A Second Order Discontinuous Galerkin Fast Sweeping Method for Eikonal Equations

Fengyan Li[1], Chi-Wang Shu[2], Yong-Tao Zhang[3], and Hongkai Zhao[4]

ABSTRACT

In this paper, we construct a second order fast sweeping method with a discontinuous Galerkin (DG) local solver for computing viscosity solutions of a class of static Hamilton-Jacobi equations, namely the Eikonal equations. Our piecewise linear DG local solver is built on a DG method developed recently [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] for the time-dependent Hamilton-Jacobi equations. The causality property of Eikonal equations is incorporated into the design of this solver. The resulting local nonlinear system in the Gauss-Seidel iterations is a simple quadratic system and can be solved explicitly. The compactness of the DG method and the fast sweeping strategy lead to fast convergence of the new scheme for Eikonal equations. Extensive numerical examples verify efficiency, convergence and second order accuracy of the proposed method.

Key Words: fast sweeping methods, discontinuous Galerkin finite element methods, second order accuracy, static Hamilton-Jacobi equations, Eikonal equations

[1]Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180, USA. E-mail: lif@rpi.edu. Research supported by NSF grant DMS-0652481.

[2]Division of Applied Mathematics, Brown University, Providence, RI 02912, USA. E-mail: shu@dam.brown.edu. Research supported by NSF grant DMS-0510345.

[3]Department of Mathematics, University of Notre Dame, Notre Dame, IN 46556-4618, USA. E-mail: yzhang10@nd.edu

[4]Department of Mathematics, University of California, Irvine, CA 92697-3875, USA. E-mail: zhao@math.uci.edu. Research partially supported by NSF grant DMS-0513073, ONR grant N00014-02-1-0090 and DARPA grant N00014-02-1-0603.

1 Introduction

In this paper, we develop a second order fast sweeping method for numerically solving an important class of static Hamilton-Jacobi (H-J) equations, namely the Eikonal equations

$$|\nabla\phi(x)| = f(x), \qquad x \in \Omega\backslash\Gamma, \tag{1.1}$$

$$\phi(x) = g(x), \qquad x \in \Gamma \subset \Omega, \tag{1.2}$$

where Ω belongs to $\mathbb{R}^2$ with Γ as its subset, e.g., the boundary, and $0 < c < f(x) < C < \infty$ for some positive c, C and $f(x)$ and $g(x)$ are Lipschitz continuous. Such equations appear in many applications, such as optimal control, differential games, level set method, image processing, computer vision, and geometric optics.

Since the boundary value problems (1.1)-(1.2) are nonlinear first order partial differential equations, we may apply the classical method of characteristics to solve these equations in phase space; namely, consider the gradient components as independent variables and solve ODE systems to follow the propagation of characteristics. Although the characteristics may never intersect in phase space, their projection into physical space may intersect so that the solution in physical space is not uniquely defined at these intersections. By mimicking the entropy condition for hyperbolic conservation laws to single out a physically relevant solution, Crandall and Lions [16] introduced the concept of viscosity solutions for H-J equations so that a unique global weak solution can be defined for such first order nonlinear equations. Moreover, monotone finite difference schemes are developed to compute such viscosity solutions stably.

There are mainly two classes of numerical methods for solving static H-J equations. The first class of numerical methods is based on reformulating the equations into suitable time-dependent problems. Osher [32] provides a natural link between static and time-dependent H-J equations by using the level-set idea and thus raising the problem one-dimensional higher. In the control framework, a semi-Lagrangian scheme is obtained for H-J equations by discretizing in time the dynamic programming principle [19, 20]. Another approach to obtaining a "time" dependent H-J equation from the static H-J equation is using the so called paraxial formulation in which a preferred spatial direction is assumed in the characteristic propagation [21, 17, 29, 36, 37]. High order numerical schemes are well developed for the time dependent H-J equation on structured and unstructured meshes [34, 25, 51, 24, 33, 7, 26, 31, 35, 1, 3, 4, 6, 8]; see a recent review on high order numerical methods for time dependent H-J equations by Shu [46]. Due to the finite speed of propagation and the CFL condition for the discrete time step size, the number of time steps has to

be of the same order as that for one of the spatial dimensions so that the solution converges in the entire domain.

The other 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 [48, 43, 22, 44, 45] is based on the Dijkstra's algorithm [18]. 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); hence two essential ingredients are needed in the 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 [50]. 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 [40] 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 $f(x)$. In the fast sweeping method [5, 55, 47, 54, 27, 28, 53, 38, 39, 52], Gauss-Seidel iterations with alternating orderings is 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 a finite number of iterations is needed [54], so that the complexity of the algorithm is $O(N)$ for a total of N grid points, 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 [53] provides a quite general framework, and it is easy to incorporate any order of accuracy and any type of numerical Hamiltonian into the framework. Much faster convergence speed than that by 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. In this paper, we develop a second order fast sweeping method based on a DG local solver for an important class of static H-J equations, namely the Eikonal equations. A very fast convergence speed is observed in the numerical experiments.

The DG method is a class of finite element methods, using discontinuous piecewise polynomials as approximations for solutions and test functions. The main difference between the DG method and the finite difference or finite volume method is that the former stores and solves for a complete polynomial in each cell, while the latter stores and solves for only one piece of information (the point value for the finite difference scheme and the cell average for the finite volume scheme) in each cell, and it must resort to a reconstruction or interpolation using a wide stencil of neighboring cells in order to achieve higher order accuracy. The DG method was first designed as a method for solving hyperbolic conservation laws containing only first order spatial derivatives, e.g. Reed and Hill [42] for solving linear equations, and Cockburn et al. [12, 11, 10, 13] for solving nonlinear equations. In recent years the DG method has also been extended to solve many other nonlinear PDEs such as the convection diffusion equations [14], KdV type dispersive wave equations [49], etc. Even though the approximation spaces of the DG method consist of discontinuous functions, the DG method can also be used to approximate continuous and even smooth solutions like in elliptic equations. The key point is that discontinuities at the cell boundary are automatically controlled to be consistent with the approximation error inside the cell for a well designed DG method. We refer to [15, 2] for more details about DG methods.

There are mainly two types of DG methods for solving H-J equations. One was originally proposed in [24, 23] which is based on the observation that the gradient of the solution ϕ of the H-J equations satisfies a hyperbolic system. The DG method combined with a least squares procedure, is then applied to solve the system to get the gradient of ϕ. The missing constant in ϕ is further recovered through the original equation. This method is later reinterpreted and simplified in [30] by using the piecewise curl-free solution space in DG method. In another recent work on DG method for H-J equation [9], the solution ϕ is solved directly. This method is more desirable in the sense that the derived hyperbolic system for $\nabla\phi$ in [24, 30] involves more unknown functions than the original scalar H-J equation which is solved directly in [9]. Moreover the solutions of H-J equations generally are smoother than those of the conservations laws.

Based on the DG discretization in [9], in this paper, we develop a second order fast sweeping method for solving Eikonal equations by incorporating the causality property of these equations into the DG local solver. The resulting local nonlinear system in the Gauss-Seidel iterations is a simple quadratic system and can be solved explicitly. The compactness of the DG method and the fast sweeping strategy lead to a very fast convergence of the new scheme for Eikonal equations. The rest of the paper is organized as follows. The algorithm is developed in Section 2, and numerical examples are given in Section 3 to demonstrate the accuracy and the fast convergence of the method. Concluding remarks are given in Section 4.

2 Numerical Formulation

Let $q = \nabla_x \phi$ and $H(q) = |q| - f(x)$, $q \in \mathbb{R}^2$, then the characteristic equations of the Eikonal equation (1.1)-(1.2) are

$$\dot{x} = \nabla_q H = \frac{q}{f} \tag{2.1}$$

$$\dot{q} = \nabla_x f \tag{2.2}$$

$$\dot{\phi} = \nabla_x \phi \cdot \dot{x} = q \cdot \nabla_q H = f(x) > 0 \tag{2.3}$$

One can see that ϕ is increasing along the characteristics, which is an important component in the design of the efficient methods for solving (1.1)-(1.2).

One of the key ingredients in designing a fast sweeping method is to design a local solver, which expresses the value at the standing mesh point in terms of its neighboring values, and (1) it is consistent with the causality of the PDE and is able to deal with possible discontinuities in the derivatives, and (2) the resulting nonlinear equation can be solved efficiently during the Gauss-Seidel iterations. The second order local solver developed in this paper is based on the second type discontinuous Galerkin method [9] reviewed in the introduction. The main idea will be presented for the cases with rectangular meshes.

We start with a partition of $\Omega = \cup_{1\le i\le I, 1\le j\le J} I_{ij}$ 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 lengths of I_i, J_j are denoted by $\Delta x_i = x_{i+1/2} - x_{i-1/2}$ and $\Delta y_j = y_{j+1/2} - y_{j-1/2}$. In this paper, we take $\Delta x_i = \Delta y_j = h$ for simplicity of the presentation. We further define the piecewise linear approximation space as

$$V_h^1 = \{v : v|_{I_{ij}} \in P^1(I_{ij}), \forall i, j\}$$

where $P^1(I_{ij})$ is the set of linear polynomials on I_{ij}. Following the idea of the method developed in [9], a numerical scheme for (1.1) can be formulated as follows: find $\phi_h \in V_h^1$, such that

$$\begin{aligned}\int_{I_{ij}} |\nabla\phi_h| w_h(x,y)dxdy + \alpha_{r,ij}\int_{J_j}[\phi_h](x_{i+\frac{1}{2}},y)w_h(x^-_{i+\frac{1}{2}},y)dy + \alpha_{l,ij}\int_{J_j}[\phi_h](x_{i-\frac{1}{2}},y)w_h(x^+_{i-\frac{1}{2}},y)dy \\ + \alpha_{t,ij}\int_{I_i}[\phi_h](x,y_{j+\frac{1}{2}})w_h(x,y^-_{j+\frac{1}{2}})dx + \alpha_{b,ij}\int_{I_i}[\phi_h](x,y_{j-\frac{1}{2}})w_h(x,y^+_{j-\frac{1}{2}})dx \\ = \int_{I_{ij}} f(x,y)w_h(x,y)dxdy, \qquad \forall\, i,j,\ \ \forall\, w_h \in V_h^1 \end{aligned} \tag{2.4}$$

Let us now explain the notations in (2.4). For the piecewise smooth function $\phi_h \in V_h^1$, $[\phi_h]$ denotes the jump of ϕ_h across the cell interface which is defined horizontally by $[\phi_h](x_{i+\frac{1}{2}},\cdot) = \phi_h(x^+_{i+\frac{1}{2}},\cdot) - \phi_h(x^-_{i+\frac{1}{2}},\cdot)$, $\forall\, i$ and vertically by $[\phi_h](\cdot,y_{j+\frac{1}{2}}) = \phi_h(\cdot,y^+_{j+\frac{1}{2}}) - \phi_h(\cdot,y^-_{j+\frac{1}{2}})$, $\forall\, j$. Here $\phi_h(x^+_{i+\frac{1}{2}},y) = \phi_h|_{I_{i+1,j}}(x_{i+\frac{1}{2}},y)$, $\phi_h(x^-_{i+\frac{1}{2}},y) = \phi_h|_{I_{ij}}(x_{i+\frac{1}{2}},y)$ for $y \in J_j$, and $\phi_h(x,y^+_{j+\frac{1}{2}}) = \phi_h|_{I_{i,j+1}}(x,y_{j+\frac{1}{2}})$, $\phi_h(x,y^-_{j+\frac{1}{2}}) = \phi_h|_{I_{ij}}(x,y_{j+\frac{1}{2}})$ for $x \in I_i$. $\alpha_{r,ij},\alpha_{l,ij},\alpha_{t,ij},\alpha_{b,ij}$ are constants which only depend on the numerical solutions in the neighboring cells of I_{ij}, and they are one of the important components of the local solver. By properly choosing these functions, we expect that the formulation (2.4) is not only accurate and stable, but also enforces the causality of the Eikonal equation therefore making it possible to design an efficient fast sweeping algorithm for solving (1.1)-(1.2).

The piecewise linear approximation $\phi_h|_{I_{ij}}$ can be written as $\phi_h|_{I_{ij}} = \bar{\phi}_{ij} + u_{ij}X_i + v_{ij}Y_j$, where $\bar{\phi}_{ij}$, u_{ij}, v_{ij} are the unknowns, and $X_i = \frac{x-x_i}{h}$, $Y_j = \frac{y-y_j}{h}$. Note that $\bar{\phi}_{ij}$ is the cell average of ϕ_h over I_{ij}.

With the consideration of the causality of the Eikonal equation (1.1)-(1.2), we choose the constants $\alpha_{r,ij}$, $\alpha_{l,ij}$, $\alpha_{t,ij}$, $\alpha_{b,ij}$ in the formulation (2.4) as following:

$$\alpha_{l,ij} = \begin{cases} \max(0, u_{i-1,j}/(hf_{i-1,j})) & \text{when } \bar{\phi}_{i-1,j} \le \bar{\phi}_{i+1,j}; \\ 0 & \text{when } \bar{\phi}_{i-1,j} > \bar{\phi}_{i+1,j}, \end{cases} \tag{2.5}$$

$$\alpha_{r,ij} = \begin{cases} 0 & \text{when } \bar{\phi}_{i-1,j} \le \bar{\phi}_{i+1,j}; \\ \min(0, u_{i+1,j}/(hf_{i+1,j})) & \text{when } \bar{\phi}_{i-1,j} > \bar{\phi}_{i+1,j}, \end{cases} \tag{2.6}$$

$$\alpha_{b,ij} = \begin{cases} \max(0, v_{i,j-1}/(hf_{i,j-1})) & \text{when } \bar{\phi}_{i,j-1} \le \bar{\phi}_{i,j+1}; \\ 0 & \text{when } \bar{\phi}_{i,j-1} > \bar{\phi}_{i,j+1}, \end{cases} \tag{2.7}$$

$$\alpha_{t,ij} = \begin{cases} 0 & \text{when } \bar{\phi}_{i,j-1} \le \bar{\phi}_{i,j+1}; \\ \min(0, v_{i,j+1}/(hf_{i,j+1})) & \text{when } \bar{\phi}_{i,j-1} > \bar{\phi}_{i,j+1}, \end{cases} \tag{2.8}$$

where $f_{i,j} = f(x_i,y_j)$. If we denote $H_1 = \frac{\partial H}{\partial \phi_x}$ and $H_2 = \frac{\partial H}{\partial \phi_y}$, one would notice that the constants $\alpha_{r,ij},\alpha_{l,ij},\alpha_{t,ij},\alpha_{b,ij}$ are approximations of $H_1(\nabla\phi_h)$ and $H_2(\nabla\phi_h)$ in the four neighboring cells of I_{ij}, with min and max operations to enforce the consistency of the causality. For instance when $\bar{\phi}_{i-1,j} \le \bar{\phi}_{i+1,j}$ and $\bar{\phi}_{i,j-1} \le \bar{\phi}_{i,j+1}$, that is when the information is propagating from the bottom-left to the top-right in the neighborhood of the cell I_{ij}, we expect the causality consistency

that $H_1(\nabla\phi_h)|_{I_{i-1,j}} \approx u_{i-1,j}/(hf_{i-1,j}) > 0$, $H_2(\nabla\phi_h)|_{I_{i,j-1}} \approx v_{i,j-1}/(hf_{i,j-1}) > 0$ for the current numerical solution in neighboring cells $I_{i-1,j}$ and $I_{i,j-1}$. So if this holds, $\alpha_{l,ij}$ and $\alpha_{b,ij}$ will be non-zeros and the scheme (2.4) becomes

$$\int_{I_{ij}} |\nabla\phi_h| w_h(x,y)dxdy + \alpha_{l,ij}\int_{J_j}[\phi_h](x_{i-\frac{1}{2}},y)w_h(x^+_{i-\frac{1}{2}},y)dy + \alpha_{b,ij}\int_{I_i}[\phi_h](x,y_{j-\frac{1}{2}})w_h(x,y^+_{j-\frac{1}{2}})dx$$
$$= \int_{I_{ij}} f(x,y)w_h(x,y)dxdy, \qquad \forall\, i,j,\ \ \forall\, w_h \in V_h^1 \tag{2.9}$$

In other words, the method reflects the fact that the solution is increasing along the characteristics.

Remark 2.1 *The choices for $\alpha_{l,ij}$, $\alpha_{r,ij}$, $\alpha_{b,ij}$ and $\alpha_{t,ij}$ in this paper are different from those used in [9]. For any given (i,j), these functions do not depend on $\phi_h|_{I_{ij}}$, and this is crucial in order to get a simple local solver for* (2.4) *which forms a building block of the fast convergence algorithm, see Section 2.1 and Section 3. On the other hand, it is indicated in [9] (on page 400) that as long as $\alpha_{l,ij}$ is within $O(h)$ perturbation of $H_1(\nabla\phi_h(x_{i-1/2},y_j))$ (similar argument goes to $\alpha_{r,ij}$, $\alpha_{b,ij}$ and $\alpha_{t,ij}$), the accuracy developed in [9] is guaranteed by truncation error analysis and this is confirmed by numerical tests. Our choice of $\alpha_{l,ij},\alpha_{r,ij}$, $\alpha_{b,ij}$ and $\alpha_{t,ij}$ satisfies such conditions therefore will maintain the accuracy results in [9].*

If $I_{ij}\cap(\partial\Omega\backslash\Gamma)$ is not empty for some (i,j), we modify (2.5)-(2.8) in this cell to carry out the outflow boundary condition. For instance in the cell I_{ij} where $x_{i+\frac{1}{2}}$ is aligned with the domain boundary and $I_{ij}\cap\Gamma$ is empty, we take $\alpha_{r,ij}=0$. Similar adjustment is made to $\alpha_{l,ij},\alpha_{t,ij},\alpha_{b,ij}$.

For any given (i,j), by taking $w_h = 1, X_i, Y_j$ on I_{ij} and $w_h = 0$ elsewhere, the DG formulation (2.4) can be converted from the integral form to the following nonlinear algebraic system:

$$\sqrt{u_{ij}^2+v_{ij}^2} + \gamma_{ij}\bar{\phi}_{ij} + \beta_{ij}u_{ij} + \lambda_{ij}v_{ij} = R_{1,ij} \tag{2.10}$$
$$12\beta_{ij}\bar{\phi}_{ij} + \zeta_{ij}u_{ij} = R_{2,ij} \tag{2.11}$$
$$12\lambda_{ij}\bar{\phi}_{ij} + \eta_{ij}v_{ij} = R_{3,ij} \tag{2.12}$$

where

$$\beta_{ij} = -\frac{1}{2}(\alpha_{r,ij}+\alpha_{l,ij}), \qquad \lambda_{ij} = -\frac{1}{2}(\alpha_{t,ij}+\alpha_{b,ij})$$
$$\gamma_{ij} = \alpha_{l,ij} - \alpha_{r,ij} + \alpha_{b,ij} - \alpha_{t,ij}$$
$$\zeta_{ij} = -3\alpha_{r,ij} + 3\alpha_{l,ij} - \alpha_{t,ij} + \alpha_{b,ij}$$
$$\eta_{ij} = -3\alpha_{t,ij} + 3\alpha_{b,ij} - \alpha_{r,ij} + \alpha_{l,ij}$$

and

$$
\begin{aligned}
R_{1,ij} =& \frac{1}{h}\int_{I_{ij}} f(x,y)dxdy - \alpha_{r,ij}(\bar{\phi}_{i+1,j} - \frac{1}{2}u_{i+1,j}) + \alpha_{l,ij}(\bar{\phi}_{i-1,j} + \frac{1}{2}u_{i-1,j}) \\
& - \alpha_{t,ij}(\bar{\phi}_{i,j+1} - \frac{1}{2}v_{i,j+1}) + \alpha_{b,ij}(\bar{\phi}_{i,j-1} + \frac{1}{2}v_{i,j-1}) \\
R_{2,ij} =& \frac{12}{h}\int_{I_{ij}} f(x,y)X_i dxdy - 6\alpha_{r,ij}(\bar{\phi}_{i+1,j} - \frac{1}{2}u_{i+1,j}) - 6\alpha_{l,ij}(\bar{\phi}_{i-1,j} + \frac{1}{2}u_{i-1,j}) - \alpha_{t,ij}u_{i,j+1} + \alpha_{b,ij}u_{i,j-1} \\
R_{3,ij} =& \frac{12}{h}\int_{I_{ij}} f(x,y)Y_j dxdy - 6\alpha_{t,ij}(\bar{\phi}_{i,j+1} - \frac{1}{2}v_{i,j+1}) - 6\alpha_{b,ij}(\bar{\phi}_{i,j-1} + \frac{1}{2}v_{i,j-1}) - \alpha_{r,ij}v_{i+1,j} + \alpha_{l,ij}v_{i-1,j}
\end{aligned}
$$

To solve the coupled nonlinear system (2.10)-(2.12) with $1 \le i \le I, 1 \le j \le J$ efficiently, we are going to propose a fast sweeping method which uses block Gauss-Seidel iterations with alternating directions of sweepings. Based on the basic steps of the fast sweeping methods in [47, 54], we first need to define a local solver to compute ϕ_h in I_{ij} based on the DG discretization (2.10)-(2.12), provided that the solution in the remaining region is known.

2.1 DG local solver

Note that $\beta_{ij}, \lambda_{ij}, \gamma_{ij}, \zeta_{ij}, \eta_{ij}$ in (2.10)-(2.12) are functions of $\alpha_{r,ij}$, $\alpha_{l,ij}$, $\alpha_{t,ij}$, and $\alpha_{b,ij}$ defined in (2.5)-(2.8), which are independent of $\phi_h|_{I_{ij}}$. In addition, $R_{1,ij}$, $R_{2,ij}$ and $R_{3,ij}$ in (2.10)-(2.12) are independent of $\phi_h|_{I_{ij}}$. These indicate that the system of equations (2.10)-(2.12) provides a local representation of $(\bar{\phi}_{ij}, u_{ij}, v_{ij})$ in terms of the unknown solution ϕ_h in the *neighboring* cells and the source data. Moreover, (2.10)-(2.12) is quadratic with respect to $(\bar{\phi}_{ij}, u_{ij}, v_{ij})$.

Given (i,j), let $\phi_h^{new}|_{I_{ij}} = \bar{\phi}_{ij}^{new} + u_{ij}^{new}X_i + v_{ij}^{new}Y_j$ denote the candidate for the update of the numerical solution ϕ_h in I_{ij} and $\phi_h|_{I_{kl}} = \bar{\phi}_{kl} + u_{kl}X_i + v_{kl}Y_j$ denote the current numerical solution in any (k,l)-th cell, then we compute ϕ_h^{new} according to the following formula

$$\sqrt{(u_{ij}^{new})^2 + (v_{ij}^{new})^2} + \gamma_{ij}\bar{\phi}_{ij}^{new} + \beta_{ij}u_{ij}^{new} + \lambda_{ij}v_{ij}^{new} = R_{1,ij}, \tag{2.13}$$

$$12\beta_{ij}\bar{\phi}_{ij}^{new} + \zeta_{ij}u_{ij}^{new} = R_{2,ij}, \tag{2.14}$$

$$12\lambda_{ij}\bar{\phi}_{ij}^{new} + \eta_{ij}v_{ij}^{new} = R_{3,ij}. \tag{2.15}$$

Note that $\beta_{ij}, \lambda_{ij}, \gamma_{ij}, \zeta_{ij}, \eta_{ij}, R_{1,ij}, R_{2,ij}, R_{3,ij}$ do not depend on the numerical solution in I_{ij}, hence (2.13)-(2.15) defines a quadratic system for $(\bar{\phi}_{ij}^{new}, u_{ij}^{new}, v_{ij}^{new})$ which can be solved explicitly. We can also show that the parameters $\gamma_{ij}, \zeta_{ij}, \eta_{ij}$ are either all zero or all positive. Since the system (2.13)-(2.15) in principle can have none or two sets of solutions of $(\bar{\phi}_{ij}^{new}, u_{ij}^{new}, v_{ij}^{new})$, we take the following strategy to decide when to *reject* or to *accept* the candidates of $(\bar{\phi}_{ij}^{new}, u_{ij}^{new}, v_{ij}^{new})$ computed from (2.13)-(2.15), based on the causality of the Eikonal equation:

1. When $\gamma_{ij}, \zeta_{ij}, \eta_{ij}$ are all zero, we will not update the solution in this cell;

2. When $\gamma_{ij}, \zeta_{ij}, \eta_{ij}$ are all positive, if we get from (2.13)-(2.15)

   - two sets of *real* solutions $(\bar{\phi}_{ij}^{new}, u_{ij}^{new}, v_{ij}^{new})$: then update $\phi_h|_{I_{ij}}$ with $\phi_h^{new}|_{I_{ij}}$ if the following conditions are satisfied:

$$\begin{cases} \bar{\phi}_{ij}^{new} \geq \bar{\phi}_{i-1,j} \text{ and } \quad u_{ij}^{new} \geq 0, \quad \text{if } \alpha_{l,ij} > 0 \\ \bar{\phi}_{ij}^{new} \geq \bar{\phi}_{i+1,j} \text{ and } \quad u_{ij}^{new} \leq 0, \quad \text{if } \alpha_{r,ij} < 0 \\ \bar{\phi}_{ij}^{new} \geq \bar{\phi}_{i,j-1} \text{ and } \quad v_{ij}^{new} \geq 0, \quad \text{if } \alpha_{b,ij} > 0 \\ \bar{\phi}_{ij}^{new} \geq \bar{\phi}_{i,j+1} \text{ and } \quad v_{ij}^{new} \leq 0, \quad \text{if } \alpha_{t,ij} < 0 \end{cases} \tag{2.16}$$

     If both sets of the solutions satisfy the above conditions, choose the one with the smaller value of $\bar{\phi}_{ij}^{new}$;

   - two sets of *complex* solutions: we will not update the solution in this cell.

Remark 2.2 *The strategy regarding accepting or rejecting the candidate solutions in* (2.16) *is to ensure the consistency of the causality. For example, if we follow the discussion after the definition of* $\alpha_{l,ij}$ *to* $\alpha_{t,ij}$ *in* (2.5)-(2.8) *when the information is propagating from the bottom-left to the top-right in the neighborhood of the cell* I_{ij}*, we expect* $\alpha_{l,ij} > 0$ *and* $\alpha_{b,ij} > 0$*, therefore we accept the candidate solution* $(\bar{\phi}_{ij}^{new}, u_{ij}^{new}, v_{ij}^{new})$ *which is consistent with the causality:* $\bar{\phi}_{ij}^{new} \geq \bar{\phi}_{i-1,j}$*,* $\bar{\phi}_{ij}^{new} \geq \bar{\phi}_{i,j-1}$ *as well as* $u_{ij}^{new} \geq 0, v_{ij}^{new} \geq 0$*. That is, along the characteristic the solution is nondecreasing. Note that comparing average values in cells does not necessarily give the correct information of the characteristics or monotonicity of the solution. That explains why oscillations can occur (see Example 7 in Section 3) if the derivative criterion is not used, i.e., the average is monotone even if the solution is not. However, the sign of derivatives become subtle near local extrema or shocks. So in* (2.16) *we use both the cell average and the derivatives of the solution to decide the acceptance or the rejection of a candidate for the solution update to ensure the causality consistency.*

2.2 Hybrid DG local solver

With the local solver defined in the previous section as the building block, the numerical experiments show that the straightforward application of the fast sweeping method as in [47, 54] based on the DG discretization (2.4) may not produce satisfactory results all the time. There are two issues that we need to pay more attention. The first is the initial data. In general iterative methods for

nonlinear problems need a good initial guess for convergence, in particular when they are based on high order discretizations. On the other hand, the fast sweeping method based on the first order Godunov scheme has been shown monotone, convergent and efficient for any initial data that are super-solution (or sub-solution) of the true viscosity solution [54]. So we use the first order scheme to provide a good initial guess for our high order method just as in [53], see Section 2.3 for the complete description.

The second issue is that we enforce a quite strong causality condition (2.16) when we update the value in a cell using the local DG solver. Near shocks, where characteristics intersect, or near points where ϕ_x or ϕ_y are close to zero, causality issue becomes more subtle. The local DG solver may not be able to provide a solution satisfying (2.16). Hence in those cells where the second order local DG solver can not provide a solution we switch back to a first order finite difference type Godunov scheme. The hope is that more accurate information propagated through the DG solver will be felt at all points.

However, those conditions in (2.16) will enforce propagation of information in the right direction and this will significantly reduce the number of iterations of our final algorithm in Section 2.4. For example, such conditions were not used in [53] and more iterations are needed in general.

One concern for hybridizing the two local solvers is a possible reduction of the accuracy. In practice the first order local solver is only used rarely as explained above. This phenomenon is observed in our numerical examples. Here we classify the convergence of our method into three cases.

C1: The first order finite difference type local solver is never used;

C2: The first order finite difference type local solver is used before the iterations converge. By the time the iterations converge, the first order local solver is not used.

C3: The first order finite difference type local solver is used in the whole process of the computation. By the time the iterations converge, the first order local solver is used in certain cells and the solution is not further updated if the iterations continue. The ratio between the number of the cells in which the first order local solver is used and the number of the total cells is decreasing when h decreases, and the numerical evidence shows that this ratio is about $O(h)$.

Remark 2.3 *Numerical examples show that in the case C3 described above, the first order local solver is used in $O(h)$ percentage of the total number of cells which are in the neighborhood of the*

shock location, i.e., where characteristics intersect, and hence it does not pollute solutions in other region. Therefore the method still achieves second order accuracy in the L^1 norm.

The hybrid local solver is defined as follows: given (i, j)

(1) When the DG local solver defined in the previous section provides an update in I_{ij}, the update is accepted;

(2) When the DG local solver defined in the previous section provides no update in I_{ij}, the following finite difference based local solver is used instead: let

$$a = \min(\bar{\phi}_{i-1,j}, \bar{\phi}_{i+1,j}),\ b = \min(\bar{\phi}_{i,j-1}, \bar{\phi}_{i,j+1}),\ f_{i,j} = f(x_i, y_j),$$

we then update the solution in the cell I_{ij} as the following:

- If $|a - b| \geq f_{i,j}h$, then

$$\bar{\phi}_{i,j}^{new} = \min(a, b) + f_{i,j}h$$

and

$$\begin{cases} u_{ij}^{new} = \bar{\phi}_{ij}^{new} - \bar{\phi}_{i-1,j},\ \ v_{ij}^{new} = 0, & \text{if } a = \min(a,b) = \bar{\phi}_{i-1,j} < \bar{\phi}_{i+1,j} \\ u_{ij}^{new} = \bar{\phi}_{i+1,j} - \bar{\phi}_{ij}^{new},\ \ v_{ij}^{new} = 0, & \text{if } a = \min(a,b) = \bar{\phi}_{i+1,j} \leq \bar{\phi}_{i-1,j} \\ u_{ij}^{new} = 0,\ \ v_{ij}^{new} = \bar{\phi}_{ij}^{new} - \bar{\phi}_{i,j-1}, & \text{if } b = \min(a,b) = \bar{\phi}_{i,j-1} < \bar{\phi}_{i,j+1} \\ u_{ij}^{new} = 0,\ \ v_{ij}^{new} = \bar{\phi}_{i,j+1} - \bar{\phi}_{ij}^{new}, & \text{if } b = \min(a,b) = \bar{\phi}_{i,j+1} \leq \bar{\phi}_{i,j-1} \end{cases}$$

- If $|a - b| < f_{i,j}h$, then

$$\bar{\phi}_{i,j}^{new} = \frac{a + b + \sqrt{2f_{i,j}^2h^2 - (a-b)^2}}{2}$$

and

$$u_{ij}^{new} = \begin{cases} \bar{\phi}_{ij}^{new} - \bar{\phi}_{i-1,j}, & \text{if } \bar{\phi}_{i-1,j} < \bar{\phi}_{i+1,j} \\ \bar{\phi}_{i+1,j} - \bar{\phi}_{ij}^{new}, & \text{if } \bar{\phi}_{i-1,j} \geq \bar{\phi}_{i+1,j} \end{cases}$$

$$v_{ij}^{new} = \begin{cases} \bar{\phi}_{ij}^{new} - \bar{\phi}_{i,j-1}, & \text{if } \bar{\phi}_{i,j-1} < \bar{\phi}_{i,j+1} \\ \bar{\phi}_{i,j+1} - \bar{\phi}_{ij}^{new}, & \text{if } \bar{\phi}_{i,j-1} \geq \bar{\phi}_{i,j+1} \end{cases}$$

Remark 2.4 *When the first order finite difference based local solver is used in this hybrid local solver, the way to compute the update for $\bar{\phi}_{ij}$ is the same as that used in [54]; and the way to compute the updates for u_{ij} and v_{ij} is based on how the derivatives ϕ_x and ϕ_y in the Eikonal equation are approximated in the Godunov finite difference discretization as in [54].*

2.3 Initial guess and boundary condition

In order to complete the fast sweeping algorithm based on the hybrid DG local solver defined above, we also need to specify how to impose the boundary conditions and how to set up the initial guess for the iteration.

To assign the boundary conditions, we pre-assign the solution in the *boundary cells* (certain cells around Γ, which will be specified in Section 3 for each example), and the solution in these cells will not be updated during the iterations. In the numerical experiments, the following approach is taken: for any I_{ij} which is identified as the *boundary cell*, $\phi_h|_{I_{ij}} \in P^1(I_{ij})$ is the least squares fit of the exact solution ϕ, satisfying

$$J_{ij}(\phi_h - \phi) = \min_{w_h \in P^1(I_{ij})} J_{ij}(w_h - \phi)$$

where

$$J_{ij}(\psi) = (\psi_{i-\frac{1}{2},j-\frac{1}{2}})^2 + (\psi_{i-\frac{1}{2},j+\frac{1}{2}})^2 + (\psi_{i+\frac{1}{2},j-\frac{1}{2}})^2 + (\psi_{i+\frac{1}{2},j+\frac{1}{2}})^2,$$

and $\psi_{i-\frac{1}{2},j-\frac{1}{2}}$ denotes $\psi(x_{i-\frac{1}{2}}, y_{j-\frac{1}{2}})$. One can easily derive the following explicit formula for computing this least squares fit $\phi_h|_{I_{ij}} = \bar{\phi}_{ij} + u_{ij}X_i + v_{ij}Y_j$ from the exact solution ϕ:

$$\bar{\phi}_{ij} = \frac{1}{4}(\phi_{i-\frac{1}{2},j-\frac{1}{2}} + \phi_{i-\frac{1}{2},j+\frac{1}{2}} + \phi_{i+\frac{1}{2},j-\frac{1}{2}} + \phi_{i+\frac{1}{2},j+\frac{1}{2}}), \tag{2.17}$$

$$u_{ij} = \frac{1}{2}(\phi_{i+\frac{1}{2},j-\frac{1}{2}} - \phi_{i-\frac{1}{2},j-\frac{1}{2}} + \phi_{i+\frac{1}{2},j+\frac{1}{2}} - \phi_{i-\frac{1}{2},j+\frac{1}{2}}), \tag{2.18}$$

$$v_{ij} = \frac{1}{2}(\phi_{i-\frac{1}{2},j+\frac{1}{2}} - \phi_{i-\frac{1}{2},j-\frac{1}{2}} + \phi_{i+\frac{1}{2},j+\frac{1}{2}} - \phi_{i+\frac{1}{2},j-\frac{1}{2}}). \tag{2.19}$$

This procedure provides second order accurate boundary conditions provided that ϕ is sufficiently smooth.

Remark 2.5 *In real implementation the exact solution is known only along some curves or at some isolated points. One can use (a) interpolation or extrapolation, or (b) ray tracing method, or (c) a first order approximation of the solution on a finer mesh, in a neighborhood of the boundary* Γ *to provide the second order boundary condition approximation. Then the method developed here can be used to compute the solution efficiently in the whole domain.*

Once the boundary condition is prescribed in the boundary cells, the initial guess of the numerical solution in the remaining region is given as follows: we compute the first order Godunov finite difference based fast sweeping approximation ϕ_{FD} [47, 54] to ϕ with the unknowns defined at $(x_{i-\frac{1}{2}}, y_{j-\frac{1}{2}})$ for all (i, j), then we take the least squares fit of ϕ_{FD} as the initial guess ϕ_h. That

is, ϕ_h is computed by replacing ϕ with ϕ_{FD} in (2.17)-(2.19). Note this initial guess is first order accurate.

2.4 Algorithm

Now by combining the hybrid DG local solver, the boundary condition treatment and the initial guess described in Section 2.1-2.3 with the block Gauss Seidel iteration with alternating sweeping orderings, we can summarize the main algorithm for solving (1.1)-(1.2):

1. Initialization: in the cells around Γ, assign the least squares fit of the exact solution or approximation solution and the solution in these cells will not be updated in the iterations; in the remaining region, initialize the solution by the least squares fit of the solution from the first order finite difference scheme, see Section 2.3.

2. Update ϕ_h on I_{ij} by block Gauss Seidel iteration with four alternating sweeping orderings:

$$(1)\ i = 1 : I; j = 1 : J(\rightrightarrows\uparrow\uparrow) \qquad (2)\ i = I : 1; j = 1 : J(\leftrightarrows\uparrow\uparrow)$$

$$(3)\ i = I : 1; j = J : 1(\leftrightarrows\downarrow\downarrow) \qquad (4)\ i = 1 : I; j = J : 1(\rightrightarrows\downarrow\downarrow)$$

   Solve $\phi_h|_{I_{ij}}^{new}$ by the hybrid DG local solver defined in Section 2.2.

3. Convergence: $\delta > 0$ is a given small number. For any sweep, if we denote the solutions before and after the sweep as ϕ_h^{old} and ϕ_h^{new}, and if

$$||\phi_h^{new} - \phi_h^{old}||_{L^1(\Omega)} < \delta, \tag{2.20}$$

   the algorithm converges and stops.

Remark 2.6 *The Gauss Seidel iteration we use is* block *Gauss Seidel as each time the numerical solution in I_{ij}, which contains three unknowns, is considered for update.*

Remark 2.7 *We use L^1 norm in determining the convergence of the algorithm in* (2.20) *since this norm is commonly used in analyzing hyperbolic problems, and it is less affected by the degeneracy in $O(h)$ percentage of the total number of cells.*

3 Numerical Examples

In this section, we demonstrate the performance of the algorithm defined in Section 2.4 through some typical two dimensional examples, and δ in the stopping criteria is taken as 10^{-14}. The numerical errors in L^1 norm, L^2 norm and L^∞ norm are presented. In order to make a fair comparison among these errors, the following normalization is used:

$$||w||_{L^1(\Omega_c)} = \frac{\int_{\Omega_c} |w(x,y)|dxdy}{|\Omega_c|}, \qquad ||w||_{L^2(\Omega_c)} = \left(\frac{\int_{\Omega_c} |w(x,y)|^2 dxdy}{|\Omega_c|}\right)^{1/2},$$

where Ω_c is the domain in which the errors are computed, and $|\Omega_c|$ denotes the area of Ω_c. We also report which situation among C1, C2 and C3 described in Section 2.2 happens for each example. In all tables, # stands for the number of sweeps needed for the convergence, and it does not include the sweep in which the convergence is noticed. For those examples whose exact solutions are known, boundary conditions are given by the least squares fit of the exact solutions as described in Section 2.3.

Example 1: $\Omega = [-1,1]^2$, $\Gamma = \{(0,0)\}$ and $f(x,y) = 1$. The exact solution

$$\phi(x,y) = \sqrt{x^2+y^2}$$

is the distance function from Γ, and all the characteristics start from the origin and propagate outward along all directions. Note the origin Γ is a singularity.

We pre-assign the boundary conditions in a 0.2-length square box around Γ in which the solution is not updated during iterations. This is the so-called wrapping technique [53, 39] which is used in order to correctly measure the order of accuracy for the point source problem. The numerical errors and convergence orders are computed in the domain without the 0.2-length box around Γ and the results are reported in Table 3.1. The same table also includes the number of sweeps needed for the convergence, and indicates which situation among C1, C2 and C3 described in Section 2.2 happens in the computation. The results show that the finite difference based local solver is never used through the whole computation (except when $n = 20$), and the proposed scheme is second order accurate. Only four sweeps are needed for the convergence.

As a comparison, in Table 3.2 we include the results by the first order Godunov finite difference based fast sweeping method [54]. One can see that our proposed algorithm provides more accurate approximations. Since the second order local solver is more expensive than the first order local solver in one cell, we further examine the efficiency of these two methods by studying the relation

Table 3.1: Example 1. $\Gamma = \{(0,0)\}$ and $f(x,y) = 1$. The boundary condition is pre-assigned in the 0.2-length square box around Γ, and the errors are computed outside the box. $n = 2/h$.

n	L^1 error	order	L^2 error	order	L^∞ error	order	#	type
20	5.08E-02	-	7.74E-02	-	2.74E-01	-	4	C2
40	4.28E-03	3.57	8.85E-03	3.13	5.02E-02	2.44	4	C1
80	4.14E-04	3.37	1.06E-03	3.07	9.28E-03	2.44	4	C1
160	5.93E-05	2.80	1.56E-04	2.76	1.99E-03	2.22	4	C1
320	1.04E-05	2.51	2.60E-05	2.59	4.62E-04	2.11	4	C1

between the number of the involved $\times$, /, and $\sqrt{\ }$ operations and the accuracy of the method. We assume the dominating computational cost is in the iterations (including the iterations in order to get the first order initial guess), therefore we choose not to count the operations related to pre-processing and post-processing the data to simplify the comparison. Note that both methods need the same number of sweeps for the iterations to converge in this example. In Figure 3.1 we plot the operations per sweep versus the L^1 errors in ϕ. The operations in initializing the proposed algorithm using the first order method has also been taken into account. It shows that in order to achieve a relatively low accuracy (lower than 10^{-2} in L^1 error for this example), the first order scheme needs fewer operations than the second order algorithm proposed here. But in order to achieve a better accuracy (higher than 5×10^{-3} in L^1 error for this example), the second order algorithm needs fewer operations than the first order algorithm, and the higher the accuracy we want to achieve, the more efficient our DG algorithm is compared to the fist order method. We want to point out that the results in Table 3.1 and Table 3.2 are computed in different ways. If we take the error in L^1 norm as an example, the errors in Table 3.1 for the DG approximation ϕ_h by the proposed algorithm are computed as follows:

$$||\phi - \phi_h||_{L^1(\Omega)} = \sum_{i=1}^{I} \sum_{j=1}^{J} \int_{I_{ij}} |\phi(x,y) - \phi_h(x,y)| dxdy$$

where $\int_{I_{ij}} |\phi(x,y) - \phi_h(x,y)| dxdy$ is further approximated by Gaussian quadrature which is exact for polynomials of degree at most 5. For the numerical solution ϕ_h resulted from the Godunov finite difference based fast sweeping method, we only know its values at the grid points $(x_{i-1/2}, y_{j-1/2})$, therefore the errors in Table 3.2 are computed by

$$||\phi - \phi_h||_{L^1(\Omega)} \approx \sum_{i=1}^{I} \sum_{j=1}^{J} |\phi(x_{i-1/2}, y_{j-1/2}) - \phi_h(x_{i-1/2}, y_{j-1/2})| \Delta x_i \Delta y_j.$$

This note also applies to Example 4 and Example 6 when similar comparison is made.

Table 3.2: Example 1. $\Gamma = \{(0,0)\}$ and $f(x,y) = 1$. The boundary condition is pre-assigned in the 0.2-length square box around Γ. The computation is by the first order Godunov finite difference fast sweeping method. $n = 2/h$.

n	L^1 error	order	L^2 error	order	L^∞ error	order	#
20	3.33E-02	-	3.86E-02	-	6.81E-02	-	4
40	1.86E-02	0.84	2.15E-02	0.85	3.79E-02	0.84	4
80	9.90E-03	0.91	1.14E-02	0.91	2.02E-02	0.91	4
160	5.12E-03	0.95	5.89E-03	0.95	1.04E-02	0.96	4
320	2.60E-03	0.98	2.99E-03	0.98	5.23E-03	0.99	4

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

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

The exact solution is

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

Though the solution is smooth, $f(0,0) = 0$ indicates that the equation is degenerate at the origin, i.e., the propagating speed is infinity.

In the computation, the boundary condition is pre-assigned either in a fixed region around Γ (0.2×0.2 square box) or in a fixed number of cells around Γ ($4h \times 4h$ square box). The results collected in Table 3.3 show that four sweeps are needed for the convergence and the second order convergence rate is achieved. The slightly lower convergence orders observed in the second part of Table 3.3, when only a region of size $O(h)$ is taken out near the source, is due to the degeneracy of the Eikonal equation ($f(0,0) = 0$ for this example) or the singularity of the solution at source point. Any numerical scheme will create larger errors near the singularity and the error incurred near the source point will propagate to and pollute the computation domain [54, 53, 39]. For this example the finite difference type local solver is used before the iterations converge and it is never used afterward, i.e. case 2 as discussed in Section 2.2.

Since our proposed algorithm is second order accurate and ϕ_h in each element is a polynomial function, the piecewise defined gradient of ϕ_h naturally provides a first order approximation for the gradient of the exact solution ϕ, see Table 3.4.

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

$$\phi(x,y) = |\sqrt{x^2+y^2} - R_s|$$

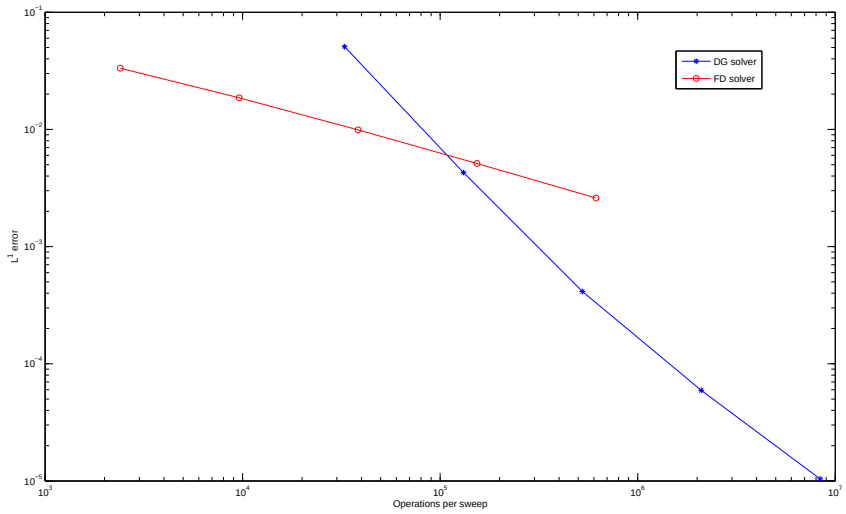

Figure 3.1: Example 1. Operations count, $\mathcal{C}$, per sweep versus the L^1 errors in ϕ. The operations per sweep when the first order finite difference solver is used: $\mathcal{C} = C_1 * n^2$; the operations per sweep when the second order DG solver is used (including obtaining the first order initial guess): $\mathcal{C} = (C_1 + C_2) * n^2$. Here $C_1 = 6$, $C_2 = 76$ are the operations involved per element by the first order finite difference local solver and the DG local solver respectively. "$\circ$" and "$\star$" along the curves are for different n with $n = 20, 40, 80, 160, 320$ from left to right. Operations including $*, /, \sqrt{}$ are counted without $+$ and $-$.

is the distance function from Γ.

The boundary condition is pre-assigned in the $2\sqrt{2}h$-distanced cells from Γ. The errors, convergence orders and the number of sweeps needed for convergence are included in Table 3.5. For this example, the boundary Γ is smooth and separates the interior and exterior regions. The origin $(0, 0)$ is the only singularity of the solution. The full second order convergence rate can been seen only when the errors are measured away from the origin. Four sweeps are needed for the convergence. The first order finite difference based local solver is used before the convergence is achieved when $n = 160, 320$, and it is never used afterward.

Example 4: $\Omega = [-1, 1]^2$, and Γ consists of two circles with centers $(0.5, 0.5)$, $(-0.5, -0.5)$ and the radius $R_s = 0.3$, and $f(x, y) = 1$. The exact solution is

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

which is the distance function from Γ.

For this example, the centers of two circles and {(x,y): x+y=0}, the equally distanced line from

Table 3.3: Example 2. The exact solution is $\phi(x,y) = -\cos(\frac{\pi}{2}x) - \cos(\frac{\pi}{2}y)$ on $[-1,1]^2$ with $\Gamma = \{(0,0)\}$. The boundary condition is pre-assigned in the L-length square box around Γ. $n = 2/h$.

n	L^1 error	order	L^2 error	order	L^∞ error	order	#	type
$L = 0.2$								
20	7.00E-03	-	1.29E-02	-	6.48E-02	-	4	C1
40	7.83E-04	3.16	1.53E-03	3.08	1.23E-02	2.40	4	C2
80	1.56E-04	2.33	2.78E-04	2.46	2.96E-03	2.05	4	C2
160	3.38E-05	2.20	5.54E-05	2.33	7.34E-04	2.01	4	C2
320	7.73E-06	2.13	1.17E-05	2.24	1.83E-04	2.00	4	C2
$L = 4h$								
20	2.10E-03	-	3.73E-03		2.64E-02	-	4	C1
40	7.83E-04	1.43	1.53E-03	1.29	1.23E-02	1.11	4	C2
80	2.95E-04	1.41	5.75E-04	1.41	5.10E-03	1.26	4	C2
160	1.01E-04	1.54	1.98E-04	1.54	1.98E-03	1.37	4	C2
320	3.23E-05	1.65	6.34E-05	1.64	7.27E-04	1.45	4	C2

the two circles Γ are singularities, i.e., where characteristics intersect. In the computation, the boundary condition is pre-assigned in the $2\sqrt{2}h$-distanced cells from Γ. The errors, convergence orders and the number of sweeps needed for convergence are included in Table 3.6. Not like in all the previous examples, the first order finite difference local solver is used all the way till the iterations converge. When the convergence is achieved, the first order local solver is used in some cells along the equally distanced line $\{(x,y) : x + y = 0\}$ where characteristics intersect. See Figure 3.2. However, the number of cells in which the first order finite difference local solver is used is no more than $O(h)$ percentage of the total number of cells. This explains the second order global convergence rate in L^1 norm and first order global convergence rate in L^∞. On the other hand, the full second order convergence rate can be seen when the errors are measured away from the centers of the two circles, and the equally distanced line from these two circles. It is interesting to note that the singularities at the centers of the two circles are quite different from the singularities at the equal distance line. At the centers infinite many characteristics from all directions intersect. On the other hand, the characteristics only intersect pairwise at the equal distance line. So we have an extra difficulty to resolve infinite many directions of characteristics at the centers. Similar to the point source scenario, a fixed region has to be taken out near the circle centers to see second order convergence, while only a $O(h)$ neighborhood of the equal distance line needs to be taken out. Although the characteristics in this example are straight lines, six sweeps are needed for convergence to machine zero instead of four sweeps due to the presence of shocks. Closer examination shows

Table 3.4: Example 2. Convergence of the gradient of ϕ_h to $\nabla\phi$. The exact solution is $\phi(x,y) = -\cos(\frac{\pi}{2}x) - \cos(\frac{\pi}{2}y)$ on $[-1,1]^2$ with $\Gamma = \{(0,0)\}$. The boundary condition is pre-assigned in the L-length square box around Γ. $n = 2/h$.

n	L^1 error	order	L^2 error	order	L^∞ error	order
$L = 0.2$						
20	1.36E-01	-	1.47E-01	-	4.54E-01	-
40	5.21E-02	1.38	5.42E-02	1.44	2.11E-01	1.11
80	2.41E-02	1.11	2.40E-02	1.18	1.06E-01	0.99
160	1.16E-02	1.05	1.12E-02	1.10	5.29E-02	1.00
320	5.68E-03	1.03	5.38E-03	1.06	2.65E-02	1.00
$L = 4h$						
20	1.02E-01	-	1.03E-01		3.36E-01	-
40	5.21E-02	0.97	5.42E-02	0.92	2.11E-01	0.67
80	2.59E-02	1.01	2.75E-02	0.98	1.27E-01	0.73
160	1.27E-02	1.03	1.36E-02	1.02	7.41E-02	0.78
320	6.19E-03	1.04	6.56E-03	1.05	4.24E-02	0.81

that by the end of the fourth sweep, the solution is settled everywhere except in a few cells near the shock location. Such cells are among those which eventually involve the first order local solver when the convergence is achieved, see Figure 3.2. Similar phenomenon occurs for the first order monotone scheme although the accuracy has already been achieved after four iterations, i.e., the modifications to balance information from different characteristics near shocks after four iterations are really small compared to the approximation error of the discretization scheme. This is discussed in [54].

Compared to the results in Table 3.7 using first order scheme [54], we see that the global L^∞ error is comparable. This is due to the fact that the first order local solver is used in certain cells, e.g., near the shocks. As shown by a constructed example in [54], at most first order accuracy can be achieved in L^∞ norm at the shock location. However, for the integral L^1 or L^2 norm we see a higher order of convergence and much smaller error than the first order scheme even including those singularities. This is due to the fact that the number of cells in which the first order local solver is used is no more than $O(h)$ percentage of the number of the total cells.

Example 5 (shape-from-shading): $\Omega = [-1,1]^2$, $\Gamma = \partial\Omega$. The solution of this example is the shape function, which has the brightness $I(x,y) = 1/\sqrt{1+f(x,y)^2}$ under vertical lighting. See [41] for details. We consider the following two cases:

Table 3.5: Example 3. Γ is a circle centered at $(0,0)$ with radius 0.5. $f(x,y)=1$. The boundary condition is pre-assigned in the cells within $2\sqrt{2}h$-distance from Γ. Global error: errors computed outside the boundary condition region; regional error: errors computed outside the boundary condition region and outside the 0.1-length box around the origin. $n=2/h$.

n	L^1 error	order	L^2 error	order	L^∞ error	order	#	type
global error								
20	1.07E-03	-	1.73E-03	-	1.73E-02	-	4	C1
40	2.94E-04	1.86	4.65E-04	1.90	8.12E-03	1.09	4	C1
80	7.68E-05	1.93	1.26E-04	1.89	4.03E-03	1.01	4	C1
160	1.98E-05	1.96	3.41E-05	1.88	2.01E-03	1.00	4	C2
320	5.06E-06	1.97	9.22E-06	1.89	1.01E-03	1.00	4	C2
regional error								
20	9.51E-04	-	1.33E-03	-	7.57E-03	-	4	C1
40	2.75E-04	1.79	3.80E-04	1.81	3.88E-03	0.96	4	C1
80	7.18E-05	1.94	9.81E-05	1.95	9.29E-04	2.06	4	C1
160	1.84E-05	1.96	2.55E-05	1.94	2.35E-04	1.98	4	C2
320	4.69E-06	1.97	6.55E-06	1.96	5.97E-05	1.98	4	C2

Case 1:

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

and the exact solution is

$$\phi(x,y)=(1-|x|)(1-|y|).$$

Note the solution is not smooth but piecewise bi-linear. With the boundary condition pre-assigned in the boundary cells (those cells such that $I_{ij}\cap\Gamma$ is not empty), the proposed scheme gives the second order accurate approximation and four sweeps are needed for the convergence, see Table 3.8. For this example, the finite difference based local solver is never used in the computation.

Case 2:

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

and the exact solution is

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

which is bi-quadratic.

Though the solution is smooth, $f(0,0)=0$ indicates that the equation is degenerate at the origin. We pre-assign the boundary condition in both the boundary cells (those cells such that $I_{ij}\cap\Gamma$ is not empty) *and* the cells inside a 0.4-length square box around the origin. The errors, convergence orders and the number of sweeps needed for convergence are included in Table 3.9. The

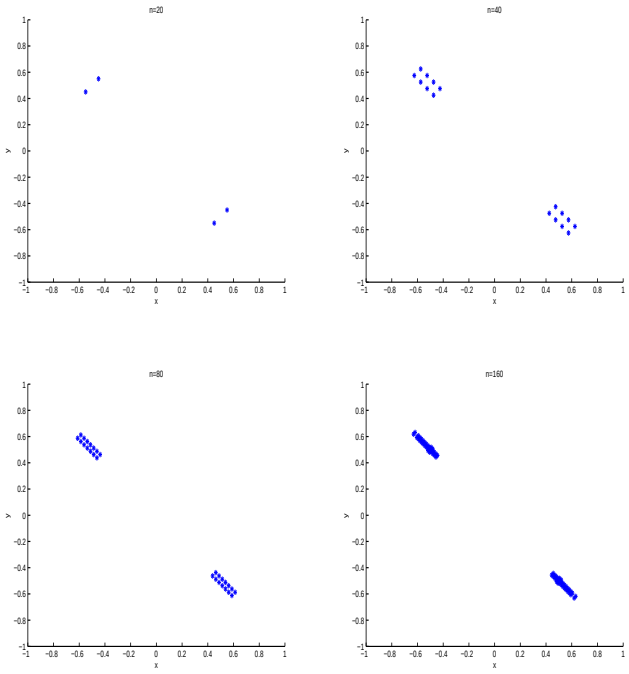

Figure 3.2: Example 4. The location of those cells where the first order local solver is used by the time the convergence is achieved.

results show that the method is second order accurate, and the convergence is achieved quickly. In the computation, the finite difference type local solver is used before the convergence is achieved, and it is never used afterward.

In Figure 3.3, we plot the numerical solutions ϕ_h on the 80×80 mesh for both the smooth and non-smooth examples.

Example 6 (shape-from-shading): $\Omega = [0,1]^2$, $\Gamma = \partial\Omega \cup \{(\frac{1}{4},\frac{1}{4}),(\frac{1}{4},\frac{3}{4}),(\frac{3}{4},\frac{1}{4}),(\frac{3}{4},\frac{3}{4}),(\frac{1}{2},\frac{1}{2})\}$, and

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

The solution for this problem is the shape function, which has the brightness $I(x,y) = 1/\sqrt{1+f(x,y)^2}$ under vertical lighting. See [41] for details. On Γ, $\phi(x,y) = g(x,y)$ is given with the following two choices:

Case 1:

$$g(x,y) = \begin{cases} 0, & (x,y) \in \partial\Omega \cup \{(1/2,1/2)\} \\ 1, & (x,y) \in \{(1/4,1/4),(3/4,3/4)\} \\ -1, & (x,y) \in \{(1/4,3/4),(3/4,1/4)\} \end{cases}$$

Table 3.6: Example 4. Γ consists of two circles with centers $(0.5, 0.5)$, $(-0.5, -0.5)$ and the radius $R_s = 0.3$. $f(x, y) = 1$. The boundary condition is pre-assigned in the cells within $2\sqrt{2}h$-distance from Γ. Global error: errors computed outside the boundary condition region; regional error: errors computed in the cells $2\sqrt{2}h$ distance away from the equally distanced line from the circles, 0.1 distance away from the circle centers, and outside the boundary condition region. †= the percentage of the cells in which the finite difference type local solver is used by the time the iterations converge. $n = 2/h$.

n	L^1 error	order	L^2 error	order	L^∞ error	order	#	type	†
global error									
20	2.75E-03	-	6.80E-03	-	7.20E-02	-	6	C3	1.000%
40	6.59E-04	2.06	1.97E-03	1.79	3.66E-02	0.98	6	C3	1.250%
80	1.63E-04	2.02	6.30E-04	1.64	1.84E-02	0.99	6	C3	0.438%
160	4.08E-05	2.00	2.11E-04	1.58	9.25E-03	0.99	6	C3	0.250%
320	1.03E-05	1.99	7.26E-05	1.54	4.63E-03	1.00	6	C3	0.137%
regional error									
20	9.74E-04	-	1.14E-03	-	3.16E-03	-	6	C3	1.000%
40	3.49E-04	1.48	4.74E-04	1.27	3.84E-03	-	6	C3	1.250%
80	9.09E-05	1.94	1.24E-04	1.93	9.31E-04	2.04	6	C3	0.438%
160	2.33E-05	1.96	3.23E-05	1.94	2.37E-04	1.97	6	C3	0.250%
320	5.90E-06	1.98	8.29E-06	1.96	6.03E-05	1.97	6	C3	0.137%

The exact solution for this case is a smooth function:

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

Case 2:

$$g(x, y) = \begin{cases} 0, & (x, y) \in \partial\Omega \\ 1, & (x, y) \in \{(1/4, 1/4), (1/4, 3/4), (3/4, 1/4), (3/4, 3/4)\} \\ 2, & (x, y) \in \{(1/2, 1/2)\} \end{cases}$$

The exact solution for this case is

$$\phi(x, y) = \begin{cases} 1 + \cos(2\pi x)\cos(2\pi y), & \text{if } |x + y - 1| < 1/2, \text{and } |x - y| < 1/2 \\ |\sin(2\pi x)\sin(2\pi y)|, & \text{otherwise} \end{cases}$$

which is non-smooth.

For these two cases, the boundary conditions are pre-assigned in the cells right next to the domain boundary, and in the cells within a 0.1-length square box around each isolated point in Γ. The results are reported in Table 3.10 and Table 3.11, and the numerical solutions on the 80×80 mesh are presented in Figure 3.4. For these examples, the first order finite difference local solver is used all the way till the iterations converge. When the convergence is achieved, the number of cells in which the first order finite difference local solver is still used is no more than $O(h)$ percentage of

Table 3.7: Example 4. Γ consists of two circles with centers $(0.5, 0.5)$, $(-0.5, -0.5)$ and the radius $R_s = 0.3$. $f(x, y) = 1$. The boundary condition is pre-assigned in the cells within $2\sqrt{2}h$-distance from Γ. The computation is by the first order Godunov finite difference fast sweeping method. $n = 2/h$.

n	L^1 error	order	L^2 error	order	L^∞ error	order	#
20	4.40E-03	-	8.21E-03	-	3.14E-02	-	7
40	3.68E-03	0.26	5.37E-03	0.61	2.38E-02	0.40	7
80	2.54E-03	0.53	3.39E-03	0.66	1.58E-02	0.59	7
160	1.49E-03	0.77	1.92E-03	0.82	9.93E-03	0.67	7
320	8.04E-04	0.89	1.02E-03	0.91	6.00E-03	0.73	7

Table 3.8: Example 5. Shape-from-shading, the non-smooth example. The boundary condition is pre-assigned in the boundary cells. $n = 2/h$.

n	L^1 error	order	L^2 error	order	L^∞ error	order	#	type
20	6.44E-04	-	8.63E-04	-	2.53E-03	-	4	C1
40	1.66E-04	1.95	2.20E-04	1.98	6.84E-04	1.89	4	C1
80	4.25E-05	1.97	5.57E-05	1.98	1.82E-04	1.91	4	C1
160	1.08E-05	1.98	1.41E-05	1.98	4.80E-05	1.92	4	C1
320	2.72E-06	1.99	3.54E-06	1.99	1.26E-05	1.93	4	C1

the number of the total cells. This means, asymptotically, the local solver is second order accurate, and this explains the second order convergence rates in L^1 norm in Table 3.10 and Table 3.11. On the other hand, the existence of a $O(h)$-size region in which the first order finite difference based local solver is used explains the lower-than-second-order convergence rates in L^2 and L^∞ norms. In both examples, the number of sweeps needed for the convergence is fairly small.

Similar to what we have seen in Example 4, although the first order local solver is used in certain cells when the convergence is achieved, our essentially second order accurate algorithm produces more accurate approximations than the first order Godunov finite difference fast sweeping method [54], see Table 3.12.

Example 7: Finally in this section, we want to use one example to show that it is necessary to check the signs of u_{ij}^{new} and v_{ij}^{new} in the local solver in Section 2.1 and Section 2.2 in order to decide which solution candidate should be accepted on I_{ij}.

We consider the example with the exact solution $\phi(x, y) = \sin(\frac{\pi}{2}x)$ on $\Omega = [0, 1]^2$, $\Gamma = \{(x, y) \mid x = 0\}$. The results computed with and without checking the sign of u_{ij}^{new} and v_{ij}^{new}

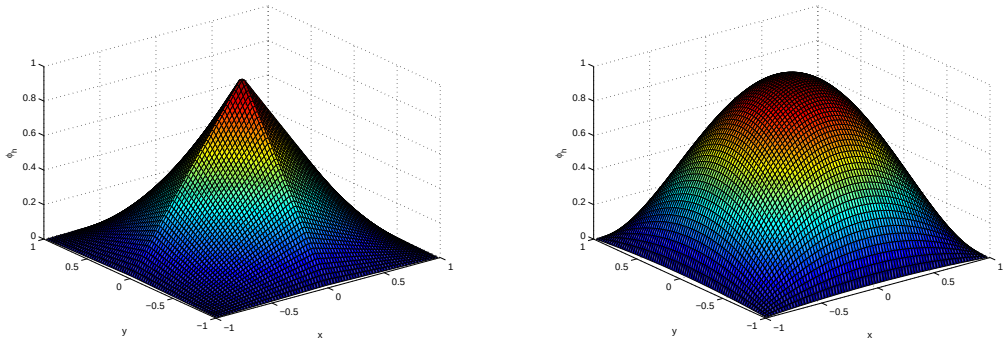

Figure 3.3: Example 5. ϕ_h in Shape-from-shading. The left one is for **case 1**, the non-smooth example and the right one is for **case 2**, the smooth example. $n = 2/h = 80$.

Table 3.9: Example 5. Shape-from-shading, the smooth example. The boundary condition is pre-assigned in both the boundary cells *and* the cells in the 0.4-length square box in the center of the domain. $n = 2/h$.

n	L^1 error	order	L^2 error	order	L^∞ error	order	#	type
20	1.55E-03	-	1.90E-03	-	7.57E-03	-	7	C2
40	3.66E-04	2.08	4.59E-04	2.05	1.97E-03	1.94	8	C2
80	8.92E-05	2.03	1.13E-04	2.02	5.24E-04	1.91	8	C2
160	2.22E-05	2.01	2.84E-05	2.00	1.36E-04	1.95	11	C2
320	5.55E-06	2.00	7.11E-06	2.00	3.58E-05	1.93	23	C2

are included in Figure 3.5 and Figure 3.6. One can see that without checking the sign of the derivative of ϕ_h (using either the hybrid local solver, or the pure DG local solver), the wrong candidate will be picked up. The plots for 1D cut of the $\bar{\phi}_{ij}$ and u_{ij}/h with fixed y in Figure 3.6 show what actually happens.

4 Concluding Remarks

We have developed a fast sweeping method based on a discontinuous Galerkin (DG) local solver for computing viscosity solutions of a class of static Hamilton-Jacobi equations, namely the Eikonal equations on rectangular grids. The compactness of the local DG solver matches well with the fast sweeping strategy, a Gauss-Seidel type of iterated method, to provide both accuracy and efficiency, which is demonstrated by extensive numerical tests. Extension to higher dimensions, higher order DG solvers, and non-uniform/triangular meshes will be studied in the future. In three dimensional case, the local solver is still very simple. We believe the advantage over first order scheme will be

Table 3.10: Example 6. Shape-from-shading, the smooth example. The boundary condition is pre-assigned in the cells right next to the domain boundary and in the 0.1-length square box around the isolated points in Γ. †= the percentage of the cells in which the finite difference type local solver is used by the time the iterations converge. $n = 1/h$.

n	L^1 error	order	L^2 error	order	L^∞ error	order	#	type	†
20	1.08E-02	-	1.84E-02	-	1.20E-01	-	8	C3	3.000%
40	1.91E-03	2.50	4.30E-03	2.10	5.38E-02	1.15	7	C3	1.750%
80	4.47E-04	2.09	1.34E-03	1.68	2.67E-02	1.01	12	C3	1.250%
160	1.09E-04	2.04	4.51E-04	1.57	1.32E-02	1.01	15	C3	0.547%
320	2.71E-05	2.01	1.54E-04	1.55	6.60E-03	1.00	16	C3	0.270%

Table 3.11: Example 6. Shape-from-shading, the non-smooth example. The boundary condition is pre-assigned in the cells right next to the domain boundary and in the 0.1-length square box around the isolated points in Γ. †= the percentage of the cells in which the finite difference type local solver is used by the time the iterations converge. $n = 1/h$.

n	L^1 error	order	L^2 error	order	L^∞ error	order	#	type	†
20	9.40E-03	-	1.16E-02	-	4.94E-02	-	7	C3	2.000%
40	2.70E-03	1.80	4.60E-03	1.33	5.13E-02	-	11	C3	2.000%
80	6.83E-04	1.99	1.16E-03	1.99	1.41E-02	1.87	11	C3	0.188%
160	1.87E-04	1.87	4.23E-04	1.46	4.67E-03	1.59	15	C3	0.094%
320	5.19E-05	1.85	1.48E-04	1.51	2.07E-03	1.17	15	C3	0.039%

more significant in higher dimensions. In order to develop the higher order DG based fast sweeping methods, one would need to first overcome the use of the first order solver in the DG local solver. The success of the fast sweeping method on triangular meshes in [38] may assist us to design the DG based fast sweeping methods for more general meshes.

References

[1] R. Abgrall, *Numerical discretization of the first-order Hamilton-Jacobi equation on triangular meshes*, Communications on Pure and Applied Mathematics, 49 (1996), 1339–1373.

[2] D.N. Arnold, F. Brezzi, B. Cockburn, and L.D. Marini, *Unified analysis of discontinuous Galerkin methods for elliptic problems*, SIAM Journal on Numerical Analysis, 39 (2001/02), 1749–1779.

[3] S. Augoula and R. Abgrall, *High order numerical discretization for Hamilton-Jacobi equations on triangular meshes*, Journal of Scientific Computing, 15 (2000), 197–229.

Table 3.12: Example 6. Shape-from-shading, both smooth and non-smooth examples. The boundary condition is pre-assigned in the cells right next to the domain boundary and in the 0.1-length square box around the isolated points in Γ. The computation is by the first order Godunov finite difference fast sweeping method. $n = 1/h$.

n	L^1 error	order	L^2 error	order	L^∞ error	order	#
smooth example							
20	6.08E-02	-	7.80E-02	-	1.47E-01	-	8
40	3.54E-02	0.78	4.38E-02	0.83	7.78E-02	0.92	8
80	1.91E-02	0.89	2.32E-02	0.92	3.97E-02	0.97	8
160	9.93E-03	0.94	1.20E-02	0.95	1.99E-02	1.00	8
320	5.08E-03	0.97	6.08E-03	0.98	9.88E-03	1.01	8
non-smooth example							
20	4.36E-02	-	6.34E-02	-	1.49E-01	-	8
40	2.63E-02	0.73	3.73E-02	0.77	8.14E-02	0.87	8
80	1.52E-02	0.79	2.12E-02	0.82	4.38E-02	0.89	8
160	8.52E-03	0.84	1.17E-02	0.86	2.33E-02	0.91	8
320	4.65E-03	0.87	6.33E-03	0.89	1.23E-02	0.92	8

[4] T. Barth and J. Sethian, *Numerical schemes for the Hamilton-Jacobi and level set equations on triangulated domains*, Journal of Computational Physics, 145 (1998), 1–40.

[5] 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.

[6] S. Bryson and D. Levy, *High-order central WENO schemes for multidimensional Hamilton-Jacobi equations*, SIAM Journal on Numerical Analysis, 41 (2003), 1339–1369.

[7] T. Cecil, J. Qian and S. Osher, *Numerical methods for high dimensional Hamilton-Jacobi equations using radial basis functions*, Journal of Computational Physics, 196 (2004), 327–347.

[8] L.-T. Cheng and Y.-H. Tsai, *Redistancing by flow of time dependent Eikonal equation*, UCLA CAM Report 07-16.

[9] 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.

[10] B. Cockburn, S. Hou and C.-W. Shu, *The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case*, Mathematics of Computation, 54 (1990), 545–581.

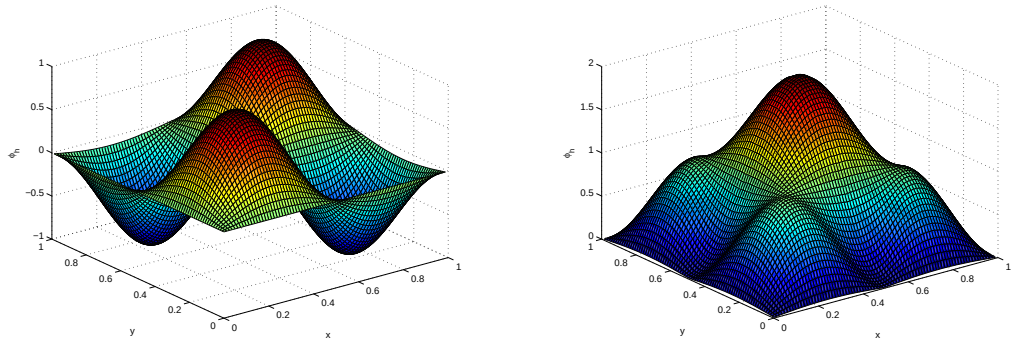

Figure 3.4: Example 6. ϕ_h in Shape-from-shading. The left one is for the smooth solution and the right one is for the non-smooth one. $n = 1/h = 80$.

[11] B. Cockburn, S.-Y. Lin and C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: one dimensional systems*, Journal of Computational Physics, 84 (1989), 90–113.

[12] B. Cockburn and C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: general framework*, Mathematics of Computation, 52 (1989), 411–435.

[13] B. Cockburn and C.-W. Shu, *The Runge-Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems*, Journal of Computational Physics, 141 (1998), 199–224.

[14] B. Cockburn and C.-W. Shu, *The local discontinuous Galerkin method for time-dependent convection-diffusion systems*, SIAM Journal on Numerical Analysis, 35 (1998), 2440–2463.

[15] B. Cockburn and C.-W. Shu, *Runge-Kutta discontinuous Galerkin methods for convection-dominated problems*, Journal of Scientific Computing, 16 (2001), 173–261.

[16] M.G. Crandall and P.L. Lions, *Viscosity solutions of Hamilton-Jacobi equations*, Transactions of American Mathematical Society, 277 (1983), 1–42.

[17] J. Dellinger and W.W. Symes, *Anisotropic finite-difference traveltimes using a Hamilton-Jacobi solver*, 67th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 1997, 1786–1789.

[18] E. W. Dijkstra, *A note on two problems in connection with graphs*, Numerische Mathematik, 1 (1959), 269–271.

[19] M. Falcone and R. Ferretti, *Discrete time high-order schemes for viscosity solutions of Hamilton-Jacobi-Bellman equations*, Numerische Mathematik, 67 (1994), 315–344.

[20] M. Falcone and R. Ferretti, *Semi-Lagrangian schemes for Hamilton-Jacobi equations, discrete representation formulae and Godunov methods*, Journal of Computational Physics, 175 (2002), 559–575.

[21] S. Gray and W. May, *Kirchhoff migration using Eikonal equation travel-times*, Geophysics, 59 (1994), 810–817.

[22] J. Helmsen, E. Puckett, P. Colella and M. Dorr, *Two new methods for simulating photolithography development in 3D*, Proceedings SPIE 2726 (1996), 253–261.

[23] C. Hu, O. Lepsky and C.-W. Shu, *The effect of the least square procedure for discontinuous Galerkin methods for Hamilton-Jacobi equations*. Discontinuous Galerkin methods (Newport, RI, 1999), Lecture Notes in Computational Science and Engineering, Springer, Berlin, 2000. 343–348.

[24] 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.

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

[26] S. Jin and Z. Xin, *Numerical passage from systems of conservation laws to Hamilton-Jacobi equations and relaxation schemes*, SIAM Journal on Numerical Analysis, 35 (1998), 2385–2404.

[27] 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.

[28] 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.

[29] S. Kim and R. Cook, *3D traveltime computation using second-order ENO scheme*, Geophysics, 64 (1999), 1867–1876.

[30] F. Li and C.-W. Shu, *Reinterpretation and simplified implementation of a discontinuous Galerkin method for Hamilton-Jacobi equations*, Applied Mathematics Letters, 18 (2005), 1204–1209.

[31] C.-T. Lin and E. Tadmor, *High-resolution non-oscillatory central schemes for approximate Hamilton-Jacobi equations*, SIAM Journal on Scientific Computing, 21 (2000), 2163–2186.

[32] S. Osher, *A level set formulation for the solution of the Dirichlet problem for Hamilton-Jacobi equations*, SIAM Journal on Mathematical Analysis, 24 (1993), 1145–1152.

[33] S. Osher and J. Sethian, *Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations*, Journal of Computational Physics, 79 (1988), 12–49.

[34] S. Osher and C.-W. Shu, *High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations*, SIAM Journal on Numerical Analysis, 28 (1991), 907–922.

[35] D. Peng, S. Osher, B. Merriman, H.-K. Zhao and M. Rang, *A PDE-based fast local level set method*, Journal of Computational Physics, 155 (1999), 410–438.

[36] J. Qian and W.W. Symes, *Paraxial Eikonal solvers for anisotropic quasi-P travel times*, Journal of Computational Physics, 174 (2001), 256–278.

[37] J. Qian and W.W. Symes, *Finite-difference quasi-P traveltimes for anisotropic media*, Geophysics, 67 (2002), 147–155.

[38] 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.

[39] 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.

[40] C. Rasch and T. Satzger, *Remarks on the O(N) implementation of the fast marching method*, submitted.

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

[42] W.H. Reed and T.R. Hill, *Triangular mesh method for the neutron transport equation*, Technical report LA-UR-73-479, Los Alamos Scientific Laboratory, Los Alamos, NM, 1973.

[43] 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.

[44] 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.

[45] 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.

[46] C.-W. Shu, *High order numerical methods for time dependent Hamilton-Jacobi equations*, IMS Lecture Notes Series, Vol 11: Mathematics and Computation in Imaging Science and Information Processing, World Scientific Publishing, Singapore, 2007, 47–91.

[47] 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.

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

[49] J. Yan and C.-W. Shu, *A local discontinuous Galerkin method for KdV type equations*, SIAM Journal on Numerical Analysis, 40 (2002), 769–791.

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

[51] 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.

[52] 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.

[53] 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.

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

[55] 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.

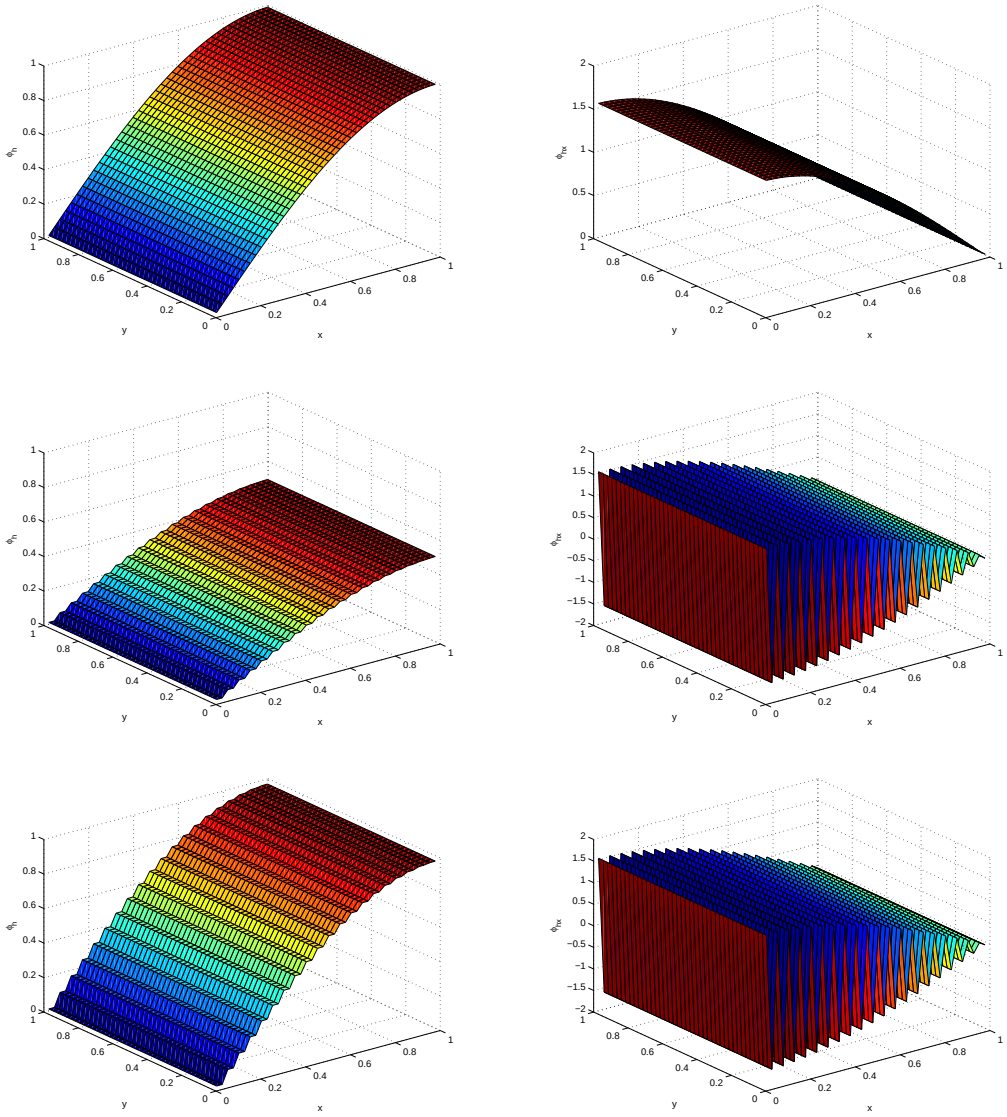

Figure 3.5: Justification for the necessity of checking the signs of u_{ij}^{new} and v_{ij}^{new} in the local solver in Section 2.1 and 2.2. The exact solution is $\phi(x, y) = \sin(\frac{\pi}{2}x)$. ϕ_h is computed when the signs of u_{ij} and v_{ij} are checked (top), when the signs are not checked (middle), and when the signs are not checked and the pure DG local solver is used (bottom). Left: ϕ_{ij}; Right: u_{ij}/h. $n = 1/h = 40$.

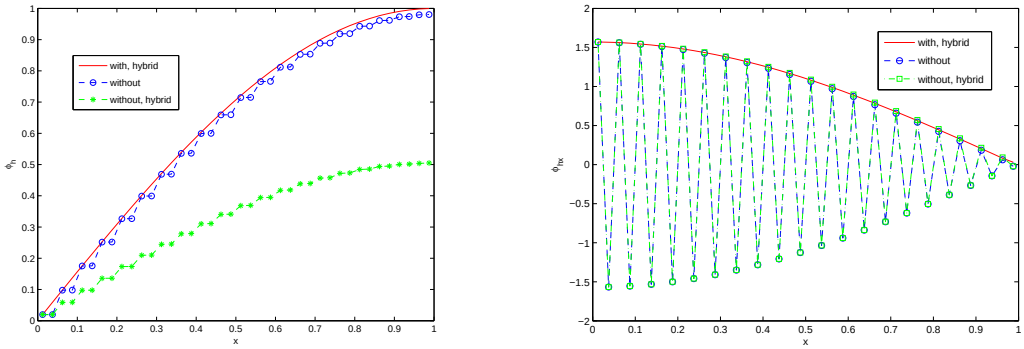

Figure 3.6: Justification for the necessity of checking the signs of u_{ij}^{new} and v_{ij}^{new} in the local solver in Section 2.1 and 2.2. The exact solution is $\phi(x,y) = \sin(\frac{\pi}{2}x)$. The 1D cut of $\bar{\phi}_{ij}$ (left) and u_{ij}/h (right). ϕ_h is computed when the signs of u_{ij} and v_{ij} are checked (with, hybrid), when the signs are not checked (without, hybrid), and when the signs are not checked and the pure DG local solver is used (without). $n = 1/h = 40$.