Uniformly accurate discontinuous Galerkin fast sweeping methods for Eikonal equations

Yong-Tao Zhang[*] Shanqin Chen[†] Fengyan Li[‡] Hongkai Zhao[§] Chi-Wang Shu[¶]

March 27, 2011

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. 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. 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.

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

[*]Department of Applied and Computational Mathematics and Statistics, University of Notre Dame, Notre Dame, IN 46556, 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.

[†]Department of Mathematical Sciences, Indiana University South Bend, South Bend, IN 46634, USA. E-mail: chen39@iusb.edu. Research supported by IU South Bend Research and Development grant.

[‡]Department 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.

[§]Department of Mathematics, University of California, Irvine, CA 92697-3875, USA. E-mail: zhao@math.uci.edu. The research is partially supported by ONR Grant N00014-02-1-0090, ARO MURI Grant W911NF-07-1-0185 and NSF Grant DMS0811254.

[¶]Division 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.

1 Introduction

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

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

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 static Hamilton-Jacobi equations

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

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 [25, 19, 6, 20, 21] 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 [27]. 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 [16] 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 $f(\mathbf{x})$, where $f_{\max}$ and $f_{\min}$ are maximal and minimal values of $f(\mathbf{x})$ in the computation domain respectively. In the fast sweeping method [1, 32, 24, 10, 31, 12, 30, 29, 14, 15, 11], 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 [31], 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 [30] 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 [26, 18]. 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 [13] for solving the Eikonal equations. The local solver is based on the P^1 (piecewise-linear) version of the 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 [31], 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 fast convergence of this DG sweeping method, however the DG local solver can *not* provide a solution for all cells. In [13], 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 [31] is used. As a result, the method in [13] 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 to indicate the information flow direction 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 [13]. Hence the numerical values on all cells after the iterations converge are the solution of the DG scheme. Both 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 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 [31]. 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}. 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 caux_{ij}=1 indicates that the information is propagating from the right neighboring cell $I_{i+1,j}$ to the cell I_{ij}. Similarly, 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 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., caux_{ij}=10 or 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 flagx(i, j) and flagy(i, j), for $1 \le i \le N$, $1 \le j \le M$.

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

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

where a, b, r_1 and r_2 are determined by the causality as follows. At interior grid points (x_i, y_j) : $i = 2, 3, \cdots, N-1$, $j = 2, 3, \cdots, M-1$, 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
    $a = \phi_{i-1,j}$,        $r_1 = d_l \cdot f_{ij}$;
    $\mathsf{caux}_{ij} = 0$;
else
    $a = \phi_{i+1,j}$,        $r_1 = d_r \cdot f_{ij}$;
    $\mathsf{caux}_{ij} = 1$.
```

Similarly,

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

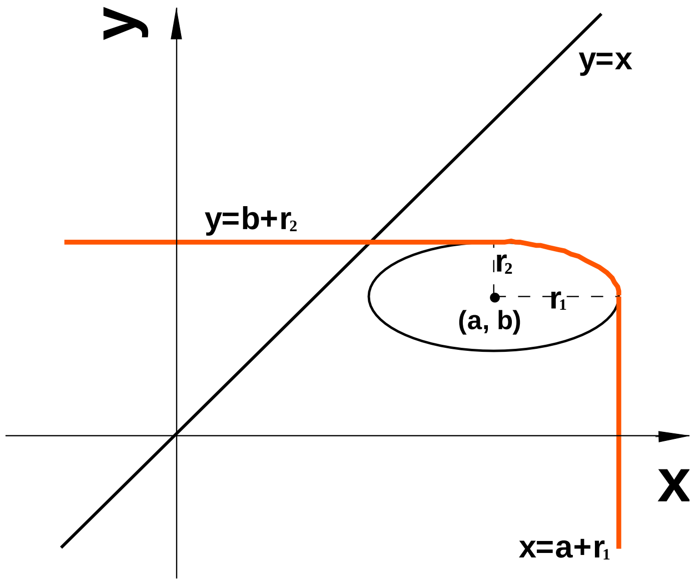

Figure 2.1: About the solution of the quadratic equation (2.1).

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

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

else

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

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

At the boundary of the computational domain, one sided difference is used. Namely, at the left boundary $i = 1$, we take $a = \phi_{2,j}$, $r_1 = d_r \cdot f_{1j}$, $\mathsf{caux}_{1j} = 1$; at the right boundary $i = N$, we take $a = \phi_{N-1,j}$, $r_1 = d_l \cdot f_{Nj}$, $\mathsf{caux}_{Nj} = 0$; at the top boundary $j = M$, we take $b = \phi_{i,M-1}$, $r_2 = d_b \cdot f_{iM}$, $\mathsf{cauy}_{iM} = 0$; at the bottom boundary $j = 1$, we take $b = \phi_{i,2}$, $r_2 = d_t \cdot f_{i1}$, $\mathsf{cauy}_{i1} = 1$.

To solve (2.1), we consider the solution of the equation (2.1) as the intersection point of the following two curves on the 2D plane:

$$\left[\left(\frac{x-a}{r_1}\right)^+\right]^2 + \left[\left(\frac{y-b}{r_2}\right)^+\right]^2 = 1, \tag{2.2}$$

$$y = x. \tag{2.3}$$

The curve represented by the equation (2.2) is composed of two lines $y = b + r_2$ and $x = a + r_1$ connected by the top-right quarter of an ellipse, as the red curve shown in Figure 2.1. The location of the curve will be different for different values of the ellipse center (a, b). But for any values of a, b, r_1 and r_2, the line $y = x$ has a unique intersection point with the red curve in Figure 2.1. To find the coordinates of the intersection point, we first consider the case that the top-right quarter of the ellipse can intersect the line $y = x$. If we fix an arbitrary value of a and let b vary, then the center of the ellipse will move along the line $x = a$. If and only if $a - r_2 < b < a + r_1$, the top-right quarter of the ellipse can intersect $y = x$. Similarly if we fix an arbitrary value of b and let a vary, then the center of the ellipse will move along the line $y = b$. If and only if $b - r_1 < a < b + r_2$, the top-right quarter of the ellipse can intersect $y = x$. In summary, the top-right quarter of the ellipse and $y = x$ will intersect if and only if $-r_2 < b - a < r_1$. In this case, the equation (2.1) is equivalent to the equation

$$\left(\frac{x-a}{r_1}\right)^2 + \left(\frac{x-b}{r_2}\right)^2 = 1. \tag{2.4}$$

Since the point should be in the top-right quarter of the ellipse, we choose the larger solution of the quadratic equation (2.4)

$$x = \frac{ar_2^2 + br_1^2 + r_1r_2\sqrt{r_1^2 + r_2^2 - (a-b)^2}}{r_1^2 + r_2^2}. \tag{2.5}$$

For the second case, if and only if $b - a \leq -r_2$, namely $a \geq b + r_2$, the line $y = x$ can intersect $y = b + r_2$ and the solution for the equation (2.1) is

$$x = b + r_2. \tag{2.6}$$

And the last case, if and only if $b - a \geq r_1$, namely $b \geq a + r_1$, the line $y = x$ can intersect $x = a + r_1$ and the solution for the equation (2.1) is

$$x = a + r_1. \tag{2.7}$$

In summary, if we 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

$$\phi_{ij} = \begin{cases} \dfrac{as_2^2 + bs_1^2 + s_1s_2\sqrt{r_1^2 + r_2^2 - (a-b)^2}}{s_1^2 + s_2^2}, & \text{if } -r_2 < b - a < r_1, \\ b + r_2, \quad \text{if } b - a \leq -r_2, \\ a + r_1, \quad \text{if } b - a \geq r_1. \end{cases} \tag{2.8}$$

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 the 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: $\mathsf{flagx(i, j)} = 10$, $\mathsf{flagy(i, j)} = 10$, for $1 \le i \le N, 1 \le j \le M$.

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 (2.8) 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:

```
If $-r_2 < b - a < r_1$, then
    flagx(i,j)=caux$_{ij}$, flagy(i,j)=cauy$_{ij}$;
If $b - a \le -r_2$, then
    flagx(i,j) = 10, flagy(i,j)=cauy$_{ij}$;
If $b - a \ge r_1$, then
    flagx(i,j)=caux$_{ij}$, flagy(i,j) = 10.
```

3. Convergence: if

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

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. ■

Remark: In this subsection, the first order finite difference fast sweeping method is used to initialize the causality arrays. Hence only solution values ϕ_{ij} on the dual mesh Θ_h are iterated. The

DG local solver of the next subsection and slope values of the solution have not been involved yet at this step. For realistic applications, if some part or the whole of the inflow boundary Γ is not on the grid lines, the Richardson extrapolation or inverse Lax-Wendroff procedure developed in [8] can be used to obtain accurate approximation to values at the grid points near the inflow boundary Γ. We will discuss more about the numerical boundary conditions in the subsection 2.2.4. We update the causality arrays in step 2 so that we can track the convergence history of the information flow directions. The values of causality arrays can also be assigned after the iteration converges if we do not need to track their convergence procedure. In step 2, the solution ϕ_{ij} is only updated if it is smaller than the current value. This makes the solution of the first order finite difference fast sweeping method is nonincreasing with each Gauss-Seidel iteration and converges monotonically to the solution of the discretized system. See [31] for the detailed proof.

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 [13]. We will emphasize their differences in the following.

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

$$V_h^1 = \{v : v|_{I_{ij}} \in P^1(I_{ij}),\ i = 1, \cdots, N,\ j = 1, \cdots, M\} \tag{2.9}$$

where $P^1(I_{ij})$ denotes all linear polynomials on I_{ij}. As in [2, 13], the DG scheme for the Eikonal equations (1.1) is defined as: find $\phi_h \in V_h^1$, such that

$$\begin{aligned}\int_{I_{ij}} |\nabla\phi_h(x,y)| v_h(x,y)dxdy \quad &+ \quad \alpha_{l,ij}\int_{y_{j-1/2}}^{y_{j+1/2}} [\phi_h](x_{i-\frac{1}{2}},y)v_h(x^+_{i-\frac{1}{2}},y)dy \\ &+ \quad \alpha_{b,ij}\int_{x_{i-1/2}}^{x_{i+1/2}} [\phi_h](x,y_{j-\frac{1}{2}})v_h(x,y^+_{j-\frac{1}{2}})dx \\ &+ \quad \alpha_{r,ij}\int_{y_{j-1/2}}^{y_{j+1/2}} [\phi_h](x_{i+\frac{1}{2}},y)v_h(x^-_{i+\frac{1}{2}},y)dy \\ &+ \quad \alpha_{t,ij}\int_{x_{i-1/2}}^{x_{i+1/2}} [\phi_h](x,y_{j+\frac{1}{2}})v_h(x,y^-_{j+\frac{1}{2}})dx \\ &= \quad \int_{I_{ij}} f(x,y)v_h(x,y)dxdy,\ i = 1, \cdots, N,\ j = 1, \cdots, M\end{aligned} \tag{2.10}$$

holds for any $v_h \in V_h^1$. 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 (2.10) is different from that in [13], and it will be described in the following subsection 2.2.1. Mathematical proof of the existence of the solutions of the local solver (2.10) with the new local causality constants in subsection 2.2.1 is still open. But our numerical experiments in this paper show that the system does give a unique solution, under the causality conditions determined by the algorithm described in the current Section 2.

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 $\eta_j = \frac{y-y_j}{l_j}$. So the unknown degrees of freedom on I_{ij} are ϕ_{ij}, u_{ij} and v_{ij}. ϕ_{ij}, u_{ij}/h_i and v_{ij}/l_j are the cell average, the slope in the x-direction and the slope in the y-direction of the linear polynomial $\phi_h(x,y)$ on I_{ij}, respectively.

Let us denote $H_1 \triangleq \frac{\partial H}{\partial \phi_x}$ and $H_2 \triangleq \frac{\partial H}{\partial \phi_y}$. The local causality 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}. The construction of the causality indicators and the calculation of the local causality constants are motivated by the idea of upwind schemes for solving hyperbolic conservation laws and the iterative framework of fast sweeping methods. The local causality constants reflect the causality / upwind information of the Eikonal equations (1.1). However due to the nonlinearity of the Eikonal equations, the causality information is unknown beforehand. On the other hand, the iterative framework of fast sweeping methods allows us to initially estimate the causality information by the first order fast sweeping iterations (see the subsection 2.1), and then iterate the causality information along with the iteration of the solution itself of the DG scheme (2.10).

The calculation of the local causality constants needs the causality information of the current iteration step. In [13], only the information of cell averages was used to determine the causality information. This may not be accurate enough since the DG local solver provides both cell averages and slopes information. By introducing the causality indicators, we can use both cell averages and slopes information to determine the causality information more accurately.

The values of causality indicators are determined initially by the first order fast sweeping iterations (see the subsection 2.1), and they are updated along with the solutions. How to update causality indicators in the iterations will be discussed in detail in the subsection 2.2.3. Numerical experiments in the section 3 show that faster convergence rate can be obtained if the causality information is captured accurately. According to the values of causality indicators in the current iteration step, the local causality constants are calculated. First let's consider the causality constant $\alpha_{l,ij}$. $\alpha_{l,ij}$ will carry causality information from the left neighboring cell $I_{i-1,j}$ in the x-direction if there is causality information comes in from that direction. If the x-direction causality indicator flagx(i,j)=1, it indicates that the information comes in from the right neighboring cell $I_{i+1,j}$ in the x-direction. If flagx(i,j)=10, then it indicates that there is no information comes into the cell I_{ij} in the x-direction. So for these two cases, $\alpha_{l,ij}$ should not carry any causality information and it will be set to 0. If flagx(i,j)=0, it indicates that information may flow in from the cell $I_{i-1,j}$. Because the values of causality indicators are assigned initially by the first order fast sweeping iterations and the initial values of slopes $u_{i-1,j}$ and $v_{i-1,j}$ on the cell $I_{i-1,j}$ are assigned to be 0, it is possible that the causality indicator flagx(i,j)=0 requires the slopes information from the cell $I_{i-1,j}$, which may not be available yet (only has initially assigned value 0). This often happens at the beginning iterations when DG local solver has not been executed on the cell $I_{i-1,j}$. Another possible situation this could happen is that the DG solution on the cell $I_{i-1,j}$ gives the slopes $u_{i-1,j} = v_{i-1,j} = 0$. This means that no information will flow out from the cell $I_{i-1,j}$. So for the case flagx(i,j)=0 and $u_{i-1,j} = v_{i-1,j} = 0$, we skip the current cell I_{ij} in the current iteration and will update the values in this cell at the later iterations when there is more information around this cell available. In addition, skipping current cell due to lack of proper updated information can save some computational cost. Finally, if flagx(i,j)=0 and the slopes information on the cell $I_{i-1,j}$ is available, i.e., at least one of $u_{i-1,j}$ and $v_{i-1,j}$ is not 0, we can compute $H_1(\nabla\phi_h)|_{I_{i-1,j}}$. In summary, the formula to calculate $\alpha_{l,ij}$ is

$$\alpha_{l,ij} = \begin{cases} \max(0, H_1(\nabla\phi_h)|_{I_{i-1,j}}) = \max\left(0, \dfrac{u_{i-1,j}l_j}{\sqrt{(u_{i-1,j}l_j)^2 + (v_{i-1,j}h_{i-1})^2}}\right), \\ \qquad\qquad \text{If flagx(i,j)=0 and } (u_{i-1,j}l_j)^2 + (v_{i-1,j}h_{i-1})^2 \neq 0; \\ \text{skip current cell}, \qquad \text{If flagx(i,j)=0 and } u_{i-1,j} = v_{i-1,j} = 0; \\ 0, \qquad \text{If flagx(i,j)=1 or flagx(i,j)=10.} \end{cases} \tag{2.11}$$

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 flagx(i,j) to be flagx(i,j)=10. By doing this, we shut down this information flow direction of the cell I_{ij} in the current iteration. This could happen when the initial causality determined by the first order fast sweeping method contradicts with the slope information obtained by the second order DG local solver. Namely, although $\mathsf{flagx(i,j)} = 0$ but $u_{i-1,j} < 0$. From our numerical experiments, we found that this situation often happens near shock locations where the characteristics from the left intersect with the characteristics from the right. Likewise,

$$\alpha_{r,ij} = \begin{cases} \min(0, H_1(\nabla\phi_h)|_{I_{i+1,j}}) = \min\left(0, \dfrac{u_{i+1,j}l_j}{\sqrt{(u_{i+1,j}l_j)^2 + (v_{i+1,j}h_{i+1})^2}}\right), \\ \qquad\qquad \text{If flagx(i,j)=1 and } (u_{i+1,j}l_j)^2 + (v_{i+1,j}h_{i+1})^2 \neq 0; \\ \text{skip current cell}, \qquad \text{If flagx(i,j)=1 and } u_{i+1,j} = v_{i+1,j} = 0; \\ 0, \qquad \text{If flagx(i,j)=0 or flagx(i,j)=10.} \end{cases} \tag{2.12}$$

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 flagx(i,j) to be flagx(i,j)=10. Similarly,

$$\alpha_{b,ij} = \begin{cases} \max(0, H_2(\nabla\phi_h)|_{I_{i,j-1}}) = \max\left(0, \dfrac{v_{i,j-1}h_i}{\sqrt{(u_{i,j-1}l_{j-1})^2 + (v_{i,j-1}h_i)^2}}\right), \\ \qquad\qquad \text{If flagy(i,j)=0 and } (u_{i,j-1}l_{j-1})^2 + (v_{i,j-1}h_i)^2 \neq 0; \\ \text{skip current cell}, \qquad \text{If flagy(i,j)=0 and } u_{i,j-1} = v_{i,j-1} = 0; \\ 0, \qquad \text{If flagy(i,j)=1 or flagy(i,j)=10.} \end{cases} \tag{2.13}$$

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 flagy(i,j) to be flagy(i,j)=10. Finally,

$$\alpha_{t,ij} = \begin{cases} \min(0, H_2(\nabla\phi_h)|_{I_{i,j+1}}) = \min\left(0, \dfrac{v_{i,j+1}h_i}{\sqrt{(u_{i,j+1}l_{j+1})^2 + (v_{i,j+1}h_i)^2}}\right), \\ \qquad\qquad \text{If flagy(i,j)=1 and } (u_{i,j+1}l_{j+1})^2 + (v_{i,j+1}h_i)^2 \neq 0; \\ \text{skip current cell}, \qquad \text{If flagy(i,j)=1 and } u_{i,j+1} = v_{i,j+1} = 0; \\ 0, \qquad \text{If flagy(i,j)=0 or flagy(i,j)=10.} \end{cases} \tag{2.14}$$

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 flagy(i,j) to be flagy(i,j)=10. If both flagx(i,j)=10 and 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 (2.10). 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 [13]. 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 (2.10) is converted from the integral form to a quadratic system:

$$\sqrt{u_{ij}^2 + v_{ij}^2 r_{ij}^2} + e_{ij}\phi_{ij} + \beta_{ij}u_{ij} + \lambda_{ij}v_{ij} = R_{1,ij} \tag{2.15}$$

$$12\beta_{ij}\phi_{ij} + d_{ij}u_{ij} = R_{2,ij} \tag{2.16}$$

$$12\lambda_{ij}\phi_{ij} + g_{ij}v_{ij} = R_{3,ij} \tag{2.17}$$

where

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

and

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

To solve this quadratic system (2.15)-(2.17), 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. flagx(i,j) $\neq 10$

   In this case, $\beta_{ij} \neq 0$ and $g_{ij} \neq 0$. From (2.16) and (2.17), we have

$$\begin{aligned}
\phi_{ij} &= a_{ij} + b_{ij}u_{ij}, \\
v_{ij} &= c_{ij} + t_{ij}u_{ij},
\end{aligned} \tag{2.18}$$

   where

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

Substitute (2.18) into (2.15), we obtain a quadratic equation

$$a_6 u_{ij}^2 + a_7 u_{ij} + a_8 = 0, \tag{2.19}$$

where

$$a_6 = a_1 - a_4^2,\ a_7 = a_2 - 2a_4a_5,\ a_8 = a_3 - a_5^2,$$

$$a_1 = 1 + (t_{ij}r_{ij})^2,\ a_2 = 2c_{ij}t_{ij}r_{ij}^2,\ a_3 = (c_{ij}r_{ij})^2,$$

$$a_4 = -(e_{ij}b_{ij} + \beta_{ij} + \lambda_{ij}t_{ij}),\ a_5 = R_{1,ij} - e_{ij}a_{ij} - \lambda_{ij}c_{ij}.$$

If the quadratic equation (2.19) 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:

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

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

The updated values for v_{ij} and ϕ_{ij} can be obtained by (2.18). If the quadratic equation (2.19) 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 (2.16)-(2.17), we have

$$\begin{aligned} u_{ij} &= \frac{R_{2,ij}}{d_{ij}}, \\ \phi_{ij} &= a_{ij} + b_{ij}v_{ij}, \end{aligned} \tag{2.20}$$

where

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

Substituting (2.20) into (2.15), we obtain a quadratic equation for v_{ij}

$$a_6 v_{ij}^2 + a_7 v_{ij} + a_8 = 0, \tag{2.21}$$

where

$$a_6 = r_{ij}^2 - a_3^2, \qquad a_7 = -2a_2a_3, \qquad a_8 = a_1^2 - a_2^2,$$

$$a_1 = \frac{R_{2,ij}}{d_{ij}}, \qquad a_2 = R_{1,ij} - e_{ij}a_{ij}, \qquad a_3 = -(\lambda_{ij} + e_{ij}b_{ij}).$$

If the quadratic equation (2.21) 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:

$$\text{if flagy(i,j)} = 0, \qquad \text{then } v_{ij} = \max(\bar{v}_1, \bar{v}_2);$$

$$\text{if flagy(i,j)} = 1, \qquad \text{then } v_{ij} = \min(\bar{v}_1, \bar{v}_2).$$

The updated values for u_{ij} and ϕ_{ij} can be obtained by (2.20). If the quadratic equation (2.21) 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 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 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 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 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
    flagx(i+1,j) = 0;
```

otherwise we do *not* update flagx(i+1,j). We would like to point out the reason that we need the current numerical values on cell (i+2, j) to be provided by DG local solver. This is because the initial iteration values are provided by the first order finite difference fast sweeping iterations, but the numerical values on the cell (i,j) have been provided by the DG local solver in the current iteration. So we need the numerical values on the cell (i+2,j) to be also provided by the DG local solver in order to have consistent information for computing the "global" causality.

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 flagx(i-1,j). If the cell (i-1, j) is a boundary cell, then we do *not* need to update flagx(i-1,j). Otherwise if the cell (i-1, j) happens to be at the boundary of the computational domain, we will update 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
    flagx(i-1,j) = 1;
```

otherwise we do *not* update flagx(i-1,j).

Cases in the y-direction are similar and the detailed description is in the Appendix.

2.2.4 Initialization of the DG local solver and boundary conditions

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 [13]. For example, if the values $\phi(x_{i\pm1/2}, y_{j\pm1/2})$ 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

$$\phi_{ij} = \frac{1}{4}\left(\phi(x_{i-1/2}, y_{j-1/2}) + \phi(x_{i+1/2}, y_{j-1/2}) + \phi(x_{i-1/2}, y_{j+1/2}) + \phi(x_{i+1/2}, y_{j+1/2})\right), \tag{2.22}$$

$$u_{ij} = \frac{1}{2}\left(\phi(x_{i+1/2}, y_{j-1/2}) - \phi(x_{i-1/2}, y_{j-1/2}) + \phi(x_{i+1/2}, y_{j+1/2}) - \phi(x_{i-1/2}, y_{j+1/2})\right), \tag{2.23}$$

$$v_{ij} = \frac{1}{2}\left(\phi(x_{i-1/2}, y_{j+1/2}) - \phi(x_{i-1/2}, y_{j-1/2}) + \phi(x_{i+1/2}, y_{j+1/2}) - \phi(x_{i+1/2}, y_{j-1/2})\right), \tag{2.24}$$

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.

Remark on boundary treatment strategies: The least squares method is used to specify the values of ϕ_{ij}, u_{ij} and v_{ij} on the boundary cells around the inflow boundary Γ, as shown in (2.22) - (2.24). Hence the values $\phi(x_{i\pm1/2}, y_{j\pm1/2})$ are needed at four corner grid points of the boundary cell I_{ij} in order to perform the least squares approximation (2.22) - (2.24). If grid points of the boundary cells are not on the boundary Γ and the values on these grid points are not available, the Richardson extrapolation or inverse Lax-Wendroff procedure developed in [8] can be used to obtain accurate approximation to these values. Richardson extrapolation procedure uses first order accurate solutions on several locally successively refined meshes to obtain high order approximations to numerical values at grid points of the boundary cell. Richardson extrapolation procedure is suitable for different types of inflow boundaries, including the source boundary consisting of a single point. The inverse Lax-Wendroff procedure repeatly uses the PDE itself to obtain high order approximations to the numerical values for the boundary grid points. This procedure can be applied to complex domains and other hyperbolic PDEs, as shown in [22]. We will briefly describe the inverse Lax-Wendroff procedure ([8]) here. More details can be found in [8, 22] and its application to fifth order fast sweeping WENO scheme [26].

We will use the following example to describe the inverse Lax-Wendroff procedure. Assume that the computational domain is $[-1, 1]^2$ and the left boundary $\Gamma = \{(x, y)|x = -1, -1 \leq y \leq 1\}$ is the inflow boundary. The solution on Γ is given as

$$\phi(-1, y) = g(y), \qquad -1 \leq y \leq 1. \tag{2.25}$$

For our second order method, we would like to obtain a second order approximation to $\phi(x_{i+1/2}, y_{j+1/2})$ which is a corner grid point of the boundary cell I_{ij}. By Taylor expansion,

$$\phi(x_{i+1/2}, y_{j+1/2}) = \phi(-1, y_{j+1/2}) + (x_{i+1/2} + 1)\phi_x(-1, y_{j+1/2}) + \mathcal{O}(h^2), \tag{2.26}$$

where $h = \max_i\{h_i\}$. Hence our desired approximation for the second order DG scheme is

$$\phi(x_{i+1/2}, y_{j+1/2}) \approx \phi(-1, y_{j+1/2}) + (x_{i+1/2} + 1)\phi_x(-1, y_{j+1/2}). \tag{2.27}$$

We already have $\phi(-1, y_{j+1/2}) = g(y_{j+1/2})$ from the boundary condition (2.25). We evaluate the Eikonal equation (1.1) and obtain

$$\sqrt{\phi_x(-1, y_{j+1/2})^2 + \phi_y(-1, y_{j+1/2})^2} = f(-1, y_{j+1/2}) \tag{2.28}$$

in which $\phi_y(-1, y_{j+1/2}) = g'(y_{j+1/2})$ and the only unknown quantity is $\phi_x(-1, y_{j+1/2})$. Solving this equation should give us $\phi_x(-1, y_{j+1/2})$. There are two roots with different signs for this quadratic equation. For this example, the positive one should be chosen to guarantee that the boundary Γ is an inflow boundary.

We would like to emphasize that we are using this simple example to fix the idea of the inverse Lax-Wendroff procedure. As shown in [8, 22, 26], this procedure can be carried out to any desired order of accuracy. Also, it can be applied to the inflow boundary Γ with very complicated curved geometries by changing the x and y partial derivatives to normal and tangential derivatives with respect to Γ. Recently, the inverse Lax-Wendroff procedure has been applied to moving boundaries in [23].

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 flagx(i,j) $\neq 1$ and 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 flagx(i,j) $= 1$ and 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 flagx(i,j) $\neq 0$ and 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 flagx(i,j) $= 0$ and 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 are candidate cells for which the DG local solver will be applied. By doing this, we can save a lot of 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). Examples include solutions which have discontinuities in their derivatives, and difficult test cases such as the examples of shape from shading arising in the application area of computer vision. Numerical results 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 are candidate cells for which the DG local solver will be applied. 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 current local system has solutions or not. The effective sweeping number can take non integer values because many cells are not updated in specific sweeps and hence are not counted towards the computation of this number. When this number is n, it means n sweepings, not $n \times 4$ sweepings.

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

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

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 [31, 30, 15, 13] when there is singularity at the source point.

Example 2. (Point source distance function problem). $\Omega = [-1,1]^2$, $\Gamma = \{(0,0)\}$ and $f(x,y) = 1$, $g = 0$. 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 $f(x,y) = \frac{\pi}{2}\sqrt{\sin^2\left(\frac{\pi}{2}x\right) + \sin^2\left(\frac{\pi}{2}y\right)}$. The exact solution is $\phi(x,y) = -\cos\left(\frac{\pi}{2}x\right) - \cos\left(\frac{\pi}{2}y\right)$. 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 $f(x,y) = \frac{\pi}{2}\sqrt{\sin^2\left(\frac{\pi}{2}x\right) + \sin^2\left(\frac{\pi}{2}y\right)}$. The exact solution is $\phi(x,y) = -\cos\left(\frac{\pi}{2}x\right) - \cos\left(\frac{\pi}{2}y\right)$. 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 $f(x, y) = 1$, $g = 0$.

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 $f(x, y) = 1$, $g = 0$. 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.

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

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

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 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).

In Table 3.8, we report the CPU time for computing causality indicators and arrays, the global CPU time, and their ratios. The regular sweeping numbers are also listed in Table 3.8. We can see that the computations of causality indicators and arrays only take about $6\% \sim 12\%$ of the global CPU time. The regular sweeping numbers are 16 for all mesh sizes hence they are independent of the mesh sizes. The pictures of the numerical solution on the 160×160 mesh are presented in the Figure 3.1.

Remark: For the same example in our previous work [13], 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 [13].

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

Table 3.8: Example 4. CPU time (Unit: seconds) and iteration numbers.

mesh	CPU for causal. info.	Global CPU	percentage	swp. #	eff. swp. #
80 × 80	0.01	0.17	5.88%	16	3.91
160 × 160	0.06	0.67	8.96%	16	3.92
320 × 320	0.31	2.77	11.19%	16	3.93
640 × 640	1.13	11.24	10.05%	16	3.93
1280 × 1280	5.21	45.23	11.52%	16	3.94

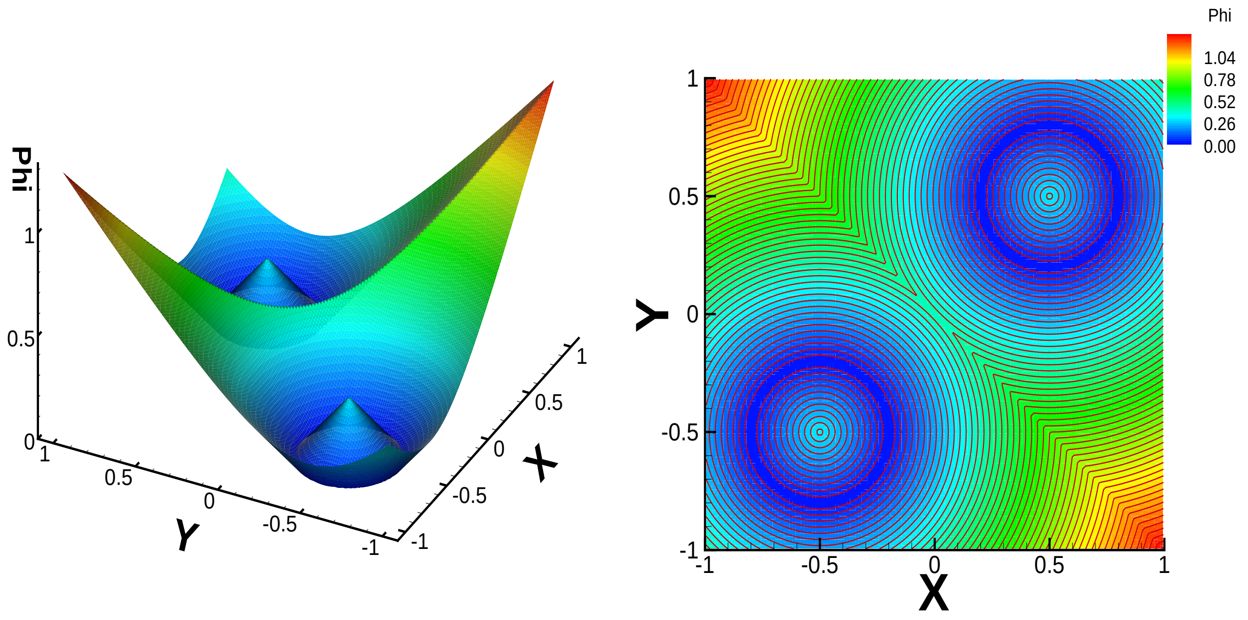

Figure 3.1: Example 4. Γ consists of two circles. The numerical solution on the 160 × 160 mesh. Left: the 3D plot; right: the contour plot.

Table 3.9: Example 5. Shape-from-shading problem I, uniform mesh.

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

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

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

$$\phi(x,y) = (1-x^2)(1-y^2). \tag{3.2}$$

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 3.9 and Table 3.10. 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

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

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 3.10: 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 [17] for details. In [9], 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 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 3.11 and Table 3.13. 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.

In Table 3.12, we report the CPU time for computing causality indicators and arrays, the global CPU time, and their ratios. The regular sweeping numbers are also listed in Table 3.12. We can see that the computations of causality indicators and arrays only take about $5\% \sim 8\%$ of the global CPU time. The regular sweeping numbers are 16 for all mesh sizes hence they are independent of the mesh sizes. The pictures of the numerical solution on the 160×160 mesh are presented in the Figure 3.2.

Remark: For the same example, the DG local solver in [13] 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

Table 3.11: 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 3.12: Example 6, uniform mesh. CPU time (Unit: seconds) and iteration numbers.

mesh	CPU for causal. info.	Global CPU	percentage	swp. #	eff. swp. #
40 × 40	0.01	0.22	4.55%	16	3.82
80 × 80	0.07	0.87	8.05%	16	3.81
160 × 160	0.18	3.40	5.29%	16	3.81
320 × 320	0.79	13.62	5.80%	16	3.80
640 × 640	3.01	54.46	5.53%	16	3.80
1280 × 1280	11.74	217.59	5.40%	16	3.80

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 [13].

Example 7 (Shape-from-shading III). Consider Eikonal equation (1.1) with

$$f(x, y) = \sqrt{(1 - |x|)^2 + (1 - |y|)^2}. \tag{3.4}$$

The computational domain $\Omega = [-1, 1] \times [-1, 1]$. $\phi(x, y) = 0$ is prescribed at the boundary of the square. As the last two examples, this is also a typical shape-from-shading problem [17] to test the high order numerical methods for Hamilton-Jacobi equations (e.g. [7, 28, 30, 13, 26]). We use this example to test the inverse Lax-Wendroff procedure for the boundary cells. Although the exact solution $\phi = (1 - |x|)(1 - |y|)$ is known, we do not use it to provide boundary values. The values of ϕ_{ij}, u_{ij} and v_{ij} on the boundary cells whose centers are within the h distance from Γ are generated by the inverse Lax-Wendroff procedure. The numerical results of convergence study are reported in

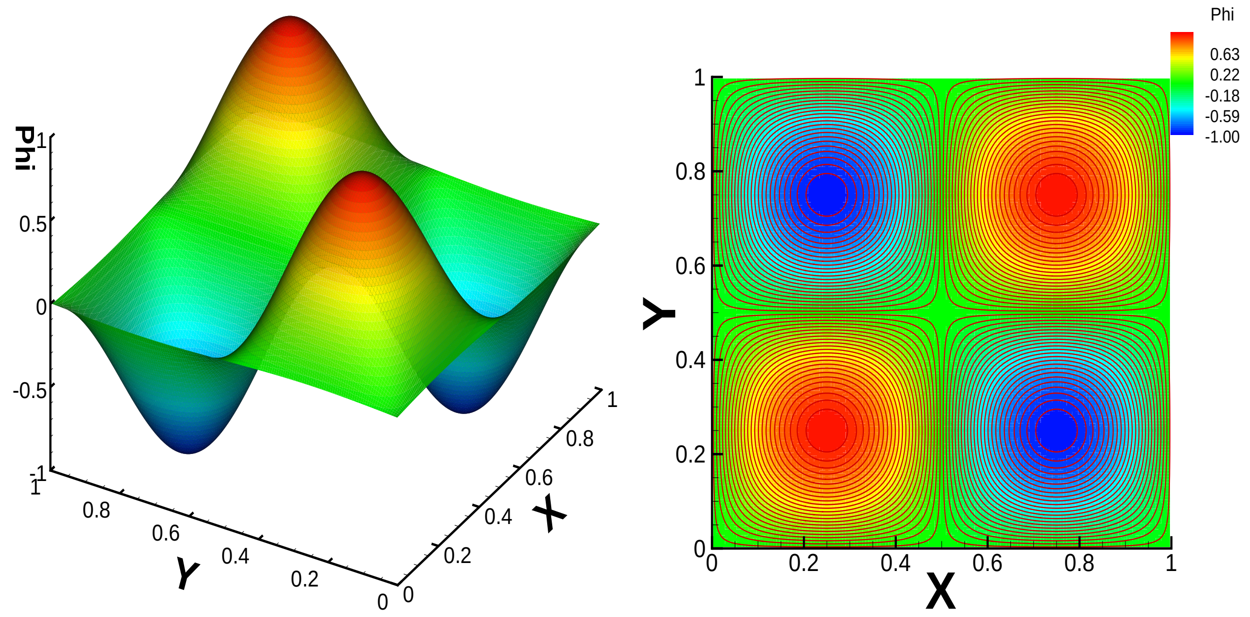

Figure 3.2: Example 6. Shape-from-shading II. The numerical solution on the 160×160 mesh. Left: the 3D plot; right: the contour plot.

Table 3.13: 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

Table 3.14: Example 7. Shape-from-shading problem III. A test for the inverse Lax-Wendroff boundary treatment.

mesh	L^1 error	order	L^∞ error	order	swp. #	eff. swp. #
20 × 20	5.40E-4	–	2.97E-3	–	8	2.00
40 × 40	1.54E-4	1.81	7.95E-4	1.90	8	2.00
80 × 80	4.11E-5	1.90	2.10E-4	1.92	8	2.00
160 × 160	1.07E-5	1.95	5.51E-5	1.93	8	2.00
320 × 320	2.72E-6	1.97	1.43E-5	1.94	8	2.00
640 × 640	6.86E-7	1.99	3.71E-6	1.95	8	2.00
1280 × 1280	1.72E-7	1.99	9.57E-7	1.95	8	2.00

Table 3.14. We can see that the second order accuracy is obtained, and 8 regular sweeping number and 2 effective sweepings are needed for the convergence regardless of the mesh sizes.

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 [13].

Appendix: Description for y-direction cases in section 2.2.3

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 flagy(i,j+1). If the cell (i, j+1) is a boundary cell, then we do *not* need to update flagy(i,j+1). Otherwise if the cell (i, j+1) happens to be at the boundary of the computational domain, we will update 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
                    flagy(i,j+1) = 0;
```

otherwise we do *not* update 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 flagy(i,j-1). If the cell (i, j-1) is a boundary cell, then we do *not* need to update flagy(i,j-1). Otherwise if the cell (i, j-1) happens to be at the boundary of the computational domain, we will update 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 flagy(i,j-1).

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] L. Huang, C.-W. Shu and M. Zhang, *Numerical boundary conditions for the fast sweeping high order WENO methods for solving the Eikonal equation*, Journal of Computational Mathematics, 26 (2008), 336–346.

[9] G.-S. Jiang and D. Peng, *Weighted ENO schemes for Hamilton-Jacobi equations*, SIAM Journal on Scientific Computing, 21 (2000), 2126–2143.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] E. Rouy and A. Tourin, *A viscosity solutions approach to shape-from-shading*, SIAM Journal on Numerical Analysis, 29 (1992), 867–884.

[18] S. Serna and J. Qian, *A stopping criterion for higher-order sweeping schemes for static Hamilton-Jacobi equations*, Journal of Computational Mathematics, 28 (2010), 552–568.

[19] 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.

[20] 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.

[21] 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.

[22] S. Tan and C.-W. Shu, *Inverse Lax-Wendroff procedure for numerical boundary conditions of conservation laws*, Journal of Computational Physics, 229 (2010), 8144–8166.

[23] S. Tan and C.-W. Shu, *A high order moving boundary treatment for compressible inviscid flows*, submitted to Journal of Computational Physics, (2010).

[24] 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.

[25] J.N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Transactions on Automatic Control, 40 (1995), 1528–1538.

[26] 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*, Journal of Scientific Computing, 45 (2010), 514–536.

[27] L. Yatziv, A. Bartesaghi and G. Sapiro, *O(N) implementation of the fast marching algorithm*, Journal of Computational Physics, 212 (2006), 393–399.

[28] Y.-T. Zhang and C.-W. Shu, *High order WENO schemes for Hamilton-Jacobi equations on triangular meshes*, SIAM Journal on Scientific Computing, 24 (2003), 1005–1030.

[29] 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.

[30] 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.

[31] H.-K. Zhao, *A fast sweeping method for Eikonal equations*, Mathematics of Computation, 74 (2005), 603–627.

[32] 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.