -\frac{1}{2}r_{ij}(\alpha_{b,ij} + \alpha_{t,ij}) \\ e_{ij} &= \alpha_{l,ij} + \alpha_{b,ij}r_{ij} - (\alpha_{r,ij} + \alpha_{t,ij}r_{ij}) \\ d_{ij} &= 3\alpha_{l,ij} + \alpha_{b,ij}r_{ij} - 3\alpha_{r,ij} - \alpha_{t,ij}r_{ij} \\ g_{ij} &= (\alpha_{l,ij} - \alpha_{r,ij}) + 3r_{ij}(\alpha_{b,ij} - \alpha_{t,ij}) \end{aligned}$$

and

$$\begin{aligned} R_{1,ij} &= \frac{1}{l_j} \int_{I_{ij}} f(x, y) dx dy - \alpha_{r,ij}(\phi_{i+1,j} - \frac{1}{2}u_{i+1,j}) + \alpha_{l,ij}(\phi_{i-1,j} + \frac{1}{2}u_{i-1,j}) \\ &\quad - \alpha_{t,ij}r_{ij}(\phi_{i,j+1} - \frac{1}{2}v_{i,j+1}) + \alpha_{b,ij}r_{ij}(\phi_{i,j-1} + \frac{1}{2}v_{i,j-1}) \\ R_{2,ij} &= \frac{12}{l_j} \int_{I_{ij}} f(x, y) \xi_i dx dy - 6\alpha_{r,ij}(\phi_{i+1,j} - \frac{1}{2}u_{i+1,j}) - 6\alpha_{l,ij}(\phi_{i-1,j} + \frac{1}{2}u_{i-1,j}) - \alpha_{t,ij}r_{ij}u_{i,j+1} + \alpha_{b,ij}r_{ij}u_{i,j-1} \\ R_{3,ij} &= \frac{12}{l_j} \int_{I_{ij}} f(x, y) \eta_j dx dy - 6\alpha_{t,ij}r_{ij}(\phi_{i,j+1} - \frac{1}{2}v_{i,j+1}) - 6\alpha_{b,ij}r_{ij}(\phi_{i,j-1} + \frac{1}{2}v_{i,j-1}) - \alpha_{r,ij}v_{i+1,j} + \alpha_{l,ij}v_{i-1,j} \end{aligned}$$

To solve this quadratic system (2.9)-(2.11), we adopt the Gauss-Seidel philosophy, namely, we use the current numerical values of neighboring cells of the cell I_{ij} . Based on the values of causality indicators, we could have the following two scenarios.

1. $\text{flag}_x(i,j) \neq 10$

In this case, $\beta_{ij} \neq 0$ and $g_{ij} \neq 0$. From (2.10) and (2.11), we have

$$\begin{aligned} \phi_{ij} &= a_{ij} + b_{ij}u_{ij}, \\ v_{ij} &= c_{ij} + t_{ij}u_{ij}, \end{aligned} \quad (2.12)$$

where

$$a_{ij} = \frac{R_{2,ij}}{12\beta_{ij}}, \quad b_{ij} = -\frac{d_{ij}}{12\beta_{ij}}, \quad c_{ij} = \frac{R_{3,ij}\beta_{ij} - \lambda_{ij}R_{2,ij}}{\beta_{ij}g_{ij}}, \quad t_{ij} = \frac{\lambda_{ij}d_{ij}}{\beta_{ij}g_{ij}}.$$

Substitute (2.12) into (2.9), we obtain a quadratic equation

$$a_6 u_{ij}^2 + a_7 u_{ij} + a_8 = 0, \quad (2.13)$$

where

$$\begin{aligned} a_6 &= a_1 - a_4^2, \quad a_7 = a_2 - 2a_4 a_5, \quad a_8 = a_3 - a_5^2, \\ a_1 &= 1 + (t_{ij} r_{ij})^2, \quad a_2 = 2c_{ij} t_{ij} r_{ij}^2, \quad a_3 = (c_{ij} r_{ij})^2, \\ a_4 &= -(e_{ij} b_{ij} + \beta_{ij} + \lambda_{ij} t_{ij}), \quad a_5 = R_{1,ij} - e_{ij} a_{ij} - \lambda_{ij} c_{ij}. \end{aligned}$$

If the quadratic equation (2.13) gives only one real solution \bar{u} , then we update the current value $u_{ij} = \bar{u}$; if it gives two real solutions \bar{u}_1 and \bar{u}_2 , then we update the solution according to the following rule:

$$\text{if flagx(i,j) = 0,} \quad \text{then } u_{ij} = \max(\bar{u}_1, \bar{u}_2);$$

$$\text{if flagx(i,j) = 1,} \quad \text{then } u_{ij} = \min(\bar{u}_1, \bar{u}_2).$$

The updated values for v_{ij} and ϕ_{ij} can be obtained by (2.12). If the quadratic equation (2.13) has no real solution, we do *not* update the current values of u_{ij} , v_{ij} and ϕ_{ij} , and skip the current cell.

2. flagx(i,j) = 10

In this case, we only use the information in the y -direction. $\alpha_{l,ij} = \alpha_{r,ij} = 0$, so from (2.10)-(2.11), we have

$$\begin{aligned} u_{ij} &= \frac{R_{2,ij}}{d_{ij}}, \\ \phi_{ij} &= a_{ij} + b_{ij} v_{ij}, \end{aligned} \quad (2.14)$$

where

$$a_{ij} = \frac{R_{3,ij}}{12\lambda_{ij}}, \quad b_{ij} = -\frac{g_{ij}}{12\lambda_{ij}}.$$

Substituting (2.14) into (2.9), we obtain a quadratic equation for v_{ij}

$$a_6 v_{ij}^2 + a_7 v_{ij} + a_8 = 0, \quad (2.15)$$

where

$$\begin{aligned} a_6 &= r_{ij}^2 - a_3^2, & a_7 &= -2a_2a_3, & a_8 &= a_1^2 - a_2^2, \\ a_1 &= \frac{R_{2,ij}}{d_{ij}}, & a_2 &= R_{1,ij} - e_{ij}a_{ij}, & a_3 &= -(\lambda_{ij} + e_{ij}b_{ij}). \end{aligned}$$

If the quadratic equation (2.15) gives only one real solution \bar{v} , then we update the current value $v_{ij} = \bar{v}$; if it gives two real solutions \bar{v}_1 and \bar{v}_2 , then we update the solution according to the following rule:

$$\begin{aligned} \text{if } \text{flagy}(i,j) = 0, & \quad \text{then } v_{ij} = \max(\bar{v}_1, \bar{v}_2); \\ \text{if } \text{flagy}(i,j) = 1, & \quad \text{then } v_{ij} = \min(\bar{v}_1, \bar{v}_2). \end{aligned}$$

The updated values for u_{ij} and ϕ_{ij} can be obtained by (2.14). If the quadratic equation (2.15) has no real solution, we do *not* update the current values of u_{ij} , v_{ij} and ϕ_{ij} , and skip the current cell.

Remark: As described in this subsection, when there are two solutions for u_{ij} (or v_{ij}) in the quadratic equation, we choose the one which is consistent with the current causality indicator values and “more upwind”. This way to pick up values gives correct numerical results in the numerical examples.

2.2.3 Update of causality arrays

If the values of u_{ij} , v_{ij} and ϕ_{ij} have been updated by the DG local solver, then we need to make the current values of causality indicators in the neighboring cells of I_{ij} consistent with the current information flow directions determined by the DG local solver. Through numerical experiments, we found that we must consider the causality information on both sides of each direction of the cells whose causality arrays may be updated. The detailed algorithm is given in the following.

In the x -direction, if ($u_{ij} > 0$ and $i < n$): this indicates that the information in the cell (i, j) is propagating to the right cell (i+1, j) and it is possible that we need to update $\text{flagx}(i+1,j)$. If the cell (i+1, j) is a boundary cell (i.e. a cell around Γ which has pre-assigned values and these values are fixed during iterations), then we do *not* need to update $\text{flagx}(i+1,j)$. Otherwise we need to look at the causality information at the right hand side of the cell (i+1, j). If the cell (i+1, j) happens to be at the boundary of the computational domain, then there is no causality information at the right hand side of the cell (i+1, j) and we just update $\text{flagx}(i+1,j) = 0$. If the cell (i+1, j) is an interior cell, then there is causality information at its right neighboring cell (i+2, j) which we need

to consider. Our numerical experiments indicate that we should update $\text{flagx}(i+1,j)$ if and only if the current numerical values on cell $(i+2, j)$ have been provided by the DG local solver (i.e., not the initial iteration values), and the “global” causality between the cell (i, j) and the cell $(i+2, j)$ is consistent with the current “local” causality for the cell $(i+1, j)$. Here the current “local” causality is just the information propagation direction indicated by the DG solution in the current iteration step and current cell. In this case, it is indicated by $u_{ij} > 0$. The “global” causality between the cell (i, j) and the cell $(i+2, j)$ is motivated by the “first arrival time” used in the first order fast sweeping method, which is defined as follows. Denote $d_l \triangleq x_{i+1} - x_i$, $d_r \triangleq x_{i+2} - x_{i+1}$,

if ((the values on cell $(i+2, j)$ are from DG solver) .and. $\phi_{ij} + d_l \cdot f_{i+1,j} < \phi_{i+2,j} + d_r \cdot f_{i+1,j}$), then

$$\text{flagx}(i+1,j) = 0;$$

otherwise we do *not* update $\text{flagx}(i+1,j)$.

Similarly if ($u_{ij} < 0$.and. $i > 1$): this indicates that the information in the cell (i, j) is propagating to the left cell $(i-1, j)$ and it is possible that we need to update $\text{flagx}(i-1,j)$. If the cell $(i-1, j)$ is a boundary cell, then we do *not* need to update $\text{flagx}(i-1,j)$. Otherwise if the cell $(i-1, j)$ happens to be at the boundary of the computational domain, we will update $\text{flagx}(i-1,j) = 1$. If the cell $(i-1, j)$ is an interior cell, then there is causality information at its left neighboring cell $(i-2, j)$ which we need to consider. Denote $d_l \triangleq x_{i-1} - x_{i-2}$, $d_r \triangleq x_i - x_{i-1}$,

if ((the values on cell $(i-2, j)$ are from DG solver) .and. $\phi_{ij} + d_r \cdot f_{i-1,j} < \phi_{i-2,j} + d_l \cdot f_{i-1,j}$), then

$$\text{flagx}(i-1,j) = 1;$$

otherwise we do *not* update $\text{flagx}(i-1,j)$.

Similarly in the y-direction, if ($v_{ij} > 0$.and. $j < m$): this indicates that the information in the cell (i, j) is propagating to the top cell $(i, j+1)$ and it is possible that we need to update $\text{flagy}(i,j+1)$. If the cell $(i, j+1)$ is a boundary cell, then we do *not* need to update $\text{flagy}(i,j+1)$. Otherwise if the cell $(i, j+1)$ happens to be at the boundary of the computational domain, we will update $\text{flagy}(i,j+1) = 0$. If the cell $(i, j+1)$ is an interior cell, then there is causality information at its top neighboring cell $(i, j+2)$ which we need to consider. Denote $d_b \triangleq y_{j+1} - y_j$, $d_t \triangleq y_{j+2} - y_{j+1}$,

if ((the values on cell $(i, j+2)$ are from DG solver) .and. $\phi_{ij} + d_b \cdot f_{i,j+1} < \phi_{i,j+2} + d_t \cdot f_{i,j+1}$), then

$$\text{flagy}(i,j+1) = 0;$$

otherwise we do *not* update $\text{flagy}(i,j+1)$.

If ($v_{ij} < 0$.and. $j > 1$): this indicates that the information in the cell (i, j) is propagating to the bottom cell (i, j-1) and it is possible that we need to update $\text{flagy}(i,j-1)$. If the cell (i, j-1) is a boundary cell, then we do *not* need to update $\text{flagy}(i,j-1)$. Otherwise if the cell (i, j-1) happens to be at the boundary of the computational domain, we will update $\text{flagy}(i,j-1) = 1$. If the cell (i, j-1) is an interior cell, then there is causality information at its bottom neighboring cell (i, j-2) which we need to consider. Denote $d_b \triangleq y_{j-1} - y_{j-2}$, $d_t \triangleq y_j - y_{j-1}$,

if ((the values on cell (i, j-2) are from DG solver) .and. $\phi_{ij} + d_t \cdot f_{i,j-1} < \phi_{i,j-2} + d_b \cdot f_{i,j-1}$), then

$$\text{flagy}(i,j-1) = 1;$$

otherwise we do *not* update $\text{flagy}(i,j-1)$.

2.2.4 Initialization of the DG local solver

To initialize the DG solver, we need to specify the values of ϕ_{ij} , u_{ij} and v_{ij} on the cells which are around the boundary Γ (these cells are called “boundary cells” and the values on boundary cells will be fixed during iterations). We use the least squares approximation of the exact or approximating boundary values to pre-assign the values of ϕ_{ij} , u_{ij} and v_{ij} on the boundary cells [12]. For example, if the values $\phi(x_{i\pm 1/2}, y_{j\pm 1/2})$ are given at four grid points of the boundary cell I_{ij} , then we can pre-assign the values of ϕ_{ij} , u_{ij} , v_{ij} as

$$\phi_{ij} = \frac{1}{4} (\phi(x_{i-1/2}, y_{j-1/2}) + \phi(x_{i+1/2}, y_{j-1/2}) + \phi(x_{i-1/2}, y_{j+1/2}) + \phi(x_{i+1/2}, y_{j+1/2})), \quad (2.16)$$

$$u_{ij} = \frac{1}{2} (\phi(x_{i+1/2}, y_{j-1/2}) - \phi(x_{i-1/2}, y_{j-1/2}) + \phi(x_{i+1/2}, y_{j+1/2}) - \phi(x_{i-1/2}, y_{j+1/2})), \quad (2.17)$$

$$v_{ij} = \frac{1}{2} (\phi(x_{i-1/2}, y_{j+1/2}) - \phi(x_{i-1/2}, y_{j-1/2}) + \phi(x_{i+1/2}, y_{j+1/2}) - \phi(x_{i+1/2}, y_{j-1/2})), \quad (2.18)$$

For the other non-boundary cells, the initial iteration values of ϕ_{ij} are the values from the first order fast sweeping iterations on the dual mesh Θ_h and the initial iteration values of u_{ij} and v_{ij} are zeros.

2.3 Algorithm summary

Now we summarize uniformly accurate DG fast sweeping methods in the following.

1. Determine the initial causality arrays by Procedure I in Section 2.1.
2. Initialize the DG local solver as described in Section 2.2.4.

3. Perform iterations on non-boundary cells with four alternating direction sweepings:

- (1) $i = 1 : N, j = 1 : M$: use the procedure described in Sections 2.2.1, 2.2.2 and 2.2.3 to update values ϕ_{ij}, u_{ij} and v_{ij} on the cells with $\text{flagx}(i,j) \neq 1$ and $\text{flagy}(i,j) \neq 1$, and update the causality arrays of their neighboring cells when it is needed;
- (2) $i = N : 1, j = 1 : M$: use the procedure described in Sections 2.2.1, 2.2.2 and 2.2.3 to update values ϕ_{ij}, u_{ij} and v_{ij} on the cells with $\text{flagx}(i,j) = 1$ and $\text{flagy}(i,j) \neq 1$, and update the causality arrays of their neighboring cells when it is needed;
- (3) $i = N : 1, j = M : 1$: use the procedure described in Sections 2.2.1, 2.2.2 and 2.2.3 to update values ϕ_{ij}, u_{ij} and v_{ij} on the cells with $\text{flagx}(i,j) \neq 0$ and $\text{flagy}(i,j) = 1$, and update the causality arrays of their neighboring cells when it is needed;
- (4) $i = 1 : N, j = M : 1$: use the procedure described in Sections 2.2.1, 2.2.2 and 2.2.3 to update values ϕ_{ij}, u_{ij} and v_{ij} on the cells with $\text{flagx}(i,j) = 0$ and $\text{flagy}(i,j) = 1$, and update the causality arrays of their neighboring cells when it is needed.

4. Convergence: if

$$\|\phi^{\text{new}} - \phi^{\text{old}}\|_{L^\infty} \leq \delta,$$

where δ is a given convergence threshold value, the iteration converges and stops. We take $\delta = 10^{-11}$ in all of numerical experiments of this paper. ■

Remark: The procedure of step 3 indicates that in each sweeping, only the cells whose causality indicator values are consistent with the current sweeping direction may be updated by the DG local solver. By doing this, we can save many computational costs since we exclude the cells where the correct characteristic information has not reached in the current sweeping. In step 4 of our algorithm, although we take $\delta = 10^{-11}$ as the threshold value to stop the iterations, when the convergence is achieved, we observe that the error $\|\phi^{\text{new}} - \phi^{\text{old}}\|_{L^\infty}$ has reached machine zero for all the cases in our numerical examples, except for the non-uniform mesh case of Example 6.

3 Numerical examples

In this section, a set of numerical examples will be presented for solving the Eikonal equations (1.1), and they demonstrate a uniform second order accuracy of the proposed method in smooth regions of the solutions, as well as the linear computational complexity. Numerical errors are calculated for non-boundary cells in all examples.

From the description of the algorithm in Section 2.3, we can see that in each sweeping, only the cells whose causality indicator values are consistent with the current sweeping direction may be updated by the DG local solver. Hence to measure the computational complexity accurately, we define the **effective sweeping number**:

$$\text{effective sweeping number} \triangleq \frac{\text{the total \# of times the DG local solver is executed}}{\text{the total \# of cells excluding the boundary cells}},$$

where “the DG local solver is executed” means that the subroutine for solving the quadratic system in the section 2.2.2 has been executed no matter whether the system has solutions or not.

Example 1. $\Omega = [-1, 1]^2$, $\Gamma = \{(0, 0)\}$, and

$$f(x, y) = \frac{\pi}{2} \sqrt{\sin^2\left(\frac{\pi}{2}x\right) + \sin^2\left(\frac{\pi}{2}y\right)}, \quad g(0, 0) = -2.$$

The exact solution is

$$\phi(x, y) = -\cos\left(\frac{\pi}{2}x\right) - \cos\left(\frac{\pi}{2}y\right).$$

To initialize the DG solver, we pre-assign the values of ϕ_{ij} , u_{ij} and v_{ij} on the cells in the fixed region $[-0.1, 0.1]^2$ around Γ . The results are listed in Table 3.1. We can see that only 2 effective sweepings are needed for convergence regardless of the mesh size and the error is uniformly second order both in L^1 and in L^∞ norms. If we pre-assign the values of ϕ_{ij} , u_{ij} and v_{ij} on the cells in the region $[-h, h]^2$ (h is the uniform grid size in this example) around Γ , we observe slightly lower accuracy orders as shown in Table 3.2. This is due to the degeneracy of the Eikonal equation for this example ($f(0, 0) = 0$). Similar phenomena was observed in our previous work [27, 26, 14, 12] when there is singularity at the source point.

Example 2. (Point source distance function problem). $\Omega = [-1, 1]^2$, $\Gamma = \{(0, 0)\}$ and $f(x, y) = 1$. We pre-assign values for the boundary cells in the domain $[-0.1, 0.1]^2$ based on the exact solution. The results are listed in Table 3.3. We can again observe that only 2 effective sweepings are needed for convergence regardless of the mesh size and the error is settling down to second order both in L^1 and in L^∞ norms for refined meshes.

Table 3.1: Example 1. $\Gamma = \{(0, 0)\}$ and $f(x, y) = \frac{\pi}{2} \sqrt{\sin^2(\frac{\pi}{2}x) + \sin^2(\frac{\pi}{2}y)}$. The exact solution is $\phi(x, y) = -\cos(\frac{\pi}{2}x) - \cos(\frac{\pi}{2}y)$. The values of ϕ_{ij} , u_{ij} and v_{ij} are pre-assigned on the cells in the fixed region $[-0.1, 0.1]^2$.

mesh	L^1 error	order	L^∞ error	order	eff. swp. number
20×20	8.03E-3	–	7.18E-2	–	2.00
40×40	1.17E-3	2.78	1.35E-2	2.41	2.00
80×80	2.48E-4	2.24	3.22E-3	2.07	2.00
160×160	5.62E-5	2.14	7.91E-4	2.02	2.00
320×320	1.32E-5	2.09	1.96E-4	2.01	2.00
640×640	3.20E-6	2.05	4.89E-5	2.01	2.00
1280×1280	7.83E-7	2.03	1.22E-5	2.00	2.00

Table 3.2: Example 1. $\Gamma = \{(0, 0)\}$ and $f(x, y) = \frac{\pi}{2} \sqrt{\sin^2(\frac{\pi}{2}x) + \sin^2(\frac{\pi}{2}y)}$. The exact solution is $\phi(x, y) = -\cos(\frac{\pi}{2}x) - \cos(\frac{\pi}{2}y)$. The values of ϕ_{ij} , u_{ij} and v_{ij} are pre-assigned on the cells in the region $[-h, h]^2$.

mesh	L^1 error	order	L^∞ error	order	eff. swp. number
20×20	8.03E-3	–	7.18E-2	–	2.00
40×40	2.92E-3	1.46	2.80E-2	1.36	2.00
80×80	9.62E-4	1.60	1.03E-2	1.44	2.00
160×160	2.95E-4	1.71	3.66E-3	1.50	2.00
320×320	8.62E-5	1.77	1.25E-3	1.55	2.00
640×640	2.44E-5	1.82	4.17E-4	1.59	2.00
1280×1280	6.76E-6	1.85	1.35E-4	1.62	2.00

Table 3.3: Example 2. The point source distance function problem.

mesh	L^1 error	order	L^∞ error	order	eff. swp. number
20×20	5.08E-2	–	2.78E-1	–	2.00
40×40	4.22E-3	3.59	5.11E-2	2.44	2.00
80×80	3.94E-4	3.42	9.45E-3	2.44	2.00
160×160	5.35E-5	2.88	2.03E-3	2.22	2.00
320×320	8.91E-6	2.59	4.72E-4	2.11	2.00
640×640	1.71E-6	2.38	1.14E-4	2.05	2.00
1280×1280	3.65E-7	2.23	2.79E-5	2.03	2.00

Table 3.4: Example 3. The distance function from the circle problem. Errors are measured in the smooth region, which is outside of $[-0.1, 0.1]^2$.

mesh	L^1 error	order	L^∞ error	order	eff. swp. number
20×20	8.65E-4	–	8.15E-3	–	2.00
40×40	2.34E-4	1.89	2.61E-3	1.64	2.00
80×80	5.96E-5	1.97	7.16E-4	1.86	2.00
160×160	1.50E-5	1.99	1.84E-4	1.96	2.00
320×320	3.76E-6	2.00	4.61E-5	1.99	2.00
640×640	9.42E-7	2.00	1.15E-5	2.00	2.00
1280×1280	2.36E-7	2.00	2.88E-6	2.00	2.00

Table 3.5: Example 3. The distance function from the circle problem. Errors are measured in the whole region.

mesh	L^1 error	order	L^∞ error	order	eff. swp. number
20×20	1.12E-3	–	1.82E-2	–	2.00
40×40	2.98E-4	1.90	9.15E-3	0.99	2.00
80×80	7.71E-5	1.95	4.57E-3	1.00	2.00
160×160	1.97E-5	1.96	2.28E-3	1.00	2.00
320×320	5.02E-6	1.97	1.14E-3	1.00	2.00
640×640	1.27E-6	1.98	5.70E-4	1.00	2.00
1280×1280	3.21E-7	1.99	2.85E-4	1.00	2.00

Example 3. $\Omega = [-1, 1]^2$, Γ is a circle with the center $(0, 0)$ and the radius 0.5, and $f(x, y) = 1$.

To initialize the DG solver, we pre-assign the values of ϕ_{ij} , u_{ij} and v_{ij} on the cells whose centers are within the $2h$ distance from Γ (h is the uniform grid size in this example). The results are listed in Tables 3.4 and 3.5. We observe as before that only 2 effective sweepings are needed for convergence regardless of the mesh size. The error is uniformly second order both in L^1 and in L^∞ norms if we measure it in smooth regions outside the circle center (see Table 3.4); or we have second order in L^1 and first order in L^∞ if we measure the error in the whole computational domain (the error in the boundary cells do not need to be included), see Table 3.5.

Example 4. Consider Eikonal equation (1.1) with $f(x, y) = 1$. The computational domain is $\Omega = [-1, 1] \times [-1, 1]$, and Γ consists of two circles of equal radius 0.3 with centers located at $(-0.5, -0.5)$ and $(0.5, 0.5)$, respectively. The exact solution is the distance function to Γ , i.e.

$$\phi(x, y) = \min(|\sqrt{(x - 0.5)^2 + (y - 0.5)^2} - 0.3|, |\sqrt{(x + 0.5)^2 + (y + 0.5)^2} - 0.3|).$$

To initialize the DG solver, we pre-assign the values of ϕ_{ij} , u_{ij} and v_{ij} on the cells whose centers

Table 3.6: Example 4. Γ consists of two circles. Errors are measured in the smooth region, which is outside of $[-0.6, -0.4]^2$, $[0.4, 0.6]^2$ and $x + y \leq 0.1$.

mesh	L^1 error	order	L^∞ error	order	eff. swp. number
20×20	1.35E-3	–	2.17E-2	–	2.79
40×40	3.48E-4	1.96	2.53E-3	3.10	3.88
80×80	9.21E-5	1.92	7.74E-4	1.71	3.91
160×160	2.38E-5	1.95	2.10E-4	1.88	3.92
320×320	6.05E-6	1.98	5.63E-5	1.90	3.93
640×640	1.52E-6	1.99	1.45E-5	1.95	3.93
1280×1280	3.83E-7	1.99	3.70E-6	1.97	3.94

Table 3.7: Example 4. Γ consists of two circles. Whole region.

mesh	L^1 error	order	L^∞ error	order	eff. swp. number
20×20	2.87E-3	–	7.91E-2	–	2.79
40×40	6.69E-4	2.10	4.00E-2	0.98	3.88
80×80	1.64E-4	2.03	2.01E-2	0.99	3.91
160×160	4.12E-5	2.00	1.01E-2	1.00	3.92
320×320	1.03E-5	1.99	5.04E-3	1.00	3.93
640×640	2.60E-6	1.99	2.52E-3	1.00	3.93
1280×1280	6.52E-7	1.99	1.26E-3	1.00	3.94

are within the $2h$ distance from Γ (h is the uniform grid size in this example). The singular set for the solution is composed of the center of each circle and the line that is of equal distance to the two circles. All of these singularities correspond to the intersection of characteristics. This is an interesting test case and our proposed algorithm converges well, see Tables 3.6 and 3.7. We observe that only about 4 effective sweepings are needed for convergence regardless of the mesh size. The error is uniformly second order both in L^1 and in L^∞ norms if we measure it in smooth regions excluding the derivative singularities (see Table 3.6); or we have second order in L^1 and first order in L^∞ if the error is measured in the whole computational domain (see Table 3.7).

Remark: For the same example in our previous work [12], the DG local solver can *not* provide a solution for all cells and the first order FD fast sweeping method is used to provide a solution for some cells near the shocks. By using the uniformly accurate DG fast sweeping methods proposed in this paper, we can see that the DG local solver can provide a solution for all cells in this example. This example shows that our methods in this paper are more robust than the previous version in [12].

Table 3.8: Example 5. Shape-from-shading problem I, uniform mesh.

mesh	L^1 error	order	L^∞ error	order	eff. swp. number
20×20	1.22E-3	–	8.29E-3	–	2.00
40×40	3.07E-4	1.99	2.12E-3	1.97	2.00
80×80	7.74E-5	1.99	5.31E-4	2.00	2.00
160×160	1.95E-5	1.99	1.33E-4	2.00	2.00
320×320	4.90E-6	1.99	3.32E-5	2.00	2.00
640×640	1.23E-6	1.99	8.28E-6	2.00	2.00
1280×1280	3.08E-7	2.00	2.07E-6	2.00	2.00

Example 5 (Shape-from-shading I). Consider Eikonal equation (1.1) with

$$f(x, y) = 2\sqrt{y^2(1-x^2)^2 + x^2(1-y^2)^2}. \quad (3.1)$$

The computational domain $\Omega = [-1, 1] \times [-1, 1]$. $\phi(x, y) = 0$ is prescribed at the boundary of the square, with the additional boundary condition $\phi(0, 0) = 1$. In [7], high order time marching DG schemes are used to calculate the solution for this problem. The exact solution is

$$\phi(x, y) = (1-x^2)(1-y^2). \quad (3.2)$$

To initialize the DG solver, we pre-assign the values of ϕ_{ij} , u_{ij} and v_{ij} on the cells whose centers are within 0.1 distance from the boundary of the square, and the cells whose centers are in the domain $[-0.1, 0.1]^2$. We perform the computation on both the uniform meshes and non-uniform meshes. The non-uniform meshes are obtained by randomly perturbing grid points of the uniform meshes in the range $[-0.1h, 0.1h] \times [-0.1h, 0.1h]$ where h is the mesh size of a uniform mesh. The results are reported in Table 3.8 and Table 3.9. We can observe that only 2 effective sweepings are needed for the convergence for uniform meshes regardless of the mesh size. For non-uniform meshes, the effective sweeping number is settling down to 3 when the mesh is refined. Uniform second order errors are obtained both in L^1 and in L^∞ norms.

Example 6 (Shape-from-shading II). Consider Eikonal equation (1.1) with

$$f(x, y) = 2\pi\sqrt{[\cos(2\pi x)\sin(2\pi y)]^2 + [\sin(2\pi x)\cos(2\pi y)]^2}. \quad (3.3)$$

The computational domain $\Omega = [0, 1] \times [0, 1]$. $\Gamma = \{(\frac{1}{4}, \frac{1}{4}), (\frac{3}{4}, \frac{3}{4}), (\frac{1}{4}, \frac{3}{4}), (\frac{3}{4}, \frac{1}{4}), (\frac{1}{2}, \frac{1}{2})\} \cup \partial\Omega$, consisting of five isolated points and the domain boundary. $g(\frac{1}{4}, \frac{1}{4}) = g(\frac{3}{4}, \frac{3}{4}) = 1$, $g(\frac{1}{4}, \frac{3}{4}) = g(\frac{3}{4}, \frac{1}{4}) = -1$, and $g(\frac{1}{2}, \frac{1}{2}) = 0$. In addition, $\phi(x, y) = 0$ is prescribed on $\partial\Omega$. The solution for this problem is the

Table 3.9: Example 5. Shape-from-shading problem I, non-uniform mesh. The mesh is obtained by randomly perturbing grid points of the uniform mesh in the range $[-0.1h, 0.1h] \times [-0.1h, 0.1h]$.

mesh	L^1 error	order	L^∞ error	order	eff. swp. number
20×20	1.23E-3	–	8.01E-3	–	2.00
40×40	3.09E-4	1.99	2.04E-3	1.97	2.00
80×80	7.79E-5	1.99	5.11E-4	2.00	2.00
160×160	1.96E-5	1.99	1.27E-4	2.01	2.00
320×320	4.95E-6	1.99	3.68E-5	1.79	2.95
640×640	1.24E-6	1.99	9.26E-6	1.99	2.95
1280×1280	3.11E-7	2.00	2.31E-6	2.01	2.95

shape function, which has the brightness $I(x, y) = 1/\sqrt{1 + f(x, y)^2}$ under vertical lighting. See [16] for details. In [8], high order time marching WENO schemes are used to calculate the solution for this problem. The exact solution is

$$\phi(x, y) = \sin(2\pi x) \sin(2\pi y).$$

To initialize the DG solver, we pre-assign the values of ϕ_{ij} , u_{ij} and v_{ij} on the cells whose centers are within 0.05 distance from the boundary of the unit square, and the cells which are in the five square boxes with length 0.05 and the centers $\{(\frac{1}{4}, \frac{1}{4}), (\frac{3}{4}, \frac{3}{4}), (\frac{1}{4}, \frac{3}{4}), (\frac{3}{4}, \frac{1}{4}), (\frac{1}{2}, \frac{1}{2})\}$. As for the example 5, we perform the computation on both the uniform meshes and non-uniform meshes. The non-uniform meshes are obtained by randomly perturbing grid points of the uniform meshes in the range $[-0.1h, 0.1h] \times [-0.1h, 0.1h]$ where h is the mesh size of a uniform mesh. The results are reported in Table 3.10 and Table 3.11. We can observe that only about 3.8 effective sweepings are needed for the convergence for uniform meshes regardless of the mesh size. For non-uniform meshes, the effective sweeping number is settling down to 4.8 when the mesh is refined. We observe a uniform second order accuracy for both the L^1 and the L^∞ norms.

Remark: For the same example, the DG local solver in [12] can *not* provide a solution for all cells and the first order FD fast sweeping method is used to provide a solution for some cells, hence even if this problem has a smooth solution, only first order accuracy is obtained in L^∞ norm. By using the uniformly accurate DG fast sweeping methods proposed in this paper, we show that the DG local solver can provide a solution for all cells and a second order accuracy is obtained in L^∞ norm in this example. Again, this example shows the improvement of the proposed algorithm over our previous work in [12].

Table 3.10: Example 6. Shape-from-shading problem II, uniform mesh.

mesh	L^1 error	order	L^∞ error	order	eff. swp. number
20×20	7.60E-3	–	5.35E-2	–	2.81
40×40	1.25E-3	2.61	9.73E-3	2.46	3.82
80×80	2.86E-4	2.13	2.40E-3	2.02	3.81
160×160	6.78E-5	2.08	6.03E-4	2.00	3.81
320×320	1.64E-5	2.05	1.51E-4	2.00	3.80
640×640	4.02E-6	2.03	3.77E-5	2.00	3.80
1280×1280	9.95E-7	2.02	9.44E-6	2.00	3.80

Table 3.11: Shape-from-shading problem II, Example 6, non-uniform mesh. The mesh is obtained by randomly perturbing grid points of the uniform mesh in the range $[-0.1h, 0.1h] \times [-0.1h, 0.1h]$.

mesh	L^1 error	order	L^∞ error	order	eff. swp. number
20×20	8.81E-3	–	1.22E-1	–	2.82
40×40	1.37E-3	2.68	2.31E-2	2.39	5.82
80×80	3.46E-4	1.99	9.76E-3	1.25	3.81
160×160	8.17E-5	2.08	2.26E-3	2.11	6.81
320×320	1.75E-5	2.22	5.39E-4	2.07	3.77
640×640	4.38E-6	2.00	1.82E-4	1.57	4.79
1280×1280	1.01E-6	2.12	2.28E-5	2.99	4.81

4 Concluding remarks

In this paper we develop a novel strategy to impose the causality in the DG solver for Eikonal equations. We design causality indicators which guide the information flow directions for the DG local solver. The values of these indicators are initially provided by the first order finite difference fast sweeping method, and they are updated during iterations along with the solution. We observe both a uniform second order accuracy in the L^∞ norm (in smooth regions) and the fast convergence speed (linear computational complexity) in the numerical examples. The uniform second order accuracy in the L^∞ norm in smooth region of the solutions shows the improvement of the proposed method over our previous work in [12].

References

- [1] M. Boué and P. Dupuis, *Markov chain approximations for deterministic control problems with affine dynamics and quadratic cost in the control*, SIAM Journal on Numerical Analysis, 36 (1999), 667–695.
- [2] Y. Cheng and C.-W. Shu, *A discontinuous Galerkin finite element method for directly solving the Hamilton-Jacobi equations*. Journal of Computational Physics, 223 (2007), 398–415.
- [3] B. Cockburn and C.-W. Shu, *Runge-Kutta discontinuous Galerkin methods for convection-dominated problems*, Journal of Scientific Computing, 16 (2001), 173–261.
- [4] M.G. Crandall and P.L. Lions, *Viscosity solutions of Hamilton-Jacobi equations*, Trans. Amer. Math. Soc., 277 (1983), 1-42.
- [5] E.W. Dijkstra, *A note on two problems in connection with graphs*, Numerische Mathematik, 1 (1959), 269–271.
- [6] J. Helmsen, E. Puckett, P. Colella and M. Dorr, *Two new methods for simulating photolithography development in 3D*, Proceedings SPIE 2726 (1996), 253–261.
- [7] C. Hu and C.-W. Shu, *A discontinuous Galerkin finite element method for Hamilton-Jacobi equations*, SIAM Journal on Scientific Computing, 20 (1999), 666–690.
- [8] G.-S. Jiang and D. Peng, *Weighted ENO schemes for Hamilton-Jacobi equations*, SIAM Journal on Scientific Computing, 21 (2000), 2126–2143.

- [9] C.Y. Kao, S. Osher and J. Qian, *Lax-Friedrichs sweeping schemes for static Hamilton-Jacobi equations*, Journal of Computational Physics, 196 (2004), 367–391.
- [10] C.Y. Kao, S. Osher and J. Qian, *Legendre-transform-based fast sweeping methods for static Hamilton-Jacobi equations on triangulated meshes*, Journal of Computational Physics, 227 (2008), 10209–10225.
- [11] C.Y. Kao, S. Osher and Y.H. Tsai, *Fast sweeping method for static Hamilton-Jacobi equations*, SIAM Journal on Numerical Analysis, 42 (2005), 2612–2632.
- [12] F. Li, C.-W. Shu, Y.-T. Zhang and H.-K. Zhao, *A second order discontinuous Galerkin fast sweeping method for Eikonal equations*, Journal of Computational Physics, 227 (2008), 8191–8208.
- [13] J. Qian, Y.-T. Zhang and H.-K. Zhao, *Fast sweeping methods for Eikonal equations on triangular meshes*, SIAM Journal on Numerical Analysis, 45 (2007), 83–107.
- [14] J. Qian, Y.-T. Zhang and H.-K. Zhao, *A fast sweeping method for static convex Hamilton-Jacobi equations*, Journal of Scientific Computing, 31 (2007), 237–271.
- [15] C. Rasch and T. Satzger, *Remarks on the $O(N)$ implementation of the fast marching method*, IMA Journal of Numerical Analysis, 29 (2009), 806–813.
- [16] E. Rouy and A. Tourin, *A viscosity solutions approach to shape-from-shading*, SIAM Journal on Numerical Analysis, 29 (1992), 867–884.
- [17] S. Serna and J. Qian, *A stopping criterion for higher-order sweeping schemes for static Hamilton-Jacobi equations*, submitted to Journal of Computational Mathematics.
- [18] J.A. Sethian, *A fast marching level set method for monotonically advancing fronts*, Proceedings of the National Academy of Sciences of the United States of America, 93 (1996), 1591–1595.
- [19] J.A. Sethian and A. Vladimirsky, *Ordered upwind methods for static Hamilton-Jacobi equations*, Proceedings of the National Academy of Sciences of the United States of America, 98 (2001), 11069–11074.
- [20] J.A. Sethian and A. Vladimirsky, *Ordered upwind methods for static Hamilton-Jacobi equations: theory and algorithms*, SIAM Journal on Numerical Analysis, 41 (2003), 325–363.

- [21] Y.-H. R. Tsai, L.-T. Cheng, S. Osher and H.-K. Zhao, *Fast sweeping algorithms for a class of Hamilton-Jacobi equations*, SIAM Journal on Numerical Analysis, 41 (2003), 673–694.
- [22] J.N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Transactions on Automatic Control, 40 (1995), 1528–1538.
- [23] T. Xiong, M. Zhang, Y.-T. Zhang and C.-W. Shu, *Fifth order fast sweeping WENO scheme for static Hamilton-Jacobi equations with accurate boundary treatment*, submitted to Journal of Scientific Computing.
- [24] L. Yatziv, A. Bartesaghi and G. Sapiro, *$O(N)$ implementation of the fast marching algorithm*, Journal of Computational Physics, 212 (2006), 393–399.
- [25] Y.-T. Zhang, H.-K. Zhao and S. Chen, *Fixed-point iterative sweeping methods for static Hamilton-Jacobi equations*, Methods and Applications of Analysis, 13 (2006), 299–320.
- [26] Y.-T. Zhang, H.-K. Zhao and J. Qian, *High order fast sweeping methods for static Hamilton-Jacobi equations*, Journal of Scientific Computing, 29 (2006), 25–56.
- [27] H.-K. Zhao, *A fast sweeping method for Eikonal equations*, Mathematics of Computation, 74 (2005), 603–627.
- [28] H. Zhao, S. Osher, B. Merriman and M. Kang, *Implicit and non-parametric shape reconstruction from unorganized points using variational level set method*, Computer Vision and Image Understanding, 80 (2000), 295–319.