Speaker: Chi-Wang Shu

Lecture: (Tuesday 11:00-12:30)

Titre: Finite difference methods for Hamilton-Jacobi equations (1)

We will discuss monotone finite difference schemes on Cartesian meshes for solving the viscosity solutions of Hamilton-Jacobi equations, in the context of treating them as building blocks for higher order schemes. We will then describe the general framework of building higher order finite difference schemes with these building blocks. We will also discuss high order time discretization.

Lecture: (Tuesday 14:00-15:30)

Titre: Finite difference methods for Hamilton-Jacobi equations (2)

We will discuss essentially non-oscillatory (ENO) and weighted ENO (WENO) schemes for solving Hamilton-Jacobi equations. We will also discuss the fast sweeping method for WENO schemes solving steady state solutions, including a recent development of high order accuracy boundary condition treatment for such methods.

Lecture: (Friday 09:00-10:30)

Titre: Discontinuous Galerkin methods for Hamilton-Jacobi equations

We will discuss the discontinuous Galerkin method of Hu and Shu for solving Hamilton-Jacobi equations, through approximating the conservation law system satisfied by the derivatives of the solution. An implementation of Li and Shu using curl-free elements allows this method to be more efficient without a least square procedure. We will also discuss a new discontinuous Galerkin method of Cheng and Shu which approximates directly the Hamilton-Jacobi equations. A fast sweeping method based on the discontinuous Galerkin method of Cheng and Shu for solving Eikonal equations will also be discussed.

HIGH ORDER NUMERICAL METHODS FOR TIME DEPENDENT HAMILTON-JACOBI EQUATIONS

Chi-Wang Shu

Division of Applied Mathematics, Brown University
Providence, Rhode Island 02912, USA
E-mail: shu@dam.brown.edu

In these lectures we review a few high order accurate numerical methods for solving time dependent Hamilton-Jacobi equations. We will start with a brief introduction of the Hamilton-Jacobi equations, the appearance of singularities as discontinuities in the derivatives of their solutions hence the necessity to introduce the concept of viscosity solutions, and first order monotone numerical schemes on structured and unstructured meshes to approximate such viscosity solutions, which can be proven convergent with error estimates. We then move on to discuss high order accurate methods which are based on the first order monotone schemes as building blocks. We describe the Essentially Non-Oscillatory (ENO) and Weighted Essentially Non-Oscillatory (WENO) schemes for structured meshes, and WENO schemes and Discontinuous Galerkin (DG) schemes for unstructured meshes.

1. Introduction and Properties of Hamilton-Jacobi Equations

In these lectures we review high order accurate numerical methods for solving time dependent Hamilton-Jacobi equations

$$\varphi_t + H(\varphi_{x_1}, ..., \varphi_{x_d}) = 0, \qquad \varphi(x, 0) = \varphi^0(x), \tag{1}$$

where H is a (usually nonlinear) function which is at least Lipschitz continuous. H could also depend on φ, x and t in some applications, however the main difficulty for numerical solutions is the nonlinear dependency of H on the gradient of φ.

Hamilton-Jacobi equations appear often in many applications. One important application of Hamilton-Jacobi equations is the area of image processing and computer vision, which is the main theme of this program at

the Institute for Mathematical Sciences (IMS) of the National University of Singapore. Other application areas include, e.g. control and differential games.

It is easy to verify that global C^1 solution does not exist for (1) in the generic situation, regardless of the smoothness of the initial condition $\varphi^0(x)$. Singularities in the form of discontinuities in the derivatives of φ would appear at a finite time in most situations, thus the solutions would be Lipschitz continuous but no longer C^1. This could be verified, at least in the one dimensional case, by observing the equivalence between the Hamilton-Jacobi equation

$$\varphi_t + H(\varphi_x) = 0, \qquad \varphi(x,0) = \varphi^0(x) \tag{2}$$

and the hyperbolic conservation law

$$u_t + H(u)_x = 0, \qquad u(x,0) = u^0(x) \tag{3}$$

if we identify $u = \varphi_x$. Singularities for the conservation law (3) are in the form of discontinuities in the solution u, thus u is bounded, with a bounded total variation, but is not continuous. The study of singularities for (3) can be performed using characteristics, see for example [23,39,25]. Such results can be directly translated to that for the Hamilton-Jacobi equation (2) by integrating u once. Discontinuities in u then become discontinuities for the derivative of φ.

This lack of global smoothness of the solution φ in (1) makes it necessary to define a "weak" solution for the PDE (1), that is, a solution φ which may not satisfy the PDE (1) pointwise at every point. In particular, we would only require that φ satisfies the PDE (1) at any point where φ has continuous first derivatives. At those points where the first derivatives of φ are not continuous, a different requirement is needed for the solution φ to be an acceptable weak solution. For the hyperbolic conservation law (3), the requirements at the discontinuities of u include the so-called Rankine-Hugoniot jump condition, which relates the moving speed of the discontinuity with its strength and is derived from an integral version of the PDE (3), and an entropy condition which singles out a unique, physically relevant weak solution from many candidates. For the Hamilton-Jacobi equation (2) or in general (1), the requirements at the discontinuities of the derivatives of φ are characterized by certain inequalities which single out the unique, physically relevant "viscosity solution" of the Hamilton-Jacobi equation. To be more precise, φ is called a viscosity sub-solution of (1) if, for any smooth function ψ, at each local maximum point $(\bar{x}, \bar{t})$ of $\varphi - \psi$,

we have the inequality

$$\psi_t(\bar{x},\bar{t}) + H(\psi_{x_1}(\bar{x},\bar{t}),...,\psi_{x_d}(\bar{x},\bar{t})) \leq 0.$$

Similarly, φ is called a viscosity super-solution of (1) if, for any smooth function ψ, at each local minimum point $(\bar{x},\bar{t})$ of $\varphi - \psi$, we have the inequality

$$\psi_t(\bar{x},\bar{t}) + H(\psi_{x_1}(\bar{x},\bar{t}),...,\psi_{x_d}(\bar{x},\bar{t})) \geq 0.$$

ψ is called the viscosity solution to (1) if it is both a viscosity sub-solution and a viscosity super-solution of (1). For more details, see for example [13].

For the purpose of numerical approximations to the Hamilton-Jacobi equation (1), we would need to pay special attention to the following properties of its viscosity solution φ:

- The viscosity solution φ may contain discontinuous derivatives. In applications, most solutions we encounter are piecewise smooth.
- The weak solution φ may not be unique. There are extra requirements at the discontinuities of the derivatives of φ to make it the unique, physically relevant viscosity solution.

For simplicity of notations we shall mostly concentrate on the two dimensional case, namely $d = 2$ in (1). In this case we will use x, y instead of x_1 and x_2. The equation (1) is then rewritten as

$$\varphi_t + H(\varphi_x, \varphi_y) = 0, \qquad \varphi(x,y,0) = \varphi^0(x,y). \tag{4}$$

2. First Order Monotone Schemes

In this section we will briefly describe first order monotone schemes for solving the Hamilton-Jacobi equation (4), both on structured meshes and on unstructured meshes. These first order monotone schemes will be used as building blocks for high order schemes to be described in the following sections.

2.1. *Monotone schemes on structured rectangular meshes*

We first consider monotone schemes on structured rectangular meshes. For simplicity of notations we will assume that the mesh is uniform in x and y. This simplification is not essential: all of the discussions below can be applied to non-uniform Cartesian meshes with obvious modifications. We

denote by Δx and Δy the mesh sizes in x and y respectively, and denote by $\varphi_{i,j}$ the numerical approximation to the viscosity solution of (4), $\varphi(x_i, y_j, t) = \varphi(i\Delta x, j\Delta y, t)$. We also use the standard notations

$$\Delta^x_{\pm}\varphi_{i,j} = \pm\left(\varphi_{i\pm1,j} - \varphi_{i,j}\right), \qquad \Delta^y_{\pm}\varphi_{i,j} = \pm\left(\varphi_{i,j\pm1} - \varphi_{i,j}\right).$$

First order monotone schemes [14] are defined as schemes of the form

$$\frac{d}{dt}\varphi_{i,j} = -\hat{H}\left(\frac{\Delta^x_-\varphi_{i,j}}{\Delta x}, \frac{\Delta^x_+\varphi_{i,j}}{\Delta x}; \frac{\Delta^y_-\varphi_{i,j}}{\Delta y}, \frac{\Delta^y_+\varphi_{i,j}}{\Delta y}\right) \tag{5}$$

where $\hat{H}$ is called a numerical Hamiltonian, which is a Lipschitz continuous function of all four arguments and is consistent with the Hamiltonian H in the PDE (4):

$$\hat{H}(u, u; v, v) = H(u, v).$$

A monotone numerical Hamiltonian $\hat{H}$ is one which is monotonically non-decreasing in the first and third arguments and monotonically non-increasing in the other two. This can be symbolically represented as

$$\hat{H}\left(\uparrow, \downarrow; \uparrow, \downarrow\right).$$

The scheme (5) with a monotone numerical Hamiltonian is called a monotone scheme. We give here the semi-discrete (continuous in time) form of the monotone scheme. The fully discrete scheme can be obtained by using forward Euler in time. It is also called a monotone scheme.

It is proven in [14] that monotone schemes have the following favorable properties:

- Monotone schemes are stable in the L^∞ norm;
- Monotone schemes are convergent to the viscosity solution of (4);
- The error between the numerical solution of a monotone scheme and the exact viscosity solution of (4), measured in the L^∞ norm, is at least half order $O(\sqrt{\Delta x})$.

The low half order error estimate is not a particular concern for viscosity solutions containing kinks (discontinuities in the first derivatives). In fact, it can be shown that for many cases, this half order error estimate is optimal. However, it is an unfortunate fact that monotone schemes cannot be higher than first order accurate *for smooth solutions*. This is indeed a serious concern, as we would hope the scheme to be high order accurate for smooth solutions, or in smooth regions of non-smooth solutions. Monotone schemes would not be able to achieve this.

The importance of monotone schemes is that they are often used as building blocks for high order schemes. All the high order schemes discussed in these lectures are built upon first order monotone schemes. Thus it is important to know a few typical monotone schemes and their relative merits.

The simplest monotone flux is the Lax-Friedrichs flux [14,32]:

$$\hat{H}^{LF}\left(u^-,u^+;v^-,v^+\right)=H\left(\frac{u^-+u^+}{2},\frac{v^-+v^+}{2}\right)-\frac{1}{2}\alpha^x\left(u^+-u^-\right)-\frac{1}{2}\alpha^y\left(v^+-v^-\right) \tag{6}$$

where

$$\alpha^x=\max_{\substack{A\le u\le B\\ C\le v\le D}}\left|H_1(u,v)\right|,\qquad \alpha^y=\max_{\substack{A\le u\le B\\ C\le v\le D}}\left|H_2(u,v)\right|. \tag{7}$$

Here $H_i(u,v)$ is the partial derivative of H with respect to the i-th argument, or the Lipschitz constant of H with respect to the i-th argument. It can be easily shown that $\hat{H}^{LF}$ is monotone for $A\le u\le B$ and $C\le v\le D$.

Another slightly different Lax-Friedrichs flux is

$$\hat{H}^{LF}\left(u^-,u^+;v^-,v^+\right)=\frac{1}{4}\left(H(u^-,v^-)+H(u^+,v^-)+H(u^-,v^+)+H(u^+,v^+)\right)-\frac{1}{2}\alpha^x\left(u^+-u^-\right)-\frac{1}{2}\alpha^y\left(v^+-v^-\right) \tag{8}$$

where α^x and α^y are chosen the same way as before by (7). This flux is also monotone for $A\le u\le B$ and $C\le v\le D$.

The Godunov type monotone flux is defined as [5]:

$$\hat{H}^{G}\left(u^-,u^+;v^-,v^+\right)=\mathrm{ext}_{u\in I(u^-,u^+)}\,\mathrm{ext}_{v\in I(v^-,v^+)}\;\;H(u,v) \tag{9}$$

where

$$I(a,b)=[\min(a,b),\max(a,b)]$$

and the function ext is defined by

$$\mathrm{ext}_{u\in I(a,b)}=\begin{cases}\min_{a\le u\le b} & \text{if}\quad a\le b,\\ \max_{b\le u\le a} & \text{if}\quad a>b.\end{cases}$$

As pointed out in [5], since in general

$$\min_u\max_v H(u,v)\ne\max_v\min_u H(u,v),$$

we will generally obtain different versions of the Godunov type fluxes $\hat{H}^G$ by changing the order of the min and the max.

The local Lax-Friedrichs flux is defined as

$$\hat{H}^{LLF}\left(u^-,u^+;v^-,v^+\right)=H\left(\frac{u^-+u^+}{2},\frac{v^-+v^+}{2}\right)$$
$$-\frac{1}{2}\alpha^x(u^-,u^+)\left(u^+-u^-\right)-\frac{1}{2}\alpha^y(v^-,v^+)\left(v^+-v^-\right) \tag{10}$$

where

$$\alpha^x(u^-,u^+)=\max_{\substack{u\in I(u^-,u^+)\\ C\le v\le D}}\left|H_1(u,v)\right|,$$
$$\alpha^y(v^-,v^+)=\max_{\substack{A\le u\le B\\ v\in I(v^-,v^+)}}\left|H_2(u,v)\right|. \tag{11}$$

It is proven in [32] that the local Lax-Friedrichs flux $\hat{H}^{LLF}$ is monotone for $A\le u\le B$ and $C\le v\le D$. The local Lax-Friedrichs flux $\hat{H}^{LLF}$ has smaller dissipation than the (global) Lax-Friedrichs flux $\hat{H}^{LF}$.

It would seem that a more local Lax-Friedrichs flux could be

$$\hat{H}^{LLLF}\left(u^-,u^+;v^-,v^+\right)=H\left(\frac{u^-+u^+}{2},\frac{v^-+v^+}{2}\right)$$
$$-\frac{1}{2}\alpha^x(u^-,u^+;v^-,v^+)\left(u^+-u^-\right)-\frac{1}{2}\alpha^y(u^-,u^+;v^-,v^+)\left(v^+-v^-\right)$$

where

$$\alpha^x(u^-,u^+;v^-,v^+)=\max_{\substack{u\in I(u^-,u^+)\\ v\in I(v^-,v^+)}}\left|H_1(u,v)\right|,$$
$$\alpha^y(u^-,u^+;v^-,v^+)=\max_{\substack{u\in I(u^-,u^+)\\ v\in I(v^-,v^+)}}\left|H_2(u,v)\right|.$$

This would be easier to compute and also would have even smaller dissipation than the local Lax-Friedrichs flux $\hat{H}^{LLF}$ defined in (7). Unfortunately, it is shown in [32] that $\hat{H}^{LLLF}$ is *not* a monotone flux.

Another very useful monotone flux is the Roe flux with entropy fix [32]:

$$\hat{H}^{RF}\left(u^-,u^+;v^-,v^+\right)=$$
$$\begin{cases} H(u^*,v^*) & \text{Case 1;}\\ H\left(\frac{u^-+u^+}{2},v^*\right)-\frac{1}{2}\alpha^x(u^-,u^+)\left(u^+-u^-\right) & \text{Case 2;}\\ H\left(u^*,\frac{v^-+v^+}{2}\right)-\frac{1}{2}\alpha^y(v^-,v^+)\left(v^+-v^-\right) & \text{Case 3;}\\ \hat{H}^{LLF}\left(u^-,u^+;v^-,v^+\right) & \text{Case 4.}\end{cases} \tag{12}$$

where Case 1 refers to the situation when $H_1(u,v)$ and $H_2(u,v)$ do not change signs in the region $u\in I(u^-,u^+)$ and $v\in I(v^-,v^+)$; Case 2 refers to the remaining situations and when $H_2(u,v)$ does not change sign in the

region $A \le u \le B$ and $v \in I(v^-, v^+)$; Case 3 refers to the remaining situations and when $H_1(u,v)$ does not change sign in the region $u \in I(u^-, u^+)$ and $C \le v \le D$; and finally Case 4 refers to all remaining situations. Here u^* and v^* are defined by upwinding

$$u^* = \begin{cases} u^-, & \text{if} \quad H_1(u,v) \ge 0; \\ u^+, & \text{if} \quad H_1(u,v) \le 0; \end{cases} \qquad v^* = \begin{cases} v^-, & \text{if} \quad H_2(u,v) \ge 0; \\ v^+, & \text{if} \quad H_2(u,v) \le 0. \end{cases}$$

This Roe flux with local Lax-Friedrichs entropy fix is easy to code and has almost as small a numerical viscosity as the (much more complicated) Godunov flux, hence it is quite popular.

All the monotone fluxes considered above apply to a general Hamiltonian H. There are also simple monotone fluxes which apply to H of certain specific forms. The most noticeable example is the Osher-Sethian flux [31], which applies to Hamiltonians of the form $H(u,v) = f(u^2, v^2)$ where f is a monotone function of each argument:

$$\hat{H}^{OS}\left(u^-, u^+, v^-, v^+\right) = f(\bar{u}^2, \bar{v}^2) \tag{13}$$

where $\bar{u}^2$ and $\bar{v}^2$ are implemented by

$$\bar{u}^2 = \begin{cases} (\min(u^-,0))^2 + (\max(u^+,0))^2, & \text{if} \quad f(\downarrow, \cdot) \\ (\min(u^+,0))^2 + (\max(u^-,0))^2, & \text{if} \quad f(\uparrow, \cdot) \end{cases}$$

$$\bar{v}^2 = \begin{cases} (\min(v^-,0))^2 + (\max(v^+,0))^2, & \text{if} \quad f(\cdot, \downarrow) \\ (\min(v^+,0))^2 + (\max(v^-,0))^2, & \text{if} \quad f(\cdot, \uparrow). \end{cases}$$

This numerical Hamiltonian is purely upwind and easy to program, hence it should be used whenever possible. However, we should point out that not all Hamiltonians H can be written in the form $f(u^2, v^2)$ with a monotone f. For example, $H(u,v) = \sqrt{au^2 + cv^2}$ is of this form for constants a and c, hence we can use the Osher-Sethian flux for it, but $H(u,v) = \sqrt{au^2 + 2buv + cv^2}$ is not of this form, hence Osher-Sethian flux does not apply and we must program a Godunov type monotone flux if we would like a purely upwind flux.

2.2. *Monotone schemes on unstructured meshes*

In many situations it is more convenient and efficient to use an unstructured mesh rather than a structured one described in the previous section. We can similarly define the concept of monotone schemes on unstructured meshes, which again serve as building stones for higher order schemes. In this section we only present the Lax-Friedrichs type monotone scheme on unstructured

meshes of Abgrall [2]. Other monotone schemes can also be defined on unstructured meshes.

The equation (4) is solved in a general domain Ω, which has a triangulation $\mathcal{T}_h$ consisting of triangles. The nodes are named by their indices $0 \leq i \leq N$, with a total of $N+1$ nodes. For every node i, we define the k_i+1 angular sectors $T_0, \cdots, T_{k_i}$ meeting at the point i; they are the inner angles at node i of the triangles having i as a vertex. The indexing of the angular sectors is ordered counterclockwise. $\vec{n}_{l+\frac{1}{2}}$ is the unit vector of the half-line $D_{l+\frac{1}{2}} = T_l \bigcap T_{l+1}$, and θ_l is the inner angle of sector T_l, $0 \leq l \leq k_i$; see Figure 1.

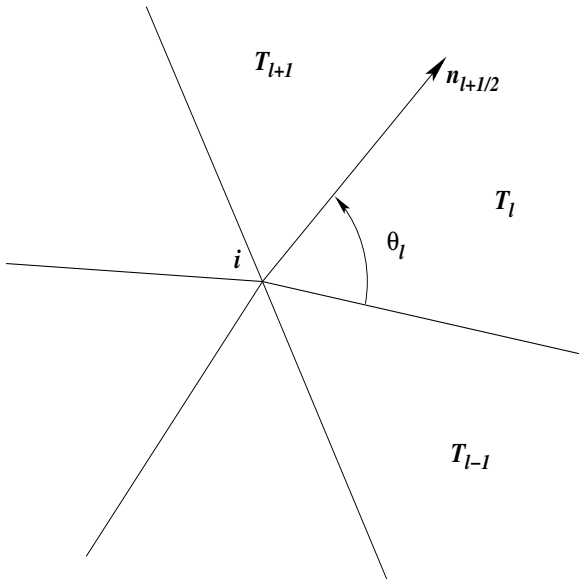

Fig. 1. Node i and its angular sectors.

We denote by φ_i the numerical approximation to the viscosity solution of (4) at node i. $(\nabla\varphi)_0, \cdots, (\nabla\varphi)_{k_i}$ will respectively represent the numerical approximation of $\nabla\varphi$ at node i in each angular sector $T_0, \cdots, T_{k_i}$.

The Lax-Friedrichs type monotone Hamiltonian for arbitrary triangulations developed by Abgrall in [2] is a generalization of the Lax-Friedrichs monotone Hamiltonian for Cartesian meshes described in the previous sec-

tion. This monotone Hamiltonian is given by

$$\hat{H}((\nabla\varphi)_0, \cdots, (\nabla\varphi)_{k_i}) = H\left(\frac{\sum_{l=0}^{k_i} \theta_l (\nabla\varphi)_l}{2\pi}\right) - \frac{\alpha}{\pi}\sum_{l=0}^{k_i} \beta_{l+\frac{1}{2}} \left(\frac{(\nabla\varphi)_l + (\nabla\varphi)_{l+1}}{2}\right)\cdot \vec{n}_{l+\frac{1}{2}} \tag{14}$$

where

$$\beta_{l+\frac{1}{2}} = \tan\left(\frac{\theta_l}{2}\right) + \tan\left(\frac{\theta_{l+1}}{2}\right)$$
$$\alpha = \max\{\max_{\substack{A\le u\le B\\ C\le v\le D}} |H_1(u,v)|, \max_{\substack{A\le u\le B\\ C\le v\le D}} |H_2(u,v)|\}.$$

Here H_1 and H_2 are again the partial derivatives of H with respect to φ_x and φ_y, respectively, or the Lipschitz constants of H with respect to φ_x and φ_y, if H is not differentiable. $[A, B]$ is the value range for $(\varphi_x)_l$, and $[C, D]$ is the value range for $(\varphi_y)_l$, over $0 \le l \le k_i$ for the local Lax-Friedrichs Hamiltonian, and over $0 \le l \le k_i$ and $0 \le i \le N$ for the global Lax-Friedrichs Hamiltonian.

The $\hat{H}$ in (14) defines a monotone Hamiltonian. It is Lipschitz continuous in all arguments and is consistent with H, i.e., $\hat{H}(\nabla\varphi, \cdots, \nabla\varphi) = H(\nabla\varphi)$. It is proven in [2] that the numerical solution of the monotone scheme using this numerical Hamiltonian converges to the viscosity solution of (4), with the same half order convergence rate in the L^∞ norm for regular triangulations, namely for such triangulations where the ratio between the radii of the smallest circle outside a triangle and the largest circle inside the triangle stays bounded during mesh refinement.

3. High Order ENO and WENO Schemes on Structured Rectangular Meshes

In this section we describe the high order ENO (essentially non-oscillatory) and WENO (weighted ENO) schemes on structured rectangular meshes for solving the two dimensional Hamilton-Jacobi equations (4). Schemes for higher spatial dimensions are similar. We will only consider spatial discretizations in this section. Time discretization will be described in section 6.

We first explain the meaning of "high order" when the solution contains possible discontinuities for its derivatives. In such situations high order

accuracy refers to a formal high order truncation error in smooth regions of the solution. Thus in general we can only expect high order accuracy in smooth regions away from derivative singularities. However, typically high order methods also have a sharper resolution for the derivative singularities. Thus high order methods are also referred to as "high resolution" schemes, especially when applied to conservation laws.

3.1. *High order ENO schemes*

High order ENO schemes for solving Hamilton-Jacobi equations were developed in [31] for the second order case and in [32] for the more general cases, based on ENO schemes for solving conservations laws [17,37,38]. We refer to the lecture notes of Shu [36] for more details of ENO and WENO schemes.

The key idea of ENO schemes is an adaptive stencil interpolation procedure, which automatically obtains information from the locally smoothest region, and hence yields a uniformly high-order essentially non-oscillatory approximation for piecewise smooth functions.

We first summarize the ENO interpolation procedure, which is used for building ENO schemes to solve the Hamilton-Jacobi equations (4). Given point values $f(x_j)$, $j = 0, \pm 1, \pm 2, ...$ of a (usually piecewise smooth) function $f(x)$ at discrete nodes x_j, we associate an r-th degree polynomial $P^{f,r}_{j+1/2}(x)$ with each interval $[x_j, x_{j+1}]$, constructed inductively as follows:

(1) We start with a first degree polynomial interpolating at the two boundary nodes of the target interval $[x_j, x_{j+1}]$ and denote the left-most point in its stencil by $k^1_{\min}$:

$$P^{f,1}_{j+1/2}(x) = f[x_j] + f[x_j, x_{j+1}](x - x_j), \qquad k^1_{\min} = j;$$

(2) If $k^{m-1}_{\min}$ and $P^{f,m-1}_{j+1/2}(x)$ are both defined, then let

$$a^{(m)} = f[x_{k^{m-1}_{\min}}, ..., x_{k^{m-1}_{\min}+m}], \qquad b^{(m)} = f[x_{k^{m-1}_{\min}-1}, ..., x_{k^{m-1}_{\min}+m-1}],$$

and

(a) If $|a^{(m)}| \geq b^{(m)}$, then $c^{(m)} = b^{(m)}$, $k^m_{\min} = k^{m-1}_{\min} - 1$; otherwise $c^{(m)} = a^{(m)}$, $k^m_{\min} = k^{m-1}_{\min}$,

(b) The ENO polynomial of the next higher degree is defined by

$$P^{f,m}_{j+1/2}(x) = P^{f,m-1}_{j+1/2}(x) + c^{(m)} \prod_{i=k^{m-1}_{\min}}^{k^{m-1}_{\min}+m-1} (x - x_i).$$

In the procedure above, $f[\cdot, \cdots, \cdot]$ are the standard Newton divided differences defined inductively as

$$f[x_i] = f(x_i); \qquad f[x_i, ..., x_{i+m}] = \frac{f[x_{i+1}, ..., x_{i+m}] - f[x_i, ..., x_{i+m-1}]}{x_{i+m} - x_i}.$$

Note that we start from the first degree polynomial $P^{f,1}$ with a stencil of two points, which would generate a first order monotone scheme in the procedure below.

Clearly, the ENO interpolation procedure starts with a base stencil containing 2 grid points, then adaptively adds one point to the stencil at each stage, which is either the left neighboring point or the right neighboring point to the current stencil depending on which would yield a smaller (in magnitude) divided difference together with points in the current stencil.

It can be shown that this ENO interpolation procedure can generate high order approximation yet avoids spurious oscillations, in the sense of yielding a total variation of the interpolant being at most $O(\Delta x^r)$ larger than the total variation of the piecewise smooth function $f(x)$ being interpolated. Thus the ENO procedure is especially suited for problems with singular but piecewise smooth solutions, such as solutions to conservation laws or Hamilton-Jacobi equations.

High order ENO schemes use monotone fluxes described in section 2.1 as building blocks and the ENO interpolation procedure described above to compute high order approximations to the left and right derivatives. The algorithm can be summarized as follows:

(1) At any node (x_i, y_j), fix j to compute along the x-direction, by using the ENO interpolation procedure, to obtain

$$u^{\pm}_{i,j} = \frac{d}{dx} P^{\varphi,r}_{i\pm 1/2,j}(x_i). \tag{15}$$

(2) Similarly, at the node (x_i, y_j), fix i to compute along the y-direction, by using the ENO interpolation procedure, to obtain

$$v^{\pm}_{i,j} = \frac{d}{dy} P^{\varphi,r}_{i,j\pm 1/2}(y_j). \tag{16}$$

(3) Form the semi-discrete r-th order ENO scheme

$$\frac{d}{dt}\varphi_{i,j} = -\hat{H}(u^-_{i,j}, u^+_{i,j}; v^-_{i,j}, v^+_{i,j}). \tag{17}$$

This semi-discrete ENO scheme will be discretized in time by the high order strong stability preserving Runge-Kutta time discretizations, to be described in section 6.

Numerical results obtained with these ENO schemes can be found in [31] and [32] and will be not be presented here.

3.2. *High order WENO schemes*

WENO schemes are designed based on ENO schemes. Both ENO and WENO schemes use the idea of adaptive stencils in the interpolation procedure based on the local smoothness of the numerical solution to automatically achieve high order accuracy and a non-oscillatory property near discontinuities. ENO uses just one (optimal in some sense) out of many candidate stencils when doing the interpolation, as is described in the previous section, while WENO uses a convex combination of all the candidate stencils, each being assigned a nonlinear weight which depends on the local smoothness of the numerical solution based on that stencil. WENO improves upon ENO in robustness, better smoothness of fluxes, better steady state convergence, better provable convergence properties, and more efficiency. For more details regarding WENO schemes, we again refer to the lecture notes [36].

High order WENO schemes for solving Hamilton-Jacobi equations were developed in [20], based on WENO schemes for solving conservations laws [30,21]. The framework of WENO schemes for solving Hamilton-Jacobi equations is similar to that of ENO schemes described in the previous section. The only difference is the interpolation procedure to obtain $u^{\pm}_{i,j}$ and $v^{\pm}_{i,j}$.

Let us look at the fifth order WENO interpolation procedure to obtain $u^{-}_{i,j}$ as an example. When the third order ENO interpolation procedure (see the previous section) is used, we can easily work out the algebra to obtain the three possible interpolations to $u^{-}_{i,j}$:

$$\begin{aligned} u^{-,0}_{i,j} &= \frac{1}{3}\frac{\Delta^+_x\varphi_{i-3,j}}{\Delta x} - \frac{7}{6}\frac{\Delta^+_x\varphi_{i-2,j}}{\Delta x} + \frac{11}{6}\frac{\Delta^+_x\varphi_{i-1,j}}{\Delta x}, \\ u^{-,1}_{i,j} &= -\frac{1}{6}\frac{\Delta^+_x\varphi_{i-2,j}}{\Delta x} + \frac{5}{6}\frac{\Delta^+_x\varphi_{i-1,j}}{\Delta x} + \frac{1}{3}\frac{\Delta^+_x\varphi_{i,j}}{\Delta x}, \\ u^{-,2}_{i,j} &= \frac{1}{3}\frac{\Delta^+_x\varphi_{i-1,j}}{\Delta x} + \frac{5}{6}\frac{\Delta^+_x\varphi_{i,j}}{\Delta x} - \frac{1}{6}\frac{\Delta^+_x\varphi_{i+1,j}}{\Delta x}, \end{aligned} \tag{18}$$

depending on which of the three possible stencils

$$\{x_{i-3}, x_{i-2}, x_{i-1}, x_i\}, \qquad \{x_{i-2}, x_{i-1}, x_i, x_{i+1}\}, \qquad \{x_{i-1}, x_i, x_{i+1}, x_{i+2}\}$$

(where y_j is omitted in the stencil as it is the same for all three stencils) are chosen by the ENO stencil choosing procedure based on the magnitudes of

the divided differences. Recall that $\Delta_x^+ \varphi_{i,j} = \varphi_{i+1,,j} - \varphi_{i,j}$ is the standard forward difference operator in x. If the third order ENO scheme is used, one of the $u_{i,j}^{-;m}$ for $m = 0, 1$ or 2 is used as $u_{i,j}^-$. The WENO procedure however uses a convex combination of all three $u_{i,j}^{-;m}$ for the final approximation $u_{i,j}^-$:

$$u_{i,j}^- = w_0 u^{-,0} + w_1 u^{-,1} + w_2 u^{-,2} \tag{19}$$

where $w_s \geq 0$ are the nonlinear weights obeying $w_0 + w_1 + w_2 = 1$. The weights w_s are chosen to satisfy the following two properties:

(1) In smooth regions, $\{w_0, w_1, w_2\}$ should be very close to the so-called optimal linear weights $\{0.1, 0.6, 0.3\}$:

$$w_0 = 0.1 + O(\Delta x^2), \qquad w_1 = 0.6 + O(\Delta x^2), \qquad w_2 = 0.3 + O(\Delta x^2),$$

which makes $u_{i,j}^-$ defined by (19) fifth order accurate in approximating $\frac{\partial \varphi}{\partial x}(x_i, y_j)$ in smooth regions;

(2) When stencil s contains a singularity (discontinuity in the x derivative) of φ, the corresponding weight w_s should be very close to zero, so that the approximation $u_{i,j}^-$ emulates an ENO approximation where "bad" stencils make no contributions. In the choice of weights in [20] $w_s = O(\Delta x^4)$ when stencil s contains a singularity.

The key ingredient in designing a nonlinear weight to satisfying the two properties listed above is a smoothness indicator, which is a measurement of how smooth the function being interpolated is inside the interpolation stencil. The recipe used in [20] is similar to that in [21] for conservation laws, namely the smoothness indicator is a scaled sum of the squares of the L^2 norms of the second and higher derivatives of the interpolation polynomial on the target interval. These smoothness indicators work out to be

$$\begin{aligned} IS_0 &= 13(a-b)^2 + 3(a-3b)^2, \\ IS_1 &= 13(b-c)^2 + 3(b+c)^2, \\ IS_2 &= 13(c-d)^2 + 3(3c-d)^2, \end{aligned}$$

where

$$a = \frac{\Delta_x^2 \varphi_{i-2,j}}{\Delta x}, \qquad b = \frac{\Delta_x^2 \varphi_{i-1,j}}{\Delta x}, \qquad c = \frac{\Delta_x^2 \varphi_{i,j}}{\Delta x}, \qquad d = \frac{\Delta_x^2 \varphi_{i+1,j}}{\Delta x} \tag{20}$$

are the second order differences, defined by $\Delta_x^2 \varphi_{i,j} = \varphi_{i+1,j} - 2\varphi_{i,j} + \varphi_{i-1,j}$. With these smoothness indicators, the nonlinear weights are then defined

by

$$w_0 = \frac{\tilde{w}_0}{\tilde{w}_0 + \tilde{w}_1 + \tilde{w}_2}, \qquad w_1 = \frac{\tilde{w}_1}{\tilde{w}_0 + \tilde{w}_1 + \tilde{w}_2}, \qquad w_2 = \frac{\tilde{w}_2}{\tilde{w}_0 + \tilde{w}_1 + \tilde{w}_2},$$

with

$$\tilde{w}_0 = \frac{1}{(\varepsilon + IS_0)^2}, \qquad \tilde{w}_1 = \frac{6}{(\varepsilon + IS_1)^2}, \qquad \tilde{w}_2 = \frac{3}{(\varepsilon + IS_2)^2},$$

where ε is a small number to prevent the denominator to become zero and is typically chosen as $\varepsilon = 10^{-6}$. Finally, after some algebraic manipulations, we obtain the fifth order WENO approximation to $u^-_{i,j}$ as

$$u^-_{i,j} = \frac{1}{12}\left(-\frac{\Delta^+_x \varphi_{i-2,j}}{\Delta x} + 7\frac{\Delta^+_x \varphi_{i-1,j}}{\Delta x} + 7\frac{\Delta^+_x \varphi_{i,j}}{\Delta x} - \frac{\Delta^+_x \varphi_{i+1,j}}{\Delta x}\right) - \Phi^{WENO}(a,b,c,d)$$

where

$$\Phi^{WENO}(a,b,c,d) = \frac{1}{3} w_0 \left(a - 2b + c\right) + \frac{1}{6}\left(w_2 - \frac{1}{2}\right)\left(b - 2c + d\right)$$

with a, b, c, d defined by (20).

By symmetry, the approximation to the right derivative $u^+_{i,j}$ is given by

$$u^+_{i,j} = \frac{1}{12}\left(-\frac{\Delta^+_x \varphi_{i-2,j}}{\Delta x} + 7\frac{\Delta^+_x \varphi_{i-1,j}}{\Delta x} + 7\frac{\Delta^+_x \varphi_{i,j}}{\Delta x} - \frac{\Delta^+_x \varphi_{i+1,j}}{\Delta x}\right) + \Phi^{WENO}(e,d,c,b)$$

with b, c, d defined by (20) and e defined by

$$e = \frac{\Delta^2_x \varphi_{i+2,j}}{\Delta x}.$$

The procedure to obtain $v^{\pm}_{i,j}$ is similar. Finally, we can form the semi-discrete fifth order WENO scheme as

$$\frac{d}{dt}\varphi_{i,j} = -\hat{H}(u^-_{i,j}, u^+_{i,j}; v^-_{i,j}, v^+_{i,j}). \tag{21}$$

This semi-discrete WENO scheme will be discretized in time by the high order strong stability preserving Runge-Kutta time discretizations, to be described in section 6. WENO schemes of different orders of accuracy can be defined along the same lines. For example, the third order WENO scheme

is given by (21) with $u^-_{i,j}$ on the left-biased stencil $\{x_{i-2}, x_{i-1}, x_i, x_{i+1}\}$ defined by

$$u^-_{i,j} = \frac{1}{2}\left(\frac{\Delta^+_x \varphi_{i-1,j}}{\Delta x} + \frac{\Delta^+_x \varphi_{i,j}}{\Delta x}\right) - \frac{w_-}{2}\left(\frac{\Delta^+_x \varphi_{i-2,j}}{\Delta x} - 2\frac{\Delta^+_x \varphi_{i-1,j}}{\Delta x} + \frac{\Delta^+_x \varphi_{i,j}}{\Delta x}\right)$$

where

$$w_- = \frac{1}{1+2r_-^2}, \qquad r_- = \frac{\varepsilon + (\Delta^2_x \varphi_{i-1,j})^2}{\varepsilon + (\Delta^2_x \varphi_{i,j})^2}.$$

By symmetry, the approximation to $u^+_{i,j}$ on the right-biased stencil $\{x_{i-1}, x_i, x_{i+1}, x_{i+2}\}$ is defined by

$$u^+_{i,j} = \frac{1}{2}\left(\frac{\Delta^+_x \varphi_{i-1,j}}{\Delta x} + \frac{\Delta^+_x \varphi_{i,j}}{\Delta x}\right) - \frac{w_+}{2}\left(\frac{\Delta^+_x \varphi_{i+1,j}}{\Delta x} - 2\frac{\Delta^+_x \varphi_{i,j}}{\Delta x} + \frac{\Delta^+_x \varphi_{i-1,j}}{\Delta x}\right)$$

where

$$w_+ = \frac{1}{1+2r_+^2}, \qquad r_+ = \frac{\varepsilon + (\Delta^2_x \varphi_{i+1,j})^2}{\varepsilon + (\Delta^2_x \varphi_{i,j})^2}.$$

Numerical results obtained with these WENO schemes can be found in [20] and will be not be presented here.

4. High Order WENO Schemes on Unstructured Meshes

In this section we describe high order WENO schemes for solving the two dimensional Hamilton-Jacobi equations (4) on unstructured triangular meshes. We will concentrate on the third order WENO scheme in [42]. For the fourth order WENO schemes, see [42] for details. We again use the first order monotone flux described in section 2.2 as building blocks.

The semi-discrete high order WENO scheme is given by:

$$\frac{d}{dt}\varphi_i(t) + \hat{H}((\nabla\varphi)_0, \cdots, (\nabla\varphi)_{k_i}) = 0 \tag{22}$$

where $\hat{H}$ is the monotone flux described in section 2.2. The WENO procedure to obtain approximations to the sectional derivatives $(\nabla\varphi)_0$, ..., $(\nabla\varphi)_{k_i}$ will be described in detail below. The semi-discrete scheme (22) will be discretized in time by the high order strong stability preserving Runge-Kutta time discretizations, to be described in section 6.

First we discuss how to construct a high-order approximation to $\nabla\varphi$ in every angular sector of every node, see Figure 1. Let P^k denote the set of two-dimensional polynomials of degree less than or equal to k. We use Lagrange interpolations as follows: given a smooth function φ, and a triangulation with triangles $\{\triangle_0, \triangle_1, \ldots, \triangle_M\}$ and nodes $\{0, 1, 2, \ldots, N\}$, we would like to construct, for each triangle $\triangle_i$, a polynomial $p(x, y) \in P^k$, such that $p(x_l, y_l) = \varphi(x_l, y_l)$, where (x_l, y_l) are the coordinates of the three nodes of the triangle $\triangle_i$ and a few neighboring nodes. $p(x, y)$ would thus be a $(k + 1)$th-order approximation to φ on the cell $\triangle_i$.

Because a kth degree polynomial $p(x, y)$ has $K = \frac{(k+1)(k+2)}{2}$ degrees of freedom, we need to use the information of at least K nodes. In addition to the three nodes of the triangle $\triangle_i$, we may take the other $K - 3$ nodes from the neighboring cells around triangle $\triangle_i$. We rename these K nodes as $S_i = \{M_1, M_2, \ldots, M_K\}$, S_i is called a big stencil for the triangle $\triangle_i$. Let (x_i, y_i) be the barycenter of $\triangle_i$. Define $\xi = (x - x_i)/h_i$, $\eta = (y - y_i)/h_i$, where $h_i = \sqrt{|\triangle_i|}$ with $|\triangle_i|$ denoting the area of the triangle $\triangle_i$, then we can write $p(x, y)$ as:

$$p(x, y) = \sum_{j=0}^{k} \sum_{s+r=j} a_{sr} \xi^s \eta^r.$$

Using the K interpolation conditions:

$$p(M_l) = \varphi(M_l), \qquad l = 1, 2, \cdots, K,$$

we get a $K \times K$ linear system for the K unknowns a_{sr}. The normalized variables ξ, η are used to make the condition number of the linear system independent of mesh sizes.

It is well known that in two and higher dimensions such interpolation problem is not always well defined. The linear system can be very ill-conditioned or even singular, in such cases we would have to add more nodes to the big stencil S_i from the neighboring cells around triangle $\triangle_i$ to obtain an over-determined linear system, and then use the least-square method to solve it. We remark that this ill-conditioning may come from both the geometric distribution of the nodes, for which we could do nothing other than changing the mesh, and from the choice of basis functions in the interpolation. For higher order methods, a closer to orthogonal basis rather than $\xi^s \eta^r$ would be preferred, such as the procedure using barycentric coordinates in [1] and [3]. However, for third and fourth order cases, $\xi^s \eta^r$ can be used for simplicity.

After we have obtained the approximation polynomial $p(x, y)$ on the triangle $\triangle_i$, ∇p will be a kth-order approximation for $\nabla\varphi$ on $\triangle_i$. Hence we get the high-order approximation $\nabla p(x_l, y_l)$ to $\nabla\varphi(x_l, y_l)$, for any one of the three vertices (x_l, y_l) of the triangle $\triangle_i$, in the relevant angular sectors.

A scheme is called linear if it is linear when applied to a linear equation with constant coefficients. We need a third-order approximation for $\nabla\varphi$ to construct a third-order linear scheme, hence we need a cubic polynomial interpolation. A cubic polynomial p^3 has 10 degrees of freedom. We will use some or all of the nodes shown in Figure 2 to form our big stencil. For extremely distorted meshes the number of nodes in Figure 2 may be less than the required 10. In such extreme cases we would need to expand the choice for the big stencil, see [42] for details. For our target triangle $\triangle_0$, which has three vertices i, j, k and the barycenter G, we need to construct a cubic polynomial p^3, then ∇p^3 will be a third-order approximation for $\nabla\varphi$ on $\triangle_0$, and the values of ∇p^3 at points i, j and k will be third-order approximations for $\nabla\varphi$ at the angular sector $\triangle_0$ of nodes i, j and k. We label the nodes of the neighboring triangles of triangle $\triangle_0$ as follows: nodes 1, 2, 3 are the nodes (other than i, j, k) of neighbors of $\triangle_0$, nodes 4, 5, 6, 7, 8, 9 (other than 1, 2, 3, i, j, k) are the nodes of the neighbors of the three neighboring triangles of $\triangle_0$. Notice that the points 4, 5, 6, 7, 8, 9 do not have to be six distinct points. For example the points 5 and 9 could be the same point.

The interpolation points are nodes taken from a sorted node set. An ordering is given in the set so that, when the nodes are chosen sequentially from it to form the big stencil S_0, the target triangle $\triangle_0$ remains central to avoid serious downwind bias which could lead to linear instability. Referring to Figure 2, the interpolation points for the polynomial p^3 include nodes i, j, k and the nodes taken from the sorted set: $W = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. The detailed procedure to determine the big stencil S_0 for the target triangle $\triangle_0$ is given below.

Procedure 1: The choice of the big stencil for the third-order scheme.

(1) Referring to Figure 2, we form a sorted node set: $W = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. In extreme cases when this set does not contain enough distinct points, we may need to add more points from the next layer of neighbors.
(2) To start with, we take $S_0 = \{i, j, k, 1, 2, 3, 4, 5, 6, 7\}$. Use this stencil S_0 to form the 10×10 interpolation coefficient matrix A.
(3) Compute the reciprocal condition number c of A. This is provided by

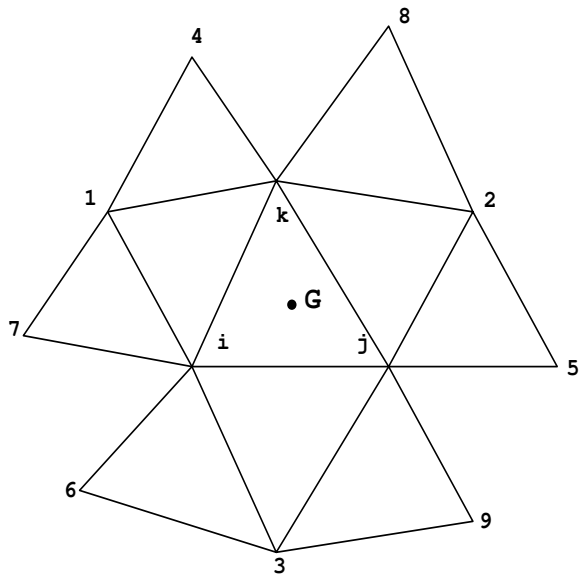

Fig. 2. The nodes used for the big stencil of the third-order scheme.

most linear solvers. If $c \geq \delta$ for some threshold δ, we have obtained the final stencil S_0. Otherwise, add the next node in W (i.e. node 8) to S_0. Use the 11 nodes in S_0 as interpolation points to get the 11×10 least square interpolation coefficient matrix A. Judge the reciprocal condition number c again. Continue in doing this until $c \geq \delta$ is satisfied. It seems that $\delta = 10^{-3}$ is a good threshold after extensive numerical experiments [42]. Notice that, since we have normalized the coordinates, this threshold does not change when the mesh is scaled uniformly in all directions. For all the triangulations tested in [42], at most 12 nodes are needed in S_0 to reach the condition $c \geq \delta$.

We now have obtained the big stencil S_0 and its associated cubic polynomial p^3. For each node (x_l, y_l) in $\triangle_0$, $\nabla p^3(x_l, y_l)$ is a third-order approximation to $\nabla\varphi(x_l, y_l)$. In order to construct a high-order WENO scheme, an important step is to obtain a high-order approximation using a linear combination of lower order approximations. We will use a linear combination of second-order approximations to get the same third-order approximation

to $\nabla\varphi(x_l, y_l)$ as $\nabla p^3(x_l, y_l)$, i.e., we require

$$\frac{\partial}{\partial x}p^3(x_l, y_l) = \sum_{s=1}^{q} \gamma_{s,x} \frac{\partial}{\partial x} p_s(x_l, y_l), \qquad \frac{\partial}{\partial y}p^3(x_l, y_l) = \sum_{s=1}^{q} \gamma_{s,y} \frac{\partial}{\partial y} p_s(x_l, y_l) \tag{23}$$

where p_s are quadratic interpolation polynomials, and $\gamma_{s,x}$ and $\gamma_{s,y}$ are the linear weights for the x-directional derivative and the y-directional derivative respectively, for $s = 1, \cdots, q$. The linear weights are constants depending only on the local geometry of the mesh. The equalities in (23) should hold for any choices of the function φ.

Notice that to get a second-order approximation for the derivatives $\nabla\varphi(x_l, y_l)$, we need a quadratic interpolation polynomial. According to the argument in [19], the cubic polynomial $p^3(x, y)$ has four more degrees of freedom than each quadratic polynomial $p_s(x, y)$, namely x^3, x^2y, xy^2, y^3. For the six degrees of freedom $1, x, y, x^2, xy, y^2$, if we take $\varphi = 1, \varphi = x, \varphi = y, \varphi = x^2, \varphi = xy$ and $\varphi = y^2$, the equalities in (23) will hold for all these cases under only one constraint each on $\gamma_{s,x}$ and $\gamma_{s,y}$, namely $\sum_{s=1}^{q} \gamma_{s,x} = 1$ and $\sum_{s=1}^{q} \gamma_{s,y} = 1$, because p^3 and p_s all reproduce these functions exactly. Hence we should only need $q \geq 5$. $q = 5$ is taken in the scheme below.

We now need $q = 5$ small stencils $\Gamma_s, s = 1, \cdots, 5$ for the target triangle $\triangle_0$, satisfying $S_0 = \bigcup_{s=1}^{5} \Gamma_s$, and every quadratic polynomial p_s is associated with a small stencil Γ_s. In the third-order scheme, the small stencils will be the same for both directions x, y and all three nodes i, j, k in $\triangle_0$. However the linear weights $\gamma_{s,x}, \gamma_{s,y}$ can be different for different nodes i, j, k and different directions x, y. Because each quadratic polynomial has six degrees of freedom, the number of nodes in Γ_s must be at least six. To build a small stencil Γ_s, we start from several candidates $\Gamma_s^{(r)}, r = 1, 2, \cdots, n_s$. These candidates are constructed by first taking a point $A_s^{(r)}$ as the "center", then finding at least six nodes from S_0 which have the shortest distances from $A_s^{(r)}$ and can generate the interpolation coefficient matrix with a good condition number, using the method of Procedure 1. We then choose the best Γ_s among $\Gamma_s^{(r)}, r = 1, \cdots, n_s$ for every $s = 1, \cdots, 5$. Here "best" means that by using this group of small stencils, the linear weights $\gamma_{s,x}, \gamma_{s,y}, s = 1, \cdots, 5$ for all three nodes i, j, k are either all positive or have the smallest possible negative values in magnitude. The details of the algorithm is described in the following procedure.

Procedure 2: The third-order linear scheme.

For every triangle $\triangle_l, l = 1, \cdots, N$, do Steps 1 to 6:

(1) Follow Procedure 1 to obtain the big stencil S_l for $\triangle_l$.

(2) For $s = 1, \cdots, 5$, find the set $W_s = \{\Gamma_s^{(r)}, r = 1, 2, \cdots, n_s\}$, which are the candidate small stencils for the s-th small stencil. We use the following method to find the $\Gamma_s^{(r)}$ in W_s: first, nodes i, j, k are always included in every $\Gamma_s^{(r)}$; then we take a point $A_s^{(r)}$ as the center of $\Gamma_s^{(r)}$, detailed below, and find at least 3 additional nodes other than i, j, k from S_l which satisfy the following two conditions: 1) they have the shortest distances from $A_s^{(r)}$; and 2) taking them and the nodes i, j, k as the interpolation points, we will obtain the interpolation coefficient matrix A with a good condition number, namely the reciprocal condition number c of A satisfies $c \geq \delta$ with the same threshold $\delta = 10^{-3}$. For the triangulations tested in [42], at most 8 nodes are used to reach this threshold value. Finally, the center of the candidate stencils $A_s^{(r)}, r = 1, \cdots, n_s; s = 1, \cdots, 5$ are taken from the nodes around $\triangle_l$ (see Figure 2) as follows:

- $A_1^{(1)}$ = point G, $n_1 = 1$;
- $A_2^{(1)}$ = node 1, $A_2^{(2)}$ = node 4, $A_2^{(3)}$ = node 7, $n_2 = 3$;
- $A_3^{(1)}$ = node 2, $A_3^{(2)}$ = node 5, $A_3^{(3)}$ = node 8, $n_3 = 3$;
- $A_4^{(1)}$ = node 3, $A_4^{(2)}$ = node 6, $A_4^{(3)}$ = node 9, $n_4 = 3$;
- $\{A_5^{(r)}\}_{r=1}^{9}$ = nodes $4, 5, 6, 7, 8, 9$ and the middle points of nodes 4 and 8, 5 and 9, 6 and 7. $n_5 \leq 9$.

(3) By taking one small stencil $\Gamma_s^{(r_s)}$ from each $W_s, s = 1, \cdots, 5$ to form a group, we obtain $n_1 \times n_2 \times \cdots \times n_5$ groups of small stencils. We eliminate the groups which contain the same small stencils, and also eliminate the groups which do not satisfy the condition

$$\bigcup_{s=1}^{5} \Gamma_s^{(r_s)} = S_l$$

According to every group $\{\Gamma_s^{(r_s)}, s = 1, \cdots, 5\}$ of small stencils, we have 5 quadratic polynomials $\{p_s^{(r_s)}\}_{s=1}^{5}$. We evaluate $\frac{\partial}{\partial x} p_s^{(r_s)}$ and $\frac{\partial}{\partial y} p_s^{(r_s)}$ at points i, j, k, to obtain second-order approximation values for $\nabla\varphi$ at the three vertices of the triangle $\triangle_l$. We remark that for practical implementation, we do not use the polynomial itself, but compute a series of constants $\{a_l\}_{l=1}^{m}$ which depend on the local geometry only, such that:

$$\frac{\partial}{\partial x} p_s^{(r_s)}(x_n, y_n) = \sum_{l=1}^{m} a_l \varphi_l \tag{24}$$

where every constant a_l corresponds to one node in the stencil $\Gamma_s^{(r_s)}$ and m is the total number of nodes in $\Gamma_s^{(r_s)}$. For every vertex (x_n, y_n) of triangle $\triangle_l$, we obtain a series of such constants. And for the y directional partial derivative, we compute the corresponding constants too.

(4) For every group $\{\Gamma_s^{(r_s)}, s = 1, \cdots, 5\}$, we form linear systems and solve them to get a series of linear weights $\gamma_{s,x}^{(r_s)}$ and $\gamma_{s,y}^{(r_s)}$ satisfying the equalities (23), for the three vertices i, j, k. Using the previous argument for combining low-order approximations to get high-order approximation, we form the linear system for $\gamma_{s,x}^{(r_s)}$ at a vertex (ξ_n, η_n) as follows (note that we use normalized variables): take $\varphi = \xi^3, \xi^2\eta, \xi\eta^2, \eta^3$ respectively, the equalities are:

$$\sum_{s=1}^{5} \gamma_{s,x}^{(r_s)} \frac{\partial}{\partial \xi} p_s^{(r_s)}(\xi_n, \eta_n) = \frac{\partial}{\partial \xi} \varphi(\xi_n, \eta_n) \tag{25}$$

where $p_s^{(r_s)}$ is the quadratic interpolation polynomial for φ, using stencil $\Gamma_s^{(r_s)}$. Again, in practical implementation, we will not use $p_s^{(r_s)}$ itself, instead we use the constants computed in the last step and equation (24) to compute the approximation for the derivatives of φ. Together with the requirement

$$\sum_{s=1}^{5} \gamma_{s,x}^{(r_s)} = 1, \tag{26}$$

we obtain a 5×5 linear system for $\gamma_{s,x}^{(r_s)}$. For $\gamma_{s,y}^{(r_s)}$, the same argument can be applied. Note that we need to compute the reciprocal condition number c for every linear system again. If $c \geq \delta$ for the same threshold $\delta = 10^{-3}$, we will accept this group of stencils as one of the remaining candidates. Otherwise, the linear system is considered to be ill-conditioned and its corresponding group of small stencils $\{\Gamma_s^{(r_s)}, s = 1, \cdots, 5\}$ is eliminated from further consideration.

(5) For each of the remaining groups $\Lambda_l = \{\Gamma_s^{(r_s)}, s = 1, \cdots, 5\}$, find the minimum value γ_l of all these linear weights $\gamma_{s,x}^{(r_s)}, \gamma_{s,y}^{(r_s)}$ of the three vertices i, j, k. Then find the group of small stencils whose γ_l is the biggest, and take this group as the final 5 small stencils for triangle $\triangle_l$. Denote them by $\Gamma_s, s = 1, \cdots, 5$. For every final small stencil $\Gamma_s, s = 1, 2, \cdots, 5$, we store the index numbers of the nodes in Γ_s, the constants in the linear combinations of node values to approximate values of $\nabla\varphi$

at points i, j, k, and the linear weights $\gamma_{s,x}$, $\gamma_{s,y}$ of the three points i, j, k.

(6) Now we have set up the necessary constants which only depend on the mesh for all triangles. To form the final linear scheme, we compute the third-order approximations $(\nabla\varphi)_0, \cdots, (\nabla\varphi)_{k_l}$ for all mesh nodes l, by the linear combinations of second-order approximations, using the prestored constants and linear weights. Then we can form the scheme (22).

We now describe the construction of WENO schemes based on non-linear weights.

We only discuss the case of WENO approximation for the x-directional derivative at vertex i of the target cell $\triangle_l$. Other cases are similar. In order to compute the non-linear weights, we need to compute the smoothness indicators first.

For a polynomial $p(x, y)$ defined on the target cell $\triangle_0$ with degree up to k, we take the smoothness indicator β as:

$$\beta = \sum_{2\leq|\alpha|\leq k} \int_{\triangle_0} |\triangle_0|^{|\alpha|-2} \left(D^\alpha p(x,y)\right)^2 dxdy \tag{27}$$

where α is a multi-index and D is the derivative operator. The smoothness indicator measures how smooth the function p is on the triangle $\triangle_0$: the smaller the smoothness indicator, the smoother the function p is on $\triangle_0$. The scaling factor in front of the derivatives renders the smoothness indicator self-similar and invariant under uniform scaling of the mesh in all directions. The smoothness indicator (27) is the same as that used for the structured mesh case discussed in the previous section.

Now we define the non-linear weights as:

$$\omega_j = \frac{\widetilde{\omega}_j}{\sum_m \widetilde{\omega}_m}, \qquad \widetilde{\omega}_j = \frac{\gamma_j}{(\varepsilon + \beta_j)^2} \tag{28}$$

where γ_j is the jth linear weight (e.g. the $\gamma_{s,x}$ in the linear schemes), β_j is the smoothness indicator for the jth interpolation polynomial $p_j(x, y)$ (the p_s in equation (23) for the third-order case) associated with the jth small stencil, and ε is again a small positive number to avoid the denominator to become 0 and is usually taken as $\varepsilon = 10^{-6}$. The final WENO approximation for the x-directional derivative at vertex i of target cell $\triangle_l$ is given by

$$(\varphi_x)_i = \sum_{j=1}^{q} \omega_j \frac{\partial}{\partial x} p_j(x_i, y_i) \tag{29}$$

where (x_i, y_i) are the coordinates of vertex i and $q = 5$ for the third-order schemes.

In the WENO schemes, the linear weights $\{\gamma_j\}_{j=1}^{q}$ depend on the local geometry of the mesh and can be negative. If $\min(\gamma_1, \cdots, \gamma_q) < 0$, we can adopt the splitting technique of treating negative weights in WENO schemes developed by Shi, Hu and Shu [34]. We omit the details of this technique and refer the readers to [34].

Again, we remark that the smoothness indicator (27) is a quadratic function of function values on nodes of the small stencil, so in practical implementation, to compute the smoothness indicator β_j for the j-th small stencil by equation (27), we do not need to use the interpolation polynomial itself, instead we use a series of constants $\{a_{rt}, r = 1, \cdots, t; t = 1, \cdots, m\}$, which can be precomputed and they depend on the mesh only, such that

$$\beta_j = \sum_{t=1}^{m} \varphi_t (\sum_{r=1}^{t} a_{rt} \varphi_r), \tag{30}$$

where m is the total number of nodes in the j-th small stencil. These constants for all smoothness indicators should be precomputed and stored once the mesh is generated.

We summarize the algorithm for the third-order WENO schemes as follows:

Procedure 3: The third-order WENO schemes.

(1) Generate a triangular mesh.
(2) Compute and store all constants which only depend on the mesh and the accuracy order of the scheme. These constants include the node index numbers of each small stencil, the coefficients in the linear combinations of function values on nodes of small stencils to approximate the derivative values and the linear weights, following Procedure 2 for the third-order case, and the constants for computing smoothness indicators in equation (30).
(3) Using the prestored constants, for each angular sector of every node i, compute the low-order approximations for $\nabla\varphi$ and the nonlinear weights, then compute the third order WENO approximation (29). Finally, form the scheme (22).

Numerical examples using the third and fourth order WENO schemes on unstructured meshes can be found in [42] and will not be presented here.

5. High Order Discontinuous Galerkin Schemes on Unstructured Meshes

Discontinuous Galerkin methods have become very popular in recent years to solve hyperbolic conservation laws because of their distinctive features, among which are the easy design of the methods with any order of accuracy and their minimal requirement on the mesh structures [12]. Adapted from these methods for conservation laws, a discontinuous Galerkin method for solving the Hamilton-Jacobi equations (1) was developed by Hu and Shu in [18] based on the equivalence between Hamilton-Jacobi equations and hyperbolic conservation laws [22,29]. See also [24]. In [18,24], the Hamilton-Jacobi equations (1) were first rewritten as a system of conservation laws

$$(w_i)_t + (H(\mathbf{w}))_{x_i} = 0, \text{in } \Omega \times [0,T], \quad \mathbf{w}(x,0) = \nabla\varphi^0(x), \tag{31}$$

where $\mathbf{w} = \nabla\varphi$. With piecewise polynomial space as the solution space, the usual discontinuous Galerkin formulation could be obtained for (31) [8,10]. Notice that $w_i, \quad i = 1, \cdots, n$ are not independent due to the restriction $\mathbf{w} = \nabla\varphi$. A least square procedure was applied in each time step (or each time stage depending on the particular time discretization used) to enforce this restriction in [18,24].

In a recent preprint by Li and Shu [27], we have given a reinterpretation and simplified implementation of the discontinuous Galerkin method for Hamilton-Jacobi equations developed in [18,24]. This was based on a recent work by Cockburn et al [9] and by Li and Shu [26], where the locally divergence-free discontinuous Galerkin methods were developed for partial differential equations with divergence-free solutions. Compared with traditional ways to solve this type of equations, the piecewise divergence-free polynomial space, which is a subspace of the standard piecewise polynomial space, is used. With minimal change in the scheme formulation (only the solution and test space is changed to a smaller space), the computational cost is reduced, the stability and the order of accuracy of the scheme are maintained. For specific applications such as the Maxwell equations [9] and the ideal magnetohydrodynamics (MHD) equations [26], this new method even improves over the traditional discontinuous Galerkin method in terms of stability and/or accuracy while saving computational costs. The idea of this approach could be applied to more general situations, by using piecewise solution space in which functions satisfy certain properties of the exact solutions (divergence-free, or curl-free, ...). The general approximation theory can guarantee no loss of accuracy when such smaller solution space is used. This observation leads to a reinterpretation and simplified implemen-

tation of the discontinuous Galerkin method for Hamilton-Jacobi equations developed in [18,24].

In this section we describe the discontinuous Galerkin method for solving the Hamilton-Jacobi equations developed in [18,24], using the reinterpretation in [27]. There are other similar or related types of discretizations for Hamilton-Jacobi equations on unstructured meshes, e.g. the schemes of Augoula and Abgrall [4] and that of Barth and Sethian [6], which will not be described in this section because of space limitations.

Starting with a regular triangulation $\mathcal{T}_h = \{K\}$ of Ω (edges denoted by e), the general discontinuous Galerkin formulation of (31) is: find $\mathbf{w} = (w_1, \cdots, w_n) \in \mathbf{V}^k$, such that

$$\frac{d}{dt}\int_K w_i v_i dx = \int_K H(\mathbf{w})(v_i)_{x_i} dx - \sum_{e\in\partial K}\int_e \hat{H}_{i,e,K} v_i ds, \quad \forall K, i = 1, \cdots, n \tag{32}$$

holds for all $\mathbf{v} = (v_1, \cdots, v_n) \in \mathbf{V}^k$, where $\mathbf{V}^k$ is the solution space which will be specified later, and $\hat{H}_{i,e,K}$ is the monotone numerical flux described in section 2.2. The strong stability preserving Runge-Kutta time discretization, to be described in section 6, could be used in time direction. Notice (32) is the formulation for the derivatives of φ in (1). To recover the missing constant in φ (e.g. the cell average of φ in each element), there are two different strategies developed in [18,24] which can be used:

(1) By requiring that

$$\int_K (\varphi_t + H(\varphi_x, \varphi_y))\, v\, dxdy \,=\, 0, \tag{33}$$

for all $v \in V_h^0$ and for all $K \in \mathcal{T}_h$, that is,

$$\int_K (\varphi_t + H(\varphi_x, \varphi_y))\, dxdy \,=\, 0, \qquad \forall K \in \mathcal{T}_h\,; \tag{34}$$

(2) By using (34) to update only one (or a few) elements, e.g., the corner element(s), then use

$$\varphi(B,t) = \varphi(A,t) + \int_A^B (\varphi_x\, dx + \varphi_y\, dy) \tag{35}$$

to determine the missing constant. The path should be taken to avoid crossing a derivative discontinuity, if possible.

We refer the readers to [18,24] for more details.

Before finalizing the scheme, we introduce the following spaces,

$$\mathbf{V}_1^k = \{(v_1, \cdots, v_n) : v_i|_K \in P^k(K), i = 1, \cdots, n, \forall K \in \mathcal{T}_h\}, \tag{36}$$

$$\mathbf{V}_2^k = \{(v_1, \cdots, v_n) : \mathbf{v}|_K = \nabla\varphi, \varphi \in P^{k+1}(K), \forall K \in \mathcal{T}_h\}, \qquad (37)$$

where $P^k(K)$ denotes the space of polynomials in K of degree at most k. It is easy to see that $\mathbf{V}_2^k \subset \mathbf{V}_1^k$. Two formulations are obtained if $\mathbf{V}^k$ in (32) is specified as follows:

- *Formulation I*: $\mathbf{V}^k = \mathbf{V}_1^k$. A single polynomial $\varphi \in P^{k+1}(K)$, up to a constant, is recovered from $\mathbf{w}$ in each element by the following least square procedure

$$\left|\left|\sum_i (\varphi_{x_i} - w_i)^2\right|\right|_{L^1(K)} = \min_{\psi \in P^{k+1}(K)} \left|\left|\sum_i (\psi_{x_i} - w_i)^2\right|\right|_{L^1(K)} \qquad (38)$$

  after each time stage. This is the method proposed by Hu and Shu in [18].
- *Formulation II*: $\mathbf{V}^k = \mathbf{V}_2^k$.

We have proven in [27] that the two formulations are mathematically equivalent. Clearly, the second formulation has several advantages over the first formulation:

(1) Formulation II allows the method of lines version of the scheme, while Formulation I does not have a method of lines version due to the least square procedure which is applied after each time step or stage. The method of lines version allows more natural and direct analysis for stability and accuracy of discontinuous Galerkin methods, e.g. the results in [24].
(2) The implementation of the algorithm is significantly simplified by using Formulation II since a smaller solution space is used and the least square procedure is completely avoided. If we characterize the computational cost of (32) per time step per element simply by the dimension of $\mathbf{V}^k|_K$, we can get

$$n_1 = dim(\mathbf{V}_1^k|_K) = n\sum_{r=0}^{k} C_{r+n-1}^{n-1}, \quad n_2 = dim(\mathbf{V}_2^k|_K) = \sum_{r=1}^{k+1} C_{r+n-1}^{n-1}.$$

For example, for the two dimensional case $n = 2$, $n_1 = (k+2)(k+1)$, $n_2 = \frac{(k+4)(k+1)}{2}$, hence $\frac{n_2}{n_1} \to \frac{1}{2}$ as $k \to \infty$; i.e. the cost is reduced to about half for higher order schemes. For the three dimensional case $n = 3$, $n_1 = \frac{k^3+6k^2+11k+6}{2}$, $n_2 = \frac{(k+1)(k^2+8k+18)}{6}$, hence $\frac{n_2}{n_1} \to \frac{1}{3}$ as $k \to \infty$; i.e. the cost is reduced to about one third for higher order schemes.

Representative numerical examples using the discontinuous Galerkin methods for solving the two dimensional Hamilton-Jacobi equations (4) will be given in section 7. More numerical examples can be found in [18,24,27].

6. High Order Strong Stability Preserving Runge-Kutta Time Discretizations

For all of the spatial discretizations discussed in the previous sections, the time variable t is left undiscretized. A popular time discretization method is the class of strong stability preserving (SSP), also referred to as total variation diminishing (TVD), high order Runge-Kutta time discretizations, see [37,35,15,16].

We start with the following ordinary differential equation (ODE)

$$\frac{d}{dt}u(t) = L(u(t), t) \tag{39}$$

resulting from a method of lines spatial discretization of a time dependent partial differential equation, such as (17), (21), (22) or (32) in the previous sections. Here $u = u(t)$ is a (usually very long) vector and $L(u, t)$ depends on u either linearly or non-linearly. In many applications $L(u, t) = L(u)$ which does not explicitly depend on t. The starting point for the SSP method is an *assumption* that the first order Euler forward time discretization to (39):

$$u^{n+1} = u^n + \Delta t L(u^n, t^n), \tag{40}$$

where u^n is an approximation to $u(t^n)$, are stable under a certain (semi) norm

$$||u^{n+1}|| \leq ||u^n|| \tag{41}$$

with a suitable time step restriction

$$\Delta t \leq \Delta t_0, \tag{42}$$

which typically depends on the spatial discretization mesh size. With this assumption, we would like to find SSP time discretization methods to (39), that are higher order accurate in time, yet still maintain the same stability condition (41). This might require a different restriction on the time step Δt than that in (42) of the form

$$\Delta t \leq c\Delta t_0, \tag{43}$$

where c is called the *CFL coefficient* of the SSP method. The objective is to find such methods with simple format, low computational cost and least restriction on the time step Δt, i.e. larger CFL coefficient c.

We remark that the strong stability assumption for the forward Euler step in (41) can be relaxed to the more general stability assumption

$$||u^{n+1}|| \leq (1 + O(\Delta t))||u^n||.$$

This general stability property is also preserved by the high order SSP time discretizations.

Runge-Kutta methods are time discretizations which can be written in several different ways. In [37], a general m stage Runge-Kutta method for (39) is written in the form:

$$\begin{aligned} u^{(0)} &= u^n, \\ u^{(i)} &= \sum_{k=0}^{i-1} \left(\alpha_{i,k} u^{(k)} + \Delta t \beta_{i,k} L(u^{(k)}, t^n + d_k \Delta t) \right), \quad i = 1, ..., m \quad (44) \\ u^{n+1} &= u^{(m)} \end{aligned}$$

where d_k are related to $\alpha_{i,k}$ and $\beta_{i,k}$ by

$$d_0 = 0, \qquad d_i = \sum_{k=0}^{i-1} (\alpha_{i,k} d_k + \beta_{i,k}), \qquad i = 1, ..., m-1.$$

Thus, we do not need to discuss the choice of d_k separately. In most ODE literatures, e.g. [7], a Runge-Kutta method is written in the form of a Butcher array. Every Runge-Kutta method in the form of (44) can be easily converted in a unique way into a Butcher array, see [37]. A Runge-Kutta method written in a Butcher array can also be rewritten into the form (44), however this conversion is in general *not* unique. This non-uniqueness in the representation (44) is exploited in the literature to seek the largest provable time steps (43) for SSP.

We always need and require that $\alpha_{i,k} \geq 0$ in (44). If this is violated no SSP methods are possible. Basically, we rely heavily on convexity arguments which would require that all $\alpha_{i,k}$'s to be non-negative.

If all the $\beta_{i,k}$'s in (44) are also nonnegative, $\beta_{i,k} \geq 0$, we have the following simple lemma, which is the backbone of SSP Runge-Kutta methods:

Lemma 4: [37] *If the forward Euler method* (40) *is stable in the sense of* (41) *under the time step restriction* (42)*, then the Runge-Kutta method* (44) *with* $\alpha_{i,k} \geq 0$ *and* $\beta_{i,k} \geq 0$ *is SSP, i.e. its solution also satisfies the same stability* (41) *under the time step restriction* (43) *with the CFL coefficient*

$$c = \min_{i,k} \frac{\alpha_{i,k}}{\beta_{i,k}}. \quad (45)$$

The most popular and successful SSP methods are those covered by Lemma 4. We will only give examples of SSP methods covered by Lemma 4 in this section. If some of the $\beta_{i,k}$'s must be negative because of accuracy constraints, there is also a way to obtain SSP methods, see [37,15,16] for details.

We list below a few popular SSP Runge-Kutta methods:

(1) A second order SSP Runge-Kutta method [37]:

$$\begin{aligned} u^{(1)} &= u^n + \Delta t L(u^n, t^n) \\ u^{n+1} &= \frac{1}{2}u^n + \frac{1}{2}u^{(1)} + \frac{1}{2}\Delta t L(u^{(1)}, t^n + \Delta t) \end{aligned} \tag{46}$$

with a CFL coefficient $c = 1$ in (43). This is just the classical Heun or modified Euler method.

(2) A third order SSP Runge-Kutta method [37]:

$$\begin{aligned} u^{(1)} &= u^n + \Delta t L(u^n, t^n) \\ u^{(2)} &= \frac{3}{4}u^n + \frac{1}{4}u^{(1)} + \frac{1}{4}\Delta t L(u^{(1)}, t^n + \Delta t) \\ u^{n+1} &= \frac{1}{3}u^n + \frac{2}{3}u^{(2)} + \frac{2}{3}\Delta t L(u^{(2)}, t^n + \frac{1}{2}\Delta t), \end{aligned} \tag{47}$$

with a CFL coefficient $c = 1$ in (43).

(3) A third order low storage SSP Runge-Kutta method [15]:

$$\begin{aligned} u^{(0)} &= u^n, \qquad du^{(0)} = 0, \\ du^{(i)} &= A_i du^{(i-1)} + \Delta t L(u^{(i-1)}, t^n + d_{i-1}\Delta t), \quad i = 1, \ldots, 3, \\ u^{(i)} &= u^{(i-1)} + B_i du^{(i)}, \qquad\qquad i = 1, \ldots, 3, \\ u^{n+1} &= u^{(3)}. \end{aligned} \tag{48}$$

with

$$
\begin{aligned}
z_1 &= \sqrt{36b^4 + 36b^3 - 135b^2 + 84b - 12} \\
z_2 &= 2b^2 + b - 2 \\
z_3 &= 12b^4 - 18b^3 + 18b^2 - 11b + 2 \\
z_4 &= 36b^4 - 36b^3 + 13b^2 - 8b + 4 \\
z_5 &= 69b^3 - 62b^2 + 28b - 8 \\
z_6 &= 34b^4 - 46b^3 + 34b^2 - 13b + 2 \\
d_0 &= 0 \\
A_1 &= 0 \\
B_1 &= b \\
d_1 &= B_1 \\
A_2 &= \frac{-z_1(6b - 4b + 1) + 3z_3}{(2b+1)z_1 - 3(b+2)(2b-1)^2} \\
B_2 &= \frac{12b(b-1)(3z_2 - z_1) - (3z_2 - z_1)^2}{144b(3b-2)(b-1)^2} \\
d_2 &= B_1 + B_2 + B_2 A_2 \\
A_3 &= \frac{-z_1 z_4 + 108(2b-1)b^5 - 3(2b-1)z_5}{24 z_1 b(b-1)^4 + 72b z_6 + 72b^6(2b-13)} \\
B_3 &= \frac{-24(3b-2)(b-1)^2}{(3z_2 - z_1)^2 - 12b(b-1)(3z_2 - z_1)}
\end{aligned}
$$

where $b = 0.924574$, with a CFL coefficient $c = 0.32$ in (43). Only u and du must be stored, resulting in two storage units for each variable. This method can be used when storage is a paramount consideration, such as in large scale three dimensional calculations.

(4) A fourth order, five stage SSP Runge-Kutta method. It can be proven [15] that all four stage, fourth order SSP Runge-Kutta scheme (44) with a nonzero CFL coefficient c in (43) must have at least one negative $\beta_{i,k}$. To obtain fourth order SSP Runge-Kutta methods with nonnegative $\beta_{i,k}$ covered by Lemma 4, we would need at least five stages. The following is a five stage, fourth order SSP Runge-Kutta method [40] with

a CFL coefficient $c = 1.508$ in (43):

$$\begin{aligned}
u^{(1)} &= u^n + 0.39175222700392\,\Delta t L(u^n, t^n)\\
u^{(2)} &= 0.44437049406734\,u^n + 0.55562950593266\,u^{(1)}\\
&\quad +0.36841059262959\,\Delta t L(u^{(1)}, t^n + 0.39175222700392\,\Delta t)\\
u^{(3)} &= 0.62010185138540\,u^n + 0.37989814861460\,u^{(2)}\\
&\quad +0.25189177424738\,\Delta t L(u^{(2)}, t^n + 0.58607968896780\,\Delta t)\\
u^{(4)} &= 0.17807995410773\,u^n + 0.82192004589227\,u^{(3)}\\
&\quad +0.54497475021237\,\Delta t L(u^{(3)}, t^n + 0.47454236302687\,\Delta t)\\
u^{n+1} &= 0.00683325884039\,u^n + 0.51723167208978\,u^{(2)}\\
&\quad +0.12759831133288\,u^{(3)}\\
&\quad +0.08460416338212\,\Delta t L(u^{(3)}, t^n + 0.47454236302687\,\Delta t)\\
&\quad +0.34833675773694\,u^{(4)}\\
&\quad +0.22600748319395\,\Delta t L(u^{(4)}, t^n + 0.93501063100924\,\Delta t).
\end{aligned} \tag{49}$$

7. A Few Numerical Examples

We will show a few numerical examples simulated by the discontinuous Galerkin method in section 5 [18] as representatives. Other examples can be found in the references listed in each sections for different numerical methods discussed in these notes.

Example 5: Two dimensional Burgers' equation:

$$\begin{cases} \varphi_t + \frac{(\varphi_x+\varphi_y+1)^2}{2} = 0, & -2 < x < 2,\ -2 < y < 2\\ \varphi(x, y, 0) = -\cos\left(\frac{\pi(x+y)}{2}\right) \end{cases} \tag{50}$$

with periodic boundary conditions.

At $t = 0.5/\pi^2$, the solution is still smooth. We use non-uniform rectangular meshes obtained from the tensor product of one dimensional nonuniform meshes via randomly shifting the cell boundaries in a uniform mesh in the range $[-0.1h, 0.1h]$ (the meshes in two directions are independent). The L^2-errors computed by a 6×6 point Gaussian quadrature in each cell are shown in Table 1.

At $t = 1.5/\pi^2$, the solution has discontinuous derivatives. Figure 3 is the graph of the numerical solution with 40×40 elements (uniform mesh).

Finally we use triangle based triangulation, the mesh with $h = \frac{1}{4}$ is shown in Figure 4. The accuracy at $t = 0.5/\pi^2$ is shown in Table 2. Similar

Table 1. Accuracy for 2D Burgers equation, non-uniform rectangular mesh, $t = 0.5/\pi^2$.

	P^1		P^2		P^3	
$N \times N$	L^2 error	order	L^2 error	order	L^2 error	order
10×10	4.47E-01	—	6.28E-02	—	1.61E-02	—
20×20	1.83E-01	1.288	1.50E-02	2.066	2.06E-03	2.966
40×40	8.01E-02	1.192	3.63E-03	2.047	3.48E-04	2.565
80×80	3.82E-02	1.068	9.17E-04	1.985	6.03E-05	2.529
160×160	1.87E-02	1.031	2.34E-04	1.970	8.58E-06	2.813

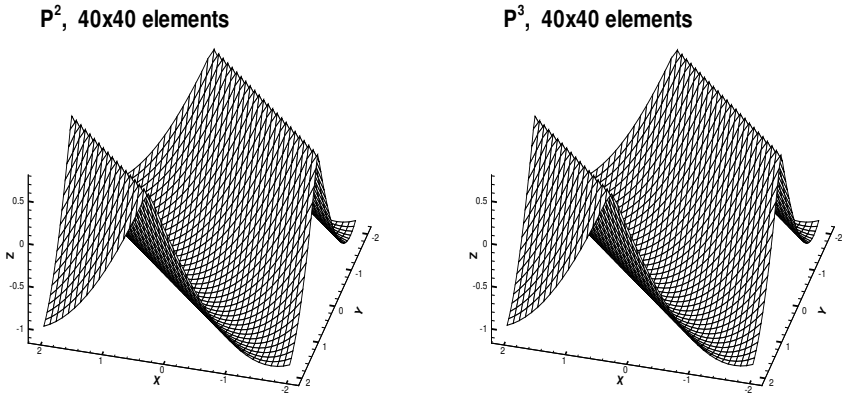

Fig. 3. Two dimension Burgers' equation, rectangular mesh, t=$1.5/\pi^2$.

accuracy pattern is observed as in the rectangular case. The result at $t = 1.5/\pi^2$, when the derivative is discontinuous, is shown in Figure 5.

Table 2. Accuracy for 2D Burgers equation, triangular mesh as those in Figure 4, $t = 0.5/\pi^2$.

	P^2		P^3	
h	L^1 error	order	L^1 error	order
1	5.48E-02	—	1.17E-02	—
1/2	1.35E-02	2.02	1.35E-03	3.12
1/4	2.94E-03	2.20	1.45E-04	3.22
1/8	6.68E-04	2.14	1.71E-05	3.08

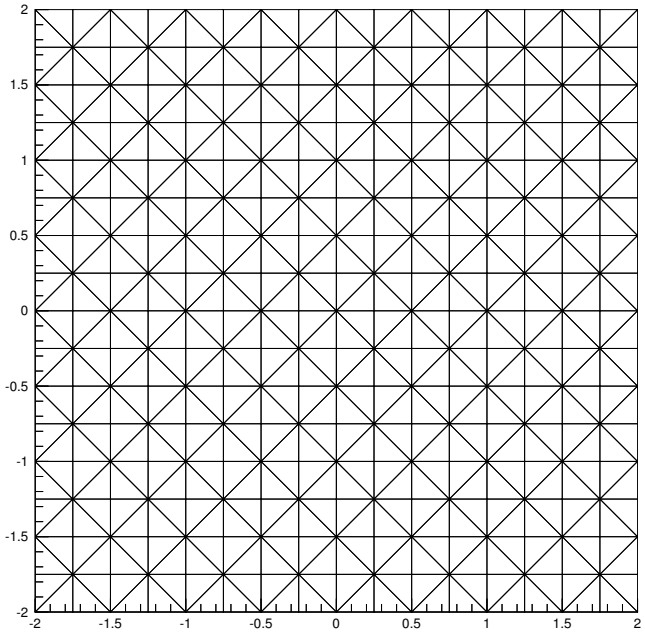

Fig. 4. Triangulation for two dimensional Burgers equation, $h = \frac{1}{4}$.

Example 6: The level set equation in a domain with a hole:

$$\begin{cases} \varphi_t + sign(\varphi_0)(\sqrt{\varphi_x^2 + \varphi_y^2} - 1) = 0, & \frac{1}{2} < \sqrt{x^2 + y^2} < 1 \\ \varphi(x, y, 0) = \varphi_0(x, y) \end{cases} \tag{51}$$

This problem is introduced in [41]. The solution φ to (51) has the same zero level set as φ_0, and the steady state solution is the distance function to that zero level curve. We use this problem to test the effects using various integration paths (35) when there is a hole in the region. Notice that the exact steady state solution is the distance function to the inner boundary of domain when boundary condition is adequately prescribed. We compute the time dependent problem to reach a steady state solution, using the exact solution for the boundary conditions of φ_x and φ_y. Four symmetric elements near the outer boundary are updated by (34), all other elements are recovered from (35) by the shortest path to the nearest one of above four elements. The results are shown in Table 3. Also shown in Table 3 is the error (difference) between the numerical solution φ thus recovered, and

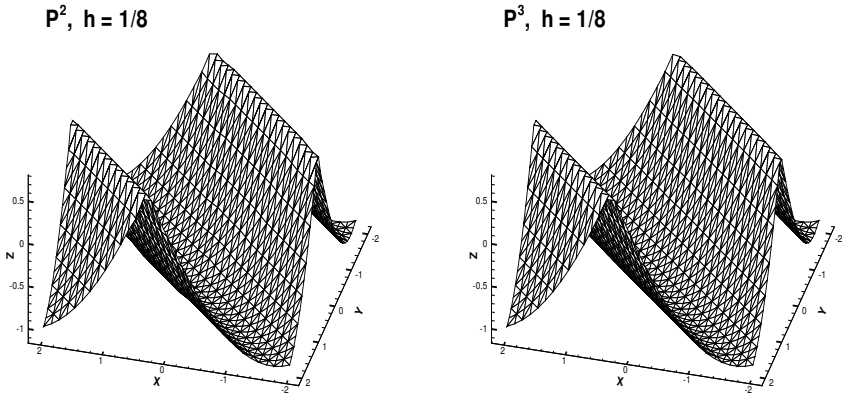

Fig. 5. Two dimension Burgers' equation, triangular mesh, t=$1.5/\pi^2$.

the value of φ after another integration along a circular path (starting and ending at the same point in (35)). We can see that the difference is small with the correct order of accuracy, further indicating that the dependency of the recovered solution φ on the integration path is on the order of the truncation errors even for such problems with holes. Finally, the mesh with 1432 triangles and the solution with 5608 triangles are shown in Figure 6.

Table 3. Errors for the level set equation, triangular mesh with P^2.

	Errors for the Solution		Errors by Integration Path	
N	L^1 error	order	L^1 error	order
403	1.02E-03	—	1.61E-04	—
1432	1.23E-04	3.05	5.84E-05	1.46
5608	1.71E-05	2.85	9.32E-06	2.65
22238	2.09E-06	3.03	1.43E-06	2.70

Example 7: The problem of a propagating surface:

$$\begin{cases} \varphi_t - (1 - \varepsilon K)\sqrt{1 + \varphi_x^2 + \varphi_y^2} = 0, \qquad 0 < x < 1, 0 < y < 1 \\ \varphi(x, y, 0) = 1 - \frac{1}{4}(\cos(2\pi x - 1))\,(\cos(2\pi y - 1)) \end{cases} \tag{52}$$

where K is the mean curvature defined by

$$K = -\frac{\varphi_{xx}(1 + \varphi_y^2) - 2\varphi_{xy}\varphi_x\varphi_y + \varphi_{yy}(1 + \varphi_x^2)}{(1 + \varphi_x^2 + \varphi_y^2)^{\frac{3}{2}}}, \tag{53}$$

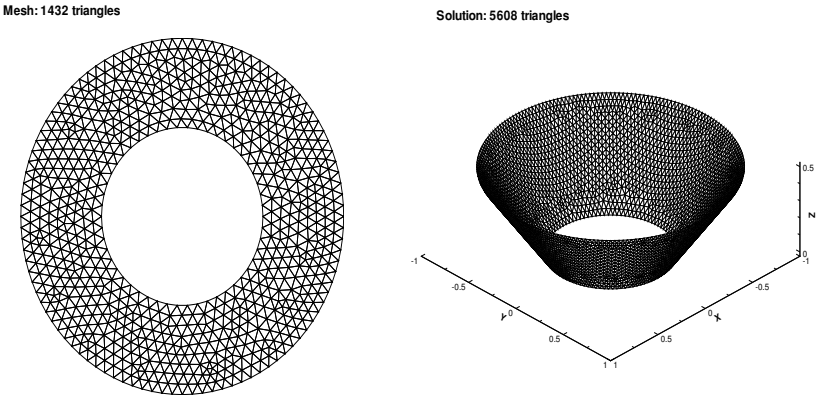

Fig. 6. The level set equation, P^2.

and ε is a small constant. Periodic boundary condition is used.

We apply the discontinuous Galerkin method, with the second derivative terms handled by the local discontinuous Galerkin techniques presented and analyzed in [11], which amounts to solving the following system

$$\begin{cases} u_t - \left(\sqrt{1+u^2+v^2} + \varepsilon \frac{p(1+v^2)-2quv+r(1+u^2)}{1+u^2+v^2}\right)_x = 0 \\ v_t - \left(\sqrt{1+u^2+v^2} + \varepsilon \frac{p(1+v^2)-2quv+r(1+u^2)}{1+u^2+v^2}\right)_y = 0 \\ p - u_x = 0 \\ q - u_y = 0 \\ r - v_y = 0 \end{cases} \tag{54}$$

using the discontinuous Galerkin method. The details of the method, especially the choices of fluxes, which are important for stability, can be found in [11].

We use a triangulation shown in Figure 7. We refine the mesh around the center of domain where the solution develops discontinuous derivatives (for the $\varepsilon = 0$ case). There are 2146 triangles and 1108 nodes in this triangulation. The solutions are displayed in Figure 8 and Figure 9, respectively, for $\varepsilon = 0$ (pure convection) and $\varepsilon = 0.1$. Notice that we shift the solution at $t = 0.0$ downward by 0.35 to show the detail of the solutions at later time.

Example 8: The problem of a propagating surface on a unit disk. The

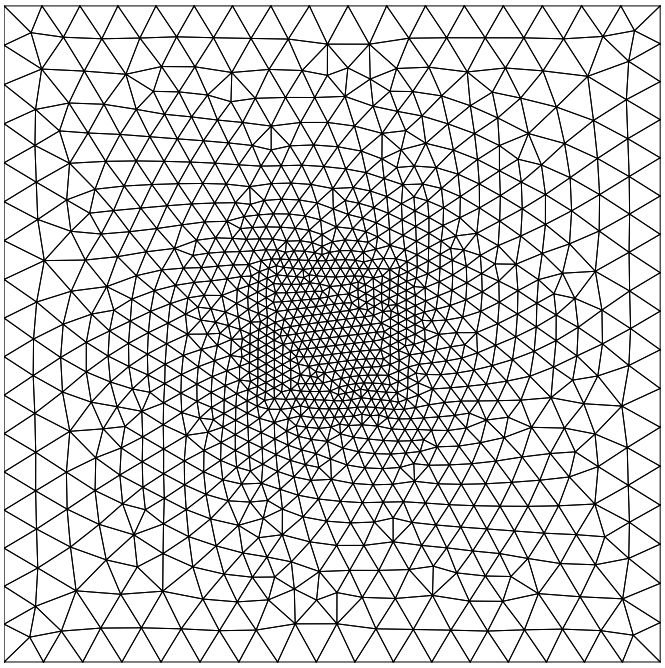

Fig. 7. Triangulation used for the propagating surfaces.

equation is the same as (52) in the previous example, but it is solved on a unit disk $x^2 + y^2 < 1$ with an initial condition

$$\varphi(x, y, 0) = \sin\left(\frac{\pi(x^2 + y^2)}{2}\right)$$

and a Neumann type boundary condition $\nabla\varphi = 0$.

It is difficult to use rectangular meshes for this problem. Instead we use the triangulation shown in Figure 10. Notice that we have again refined the mesh near the center of the domain where the solution develops discontinuous derivatives. There are 1792 triangles and 922 nodes in this triangulation. The solutions with $\varepsilon = 0$ are displayed in Figure 11. Notice that the solution at $t = 0$ is shifted downward by 0.2 to show the detail of the solution at later time.

The solution with $\varepsilon = 0.1$ are displayed in Figure 12. Notice that the solution at $t = 0$ is again shifted downward by 0.2 to show the detail of the solution at later time.

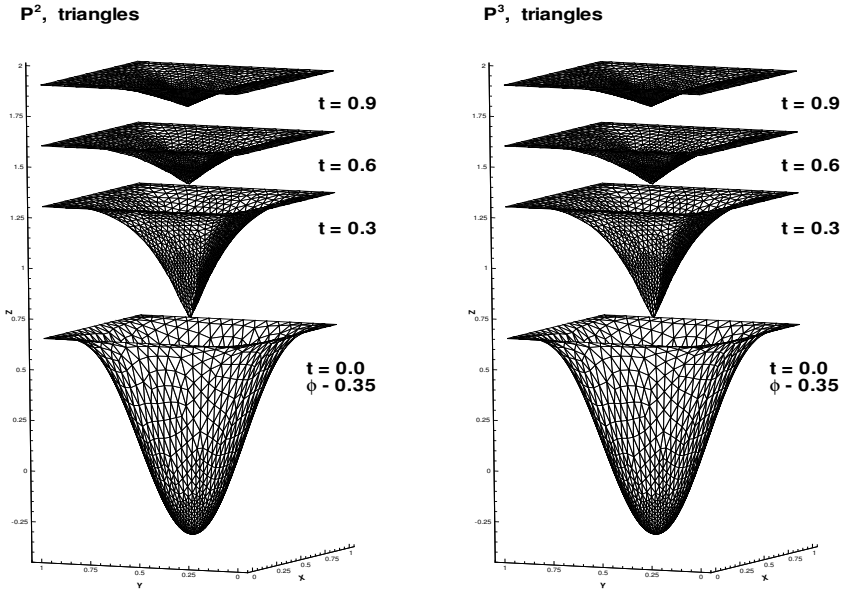

Fig. 8. Propagating surfaces, triangular mesh, $\varepsilon = 0$.

Example 9: A problem from optimal control [32]:

$$\begin{cases} \varphi_t + (\sin y)\varphi_x + (\sin x + \text{sign}(\varphi_y))\varphi_y - \frac{1}{2}\sin^2 y - (1 - \cos x) = 0, \\ \qquad\qquad -\pi < x < \pi,\ -\pi < y < \pi \\ \varphi(x, y, 0) = 0 \end{cases} \tag{55}$$

with periodic boundary conditions. We use a uniform rectangular mesh of 40×40 elements. The solution at $t = 1$ is shown in Figure 13, while the optimal control $w = \text{sign}(\varphi_y)$ is shown in Figure 14.

Notice that the discontinuous Galerkin method computes $\nabla\varphi$ as an independent variable. It is very desirable for those problems in which the most interesting features are contained in the first derivatives of φ, as in this optimal control problem.

Example 10: A problem from computer vision [33]:

$$\begin{cases} \varphi_t + I(x, y)\sqrt{1 + \varphi_x^2 + \varphi_y^2} - 1 = 0, \qquad -1 < x < 1,\ -1 < y < 1 \\ \varphi(x, y, 0) = 0 \end{cases} \tag{56}$$

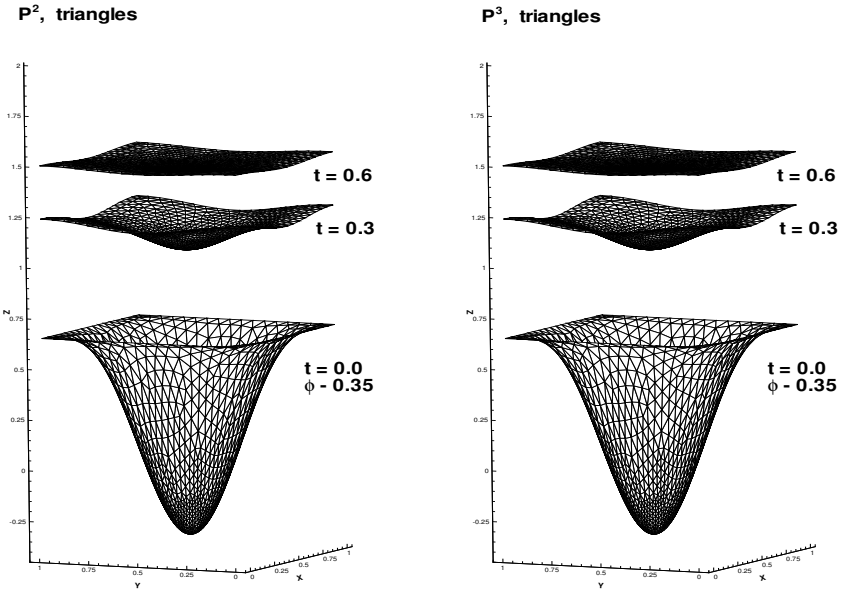

Fig. 9. Propagating surfaces, triangular mesh, $\varepsilon = 0.1$.

with $\varphi = 0$ as the boundary condition. The steady state solution of this problem is the shape lighted by a source located at infinity with vertical direction. The solution is not unique if there are points at which $I(x, y) = 1$. Conditions must be prescribed at those points where $I(x, y) = 1$. Since our method is a finite element method, we need to prescribe suitable conditions at the correspondent elements. We take

$$I(x, y) = 1/\sqrt{1 + (1 - |x|)^2 + (1 - |y|)^2} \tag{57}$$

The exact steady solution is $\varphi(x, y, \infty) = (1 - |x|)(1 - |y|)$. We use a uniform rectangular mesh of 40×40 elements. We impose the exact boundary conditions for $u = \varphi_x, v = \varphi_y$ from the above exact steady solution, and take the exact value at one point (the lower left corner) to recover φ. The results for P^2 and P^3 are presented in Figure 15, while Figure 16 contains the history of iterations to the steady state.

Next we take

$$I(x, y) = 1/\sqrt{1 + 4y^2(1 - x^2)^2 + 4x^2(1 - y^2)^2} \tag{58}$$

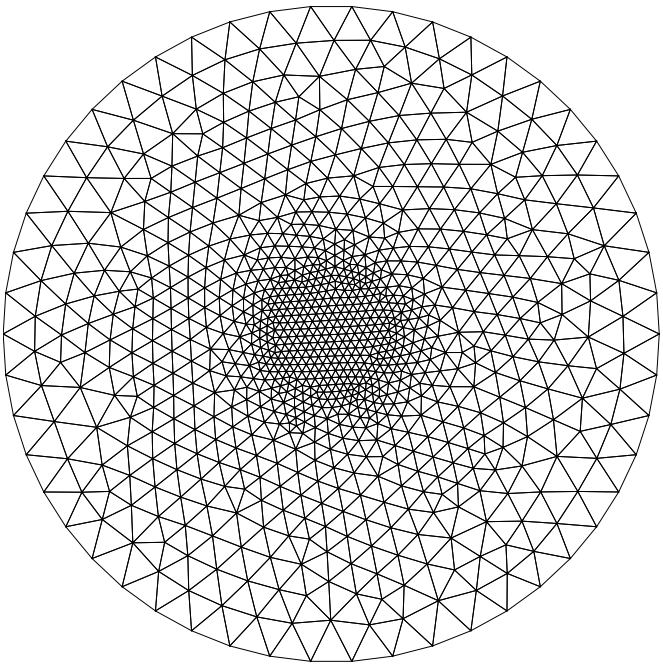

Fig. 10. Triangulation for the propagating surfaces on a disk.

The exact steady solution is $\varphi(x, y, \infty) = (1 - x^2)(1 - y^2)$. We again use a uniform rectangular mesh of 40×40 elements and impose the exact boundary conditions for $u = \varphi_x, v = \varphi_y$ from the above exact steady solution, and take the exact value at one point (the lower left corner) to recover φ. A continuation method is used, with the steady solution using

$$I_\varepsilon(x, y) = 1/\sqrt{1 + 4y^2(1 - x^2)^2 + 4x^2(1 - y^2)^2 + \varepsilon} \tag{59}$$

for bigger ε as the initial condition for smaller ε. The sequence of ε used are $\varepsilon = 0.2, 0.05, 0$. The results for P^2 and P^3 are presented in Figure 17.

8. Concluding Remarks

We have briefly surveyed the properties of Hamilton-Jacobi equations and a few numerical schemes for solving these equations. Because of space limitations, there are many related topics that we have not discussed, for example the class of central non-oscillatory schemes (e.g. [28]), techniques for efficiently solving steady state Hamilton-Jacobi equations, etc.

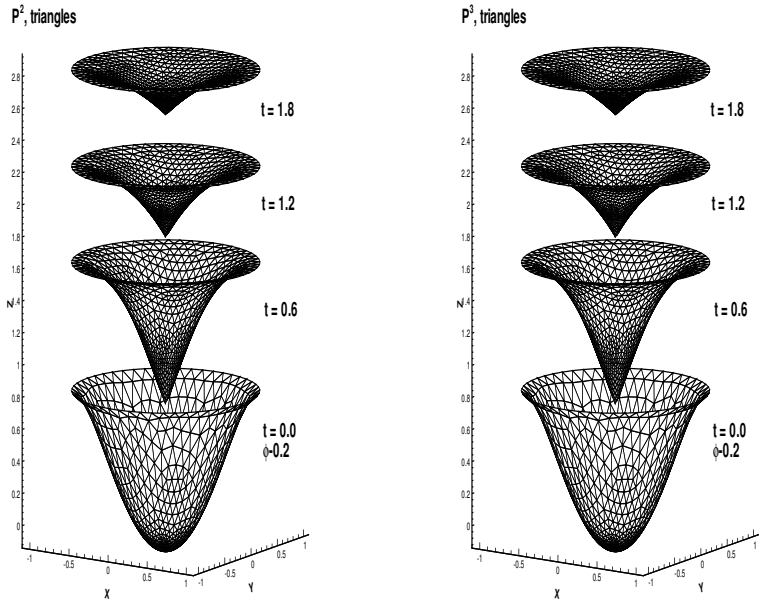

Fig. 11. Propagating surfaces on a disk, triangular mesh, $\varepsilon = 0$.

References

1. R. Abgrall, *On essentially non-oscillatory schemes on unstructured meshes: analysis and implementation*, Journal of Computational Physics, 114 (1994), 45-54.
2. R. Abgrall, *Numerical discretization of the first-order Hamilton-Jacobi equation on triangular meshes*, Communications on Pure and Applied Mathematics, 49 (1996), 1339-1373.
3. R. Abgrall and Th. Sonar, *On the use of Muehlbach expansions in the recovery step of ENO methods*, Numerische Mathematik, 76 (1997), 1-25.
4. S. Augoula and R. Abgrall, *High order numerical discretization for Hamilton-Jacobi equations on triangular meshes*, Journal of Scientific Computing, 15 (2000), 197-229.
5. M. Bardi and S. Osher, *The non-convex multi-dimensional Riemann problem for Hamilton-Jacobi equations*, SIAM Journal on Mathematical Analysis, 22 (1991), 344-351.
6. 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.
7. J. C. Butcher, *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*, John Wiley, New York, 1987.

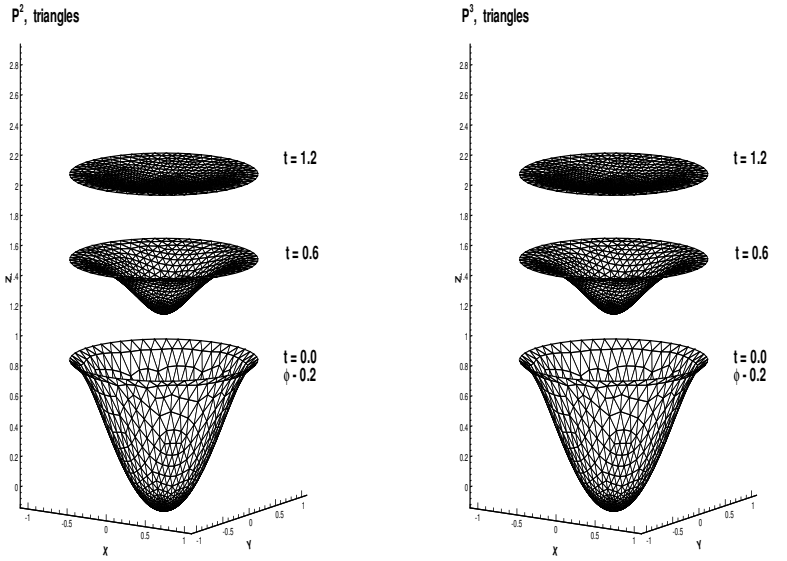

Fig. 12. Propagating surfaces on a disk, triangular mesh, $\varepsilon = 0.1$.

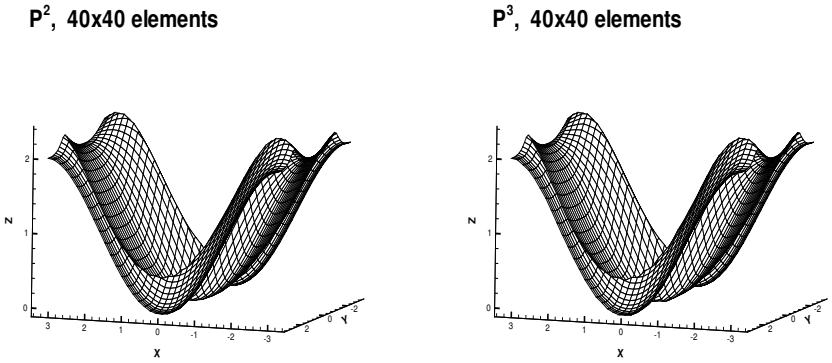

Fig. 13. Control problem, $t = 1$.

8. 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.

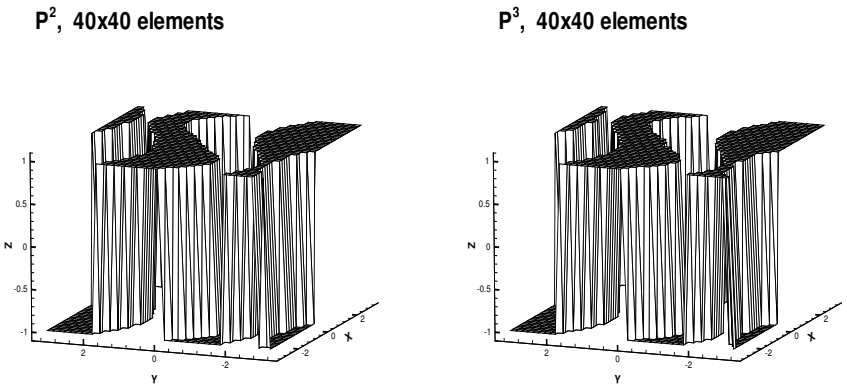

Fig. 14. Control problem, $t = 1, w = \text{sign}(\varphi_y)$.

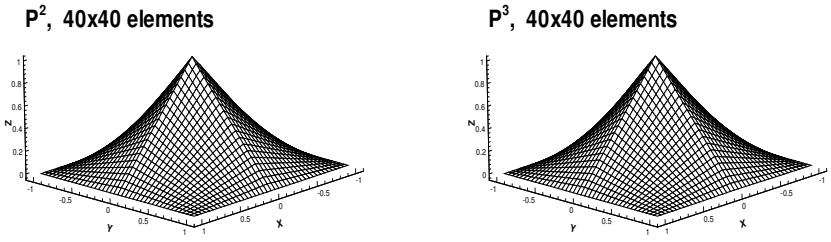

Fig. 15. Computer vision problem, $\varphi(x, y, \infty) = (1 - |x|)(1 - |y|)$.

9. B. Cockburn, F. Li and C.-W. Shu, *Locally divergence-free discontinuous Galerkin methods for the Maxwell equations*, Journal of Computational Physics, to appear.
10. 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.

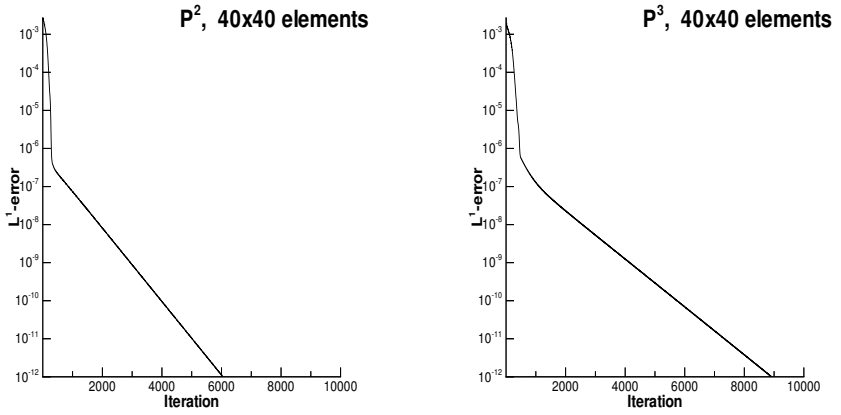

Fig. 16. Computer vision problem, history of iterations.

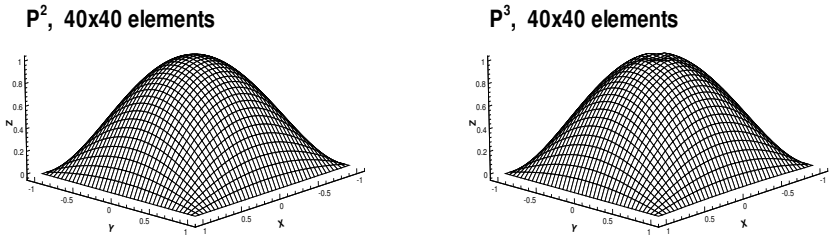

Fig. 17. Computer vision problem, $\varphi(x, y, \infty) = (1 - x^2)(1 - y^2)$.

11. 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.
12. B. Cockburn and C.-W. Shu, *Runge-Kutta discontinuous Galerkin methods for convection-dominated problems*, Journal of Scientific Computing, 16 (2001), 173-261.

13. M. Crandall and P. L. Lions, *Viscosity solutions of Hamilton-Jacobi equations*, Transactions of American Mathematical Society, 277 (1983), 1-42.
14. M. Crandall and P. L. Lions, *Monotone difference approximations for scalar conservation laws*, Mathematics of Computation, 34 (1984), 1-19.
15. S. Gottlieb and C.-W. Shu, *Total variation diminishing Runge-Kutta schemes*, Mathematics of Computation, 67 (1998), 73-85.
16. S. Gottlieb, C.-W. Shu and E. Tadmor, *Strong stability-preserving high-order time discretization methods*, SIAM Review, 43 (2001), 89-112.
17. A. Harten, B. Engquist, S. Osher and S. Chakravathy, *Uniformly high order accurate essentially non-oscillatory schemes, III*, Journal of Computational Physics, 71 (1987), 231-303.
18. C. Hu and C.-W. Shu, *A discontinuous Galerkin finite element method for Hamilton-Jacobi equations*, SIAM Journal on Scientific Computing, 21 (1999), 666-690.
19. C. Hu and C.-W. Shu, *Weighted Essentially Non-Oscillatory Schemes on Triangular Meshes*, Journal of Computational Physics, 150 (1999), 97-127.
20. G. Jiang and D.-P. Peng, *Weighted ENO schemes for Hamilton-Jacobi equations*, SIAM Journal on Scientific Computing, 21 (2000), 2126-2143.
21. G. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, Journal of Computational Physics, 126 (1996), 202-228.
22. 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.
23. P. D. Lax, *Hyperbolic Systems of Conservation Laws and the Mathematical Theory of Shock Waves*, SIAM Regional Conference series in Applied Mathematics, SIAM, Philadelphia, 1973.
24. O. Lepsky, C. Hu and C.-W. Shu, *Analysis of the discontinuous Galerkin method for Hamilton-Jacobi equations*, Applied Numerical Mathematics, 33 (2000), 423-434.
25. R. J. LeVeque, *Numerical Methods for Conservation Laws*, Birkhauser Verlag, Basel, 1992.
26. F. Li and C.-W. Shu, *Locally divergence-free discontinuous Galerkin methods for MHD equations*, Journal of Scientific Computing, to appear.
27. F. Li and C.-W. Shu, *Reinterpretation and simplified implementation of a discontinuous Galerkin method for Hamilton-Jacobi equations*, submitted to Journal of Hyperbolic Differential Equations.
28. 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.
29. P. L. Lions, *Generalized Solutions of Hamilton-Jacobi Equations*, Pitman, Boston, 1982.
30. X.-D. Liu, S. Osher and T. Chan, *Weighted essentially non-oscillatory schemes*, Journal of Computational Physics, 115 (1994), 200-212.
31. 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.

32. S. Osher and C.-W. Shu, *High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations*, SIAM Journal on Numerical Analysis, 28 (1991), 907-922.
33. E. Rouy and A. Tourin, *A viscosity solutions approach to shape-from-shading*, SIAM Journal on Numerical Analysis, 29 (1992), 867-884.
34. J. Shi, C. Hu and C.-W. Shu, *A technique of treating negative weights in WENO schemes*, Journal of Computational Physics, 175 (2002), 108-127.
35. C.-W. Shu, *Total-Variation-Diminishing time discretizations*, SIAM Journal on Scientific and Statistical Computing, 9 (1988), 1073-1084.
36. C.-W. Shu, *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws*, in *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*, B. Cockburn, C. Johnson, C.-W. Shu and E. Tadmor (Editor: A. Quarteroni), Lecture Notes in Mathematics, volume 1697, Springer, Berlin, 1998, 325-432.
37. C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes*, Journal of Computational Physics, 77 (1988), 439-471.
38. C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes II*, Journal of Computational Physics, 83 (1989), 32-78.
39. J. Smoller, *Shock Waves and Reaction-Diffusion Equations*, Springer-Verlag, New York, 1983.
40. R. Spiteri and S. Ruuth, *A new class of optimal high-order strong-stability-preserving time discretization methods*, SIAM Journal on Numerical Analysis, 40 (2002), 469-491.
41. M. Sussman, P. Smereka and S. Osher, *A level set approach for computing solution to incompressible two-phase flow*, Journal of Computational Physics, 114 (1994), 146-159.
42. 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.

SIAM J. SCI. COMPUT.
Vol. 21, No. 2, pp. 666–690

A DISCONTINUOUS GALERKIN FINITE ELEMENT METHOD FOR HAMILTON–JACOBI EQUATIONS*

CHANGQING HU† AND CHI-WANG SHU†

Abstract. In this paper, we present a discontinuous Galerkin finite element method for solving the nonlinear Hamilton–Jacobi equations. This method is based on the Runge–Kutta discontinuous Galerkin finite element method for solving conservation laws. The method has the flexibility of treating complicated geometry by using arbitrary triangulation, can achieve high-order accuracy with a local, compact stencil, and is suited for efficient parallel implementation. One- and two-dimensional numerical examples are given to illustrate the capability of the method. At least kth order of accuracy is observed for smooth problems when kth degree polynomials are used, and derivative singularities are resolved well without oscillations, even without limiters.

Key words. Hamilton–Jacobi equations, discontinuous Galerkin, high-order accuracy

AMS subject classifications. 65M60, 70H20

PII. S1064827598337282

1. Introduction.

In this paper, we consider the numerical solutions of Hamilton–Jacobi (HJ) equations

$$\varphi_t + H(\varphi_{x_1}, \ldots, \varphi_{x_d}) = 0, \qquad \varphi(x,0) = \varphi^0(x). \tag{1.1}$$

As is well known, the solutions to (1.1) are Lipschitz continuous but may have discontinuous derivatives, regardless of the smoothness of the initial condition $\varphi^0(x)$. The nonuniqueness of such solutions also necessitates the definition of viscosity solutions, to single out a unique, practically relevant solution. See Crandall and Lions [12].

A widely applied class of numerical schemes for (1.1) is the class of finite difference schemes. In [13], Crandall and Lions proved the convergence of monotone finite difference schemes to the viscosity solutions of (1.1). Unfortunately, monotone schemes are at most first-order accurate, measured by local truncation errors in smooth regions of the solution. In Osher and Sethian [23] and Osher and Shu [24], a class of high-order essentially nonoscillatory (ENO) schemes was introduced, which were adapted from the methodologies for hyperbolic conservation laws [15, 26, 27]. The numerical results in [23] and [24] indicate convergence to the viscosity solutions of (1.1) with high-order accuracy in smooth regions and with sharp resolution of the discontinuous derivatives. Recently, weighted ENO (WENO) schemes for conservation laws [21, 18] have also been adapted to the HJ equations (1.1) [16].

Finite difference methods require a structured mesh and hence are difficult to apply to complicated geometry or for adaptive mesh refinements. Finite volume schemes, which are based on arbitrary triangulation, are thus attractive in such cases. First-order monotone-type finite volume schemes and their second extensions were studied in [2]. A second-order ENO-type finite volume scheme is developed in [20]. However,

*Received by the editors April 13, 1998; accepted for publication (in revised form) November 11, 1998; published electronically October 14, 1999. The research of this paper was supported in part by ARO grant DAAG55-97-1-0318, NSF grants DMS-9500814 and DMS-9804985, ECS-9627849, and INT-9601084, NASA Langley grant NAG-1-1145 and contract NAS1-12070 while the second author was in residence at ICASE, NASA Langley Research Center, and AFOSR grant F49620-96-1-0150.
http://www.siam.org/journals/sisc/21-2/33728.html

†Division of Applied Mathematics, Brown University, Providence, RI 02912 (hu@cfm.brown.edu, shu@cfm.brown.edu).

higher-order finite volume schemes face the problem of reconstruction on arbitrary triangulation, which is quite complicated (see, e.g., [1]). More recently, a continuous finite element method for solving (1.1) was proposed in [3].

The Runge–Kutta discontinuous Galerkin method [6, 7, 8, 9, 10] is a method devised to numerically solve the conservation law (CL)

$$u_t + f_1(u)_{x_1} + \cdots + f_d(u)_{x_d} = 0, \qquad u(x,0) = u^0(x). \tag{1.2}$$

The method has the following attractive properties:

- It can be designed for any order of accuracy in space and time. In fact, p-version or spectral element–type version can be designed [22].
- It is an explicit method, thus efficient for solving the hyperbolic problem (1.2). No global linear or nonlinear systems must be solved.
- It combines the flexibility of finite element methods in the easy handling of complicated geometry, with the high resolution property for discontinuous solutions of finite difference and finite volume methods through monotone fluxes or approximate Riemann solvers applied at the element interfaces and limiters.
- It has nice stability properties: a local cell entropy inequality for the square entropy can be proven [17] for general triangulation for any scalar nonlinear conservation laws (1.2) in any spatial dimensions and for any order of accuracy, without the need of nonlinear limiters. This implies nonlinear L^2 stability and entropy consistency even for discontinuous solutions.
- The method is highly compact: the evolution of the information in any element depends only on the information of itself and its immediate neighbors, regardless of the order of accuracy. This is in contrast with high-order finite volume schemes which must use wide stencils for the high-order reconstruction. This compactness is responsible for the efficient parallel implementation of the method; see, e.g., [4].

For details of the Runge–Kutta discontinuous Galerkin method, see the references listed above and also the review paper of Cockburn [5]. This method can also be generalized to solve problems containing higher derivatives [11], which is important for some of our applications in section 4.

It is well known that the HJ equation (1.1) is closely related to the CL (1.2); in fact in one-space dimension $d = 1$ they are equivalent if one takes $\varphi = u_x$. It is thus not surprising that many successful numerical methods for the HJ equation (1.1) are adapted from those for the conservation law (1.2). The high-order finite difference ENO methods in [23] and [24] are such examples.

In this paper we adapt the Runge–Kutta discontinuous Galerkin method to solve the HJ equation (1.1). In section 2 the algorithm in one-space dimension is described and discussed. In section 3 the algorithm for two-space dimensions is developed. Numerical examples of one and two spatial dimensions are presented in section 4. At least kth order of accuracy is observed for smooth problems when kth degree polynomials are used, and derivative singularities are resolved well without oscillations, even without limiters. Concluding remarks are given in section 5.

2. One-dimensional case. In one-space dimension, (1.1) becomes

$$\varphi_t + H(\varphi_x) = 0, \qquad \varphi(x,0) = \varphi^0(x). \tag{2.1}$$

This is a relatively easy case because (2.1) is equivalent to the CL

$$u_t + H(u)_x = 0, \qquad u(x,0) = u^0(x) \tag{2.2}$$

if we identify $u = \varphi_x$.

Assuming (2.1) is solved in the interval $a \le x \le b$ and it is divided into the following cells,

$$a = x_{\frac{1}{2}} < x_{\frac{3}{2}} < \cdots < x_{N+\frac{1}{2}} = b, \tag{2.3}$$

we denote

$$I_j = (x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}),\ x_j = \frac{1}{2}\left(x_{j-\frac{1}{2}} + x_{j+\frac{1}{2}}\right)\ h_j = x_{j+\frac{1}{2}} - x_{j-\frac{1}{2}},\ h = \max_j h_j, \tag{2.4}$$

and define the following approximation space

$$V_h^k = \left\{v : v|_{I_j} \in P^k(I_j), j = 1, \ldots, N\right\}. \tag{2.5}$$

Here $P^k(I_j)$ is the set of all polynomials of degree at most k on the cell I_j.

A kth-order discontinuous Galerkin scheme for the one-dimensional HJ equation (2.1) can then be defined as follows: find $\varphi \in V_h^k$ such that

$$\int_{I_j} \varphi_{xt} v\, dx - \int_{I_j} H(\varphi_x) v_x\, dx + \hat{H}_{j+\frac{1}{2}} v^-_{j+\frac{1}{2}} - \hat{H}_{j-\frac{1}{2}} v^+_{j-\frac{1}{2}} = 0, \qquad j = 1, \ldots, N, \tag{2.6}$$

$\forall\, v \in V_h^{k-1}$. Here

$$\hat{H}_{j+\frac{1}{2}} = \hat{H}\left((\varphi_x)^-_{j+\frac{1}{2}}, (\varphi_x)^+_{j+\frac{1}{2}}\right) \tag{2.7}$$

is a monotone flux, i.e., $\hat{H}$ is nondecreasing in the first argument and nonincreasing in the second. Symbolically $\hat{H}(\uparrow, \downarrow)$, is Lipschitz continuous in both arguments and is consistent, i.e., $\hat{H}(u, u) = H(u)$. We will mainly use the simple (local) Lax–Friedrichs flux

$$\hat{H}(u^-, u^+) = H\left(\frac{u^- + u^+}{2}\right) - \frac{1}{2}\alpha\,(u^+ - u^-), \tag{2.8}$$

where $\alpha = \max_u |H'(u)|$ with the maximum taken over the range covered by u^- and u^+. For other monotone fluxes, e.g., the Godunov flux, see [24]. Notice that the method described above is exactly the discontinuous Galerkin method for the CL equation (2.2) satisfied by the derivative $u = \varphi_x$. This only determines φ for each element up to a constant, since it is only a scheme for φ_x. The missing constant can be obtained in one of the following two ways:

1. Require that

$$\int_{I_j} (\varphi_t + H(\varphi_x))\, v\, dx = 0, \qquad j = 1, \ldots, N, \tag{2.9}$$

for all $v \in V_h^0$, that is,

$$\int_{I_j} (\varphi_t + H(\varphi_x))\, dx = 0, \qquad j = 1, \ldots, N. \tag{2.10}$$

2. Use (2.10) to update only one (or a few) elements, e.g., the left-most element I_1; then use

$$\varphi(x_j, t) = \varphi(x_1, t) + \int_{x_1}^{x_j} \varphi_x(x, t)\, dx \tag{2.11}$$

to determine the missing constant for the cell I_j.

We remark that, in the second approach, the recovered values of φ are dependent upon the choice of the starting point x_1. However, this difference is on the level of truncation errors and does not affect the order of accuracy. See Examples 4.4 and 4.6 in section 4. Both approaches are used in our numerical experiments. They perform similarly for smooth problems, with the first approach giving slightly better results. However, it is our numerical experience that, when there are singularities in the derivatives, the first approach will often produce dents and bumps when the integral path in time passes through the singularities at some earlier time. The philosophy of using the second approach is that one could update only a few elements whose time integral paths do not cross derivative singularities. The numerical results shown in section 4 are obtained with the second approach.

About the stability of the method proposed above, we can quote the following result of Jiang and Shu [17]. Here we assume compact support or periodic boundary condition for φ.

LEMMA 2.1 (see [17]). *The following L^2 stability result for the derivative φ_x holds for the discontinuous Galerkin method defined above, of any order of accuracy k applied to any nonlinear HJ equation* (2.1):

$$\frac{d}{dt}\int_a^b \varphi_x^2\,dx \le 0. \tag{2.12}$$

For a finite interval $[a,b]$, (2.12) trivially implies total variation bounded (TVB) property for the numerical solution φ:

$$TV(\varphi) = \int_a^b |\varphi_x|\,dx \le \sqrt{b-a}\sqrt{\int_a^b \left(\frac{d}{dx}\varphi^0(x)\right)^2 dx}. \tag{2.13}$$

This is a rather strong stability result, considering that it applies even if the derivative of the solution φ_x develops discontinuities, no limiter has been added to the numerical scheme, and the scheme can be of arbitrary high order in accuracy. It also implies convergence of at least a subsequence of the numerical solution φ when $h \to 0$. However, this stability result is not strong enough to imply that the limit solution is the viscosity solution of (2.1).

Up to now we have only described the spatial discretization and have left the time variable t continuous (the method of lines approach). Time discretization is by the total variation diminishing (TVD) high-order Runge–Kutta methods developed in [26]; see also [14]. The second- and third-order versions used in this paper are as follows: for solving the method of lines ODE

$$\varphi_t = L(\varphi), \tag{2.14}$$

the second-order TVD Runge–Kutta method is given by

$$\begin{aligned} \varphi^{(1)} &= \varphi^n + \Delta t L(\varphi^n), \\ \varphi^{n+1} &= \frac{1}{2}\varphi^n + \frac{1}{2}\varphi^{(1)} + \frac{1}{2}\Delta t L(\varphi^{(1)}), \end{aligned} \tag{2.15}$$

and the third-order TVD Runge–Kutta method is given by

$$\begin{aligned} \varphi^{(1)} &= \varphi^n + \Delta t L(\varphi^n), \\ \varphi^{(2)} &= \frac{3}{4}\varphi^n + \frac{1}{4}\varphi^{(1)} + \frac{1}{4}\Delta t L(\varphi^{(1)}), \\ \varphi^{n+1} &= \frac{1}{3}\varphi^n + \frac{2}{3}\varphi^{(2)} + \frac{2}{3}\Delta t L(\varphi^{(2)}). \end{aligned} \tag{2.16}$$

These Runge–Kutta methods will also be used for the two-dimensional case discussed in the next section.

3. Two-dimensional case. We consider in this section the case of two spatial dimensions. The algorithm in more spatial dimensions is similar. This time, the scalar HJ equation

$$\varphi_t + H(\varphi_x, \varphi_y) = 0, \qquad \varphi(x, y, 0) = \varphi^0(x, y) \tag{3.1}$$

is in some sense equivalent to the following CL *system*

$$u_t + H(u, v)_x = 0, \qquad v_t + H(u, v)_y = 0, \qquad (u, v)(x, y, 0) = (u, v)^0(x, y) \tag{3.2}$$

if we identify

$$(u, v) = (\varphi_x, \varphi_y). \tag{3.3}$$

For example, a vanishing viscosity solution of (3.1) corresponds, via (3.3), to a vanishing viscosity solution of (3.2), and vice versa [19]. However, (3.2) is not a strictly hyperbolic system, which may cause problems in its numerical solution if we treat u and v as independent variables. Instead, we would still like to use φ as our solution variable (a polynomial) and take its derivatives as u and v.

Assume we are solving (3.1) in the domain Ω, which has a triangulation $\mathcal{T}_h$ consisting of triangles or general polygons of maximum size (diameter) h, with the following approximation space,

$$V_h^k = \left\{ v : v|_K \in P^k(K) \ \ \forall\, K \in \mathcal{T}_h \right\}, \tag{3.4}$$

where $P^k(K)$ is again the set of all polynomials of degree at most k on the cell K. We propose a discontinuous Galerkin method for (3.1) as follows: find $\varphi \in V_h^k$ such that

$$\int_K \varphi_{xt} v \, dxdy - \int_K H(\varphi_x, \varphi_y) v_x \, dxdy + \sum_{e \in \partial K} \int_e \hat{H}_{1,e,K} \, v \, d\Gamma = 0 \tag{3.5}$$

and

$$\int_K \varphi_{yt} v \, dxdy - \int_K H(\varphi_x, \varphi_y) v_y \, dxdy + \sum_{e \in \partial K} \int_e \hat{H}_{2,e,K} \, v \, d\Gamma = 0 \tag{3.6}$$

for all $v \in V_h^{k-1}$ and all $K \in \mathcal{T}_h$, *in a least square sense*. Here the numerical flux is

$$\hat{H}_{i,e,K} = \hat{H}_{i,e,K} \left((\nabla\varphi)^{int(K)}, (\nabla\varphi)^{ext(K)} \right), \qquad i = 1, 2, \tag{3.7}$$

where the superscript $\mathrm{int}(K)$ implies that the value is taken from within the element K, and the superscript $\mathrm{ext}(K)$ implies that the value is taken from outside the element K and within the neighboring element K' sharing the edge e with K. The flux (3.7) satisfies the following properties:

1. $\hat{H}_{i,e,K}$ is Lipschitz continuous with respect to all its arguments.
2. Consistency:

$$\hat{H}_{i,e,K} \left(\nabla\varphi, \nabla\varphi \right) = H(\nabla\varphi) \, n_i,$$

where $n = (n_1, n_2)$ is the unit outward normal to the edge e of the element K.

3. Conservation:

$$\hat{H}_{i,e,K}\left((\nabla\varphi)^{int(K)},(\nabla\varphi)^{ext(K)}\right)=-\hat{H}_{i,e,K'}\left((\nabla\varphi)^{int(K')},(\nabla\varphi)^{ext(K')}\right),$$

where $K\cap K'=e$.

We will again mainly use the simple (local) Lax–Friedrichs flux

$$\hat{H}_{1,e,K}((u,v)^-,(u,v)^+)=H\left(\frac{u^-+u^+}{2},\frac{v^-+v^+}{2}\right)n_1-\frac{1}{2}\alpha(u^+-u^-) \tag{3.8}$$

and

$$\hat{H}_{2,e,K}((u,v)^-,(u,v)^+)=H\left(\frac{u^-+u^+}{2},\frac{v^-+v^+}{2}\right)n_2-\frac{1}{2}\beta(v^+-v^-), \tag{3.9}$$

where $\alpha=\max_{u,v}|\frac{\partial H(u,v)}{\partial u}|$ and $\beta=\max_{u,v}|\frac{\partial H(u,v)}{\partial v}|$, with the maximum being taken over the relevant (local) range.

Notice that (3.5)–(3.6) is exactly the discontinuous Galerkin method for the CL system (3.2) satisfied by the derivatives $(u,v)=\nabla\varphi$. For a rectangular mesh and for $k=1$, this recovers the (local) Lax–Friedrichs monotone scheme [24] for (3.1) if we identify $u_{ij}=\frac{\varphi_{i+1,j}-\varphi_{i-1,j}}{2\Delta x}$ and $v_{ij}=\frac{\varphi_{i,j+1}-\varphi_{i,j-1}}{2\Delta y}$.

Of course, (3.5)–(3.6) have more equations than the number of degrees of freedoms (an overdetermined system) for $k>1$, thus a least square solution is needed. In practice, the least square procedure is performed as follows: we first evolve (3.2) for one time step (one inner stage for high-order Runge–Kutta methods), using the discontinuous Galerkin method (3.5)–(3.6) with $u=\varphi_x$ and $v=\varphi_y$; then φ at the next time level (stage) is obtained (up to a constant) by least square:

$$||(\varphi_x-u)^2+(\varphi_y-v)^2||_{L^1(K)}=\min_{\psi\in P^k(K)}||(\psi_x-u)^2+(\psi_y-v)^2||_{L^1(K)}.$$

This determines φ for each element up to a constant, since it is only a scheme for $\nabla\varphi$. The missing constant can again be obtained in one of the following two ways:

1. Require that

$$\int_K(\varphi_t+H(\varphi_x,\varphi_y))\,v\,dxdy\;=\;0 \tag{3.10}$$

for all $v\in V_h^0$ and for all $K\in\mathcal{T}_h$, that is,

$$\int_K(\varphi_t+H(\varphi_x,\varphi_y))\;dxdy\;=\;0\qquad\forall\,K\in\mathcal{T}_h\,. \tag{3.11}$$

2. Use (3.11) to update only one (or a few) element(s), e.g., the corner element(s); then use

$$\varphi(B,t)=\varphi(A,t)+\int_A^B(\varphi_x\,dx+\varphi_y\,dy) \tag{3.12}$$

to determine the missing constant. The path should be taken to avoid crossing a derivative discontinuity, if possible.

TABLE 4.1
Accuracy for one-dimensional Burgers equation (uniform mesh), $t = 0.5/\pi^2$.

	P^1		P^2		P^3		P^4	
N	L^1 error	order	L^1 error	order	L^1 error	order	L^1 error	order
10	0.17E+00	—	0.14E-02	—	0.21E-03	—	0.57E-05	—
20	0.78E-01	1.12	0.18E-03	2.92	0.13E-04	3.94	0.73E-06	2.97
40	0.35E-01	1.16	0.24E-04	2.97	0.75E-06	4.17	0.32E-07	4.52
80	0.16E-01	1.12	0.28E-05	3.08	0.43E-07	4.12	0.12E-08	4.79
160	0.76E-02	1.02	0.31E-06	3.19	0.25E-08	4.10	0.48E-10	4.59

	P^1		P^2		P^3		P^4	
N	L^∞ error	order	L^∞ error	order	L^∞ error	order	L^∞ error	order
10	0.29E+00	—	0.24E-02	—	0.69E-03	—	0.13E-04	—
20	0.13E+00	1.13	0.33E-03	2.88	0.61E-04	3.51	0.16E-05	2.99
40	0.58E-01	1.15	0.37E-04	3.15	0.58E-05	3.39	0.13E-06	3.64
80	0.27E-01	1.11	0.48E-05	2.97	0.38E-06	3.93	0.59E-08	4.44
160	0.13E-01	1.07	0.59E-06	3.00	0.23E-07	4.07	0.25E-09	4.57

We remark again that, in the second approach, the recovered values of φ are dependent upon the choice of the starting point A as well as the integration path. However, this difference is on the level of truncation errors and does not affect the order of accuracy. See Examples 4.4 and 4.6 in section 4. It is important here that φ is a single function and u and v are just its x and y derivatives. Otherwise, the second approach would be questionable in effectively recovering φ.

In section 4 we will show only numerical results obtained with the second approach; see the remarks in the previous section for the one-dimensional case.

We remark that the procedure discussed above is easily implemented in any triangulation, e.g., for both rectangles and triangles.

4. Numerical examples. We now present some results attained by the methods above.

EXAMPLE 4.1. *One-dimensional Burgers equation:*

$$\begin{cases} \varphi_t + \frac{(\varphi_x+1)^2}{2} = 0, & -1 < x < 1, \\ \varphi(x,0) = -\cos(\pi x) \end{cases} \tag{4.1}$$

with periodic boundary conditions.

The local Lax–Friedrichs flux (2.8) is used. At $t = 0.5/\pi^2$, the solution is still smooth. We list the errors and the numerical orders of accuracy in Table 4.1. We observe that, except for the P^1 case which seems to be only first order, P^k for $k > 1$ seems to provide close to $(k+1)$th-order accuracy. The meshes used are all uniform, and errors are computed at the middle point of each interval.

To investigate the accuracy problem further, we use nonuniform meshes obtained by randomly shifting the cell boundaries in a uniform mesh in the range $[-0.1h, 0.1h]$. In order to avoid possible superconvergence at cell centers, we also give the "real" L^2 error (computed by a six-point Gaussian quadrature in each cell). The results are shown in Table 4.2.

At $t = 3.5/\pi^2$, the solution has developed a discontinuous derivative. In Figure 4.1, we show the sharp corner-like numerical solution with 41 elements obtained with P^k for $k = 1, 2, 3, 4$ with a uniform mesh. Here and below, the solid line is the exact solution, the circles are numerical solutions (only one point per element is drawn).

TABLE 4.2
Accuracy for one-dimensional Burgers equation (nonuniform mesh), $t = 0.5/\pi^2$.

	P^1		P^2		P^3		P^4	
N	L^2 error	order	L^2 error	order	L^2 error	order	L^2 error	order
10	0.74E+00	—	0.34E-02	—	0.32E-03	—	0.53E-04	—
20	0.34E+00	1.11	0.51E-03	2.76	0.24E-04	3.72	0.20E-05	4.71
40	0.15E+00	1.19	0.65E-04	2.96	0.17E-05	3.82	0.71E-07	4.84
80	0.67E-01	1.17	0.90E-05	2.86	0.13E-06	3.72	0.20E-08	5.15
160	0.31E-01	1.13	0.11E-05	3.02	0.81E-08	4.02	0.74E-10	4.76

	P^1		P^2		P^3		P^4	
N	L^1 error	order	L^1 error	order	L^1 error	order	L^1 error	order
10	0.53E+00	—	0.17E-02	—	0.23E-03	—	0.30E-05	—
20	0.24E+00	1.13	0.21E-03	3.05	0.14E-04	4.01	0.40E-06	2.89
40	0.11E+00	1.19	0.26E-04	2.99	0.78E-06	4.20	0.16E-07	4.65
80	0.47E-01	1.17	0.37E-05	2.82	0.47E-07	4.05	0.61E-09	4.70
160	0.21E-01	1.13	0.41E-06	3.16	0.27E-08	4.15	0.26E-10	4.56

	P^1		P^2		P^3		P^4	
N	L^∞ error	order	L^∞ error	order	L^∞ error	order	L^∞ error	order
10	0.62E+00	—	0.36E-02	—	0.69E-03	—	0.11E-04	—
20	0.29E+00	1.11	0.47E-03	2.94	0.61E-04	3.52	0.16E-05	2.81
40	0.13E+00	1.16	0.67E-04	2.80	0.47E-05	3.70	0.13E-06	3.64
80	0.58E-01	1.14	0.17E-04	2.01	0.62E-06	2.91	0.59E-08	4.45
160	0.27E-01	1.11	0.19E-05	3.11	0.31E-07	4.32	0.33E-09	4.17

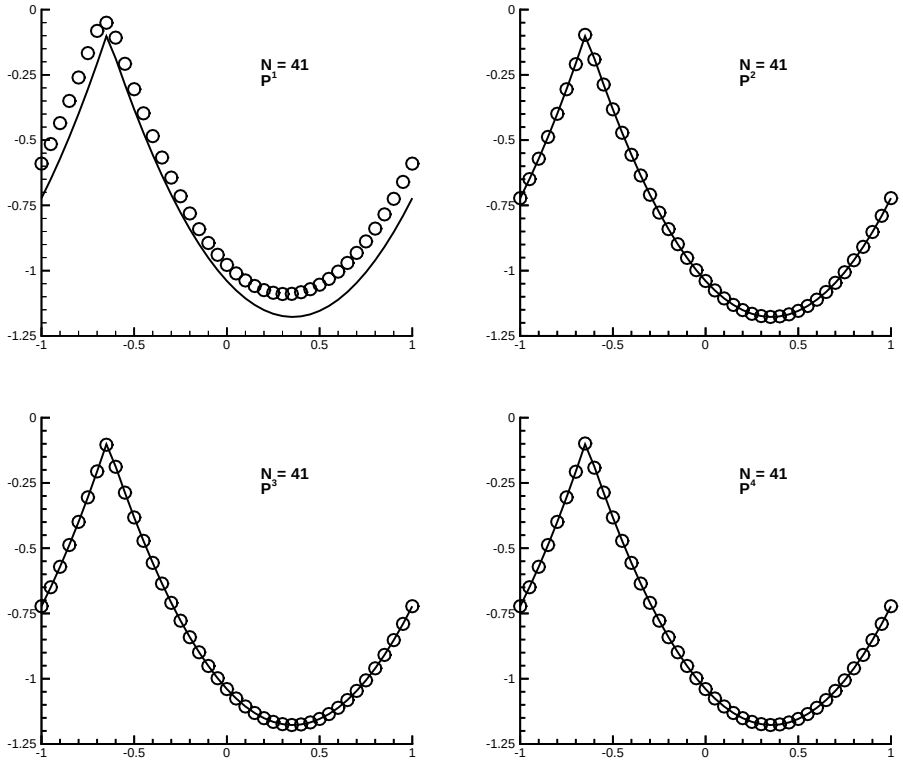

FIG. 4.1. *One-dimensional Burgers equation, $t = 3.5/\pi^2$.*

TABLE 4.3
Accuracy for one-dimensional nonconvex, $H(u) = -\cos(u+1)$, $t = 0.5/\pi^2$.

	P^1		P^2		P^3		P^4	
N	L^1 error	order	L^1 error	order	L^1 error	order	L^1 error	order
10	0.84E-01	—	0.10E-02	—	0.34E-03	—	0.24E-04	—
20	0.36E-01	1.23	0.15E-03	2.75	0.30E-04	3.49	0.13E-05	4.28
40	0.15E-01	1.26	0.21E-04	2.84	0.15E-05	4.33	0.59E-07	4.42
80	0.68E-02	1.14	0.27E-05	2.97	0.94E-07	4.00	0.21E-08	4.78

	P^1		P^2		P^3		P^4	
N	L^∞ error	order	L^∞ error	order	L^∞ error	order	L^∞ error	order
10	0.18E+00	—	0.15E-02	—	0.11E-02	—	0.99E-04	—
20	0.73E-01	1.31	0.27E-03	2.43	0.22E-03	2.35	0.13E-04	2.95
40	0.31E-01	1.24	0.47E-04	2.54	0.18E-04	3.63	0.59E-06	4.44
80	0.14E-01	1.16	0.85E-05	2.47	0.14E-05	3.75	0.26E-07	4.49

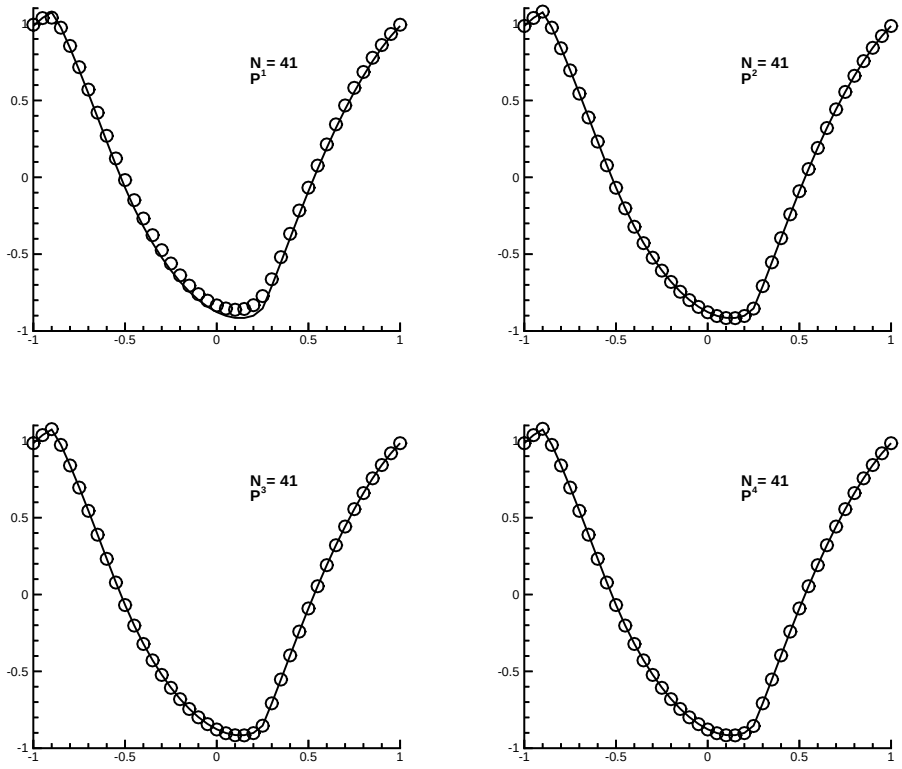

FIG. 4.2. *One dimension, nonconvex, $H(u) = -\cos(u+1)$, $t = 1.5/\pi^2$.*

EXAMPLE 4.2. *One-dimensional equation with a nonconvex flux:*

$$\begin{cases} \varphi_t - \cos(\varphi_x + 1) = 0, & -1 < x < 1, \\ \varphi(x, 0) = -\cos(\pi x) \end{cases} \tag{4.2}$$

with periodic boundary conditions.

The local Lax–Friedrichs flux (2.8) and uniform meshes are used. At $t = 0.5/\pi^2$, the solution is still smooth. The accuracy of the numerical solution is listed in Table 4.3. We observe similar accuracy as in the previous example.

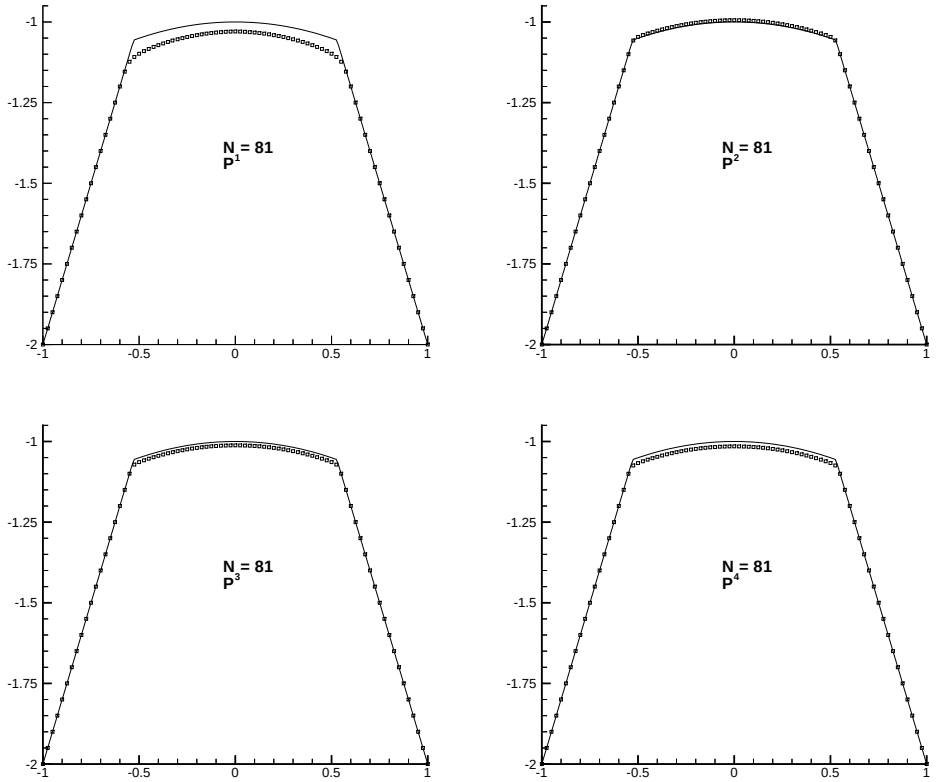

FIG. 4.3. *One-dimensional Riemann problem, local Lax–Friedrichs flux,* $H(u) = \frac{1}{4}(u^2-1)(u^2-4), t = 1$.

At $t = 1.5/\pi^2$, the solution has developed corner-like discontinuity in the derivative. The numerical result with 41 elements is shown in Figure 4.2.

EXAMPLE 4.3. *The one-dimensional Riemann problem with a nonconvex flux:*

$$\begin{cases} \varphi_t + \frac{1}{4}(\varphi_x^2 - 1)(\varphi_x^2 - 4) = 0, & -1 < x < 1, \\ \varphi(x, 0) = -2|x|. \end{cases} \tag{4.3}$$

For this test problem, the discontinuous Galerkin method, as described in section 2 and applied in the previous two examples, fails to "open up" the initial single discontinuity in the derivative sufficiently to generate the correct entropy solution. We have found out that a nonlinear total variation bounded limiting, described in detail in [7] for this one-dimensional case, is needed for convergence toward the entropy solution. This and the two-dimensional Riemann problem in Example 4.7 below are the only two examples in this paper in which we use the nonlinear limiting. We remark that for the finite difference schemes, such nonlinear limiting or the adaptive stencil in ENO is needed in most cases in order to enforce stability and to obtain nonoscillatory results.

Numerical results at $t = 1$ with 81 elements, using the local Lax–Friedrichs flux (2.8), is shown in Figure 4.3. The results of using the Godunov flux is shown in Figure 4.4. We can see that while for P^1, the results of using two different monotone fluxes are significantly different in resolution, this difference is greatly reduced for higher

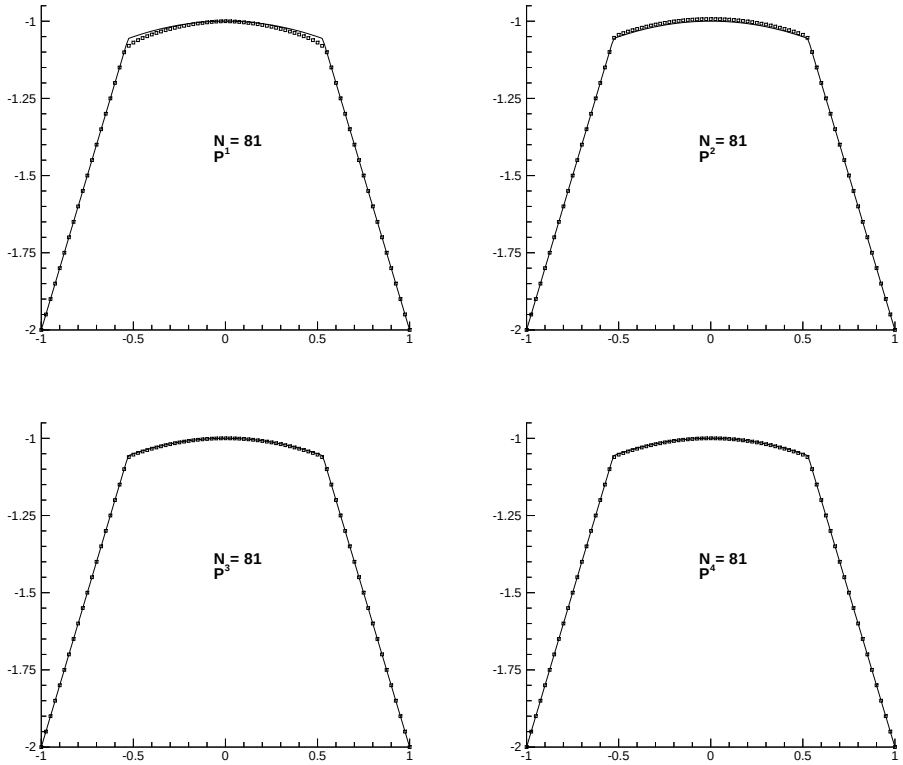

FIG. 4.4. *One-dimensional Riemann problem, Godunov flux,* $H(u) = \frac{1}{4}(u^2-1)(u^2-4), t = 1$.

order of accuracy. In most of the high-order cases, the simple local Lax–Friedrichs flux (2.8) should be good enough.

EXAMPLE 4.4. *Two-dimensional Burgers equation:*

$$\begin{cases} \varphi_t + \frac{(\varphi_x+\varphi_y+1)^2}{2} = 0, & -2 < x < 2,\ -2 < y < 2, \\ \varphi(x,y,0) = -\cos\left(\frac{\pi(x+y)}{2}\right) \end{cases} \tag{4.4}$$

with periodic boundary conditions.

We first use uniform rectangular meshes and the local Lax–Friedrichs flux (3.8)–(3.9). At $t = 0.5/\pi^2$, the solution is still smooth. The errors (computed at the center of the cells) and orders of accuracy are listed in Table 4.4. It seems that only kth order of accuracy is achieved when φ is a piecewise polynomial of degree k.

Next, as in the one-dimensional case, we use nonuniform rectangular meshes obtained from the tensor product of one-dimensional nonuniform meshes (the meshes in two directions are independent). Again, we give the "real" L^2-errors computed by a 6×6 point Gaussian quadrature as well. The results are shown in Table 4.5.

The results in the previous two tables are obtained by updating the element at the lower left corner with time, then taking an integration path consisting of line segments starting from the corner and parallel to the x-axis first and then vertically to the point. To further address the issue of the dependency of the computed values of the solution φ on the integration path and starting point, we use another path which starts vertically, then runs parallel with the x-axis to reach the point. In Table 4.6,

TABLE 4.4
Accuracy for two-dimensional Burgers equation, uniform rectangular mesh, $t = 0.5/\pi^2$.

	P^1		P^2		P^3	
$N \times N$	L^1 error	order	L^1 error	order	L^1 error	order
10×10	8.09E-02	—	8.62E-03	—	3.19E-03	—
20×20	3.36E-02	1.268	1.72E-03	2.325	3.49E-04	3.192
40×40	1.48E-02	1.183	3.93E-04	2.130	6.64E-05	2.394
80×80	6.88E-03	1.105	9.74E-05	2.013	1.14E-05	2.542
160×160	3.31E-03	1.056	2.45E-05	1.991	1.68E-06	2.763

	P^1		P^2		P^3	
$N \times N$	L^∞ error	order	L^∞ error	order	L^∞ error	order
10×10	2.62E-01	—	3.56E-02	—	8.65E-03	—
20×20	1.14E-01	1.201	8.40E-03	2.083	1.16E-03	2.899
40×40	5.00E-02	1.189	2.02E-03	2.056	1.98E-04	2.551
80×80	2.39E-02	1.065	4.92E-04	2.038	3.13E-05	2.661
160×160	1.16E-02	1.043	1.21E-04	2.024	4.41E-06	2.827

TABLE 4.5
Accuracy for two-dimensional Burgers equation, nonuniform rectangular mesh, $t = 0.5/\pi^2$.

	P^1		P^2		P^3	
$N \times N$	L^2 error	order	L^2 error	order	L^2 error	order
10×10	4.47E-01	—	6.28E-02	—	1.61E-02	—
20×20	1.83E-01	1.288	1.50E-02	2.066	2.06E-03	2.966
40×40	8.01E-02	1.192	3.63E-03	2.047	3.48E-04	2.565
80×80	3.82E-02	1.068	9.17E-04	1.985	6.03E-05	2.529
160×160	1.87E-02	1.031	2.34E-04	1.970	8.58E-06	2.813

	P^1		P^2		P^3	
$N \times N$	L^1 error	order	L^1 error	order	L^1 error	order
10×10	8.16E-02	—	9.16E-03	—	3.39E-03	—
20×20	3.41E-02	1.259	2.09E-03	2.132	4.12E-04	3.041
40×40	1.50E-02	1.185	5.21E-04	2.004	7.03E-05	2.551
80×80	7.16E-03	1.067	1.42E-04	1.875	1.24E-05	2.503
160×160	3.50E-03	1.033	3.85E-05	1.883	1.76E-06	2.817

	P^1		P^2		P^3	
$N \times N$	L^∞ error	order	L^∞ error	order	L^∞ error	order
10×10	2.83E-01	—	4.68E-02	—	1.00E-02	—
20×20	1.25E-01	1.179	1.23E-02	1.928	1.39E-03	2.847
40×40	5.74E-02	1.123	3.54E-03	1.797	2.29E-04	2.602
80×80	2.78E-02	1.046	1.15E-03	1.622	5.11E-05	2.164
160×160	1.42E-02	0.969	2.72E-04	2.080	7.16E-06	2.835

we list the *difference* of two recovered solutions φ from these two different integration paths, for the nonuniform mesh cases. We can see that these differences are at the levels of local truncation errors and decay in the same order as the errors. Thus the choice of integration path in recovering φ does not affect accuracy.

At $t = 1.5/\pi^2$, the solution has discontinuous derivatives. Figure 4.5 is the graph of the numerical solution with 40×40 elements (uniform mesh).

Finally, we use triangle-based triangulation; the mesh with $h = \frac{1}{4}$ is shown in Figure 4.6. The accuracy at $t = 0.5/\pi^2$ is shown in Table 4.7. Similar accuracy pattern is observed as in the rectangular case. The result at $t = 1.5/\pi^2$, when the derivative is discontinuous, is shown in Figure 4.7.

TABLE 4.6
Differences of the solution φ recovered by two different integration paths, nonuniform mesh, Burgers equation.

	P^1		P^2		P^3	
$N \times N$	L^1 error	order	L^1 error	order	L^1 error	order
10×10	8.61E-03	—	2.90E-03	—	1.15E-03	—
20×20	4.64E-03	0.892	1.28E-03	1.180	2.44E-04	2.237
40×40	2.54E-03	0.869	4.12E-04	1.635	3.76E-05	2.698
80×80	1.81E-03	0.489	1.39E-04	1.568	6.71E-06	2.486
160×160	1.09E-03	0.732	3.66E-05	1.925	8.79E-07	2.932

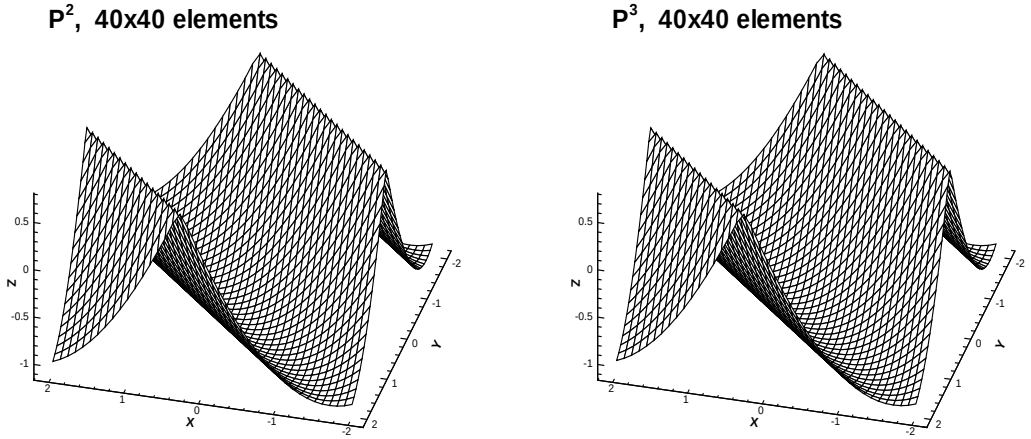

FIG. 4.5. *Two-dimensional Burgers equation, rectangular mesh, $t = 1.5/\pi^2$.*

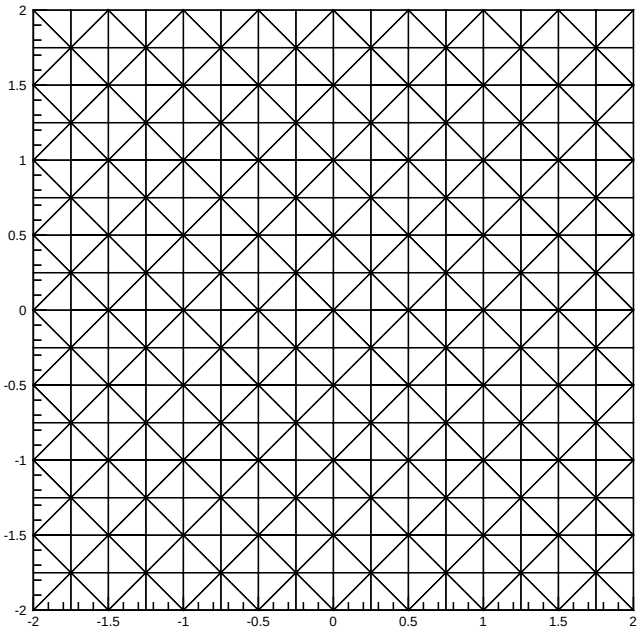

FIG. 4.6. *Triangulation for two-dimensional Burgers equation, $h = \frac{1}{4}$.*

TABLE 4.7
Accuracy for two-dimensional Burgers equation, triangular mesh, $t = 0.5/\pi^2$.

	P^2				P^3			
h	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
1	5.48E-02	—	1.52E-01	—	1.17E-02	—	2.25E-02	—
1/2	1.35E-02	2.02	6.26E-02	1.28	1.35E-03	3.12	4.12E-03	2.45
1/4	2.94E-03	2.20	1.55E-02	2.01	1.45E-04	3.22	4.31E-04	3.26
1/8	6.68E-04	2.14	3.44E-03	2.17	1.71E-05	3.08	7.53E-05	2.52

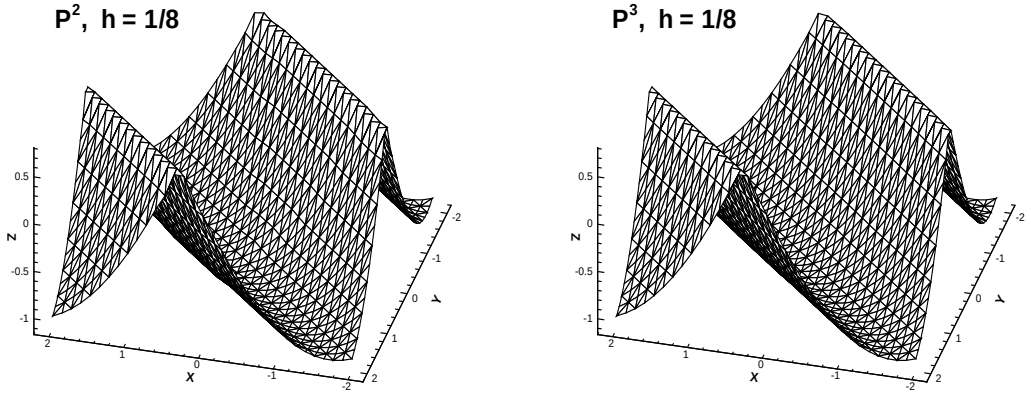

FIG. 4.7. *Two-dimensional Burgers equation, triangular mesh, $t = 1.5/\pi^2$.*

EXAMPLE 4.5. *Two-dimensional equation:*

$$\begin{cases} \varphi_t - \cos(\varphi_x + \varphi_y + 1) = 0, & -2 < x < 2,\ -2 < y < 2, \\ \varphi(x, y, 0) = -\cos\left(\frac{\pi(x+y)}{2}\right) \end{cases} \tag{4.5}$$

with periodic boundary conditions.

For this example we use uniform rectangular meshes. The local Lax–Friedrichs flux (3.8)–(3.9) is used. The solution is smooth at $t = 0.5/\pi^2$. The accuracy of the numerical solution is shown in Table 4.8.

The solution has developed a discontinuous derivative at $t = 1.5/\pi^2$. Results with 40×40 elements are shown in Figure 4.8.

EXAMPLE 4.6. *The level set equation in a domain with a hole:*

$$\begin{cases} \varphi_t + \operatorname{sign}(\varphi_0)\left(\sqrt{\varphi_x^2 + \varphi_y^2} - 1\right) = 0, & \frac{1}{2} < \sqrt{x^2 + y^2} < 1, \\ \varphi(x, y, 0) = \varphi_0(x, y). \end{cases} \tag{4.6}$$

This problem is introduced in [28]. The solution φ to (4.6) has the same zero level set as φ_0, and the steady state solution is the distance function to that zero level curve. We use this problem to test the effects using various integration paths (3.12) when there is a hole in the region. Notice that the exact steady state solution is the distance function to the inner boundary of domain when the boundary condition is adequately prescribed. We compute the time-dependent problem to reach a steady state solution, using the exact solution for the boundary conditions of φ_x and φ_y. Four symmetric elements near the outer boundary are updated by (3.11); all other elements are recovered from (3.12) by the shortest path to the nearest one of above

TABLE 4.8
Accuracy for two dimensions, $H(u,v) = -\cos(u+v+1), t = 0.5/\pi^2$.

	P^1		P^2		P^3	
$N \times N$	L^1 error	order	L^1 error	order	L^1 error	order
10×10	6.47E-02	—	8.31E-03	—	1.35E-02	—
20×20	2.54E-02	1.349	1.93E-03	2.106	1.57E-03	3.104
40×40	1.05E-02	1.274	4.58E-04	2.075	2.39E-04	2.716
80×80	4.74E-03	1.147	1.13E-04	2.019	2.89E-05	3.048
160×160	2.23E-03	1.088	2.83E-05	1.997	4.38E-06	2.722

	P^1		P^2		P^3	
$N \times N$	L^∞ error	order	L^∞ error	order	L^∞ error	order
10×10	1.47E-01	—	1.88E-02	—	2.36E-02	—
20×20	6.75E-02	1.123	7.34E-03	1.357	3.44E-03	2.778
40×40	2.65E-02	1.349	1.83E-03	2.004	4.59E-04	2.906
80×80	1.18E-02	1.167	4.55E-04	2.008	5.78E-05	2.989
160×160	2.23E-03	1.088	1.13E-04	2.010	8.54E-06	2.759

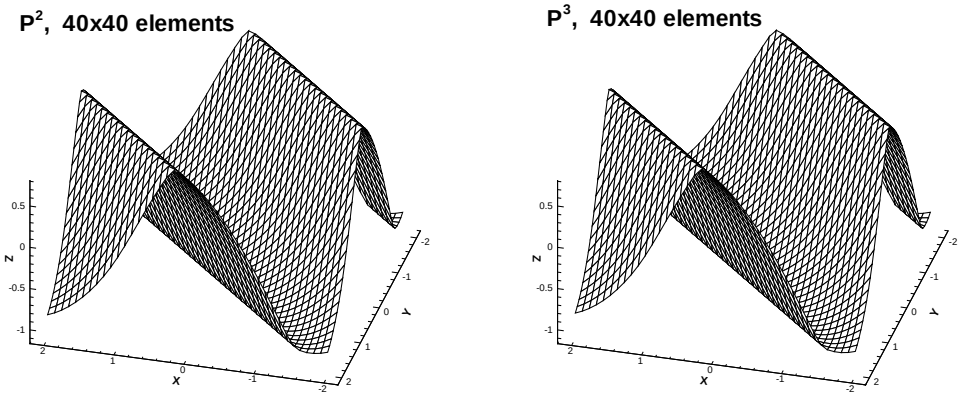

FIG. 4.8. *Two dimensions,* $H(u,v) = -\cos(u+v+1), t = 1.5/\pi^2$.

four elements. The results are shown in Table 4.9. Also shown in Table 4.9 is the error (difference) between the numerical solution φ thus recovered and the value of φ after another integration along a circular path (starting and ending at the same point in (3.12)). We can see that the difference is small with the correct order of accuracy, further indicating that the dependency of the recovered solution φ on the integration path is on the order of the truncation errors even for such problems with holes. Finally, the mesh with 1432 triangles and the solution with 5608 triangles are shown in Figure 4.9.

EXAMPLE 4.7. *Two-dimensional Riemann problem:*

$$\begin{cases} \varphi_t + \sin(\varphi_x + \varphi_y) = 0, & -1 < x < 1, -1 < y < 1, \\ \varphi(x,y,0) = \pi(|y| - |x|). \end{cases} \tag{4.7}$$

For this example, we use a uniform rectangular mesh with 40×40 elements. The local Lax–Friedrichs flux (3.8)–(3.9) is used. As was mentioned in Example 4.3, we have found out that a nonlinear limiting [9, 10] is needed for convergence toward an entropy solution. We show the numerical solution at $t = 1$ in Figure 4.10.

TABLE 4.9
Errors for the level set equation, triangular mesh with P^2.

	Errors for the solution				Errors by integration path			
N	L^1 error	order	L^∞ error	order	L^1 error	order	L^∞ error	order
403	1.02E-03	—	1.32E-03	—	1.61E-04	—	5.71E-04	—
1432	1.23E-04	3.05	2.73E-04	2.27	5.84E-05	1.46	1.68E-04	1.78
5608	1.71E-05	2.85	3.18E-05	3.10	9.32E-06	2.65	4.36E-05	1.95
22238	2.09E-06	3.03	5.01E-06	2.67	1.43E-06	2.70	6.63E-06	2.72

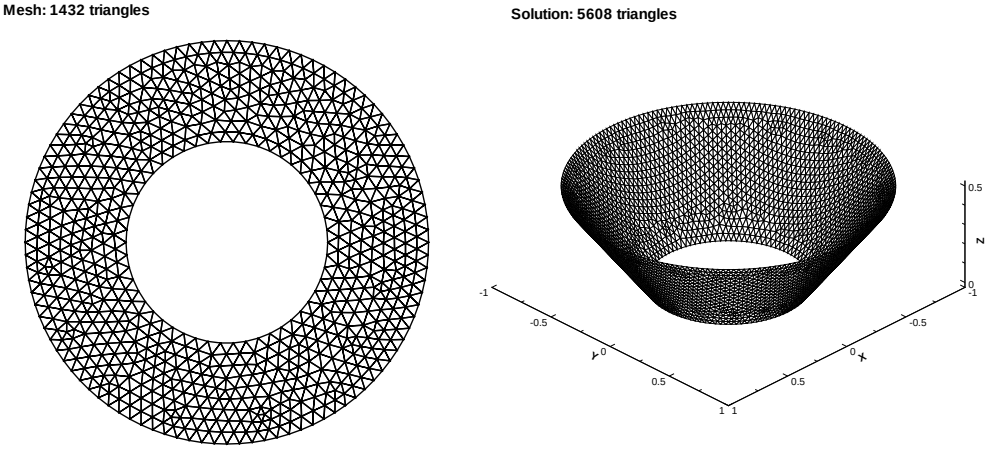

FIG. 4.9. *The level set equation, P^2.*

EXAMPLE 4.8. *The problem of a propagating surface:*

$$\begin{cases} \varphi_t - (1-\varepsilon K)\sqrt{1+\varphi_x^2+\varphi_y^2} = 0, \qquad 0<x<1,\, 0<y<1, \\ \varphi(x,y,0) = 1 - \frac{1}{4}(\cos 2\pi x - 1)\,(\cos 2\pi y - 1), \end{cases} \tag{4.8}$$

where K is the mean curvature defined by

$$K = -\frac{\varphi_{xx}(1+\varphi_y^2) - 2\varphi_{xy}\varphi_x\varphi_y + \varphi_{yy}(1+\varphi_x^2)}{(1+\varphi_x^2+\varphi_y^2)^{\frac{3}{2}}}, \tag{4.9}$$

and ε is a small constant. Periodic boundary condition is used.

This problem was studied in [23] by using the finite difference ENO schemes. We apply the discontinuous Galerkin method, with the second derivative terms handled by the local discontinuous Galerkin techniques presented and analyzed in [11], which amounts to solving the system

$$\begin{cases} u_t - \left(\sqrt{1+u^2+v^2} + \varepsilon\frac{p(1+v^2)-2quv+r(1+u^2)}{1+u^2+v^2}\right)_x = 0, \\ v_t - \left(\sqrt{1+u^2+v^2} + \varepsilon\frac{p(1+v^2)-2quv+r(1+u^2)}{1+u^2+v^2}\right)_y = 0, \\ p - u_x = 0, \\ q - u_y = 0, \\ r - v_y = 0, \end{cases} \tag{4.10}$$

using the discontinuous Galerkin method. The details of the method, especially the choices of fluxes, which are important for stability, can be found in [11].

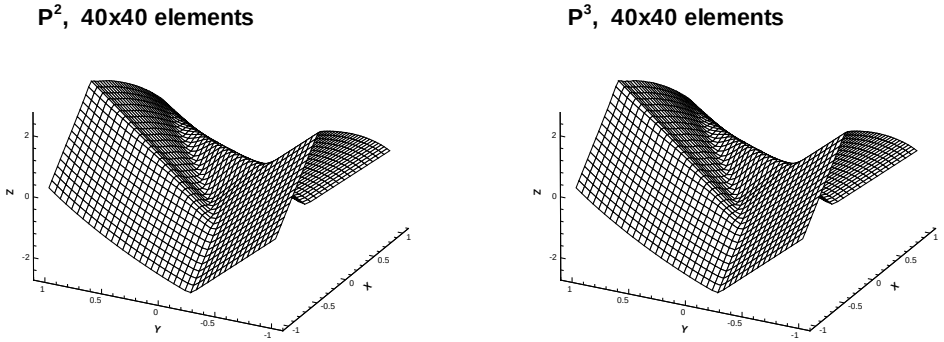

FIG. 4.10. *Two-dimensional Riemann problem,* $H(u, v) = \sin(u + v)$, $t = 1$.

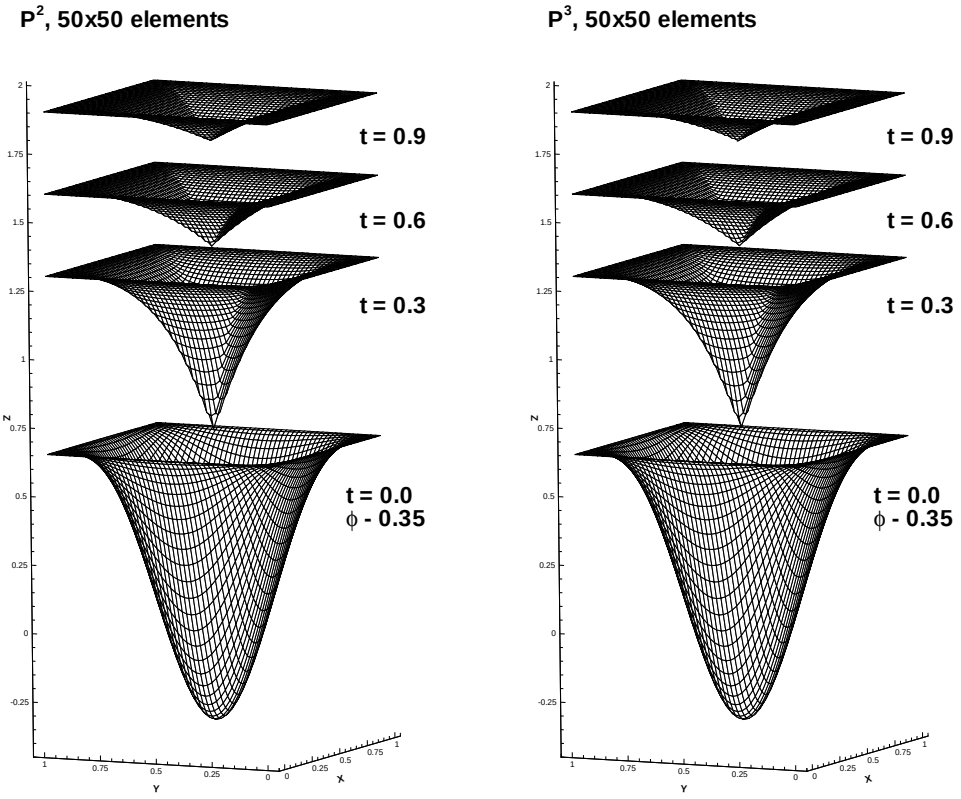

FIG. 4.11. *Propagating surfaces, rectangular mesh,* $\varepsilon = 0$.

We first use a uniform rectangular mesh of 50×50 elements and the local Lax–Friedrichs flux (3.8)–(3.9). The results of $\varepsilon = 0$ (pure convection) and $\varepsilon = 0.1$ are presented in Figure 4.11 and Figure 4.12, respectively. Notice that the surface at $t = 0$ is shifted downward by 0.35 in order to show the detail of the solution at $t = 0.3$.

Next we use a triangulation shown in Figure 4.13. We refine the mesh around the center of domain where the solution develops discontinuous derivatives (for the $\varepsilon = 0$ case). There are 2146 triangles and 1108 nodes in this triangulation. The solutions

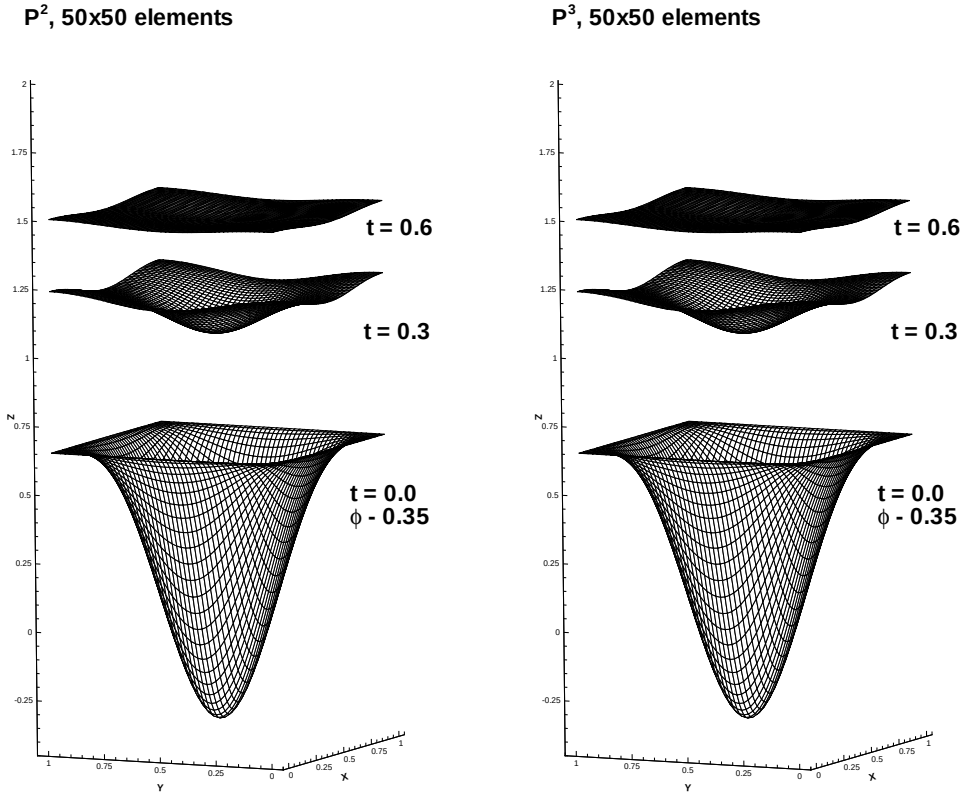

FIG. 4.12. *Propagating surfaces, rectangular mesh,* $\varepsilon = 0.1$.

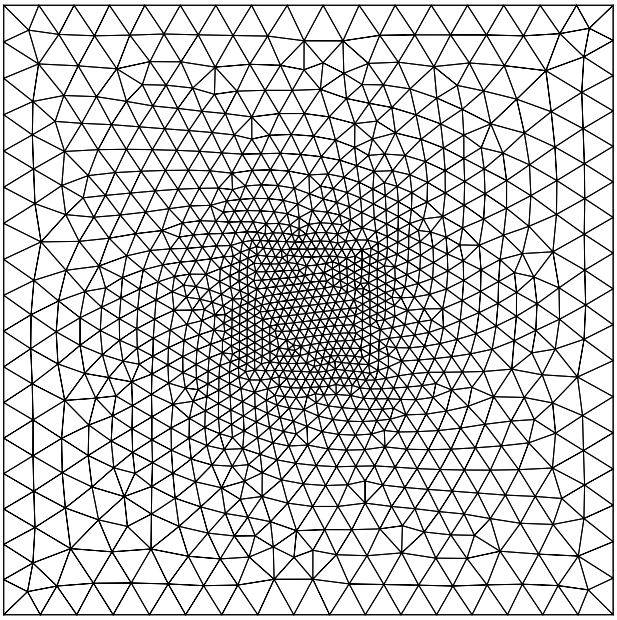

FIG. 4.13. *Triangulation used for the propagating surfaces.*

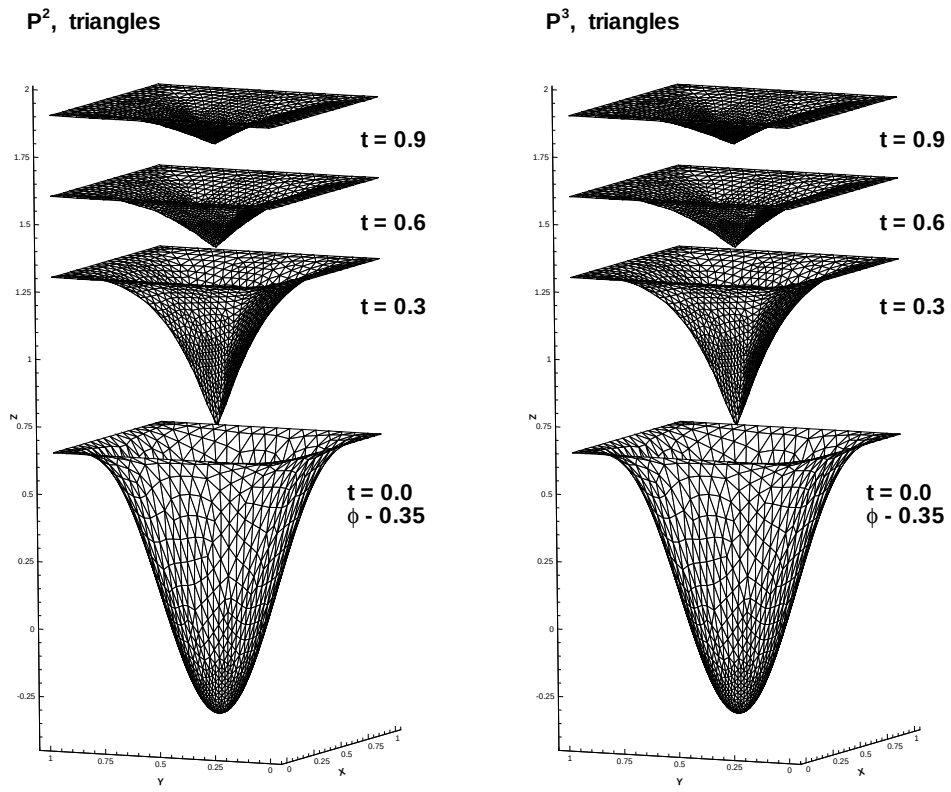

FIG. 4.14. *Propagating surfaces, triangular mesh,* $\varepsilon = 0$.

are displayed in Figure 4.14 and Figure 4.15, respectively, for $\varepsilon = 0$ (pure convection) and $\varepsilon = 0.1$. Notice that we again shift the solution at $t = 0.0$ downward by 0.35 to show the detail of the solutions at later time.

EXAMPLE 4.9. *The problem of a propagating surface on a unit disk. The equation is the same as* (4.8) *in the previous example, but it is solved on a unit disk* $x^2 + y^2 < 1$ *with an initial condition*

$$\varphi(x, y, 0) = \sin\left(\frac{\pi(x^2 + y^2)}{2}\right)$$

and a Neumann-type boundary condition $\nabla\varphi = 0$.

It is difficult to use rectangular meshes for this problem. Instead we use the triangulation shown in Figure 4.16. Notice that we have again refined the mesh near the center of the domain where the solution develops discontinuous derivatives. There are 1792 triangles and 922 nodes in this triangulation. The solutions with $\varepsilon = 0$ are displayed in Figure 4.17. Notice that the solution at $t = 0$ is shifted downward by 0.2 to show the detail of the solution at later time.

The solutions with $\varepsilon = 0.1$ are displayed in Figure 4.18. Notice that the solution at $t = 0$ is again shifted downward by 0.2 to show the detail of the solution at later time.

EXAMPLE 4.10. *A problem from optimal control* [24]:

$$\text{(4.11)} \quad \begin{cases} \varphi_t + (\sin y)\varphi_x + (\sin x + \operatorname{sign}(\varphi_y))\varphi_y - \frac{1}{2}\sin^2 y - (1 - \cos x) = 0, \\ \varphi(x, y, 0) = 0 \end{cases}$$

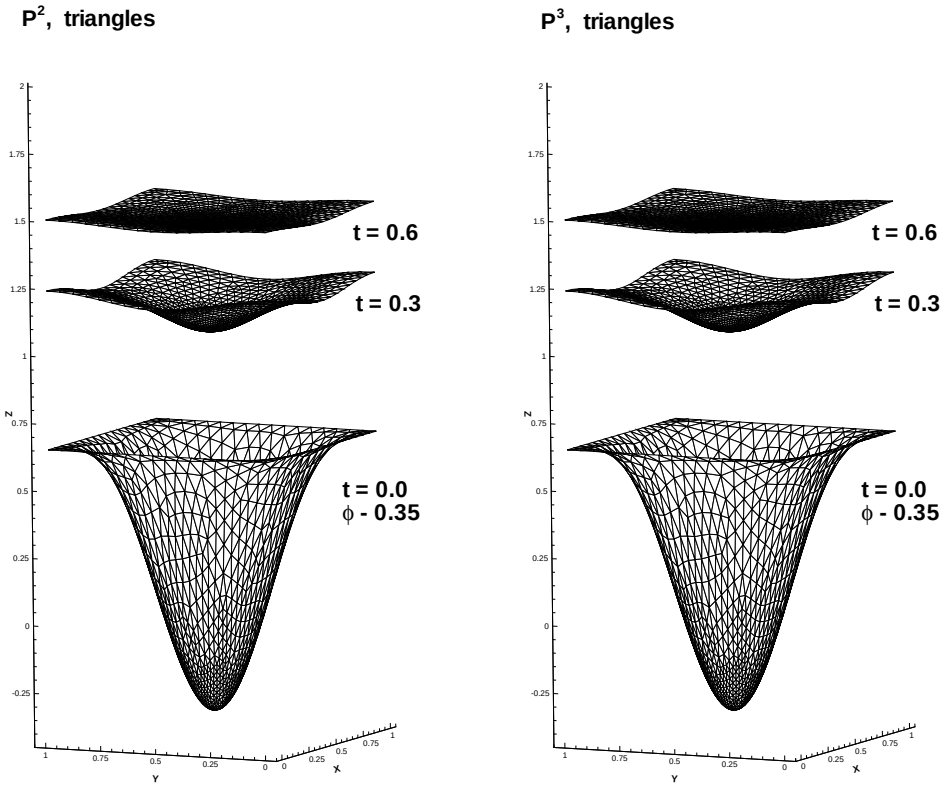

FIG. 4.15. *Propagating surfaces, triangular mesh,* $\varepsilon = 0.1$.

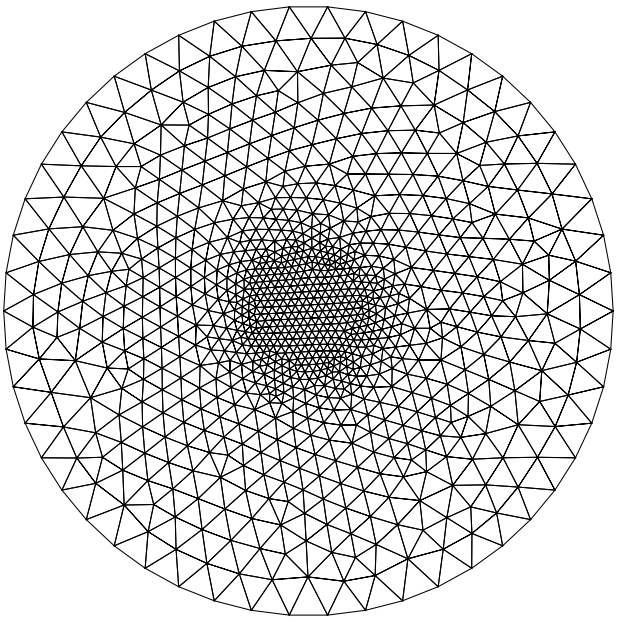

FIG. 4.16. *Triangulation for the propagating surfaces on a disk.*

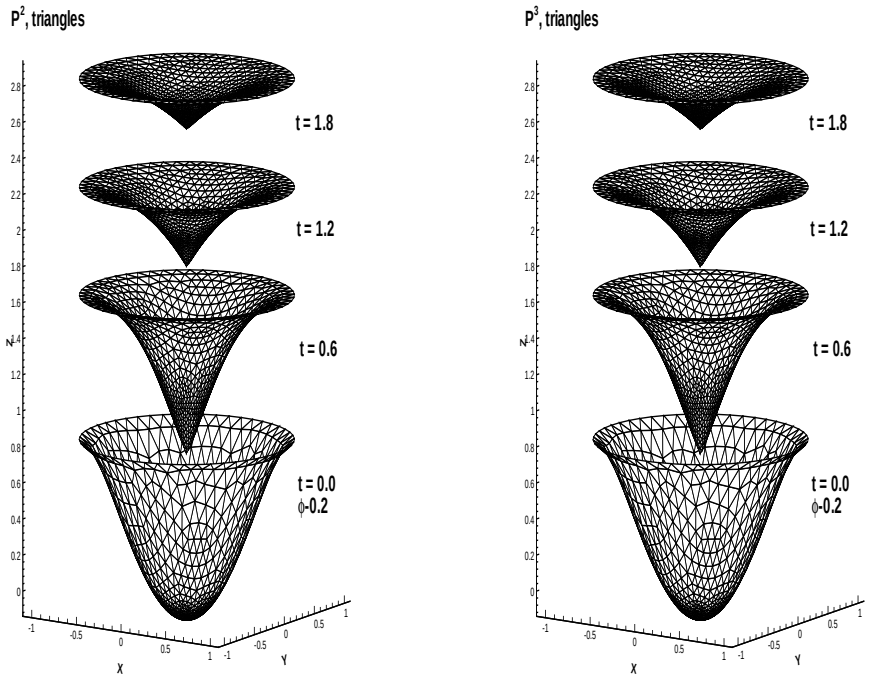

FIG. 4.17. *Propagating surfaces on a disk, triangular mesh,* $\varepsilon = 0$.

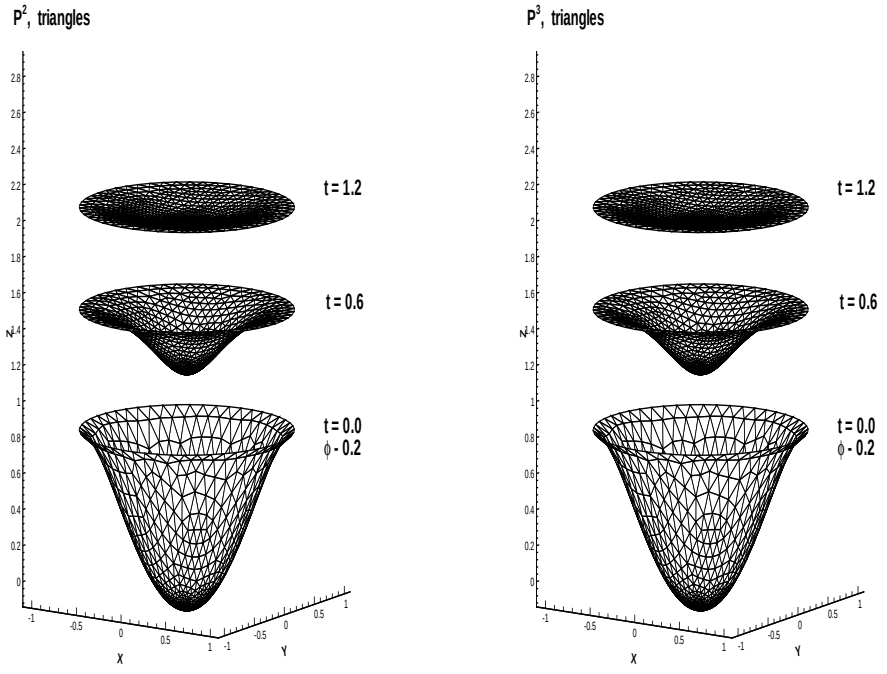

FIG. 4.18. *Propagating surfaces on a disk, triangular mesh,* $\varepsilon = 0.1$.

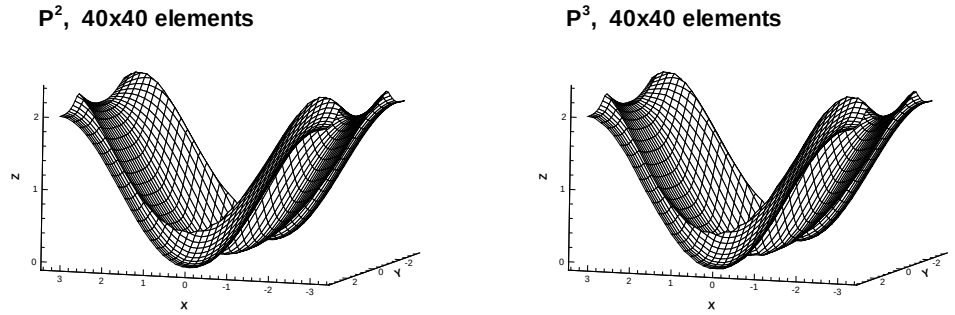

FIG. 4.19. *Control problem,* $t = 1$.

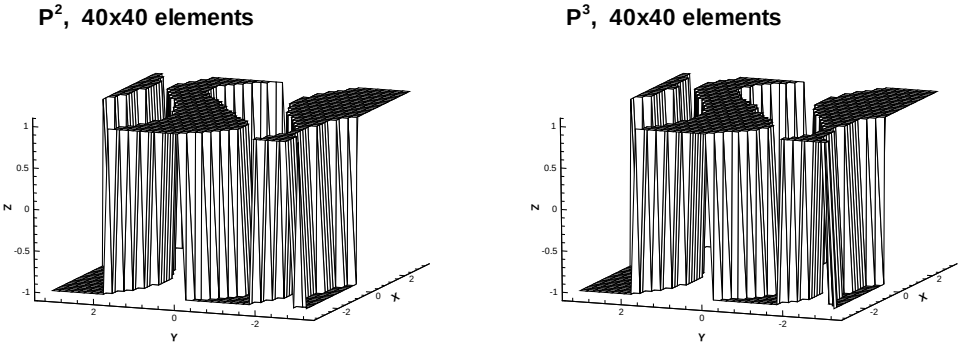

FIG. 4.20. *Control problem,* $t = 1, w = \text{sign}(\varphi_y)$.

in the domain $-\pi < x < \pi$, $-\pi < y < \pi$ *with periodic boundary conditions. We use a uniform rectangular mesh of* 40×40 *elements and the local Lax–Friedrichs flux* (3.8)–(3.9). *The solution at* $t = 1$ *is shown in Figure* 4.19, *while the optimal control* $w = \text{sign}(\varphi_y)$ *is shown in Figure* 4.20.

Notice that our method computes $\nabla\varphi$ as an independent variable. It is very desirable for those problems in which the most interesting features are contained in the first derivatives of φ, as in this optimal control problem.

EXAMPLE 4.11. *A problem from computer vision* [25]:

$$\text{(4.12)} \quad \begin{cases} \varphi_t + I(x,y)\sqrt{1+\varphi_x^2+\varphi_y^2} - 1 = 0, & -1 < x < 1,\ -1 < y < 1, \\ \varphi(x,y,0) = 0 \end{cases}$$

with $\varphi = 0$ *as the boundary condition.*

The steady state solution of this problem is the shape lighted by a source located at infinity with vertical direction. The solution is not unique if there are points at which $I(x,y) = 1$. Conditions must be prescribed at those points where $I(x,y) = 1$. Since our method is a finite element method, we need to prescribe suitable conditions at the correspondent elements. We take

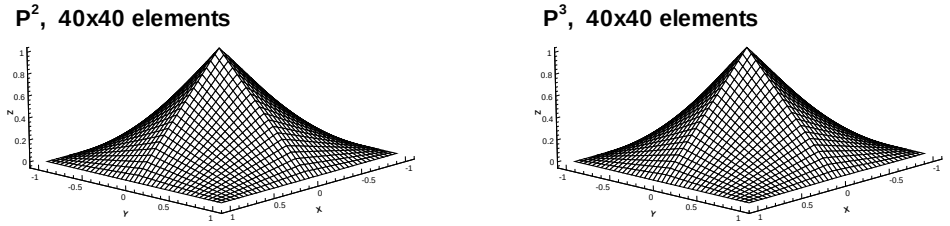

FIG. 4.21. *Computer vision problem,* $\varphi(x, y, \infty) = (1 - |x|)(1 - |y|)$.

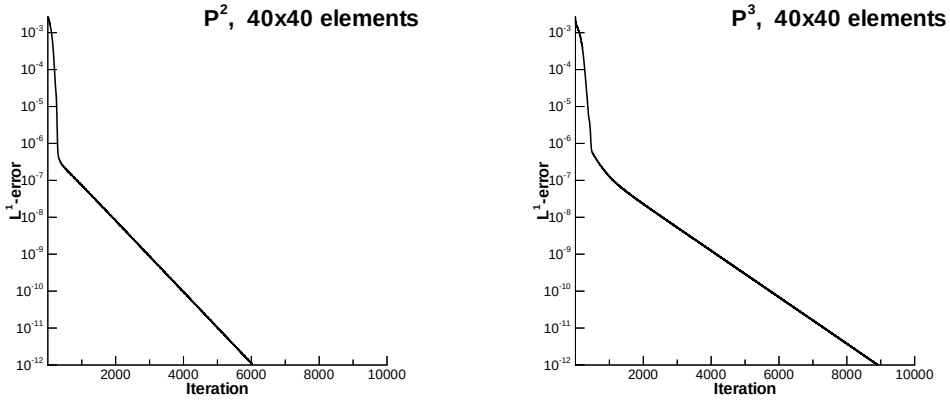

FIG. 4.22. *Computer vision problem, history of iterations.*

$$I(x, y) = 1/\sqrt{1 + (1 - |x|)^2 + (1 - |y|)^2}. \tag{4.13}$$

The exact steady solution is $\varphi(x, y, \infty) = (1 - |x|)(1 - |y|)$. We use a uniform rectangular mesh of 40×40 elements and the local Lax–Friedrichs flux (3.8)–(3.9). We impose the exact boundary conditions for $u = \varphi_x, v = \varphi_y$ from the above exact steady solution and take the exact value at one point (the lower left corner) to recover φ. The results for P^2 and P^3 are presented in Figure 4.21, while Figure 4.22 contains the history of iterations to the steady state.

Next we take

$$I(x, y) = 1/\sqrt{1 + 4y^2(1 - x^2)^2 + 4x^2(1 - y^2)^2}. \tag{4.14}$$

The exact steady solution is $\varphi(x, y, \infty) = (1 - x^2)(1 - y^2)$. We again use a uniform rectangular mesh of 40×40 elements, the local Lax–Friedrichs flux (3.8)–(3.9), impose the exact boundary conditions for $u = \varphi_x, v = \varphi_y$ from the above exact steady solution, and take the exact value at one point (the lower left corner) to recover φ. A continuation method is used, with the steady solution using

$$I_\varepsilon(x, y) = 1/\sqrt{1 + 4y^2(1 - x^2)^2 + 4x^2(1 - y^2)^2 + \varepsilon} \tag{4.15}$$

for bigger ε as the initial condition for smaller ε. The sequence of ε used are $\varepsilon = 0.2, 0.05, 0$. The results for P^2 and P^3 are presented in Figure 4.23.

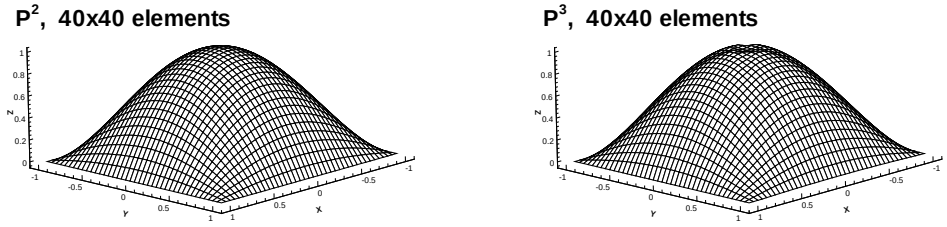

FIG. 4.23. *Computer vision problem,* $\varphi(x, y, \infty) = (1 - x^2)(1 - y^2)$.

5. Concluding remarks. We have developed and tested a class of discontinuous Galerkin methods for solving nonlinear HJ equations. These methods have the advantage of easy handling of complicated geometry and local mesh refinements, as well as efficient parallel implementations. Numerical examples in one- and two-space dimensions are shown to illustrate the capability of the methods.

REFERENCES

[1] R. ABGRALL, *On essentially non-oscillatory schemes on unstructured meshes: Analysis and implementation*, J. Comput. Phys., 114 (1994), pp. 45–58.
[2] R. ABGRALL, *Numerical discretization of the first-order Hamilton-Jacobi equation on triangular meshes*, Comm. Pure Appl. Math., 49 (1996), pp. 1339–1373.
[3] T. BARTH AND J. SETHIAN, *Numerical schemes for the Hamilton-Jacobi and level set equations on triangulated domains*, J. Comput. Phys., 145 (1998), pp. 1–40.
[4] R. BISWAS, K.D. DEVINE, AND J. FLAHERTY, *Parallel, adaptive finite element methods for conservation laws*, Appl. Numer. Math., 14 (1994), pp. 255–283.
[5] B. COCKBURN, *An introduction to the discontinuous Galerkin method for convection-dominated problems*, in Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, B. Cockburn, C. Johnson, C-W. Shu, E. Tadmor, and A. Quarteroni, eds., Lecture Notes in Math. 1697, CIME subseries, Springer-Verlag, New York, 1998, pp. 151–268.
[6] B. COCKBURN AND C.-W. SHU, *The Runge-Kutta local projection* P^1*-discontinuous Galerkin method for scalar conservation laws*, RAIRO Math. Modél. Anal. Numér., 25 (1991), pp. 337–361.
[7] B. COCKBURN AND C.-W. SHU, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws* II: *General framework*, Math. Comp., 52 (1989), pp. 411–435.
[8] 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*, J. Comput. Phys., 84 (1989), pp. 90–113.
[9] B. COCKBURN, S. HOU, AND C.-W. SHU, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws* IV: *The multidimensional case*, Math. Comp., 54 (1990), pp. 545–581.
[10] B. COCKBURN AND C.-W. SHU, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws* V: *Multidimensional systems*, J. Comput. Phys., 141 (1998), pp. 199–224.
[11] B. COCKBURN AND C.-W. SHU, *The local discontinuous Galerkin method for time-dependent convection-diffusion systems*, SIAM J. Numer. Anal., 35 (1998), pp. 2440–2463.
[12] M. CRANDALL AND P.L. LIONS, *Viscosity solutions of Hamilton-Jacobi equations*, Trans. Amer. Math. Soc., 277 (1983), pp. 1–42.
[13] M. CRANDALL AND P.L. LIONS, *Two approximations of solutions of Hamilton-Jacobi equations*, Math. Comp., 43 (1984), pp. 1–19.
[14] S. GOTTLIEB AND C.-W. SHU, *Total variation diminishing Runge-Kutta schemes*, Math. Comp., 67 (1998), pp. 73–85.
[15] A. HARTEN, B. ENGQUIST, S. OSHER, AND S. CHAKRAVARTHY, *Uniformly high order essentially non-oscillatory schemes*, III, J. Comput. Phys., 71 (1987), pp. 231–303.
[16] G. JIANG AND D. PENG, *Weighted ENO schemes for Hamilton-Jacobi equations*, SIAM J. Sci. Comput., to appear.

[17] G. Jiang and C.-W. Shu, *On cell entropy inequality for discontinuous Galerkin methods*, Math. Comp., 62 (1994), pp. 531–538.
[18] G. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, J. Comput. Phys., 126 (1996), pp. 202–228.
[19] S. Jin and Z. Xin, *Numerical passage from systems of conservation laws to Hamilton-Jacobi equations, and relaxation schemes*, SIAM J. Numer. Anal., 35 (1998), pp. 2385–2404.
[20] F. Lafon and S. Osher, *High order two dimensional nonoscillatory methods for solving Hamilton-Jacobi scalar equations*, J. Comput. Phys., 123 (1996), pp. 235–253.
[21] X.-D. Liu, S. Osher, and T. Chan, *Weighted essentially nonoscillatory schemes*, J. Comput. Phys., 115 (1994), pp. 200–212.
[22] I. Lomtev and G. Karniadakis, *A discontinuous Galerkin method for the Navier-Stokes equations*, Internat. J. Numer. Methods Fluids, 29 (1999), pp. 587–603.
[23] S. Osher and J. Sethian, *Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.
[24] S. Osher and C.-W. Shu, *High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations*, SIAM J. Numer. Anal., 28 (1991), pp. 907–922.
[25] E. Rouy and A. Tourin, *A viscosity solutions approach to shape-from-shading*, SIAM J. Numer. Anal., 29 (1992), pp. 867–884.
[26] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes*, J. Comput. Phys., 77 (1988), pp. 439–471.
[27] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes* II, J. Comput. Phys., 83 (1989), pp. 32–78.
[28] M. Sussman, P. Smereka, and S. Osher, *A level set approach for computing solution to incompressible two-phase flow*, J. Comput. Phys., 114 (1994), pp. 146–159.

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Applied Mathematics Letters 18 (2005) 1204–1209
Applied Mathematics Letters
www.elsevier.com/locate/aml

Reinterpretation and simplified implementation of a discontinuous Galerkin method for Hamilton–Jacobi equations

Fengyan Li[1], Chi-Wang Shu*

Division of Applied Mathematics, Brown University, Providence, RI 02912, United States

Received 7 July 2004; accepted 24 October 2004

Abstract

In this note, we reinterpret a discontinuous Galerkin method originally developed by Hu and Shu [C. Hu, C.-W. Shu, A discontinuous Galerkin finite element method for Hamilton–Jacobi equations, SIAM Journal on Scientific Computing 21 (1999) 666–690] (see also [O. Lepsky, C. Hu, C.-W. Shu, Analysis of the discontinuous Galerkin method for Hamilton–Jacobi equations, Applied Numerical Mathematics 33 (2000) 423–434]) for solving Hamilton–Jacobi equations. With this reinterpretation, numerical solutions will automatically satisfy the curl-free property of the exact solutions inside each element. This new reinterpretation allows a method of lines formulation, which renders a more natural framework for stability analysis. Moreover, this reinterpretation renders a significantly simplified implementation with reduced cost, as only a smaller subspace of the original solution space in [C. Hu, C.-W. Shu, A discontinuous Galerkin finite element method for Hamilton–Jacobi equations, SIAM Journal on Scientific Computing 21 (1999) 666–690; O. Lepsky, C. Hu, C.-W. Shu, Analysis of the discontinuous Galerkin method for Hamilton–Jacobi equations, Applied Numerical Mathematics 33 (2000) 423–434] is used and the least squares procedure used in [C. Hu, C.-W. Shu, A discontinuous Galerkin finite element method for Hamilton–Jacobi equations, SIAM Journal on Scientific Computing 21 (1999) 666–690; O. Lepsky, C. Hu, C.-W. Shu, Analysis of the discontinuous Galerkin method for Hamilton–Jacobi equations, Applied Numerical Mathematics 33 (2000) 423–434] is completely avoided.

MSC: 65M60; 70H20

Keywords: Hamilton–Jacobi equations; Discontinuous Galerkin method

* Corresponding author. Tel.: +1 401 863 2357; fax: +1 401 863 1355.
E-mail addresses: fengyan-l@dam.brown.edu, fli@math.sc.edu (F. Li), shu@dam.brown.edu (C.-W. Shu).
[1] Current address: Department of Mathematics, University of South Carolina, Columbia, SC 29308, United States.

0893-9659/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.aml.2004.10.009

1. Introduction

Hamilton–Jacobi equations

$$\phi_t + H(\phi_{x_1}, \ldots, \phi_{x_n}) = 0, \text{ in } \Omega \times [0, T] \subset \mathbb{R}^n \times \mathbb{R}, \qquad \phi(x, 0) = \phi^0(x) \tag{1.1}$$

arise in many areas of application, including the calculus of variations, control theory, image processing, etc. See, e.g., [14]. Viscosity solutions of Hamilton–Jacobi equations are studied [6,7,14] to single out the practically relevant solutions. These viscosity solutions are Lipschitz continuous, and may have discontinuous derivatives, regardless of the smoothness of the initial conditions.

Many numerical schemes have been developed for solving Hamilton–Jacobi equations, for example the widely used finite difference schemes [7,15,16,10] which require structured meshes; and the finite volume schemes, which are based on arbitrary triangulation but rely on quite complicated reconstruction procedures [1,18] in order to obtain higher order approximations.

Discontinuous Galerkin methods have become very popular in recent years for solving hyperbolic conservation laws because of their distinctive features, among which are the easy design of the methods with any order of accuracy and their minimal requirement on the mesh structures [5]. Adapted from these methods for conservation laws, a discontinuous Galerkin method for solving Hamilton–Jacobi equations (1.1) was developed by Hu and Shu in [9] on the basis of the equivalence between Hamilton–Jacobi equations and hyperbolic conservation laws [11,14]. See also [12]. In [9,12], the Hamilton–Jacobi equations (1.1) were first rewritten as a system of conservation laws

$$(w_i)_t + (H(\mathbf{w}))_{x_i} = 0, \text{ in } \Omega \times [0, T], \qquad \mathbf{w}(x, 0) = \nabla\phi^0(x), \tag{1.2}$$

where $\mathbf{w} = \nabla\phi$. With piecewise polynomial space as the solution space, the standard discontinuous Galerkin formulation could be obtained for (1.2) [2,4]. Notice that $w_i, i = 1, \ldots, n$, are not independent due to the restriction $\mathbf{w} = \nabla\phi$. A least squares procedure was then applied in each time step (or each time stage depending on the particular time discretization used) to enforce this restriction.

In recent work by Cockburn et al. [3] and by Li and Shu [13], the locally divergence-free discontinuous Galerkin methods were developed for partial differential equations with divergence-free solutions. Unlike in traditional ways of solving such equations, the piecewise divergence-free polynomial space, which is a subspace of the standard piecewise polynomial space, is used. With minimal change in the scheme formulation (only the solution and test space is changed to a smaller space), the computational cost is reduced, while the stability and the order of accuracy of the scheme are maintained. For specific applications such as the Maxwell equations [3] and the ideal magnetohydrodynamics (MHD) equations [13], this new method even improves on the traditional discontinuous Galerkin method in terms of stability and/or accuracy while saving computational costs. The idea of this approach could be applied to more general situations, by using piecewise solution space in which functions satisfy certain properties of the exact solutions (divergence-free, or curl-free, ...). The general approximation theory can guarantee no loss of accuracy when such a smaller solution space is used. This observation leads to a reinterpretation and simplified implementation of the discontinuous Galerkin method for Hamilton–Jacobi equations developed in [9,12].

This note is organized as follows. In Section 2, we reinterpret the discontinuous Galerkin method for Hamilton–Jacobi equations developed in [9,12] and describe the advantages of the new formulation. In Section 3, we provide a numerical example to demonstrate that the new formulation, while saving computational cost, gives identical results to those in [9,12], as expected.

2. A reinterpretation

Starting with a regular triangulation $\mathcal{T}_h = \{K\}$ of Ω (edges denoted by e), the general discontinuous Galerkin formulation of (1.2) is: find $\mathbf{w} = (w_1, \ldots, w_n) \in \mathbf{V}^k$ such that

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_K w_i v_i \mathrm{d}x = \int_K H(\mathbf{w})(v_i)_{x_i}\mathrm{d}x - \sum_{e\in\partial K}\int_e \hat{H}_{i,e,K} v_i \mathrm{d}s, \qquad \forall K, i = 1, \ldots, n, \tag{2.1}$$

holds for all $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbf{V}^k$, where $\mathbf{V}^k$ is the solution space which will be specified later, and $\hat{H}_{i,e,K}$ is the numerical flux which is an approximating Riemann solver (see [16] for more details). The strong stability preserving Runge–Kutta time discretization [17,8] could be used in the time direction. Notice that (2.1) is the formulation for the derivatives of ϕ in (1.1). To recover the missing constant in ϕ (e.g. the cell average of ϕ in each element), there are two different strategies developed in [9,12] which can be used. We refer the readers to [9,12] for the details.

Before finalizing the scheme, we introduce the following spaces:

$$\mathbf{V}_1^k = \{(v_1, \ldots, v_n) : v_i|_K \in P^k(K), i = 1, \ldots, n, \forall K \in \mathcal{T}_h\}, \tag{2.2}$$

$$\mathbf{V}_2^k = \{(v_1, \ldots, v_n) : \mathbf{v}|_K = \nabla\phi, \phi \in P^{k+1}(K), \forall K \in \mathcal{T}_h\}, \tag{2.3}$$

where $P^k(K)$ denotes the space of polynomials in K of degree at most k. It is easy to see that $\mathbf{V}_2^k \subset \mathbf{V}_1^k$. Two formulations are obtained if $\mathbf{V}^k$ in (2.1) is specified as follows:

- *Formulation I*: $\mathbf{V}^k = \mathbf{V}_1^k$. A single polynomial $\phi \in P^{k+1}(K)$, up to a constant, is further recovered from $\mathbf{w}$ in each element by the following least squares procedure:

$$\left\| \sum_i (\phi_{x_i} - w_i)^2 \right\|_{L^1(K)} = \min_{\psi \in P^{k+1}(K)} \left\| \sum_i (\psi_{x_i} - w_i)^2 \right\|_{L^1(K)} \tag{2.4}$$

  after each time stage. This is the method proposed by Hu and Shu in [9].
- *Formulation II*: $\mathbf{V}^k = \mathbf{V}_2^k$.

Proposition. *Formulations I and II give identical results.*

Proof. There are two key components in the proof which match each other in a very good way. First, (2.1) is a Galerkin formulation. Second, the least squares procedure is essentially an L^2 projection.

Notice that (2.1) could be written compactly as

$$\left(\frac{\partial \mathbf{w}}{\partial t}, \mathbf{v}\right) = \langle \bar{L}(\mathbf{w}), \mathbf{v}\rangle, \tag{2.5}$$

where

$$(\mathbf{u}, \mathbf{v}) = \sum_{K\in\mathcal{T}_h, i}\int_K u_i v_i \mathrm{d}x,$$

$$\langle \bar{L}(\mathbf{w}), \mathbf{v}\rangle = \sum_{K\in\mathcal{T}_h}\left\{\int_K H(\mathbf{w})\nabla\cdot\mathbf{v}\mathrm{d}x - \sum_{e\in\partial K, i}\int_e \hat{H}_{i,e,K} v_i \mathrm{d}s\right\}.$$

First we consider the forward Euler time discretization for (2.5). Starting with $\mathbf{w}^m \in \mathbf{V}_2^k$, we want to find $\mathbf{w}^{m+1}$ so that

$$(\mathbf{w}^{m+1}, \mathbf{v}) = \Delta t \langle \bar{L}(\mathbf{w}^m), \mathbf{v} \rangle + (\mathbf{w}^m, \mathbf{v}) \equiv \langle L(\mathbf{w}^m), \mathbf{v} \rangle, \tag{2.6}$$

holds for some test function $\mathbf{v}$.

Formulation II then becomes: find $\mathbf{w}_2^{m+1} \in \mathbf{V}_2^k$ such that

$$(\mathbf{w}_2^{m+1}, \mathbf{v}) = \langle L(\mathbf{w}^m), \mathbf{v} \rangle, \qquad \forall \mathbf{v} \in \mathbf{V}_2^k. \tag{2.7}$$

For Formulation I, we first obtain $\bar{\mathbf{w}}_1^{m+1} \in \mathbf{V}_1^k$, which satisfies

$$(\bar{\mathbf{w}}_1^{m+1}, \mathbf{v}) = \langle L(\mathbf{w}^m), \mathbf{v} \rangle, \qquad \forall \mathbf{v} \in \mathbf{V}_1^k, \tag{2.8}$$

and then the least squares procedure provides us with $\mathbf{w}_1^{m+1} \in \mathbf{V}_2^k$ via

$$(\mathbf{w}_1^{m+1}, \mathbf{v}) = (\bar{\mathbf{w}}_1^{m+1}, \mathbf{v}) = \langle L(\mathbf{w}^m), \mathbf{v} \rangle = (\mathbf{w}_2^{m+1}, \mathbf{v}) \qquad \forall \mathbf{v} \in \mathbf{V}_2^k \tag{2.9}$$

where the second equality holds because of (2.8) and $\mathbf{V}_2^k \subset \mathbf{V}_1^k$, and the last one is from (2.7). Therefore $\mathbf{w}_1^{m+1} = \mathbf{w}_2^{m+1}$ and hence Formulations I and II give the same numerical results. The conclusion can go directly to general explicit Runge–Kutta time discretizations, as these Runge–Kutta methods can be written as the linear combination of forward Euler methods (with a proper time step) and the L^2 projection is a linear operator. □

Remarks. Although the two formulations provide identical results by the previous proposition, Formulation II does have several advantages over Formulation I:

1. Formulation II allows the method of lines version of the scheme, while Formulation I does not have a method of lines version due to the least squares procedure which is applied after each time step or stage. The method of lines version allows more natural and direct analysis for stability and accuracy of discontinuous Galerkin methods, e.g. the results in [12].
2. The implementation of the algorithm is significantly simplified by using Formulation II since a smaller solution space is used and the least squares procedure is completely avoided. If we characterize the computational cost of (2.1) per time step per element simply by the dimension of $\mathbf{V}^k|_K$, we can get

$$n_1 = \dim(\mathbf{V}_1^k|_K) = n \sum_{r=0}^{k} C_{r+n-1}^{n-1}, \qquad n_2 = \dim(\mathbf{V}_2^k|_K) = \sum_{r=1}^{k+1} C_{r+n-1}^{n-1}.$$

For example, for the two-dimensional case $n = 2$, $n_1 = (k+2)(k+1)$, $n_2 = \frac{(k+4)(k+1)}{2}$; hence $\frac{n_2}{n_1} \to \frac{1}{2}$ as $k \to \infty$, i.e. the cost is reduced to about half for higher order schemes. For the three-dimensional case $n = 3$, $n_1 = \frac{k^3+6k^2+11k+6}{2}$, $n_2 = \frac{(k+1)(k^2+8k+18)}{6}$; hence $\frac{n_2}{n_1} \to \frac{1}{3}$ as $k \to \infty$, i.e. the cost is reduced to about one third for higher order schemes.

3. A numerical example

We only present one numerical example here, the two-dimensional Burgers equation, to show the identicality of the numerical results from Formulations I and II. Consider

$$\phi_t + \frac{(\phi_x + \phi_y + 1)^2}{2} = 0$$

on $[-2, 2] \times [-2, 2]$ with $\phi(x, y, 0) = -\cos(\frac{\pi(x+y)}{2})$ and periodic boundary conditions. In Table 3.1 we show errors and orders of accuracy at $t = 0.5/\pi^2$, when the exact solution is still smooth. One

Table 3.1
L^∞ errors and orders of accuracy for the 2D Burgers equations, non-uniform rectangular meshes, $t = \frac{0.5}{\pi^2}$

$N \times N$	Error for Formulation I	Error for Formulation II	Order
	$k = 2$		
10×10	4.677844483874727E−2	4.677844483874738E−2	
20×20	1.227397975233135E−2	1.227397975233091E−2	1.93
40×40	3.539268278996754E−3	3.539268278997088E−3	1.80
80×80	1.153353110058708E−3	1.153353110058597E−3	1.62
160×160	2.722372199789325E−4	2.722372199783774E−4	2.08
	$k = 3$		
10×10	1.001317293031612E−2	1.001317293031612E−2	
20×20	1.394123796094382E−3	1.394123796094604E−3	2.85
40×40	2.294961878530621E−4	2.294961878529511E−4	2.60
80×80	5.113292444847151E−5	5.113292444935968E−5	2.16
160×160	7.157782964561932E−6	7.157782965006021E−6	2.84

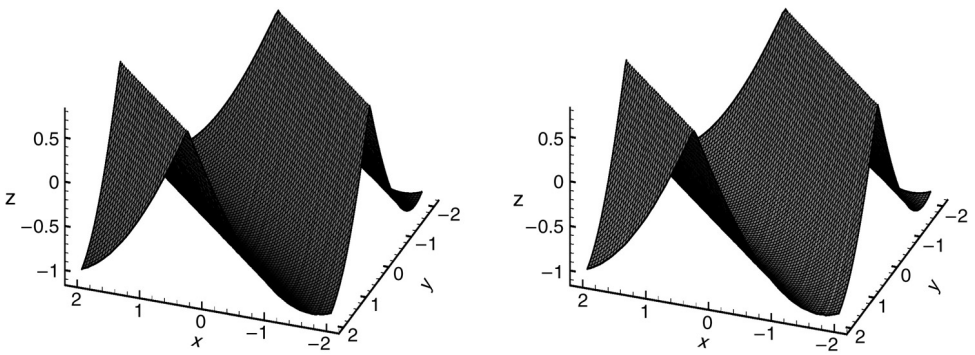

Fig. 3.1. 2D Burgers equations, non-uniform rectangular meshes, $t = \frac{1.5}{\pi^2}$. Left: $k = 2$; right: $k = 3$.

can see that the differences in error between solutions from the two formulations are just at machine error level. We also plot the numerical solutions for ϕ at $t = 1.5/\pi^2$ in Fig. 3.1 when the solution has discontinuous derivatives, obtained by Formulation II. The computation is performed on non-uniform rectangular meshes, which are obtained by a random perturbation of 10% from uniform meshes. The local Lax–Friedrichs numerical flux [9] and the Runge–Kutta time discretization of compatible accuracy order are used. The constant in ϕ is recovered first along $y = -2$ from $(-2, -2)$, then along $x = \textit{constant}$ from bottom to top. See [9] for details.

Acknowledgements

This research was supported by ARO grants DAAD19-00-1-0405 and W911NF-04-1-0291, NSF grant DMS-0207451 and AFOSR grant F49620-02-1-0113.

References

[1] R. Abgrall, On essentially non-oscillatory schemes on unstructured meshes: Analysis and implementation, Journal of Computational Physics 114 (1994) 45–58.

[2] B. Cockburn, S. Hou, 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.
[3] B. Cockburn, F. Li, C.-W. Shu, Locally divergence-free discontinuous Galerkin methods for the Maxwell equations, Journal of Computational Physics 194 (2004) 588–610.
[4] B. Cockburn, C.-W. Shu, The Runge–Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems, Journal of Computational Physics 141 (1998) 199–224.
[5] B. Cockburn, C.-W. Shu, Runge–Kutta discontinuous Galerkin methods for convection-dominated problems, Journal of Scientific Computing 16 (2001) 173–261.
[6] M. Crandall, P.L. Lions, Viscosity solutions of Hamilton–Jacobi equations, Transactions of the American Mathematical Society 277 (1983) 1–42.
[7] M. Crandall, P.L. Lions, Two approximations of solutions of Hamilton–Jacobi equations, Mathematics of Computation 43 (1984) 1–19.
[8] S. Gottlieb, C.-W. Shu, E. Tadmor, Strong stability-preserving high-order time discretization methods, SIAM Review 43 (2001) 89–112.
[9] C. Hu, C.-W. Shu, A discontinuous Galerkin finite element method for Hamilton–Jacobi equations, SIAM Journal on Scientific Computing 21 (1999) 666–690.
[10] G. Jiang, D. Peng, Weighted ENO schemes for Hamilton–Jacobi equations, SIAM Journal on Scientific Computing 21 (1999) 2126–2143.
[11] S. Jin, 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.
[12] O. Lepsky, C. Hu, C.-W. Shu, Analysis of the discontinuous Galerkin method for Hamilton–Jacobi equations, Applied Numerical Mathematics 33 (2000) 423–434.
[13] F. Li, C.-W. Shu, Locally divergence-free discontinuous Galerkin methods for MHD equations, Journal of Scientific Computing (in press).
[14] P.L. Lions, Generalized Solutions of Hamilton–Jacobi Equations, Pitman, Boston, 1982.
[15] S. Osher, J. Sethian, Fronts propagating with curvature dependent speed: Algorithms based on Hamilton–Jacobi formulations, Journal of Computational Physics 79 (1988) 12–49.
[16] S. Osher, C.-W. Shu, High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations, SIAM Journal on Numerical Analysis 28 (1991) 907–922.
[17] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, Journal of Computational Physics 77 (1988) 439–471.
[18] Y.-T. Zhang, C.-W. Shu, High order WENO schemes for Hamilton–Jacobi equations on triangular meshes, SIAM Journal on Scientific Computing 24 (2003) 1005–1030.

Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 223 (2007) 398–415
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A discontinuous Galerkin finite element method for directly solving the Hamilton–Jacobi equations

Yingda Cheng, Chi-Wang Shu *

Division of Applied Mathematics, Brown University, Box F, Providence, RI 02912, United States

Received 10 May 2006; received in revised form 14 September 2006; accepted 19 September 2006
Available online 2 November 2006

Abstract

In this paper, we propose a new discontinuous Galerkin finite element method to solve the Hamilton–Jacobi equations. Unlike the discontinuous Galerkin method of [C. Hu, C.-W. Shu, A discontinuous Galerkin finite element method for Hamilton–Jacobi equations, SIAM Journal on Scientific Computing 21 (1999) 666–690.] which applies the discontinuous Galerkin framework on the conservation law system satisfied by the derivatives of the solution, the method in this paper applies directly to the solution of the Hamilton–Jacobi equations. For the linear case, this method is equivalent to the traditional discontinuous Galerkin method for conservation laws with source terms. Thus, stability and error estimates are straightforward. For the nonlinear convex Hamiltonians, numerical experiments demonstrate that the method is stable and provides the optimal $(k+1)$th order of accuracy for smooth solutions when using piecewise kth degree polynomials. Singularities in derivatives can also be resolved sharply if the entropy condition is not violated. Special treatment is needed for the entropy violating cases. Both one and two-dimensional numerical results are provided to demonstrate the good qualities of the scheme.

Keywords: Hamilton–Jacobi equations; Discontinuous Galerkin; High order accuracy; Convex Hamiltonian

1. Introduction

In this paper, we consider the numerical solution of the Hamilton–Jacobi equation

$$\varphi_t + H(\varphi_{x_1}, \ldots, \varphi_{x_d}, x_1, \ldots, x_d) = 0, \qquad \varphi(x,0) = \varphi^0(x) \tag{1.1}$$

Here d is the space dimension. In this paper, we will only consider the linear or convex Hamilton–Jacobi equations, namely the Hamiltonian H is a linear or convex function of φ_{x_i}. The solutions to the above equations are Lipschitz continuous but may admit discontinuous derivatives. For linear case with discontinuous coefficients or the nonlinear case, this is true even if the initial condition is smooth, and the solutions are also non-unique. We are only interested in the viscosity solution [6], which is the unique practically relevant solution and

* Corresponding author. Tel.: +1 401 863 2549; fax: +1 401 863 1355.
E-mail addresses: ycheng@dam.brown.edu (Y. Cheng), shu@dam.brown.edu (C.-W. Shu).

0021-9991/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2006.09.012

satisfies the entropy condition. Fortunately, this entropy condition can be expressed in a simple form when the Hamiltonian is linear or convex.

Essentially non-oscillatory (ENO) or weighted ENO (WENO) finite difference schemes have been developed to solve the Hamilton–Jacobi equation (1.1), see, e.g. [16,10,19]. These finite difference methods work quite efficiently for Cartesian meshes, however, on unstructured meshes the scheme is quite complicated [19]. Alternatively, the Runge–Kutta discontinuous Galerkin (DG) finite element method, originally devised to solve the conservation laws [3,4,2,1,5], has the advantage of flexibility for arbitrarily unstructured meshes, with a compact stencil, and with the ability to easily achieve arbitrary order of accuracy. In [9], Hu and Shu proposed a discontinuous Galerkin method to solve the Hamilton–Jacobi equation (1.1). They use the fact that the derivatives of the solution φ satisfy a conservation law system, and apply the usual discontinuous Galerkin method on this system to advance the derivatives of φ. The solution φ itself is then recovered from these derivatives by a least square procedure for multi-dimensional cases and with an independent evolution of the cell averages of φ. Later, Li and Shu [14] reinterpreted the method of Hu and Shu by using a curl-free subspace for the discontinuous Galerkin method. The algorithm in [14] is mathematically equivalent to that in [9], but the least square procedure is avoided and the computational cost is reduced for multi-dimensional calculations. The method in [9,14] works well numerically, with provable stability results for certain special cases [9,12]. However, since this method is based on the conservation law system satisfied by the derivatives of φ, a scalar problem (1.1) is converted to a system for the multi-dimensional case, which is moreover only weakly hyperbolic at some points. This seems to have made the algorithm indirect and complicated. It is, therefore, desirable to design a discontinuous Galerkin method which solves directly the solution φ to the Hamilton–Jacobi equation (1.1). In this paper, we develop such a discontinuous Galerkin method.

This paper is organized as follows: in Section 2, we describe the formulation of our scheme for the one-dimensional case. Theoretical analysis for the linear case is provided. In Section 3, we generalize the scheme to two space dimensions. Numerical results of both one and two dimensions are presented in Section 4. Finally, in Section 5, concluding remarks are given.

2. One-dimensional case

For the simple one-dimensional case, (1.1) becomes

$$\varphi_t + H(\varphi_x, x) = 0 \qquad \varphi(x,0) = \varphi^0(x) \tag{2.1}$$

and we consider only the case where $H(\varphi_x, x)$ is a linear or convex function of φ_x. If we want to solve this equation on the interval $[a,b]$, first we divide it into N cells as follows:

$$a = x_{\frac{1}{2}} < x_{\frac{3}{2}} < \ldots < x_{N+\frac{1}{2}} = b \tag{2.2}$$

We denote

$$I_j = (x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}), \qquad x_j = \frac{1}{2}\left(x_{j-\frac{1}{2}} + x_{j+\frac{1}{2}}\right), \qquad I_{j+1/2} = \left[x_j, x_{j+1}\right] \tag{2.3}$$

and

$$\Delta x_j = x_{j+\frac{1}{2}} - x_{j-\frac{1}{2}}, \qquad h = \max_j \Delta x_j \tag{2.4}$$

Now, we define the approximation space as

$$V_h^k = \{v : v|_{I_j} \in P^k(I_j), \quad j = 1, \ldots, N\} \tag{2.5}$$

where $P^k(I_j)$ denotes all polynomials of degree at most k on I_j. We now formulate our scheme for (2.1) and give theoretical analysis for the linear case.

2.1. Formulation of the scheme

Let us denote $H_1 = \frac{\partial H}{\partial \varphi_x}$. If H_1 is always non-negative, then we can define the upwind version of our scheme as: find $\varphi_h(x,t) \in V_h^k$, such that

$$\int_{I_j}(\partial_t\varphi_h(x,t)+H(\partial_x\varphi_h(x,t),x))v_h(x)\mathrm{d}x+\max_{x\in I_{j-1/2}}H_1(\partial_x\varphi_h,x_{j-1/2})[\varphi_h]_{j-\frac{1}{2}}(v_h)^+_{j-\frac{1}{2}}=0,\qquad j=1,\ldots,N \tag{2.6}$$

holds for any $v_h\in V_h^k$. Here $[\varphi_h]_{j-\frac{1}{2}}=\varphi_h(x^+_{j-\frac{1}{2}},t)-\varphi_h(x^-_{j-\frac{1}{2}},t)$ denotes the jump of φ_h at the cell interface $x_{j-\frac{1}{2}}$. Our definition of the scheme is motivated by the usual discontinuous Galerkin method for the linear case, see the next subsection 2.2.

For general H, our scheme is formulated as follows: find $\varphi_h(x,t)\in V_h^k$, such that

$$\begin{aligned}&\int_{I_j}(\partial_t\varphi_h(x,t)+H(\partial_x\varphi_h(x,t),x))v_h(x)\mathrm{d}x\\&+\frac{1}{2}\left(\min_{x\in I_{j+1/2}}H_1(\partial_x\varphi_h,x_{j+1/2})-\left|\min_{x\in I_{j+1/2}}H_1(\partial_x\varphi_h,x_{j+1/2})\right|\right)[\varphi_h]_{j+\frac{1}{2}}(v_h)^-_{j+\frac{1}{2}}\\&+\frac{1}{2}\left(\max_{x\in I_{j-1/2}}H_1(\partial_x\varphi_h,x_{j-1/2})+\left|\max_{x\in I_{j-1/2}}H_1(\partial_x\varphi_h,x_{j-1/2})\right|\right)[\varphi_h]_{j-\frac{1}{2}}(v_h)^+_{j-\frac{1}{2}}\\&=0,\qquad j=1,\ldots,N\end{aligned} \tag{2.7}$$

holds for any $v_h\in V_h^k$. This is a Roe type generalization of the upwind scheme (2.6).

In the schemes (2.6) and (2.7), we need the reconstructed information of $\partial_x\varphi_h$ on the cells $I_{j-1/2}$ and $I_{j+1/2}$. Notice that these cells include the points $x_{j-\frac{1}{2}}$ and $x_{j+\frac{1}{2}}$, respectively, where the numerical solution $\varphi_h(x,t)$ is discontinuous. We use an L^2 reconstruction technique as follows. We define a polynomial $w_{j+\frac{1}{2}}(x)\in P^{2k+1}$ on $I_j\cup I_{j+1}$, such that

$$\int_{I_j}\varphi_h v\mathrm{d}x=\int_{I_j}w_{j+\frac{1}{2}}v\mathrm{d}x \tag{2.8}$$

for any $v\in P^k$ on I_j, and

$$\int_{I_{j+1}}\varphi_h v\mathrm{d}x=\int_{I_{j+1}}w_{j+\frac{1}{2}}v\mathrm{d}x \tag{2.9}$$

for any $v\in P^k$ on I_{j+1}. Then we use $\partial_x\varphi_h=\partial_x w_{j+\frac{1}{2}}$ on $I_{j+1/2}$ when taking the maximum or minimum in (2.6) and (2.7).

In the case of a uniform mesh and piecewise constant polynomials ($k=0$), the reconstructed derivative becomes

$$\partial_x\varphi_h=\frac{\varphi_{j+1}-\varphi_j}{h} \tag{2.10}$$

on $I_{j+1/2}$. This agrees with our intuitive definition of $\partial_x\varphi_h$. For a practical implementation, once a local basis is chosen, the coefficients of $w_{j+\frac{1}{2}}(x)$ are linear combinations of the coefficients of $\varphi_h|_{I_j}$ and of $\varphi_h|_{I_{j+1}}$. These linear combination coefficients can be pre-computed to save computational cost.

We would like to remark that in (2.7), the last two terms involving the jumps of φ_h are added for stability, whereas the first integral term guarantees the accuracy of our scheme. The purpose of taking the maximum and minimum is to obtain better stability by adding more viscosity, while still maintaining accuracy since these maximum and minimum values are a $O(h)$ perturbation from $H_1(\partial_x\varphi_h(x_{j+1/2},t),x_{j+1/2})$, which guarantees accuracy according to truncation error analysis and numerical tests.

For linear Hamiltonians with discontinuous coefficients or nonlinear Hamiltonians, since our scheme is of Roe type, it may generate entropy violating solutions. We have, therefore, adopted the following entropy correction procedure:

1. For each cell I_j, determine if it is a potentially entropy violating cell. We will provide the criteria for this determination in the numerical Section 4.
   If the cell I_j is marked as a potentially entropy violating cell, then use Step 2 below to update φ_h in this cell; otherwise, update φ_h by (2.6) or (2.7).

2. Update φ_h by the DG method of Hu and Shu [9], namely, recover $\partial_x\varphi_h$ by taking the derivative of φ_h, then compute $\partial_t(\partial_x\varphi_h)$ by the usual DG method for the conservation law satisfied by $\partial_x\varphi_h$. This will determine φ_h up to a constant. The missing constant is obtained by requiring

$$\int_{I_j}(\partial_t\varphi_h(x,t)+H(\partial_x\varphi_h(x,t),x))\mathrm{d}x=0. \tag{2.11}$$

The entropy correction in Step 2 bears comparable computational cost as our scheme (2.6) or (2.7) for this one-dimensional case. It is our experience that such entropy corrections are needed only in very few isolated cells, see Section 4.

2.2. Theoretical analysis

We first consider a linear Hamilton–Jacobi equation

$$\varphi_t+a(x)\varphi_x=0 \tag{2.12}$$

and assume, for the time being, that $a(x)$ is a smooth function. If $a(x)>0$, our scheme (2.6), after replacing the maximum by the point value at $x_{j-\frac{1}{2}}$, becomes finding $\varphi_h(x,t)\in V_h^k$, such that

$$\int_{I_j}(\partial_t\varphi_h(x,t)+a(x)(\partial_x\varphi_h(x,t)))v_h(x)\mathrm{d}x+a(x_{j-\frac{1}{2}})[\varphi_h]_{j-\frac{1}{2}}(v_h)^+_{j-\frac{1}{2}}=0,\quad j=1,\ldots,N \tag{2.13}$$

holds for any $v_h\in V_h^k$. After integration by parts, this is equivalent to

$$\begin{aligned}&\int_{I_j}\partial_t\varphi_h(x,t)v_h(x)\mathrm{d}x-\int_{I_j}a(x)\varphi_h(x,t)\partial_x v_h(x)\mathrm{d}x+(a\varphi_h)^-_{j+\frac{1}{2}}(v_h)^-_{j+\frac{1}{2}}-(a\varphi_h)^-_{j-\frac{1}{2}}(v_h)^+_{j-\frac{1}{2}}\\&\quad=\int_{I_j}a_x(x)\varphi_h(x,t)v_h(x)\mathrm{d}x\end{aligned} \tag{2.14}$$

We observe that the scheme (2.14) is the standard DG scheme for conservation laws with source terms using upwind fluxes [4] for the equation

$$\varphi_t+(a(x)\varphi)_x=a_x(x)\varphi \tag{2.15}$$

which is equivalent to (2.12). Similarly, for general $a(x)$, our scheme (2.7), after replacing the maximum and minimum by the point value at $x_{j-\frac{1}{2}}$ and $x_{j+\frac{1}{2}}$, respectively, becomes finding $\varphi_h(x,t)\in V_h^k$, such that

$$\begin{aligned}&\int_{I_j}(\partial_t\varphi_h(x,t)+a(x)(\partial_x\varphi_h(x,t)))v_h(x)\mathrm{d}x+\frac{1}{2}(a(x_{j+\frac{1}{2}})-|a(x_{j+\frac{1}{2}})|)[\varphi_h]_{j+\frac{1}{2}}(v_h)^-_{j+\frac{1}{2}}\\&\quad+\frac{1}{2}(a(x_{j-\frac{1}{2}})+|a(x_{j-\frac{1}{2}})|)[\varphi_h]_{j-\frac{1}{2}}(v_h)^+_{j-\frac{1}{2}}=0\end{aligned} \tag{2.16}$$

holds for any $v_h\in V_h^k$. After integration by parts, this is equivalent to

$$\int_{I_j}\partial_t\varphi_h(x,t)v_h(x)\mathrm{d}x-\int_{I_j}a(x)\varphi_h(x,t)\partial_x v_h(x)\mathrm{d}x+\widehat{a\varphi_h}_{j+\frac{1}{2}}(v_h)^-_{j+\frac{1}{2}}-\widehat{a\varphi_h}_{j-\frac{1}{2}}(v_h)^+_{j-\frac{1}{2}}=\int_{I_j}a_x(x)\varphi_h(x,t)v_h(x)\mathrm{d}x \tag{2.17}$$

where

$$\widehat{a\varphi_h}_{j+\frac{1}{2}}=\frac{1}{2}(a(x_{j+\frac{1}{2}})+|a(x_{j+\frac{1}{2}})|)(\varphi_h)^-_{j+\frac{1}{2}}+\frac{1}{2}(a(x_{j+\frac{1}{2}})-|a(x_{j+\frac{1}{2}})|)(\varphi_h)^+_{j+\frac{1}{2}}$$

denotes the Roe flux. This is the standard DG scheme for conservation laws with source terms using Roe fluxes [4] for Eq. (2.15) which is equivalent to (2.12). We can, therefore, use the standard techniques in the analysis for the DG schemes to obtain the following theoretical results.

Proposition 2.1. *Suppose there is a constant β such that the derivative of $a(x)$ satisfies $a_x(x) < \beta$ for $x \in [a,b]$, then we have the following L^2 stability for our scheme (2.16):*

$$\|\varphi_h(t)\|_{L^2} \leqslant e^{\beta t/2}\|\varphi_h(0)\|_{L^2}$$

Proof. This follows from the standard proof of the cell entropy inequality for DG schemes applied to scalar conservation laws [11]. On each cell I_j, we can prove as in [11]

$$\int_{I_j}(\varphi_h)_t\varphi_h \mathrm{d}x - \int_{I_j} a_x\frac{\varphi_h^2}{2}\mathrm{d}x + \hat{F}_{j+\frac{1}{2}} - \hat{F}_{j-\frac{1}{2}} + \Theta_j = 0$$

for some entropy flux $\hat{F}_{j+\frac{1}{2}}$ and $\Theta_j \geqslant 0$. Summing over j, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_a^b \frac{\varphi_h^2}{2}\mathrm{d}x \leqslant \int_a^b a_x\frac{\varphi_h^2}{2}\mathrm{d}x$$

Since $a_x < \beta$, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_a^b \varphi_h^2\mathrm{d}x \leqslant \beta\int_a^b \varphi_h^2\mathrm{d}x$$

Integrating over t finishes the proof. □

Proposition 2.2. *If $a(x)$ and the solution φ of (2.12) are smooth and the scheme (2.16) with the finite element space (2.5) is used, then we have the following optimal L^2 error estimate*

$$\|\varphi_h(t) - \varphi(t)\|_{L^2} \leqslant Ch^{k+1}$$

Proof. The proof is similar to that for standard DG schemes. The optimal $(k+1)$th order of convergence is obtained through a special projection in the proof, see for example [18] for the details. □

For nonlinear problems, we notice that our scheme is consistent only when $k \geqslant 1$. For example, for the Burgers' equation

$$\varphi_t + \frac{\varphi_x^2}{2} = 0, \tag{2.18}$$

with $\varphi_x \geqslant 0$, our upwind scheme (2.6) with piecewise constant space (the space (2.5) with $k = 0$) gives

$$(\varphi_j)_t + \left(\frac{\varphi_j - \varphi_{j-1}}{\Delta x_j}\right)^2 = 0 \tag{2.19}$$

where $\varphi = \varphi_j$ on cell I_j. This is clearly consistent with a different equation $\varphi_t + \varphi_x^2 = 0$ and is inconsistent with the Burgers' Eq. (2.18). If $k \geqslant 1$, we can prove that our scheme is consistent, which is also verified by numerical experiments in Section 4. For example, the P^1 upwind scheme (2.6) to solve the equation $\varphi_t + H(\varphi_x) = 0$ is

$$\left(\varphi_j^0\right)_t + H\left(\frac{\varphi_j^1}{\Delta x_j}\right) + \alpha_j\frac{[\varphi_h]_{j-\frac{1}{2}}}{\Delta x_j} = 0 \tag{2.20}$$

where $\varphi = \varphi_j^0 + \varphi_j^1\frac{x-x_j}{\Delta x_j}$ on cell I_j. Since $[\varphi_h]_{j-\frac{1}{2}} = O(h^2)$ for smooth functions, the scheme is consistent.

2.3. Time discretization

Up to now, we have taken the method of lines approach and have left t continuous. We can use total variation diminishing (TVD) high-order Runge–Kutta methods [17] to solve the method of lines ODE

$$\varphi_t = L(\varphi). \tag{2.21}$$

The third-order TVD Runge–Kutta method that we use in this paper is given by

$$\begin{aligned}\varphi^{(1)} &= \varphi^n + \Delta t L(\varphi^n)\\ \varphi^{(2)} &= \frac{3}{4}\varphi^n + \frac{1}{4}\varphi^{(1)} + \frac{1}{4}\Delta t L(\varphi^{(1)})\\ \varphi^{n+1} &= \frac{1}{3}\varphi^n + \frac{2}{3}\varphi^{(2)} + \frac{2}{3}\Delta t L(\varphi^{(2)})\end{aligned} \tag{2.22}$$

Detailed description of the TVD Runge–Kutta method can be found in [17], see also [7,8].

3. Two-dimensional case

In this section, we consider the case of two spatial dimensions. The equation is given by

$$\varphi_t + H(\varphi_x, \varphi_y, x, y) = 0, \qquad \varphi(x, y, 0) = \varphi^0(x, y) \tag{3.1}$$

and we again only consider the case where $H(\varphi_x,\varphi_y,x,y)$ is a linear or convex function of φ_x and φ_y. For simplicity of presentation, we consider in this paper only rectangular domains and cells, although our method can be easily defined on general triangulations as other DG methods. Suppose Eq. (3.1) is solved on the domain $[a,b]\times[c,d]$. We use rectangular meshes defined as

$$a = x_{\frac{1}{2}} < x_{\frac{3}{2}} < \ldots < x_{N_x+\frac{1}{2}} = b, \qquad c = y_{\frac{1}{2}} < y_{\frac{3}{2}} < \ldots < y_{N_y+\frac{1}{2}} = d \tag{3.2}$$

and

$$\begin{aligned}&I_{i,j} = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}] \times [y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}], \quad J_i = [x_{i-1/2}, x_{i+1/2}], \quad K_j = [y_{j-1/2}, y_{j+1/2}]\\ &J_{i+1/2} = [x_i, x_{i+1}], \quad K_{j+1/2} = [y_j, y_{j+1}], \quad i = 1, \ldots, N_x, \quad j = 1, \ldots, N_y\end{aligned} \tag{3.3}$$

We define the approximation space as

$$V_h^k = \{v : v|_{I_{i,j}} \in P^k(I_{i,j}), i = 1, \ldots, N_x, \quad j = 1, \ldots, N_y\} \tag{3.4}$$

where $P^k(I_{i,j})$ denotes all polynomials of degree at most k on $I_{i,j}$.

Let us denote $H_1 = \frac{\partial H}{\partial \varphi_x}$ and $H_2 = \frac{\partial H}{\partial \varphi_y}$. We define our scheme as: find $\varphi_h(x,t) \in V_h^k$, such that

$$\begin{aligned}&\int_{I_{i,j}} (\partial_t \varphi_h(x,y,t) + H(\partial_x \varphi_h(x,y,t), \partial_y \varphi_h(x,y,t), x, y)) v_h(x,y) \mathrm{d}x\mathrm{d}y\\ &+ \frac{1}{2}\int_{K_j} \left(\min_{x\in J_{i+1/2}} H_1(\partial_x \varphi_h, \overline{\partial_y \varphi_h}, x_{i+1/2}, y) - \left| \min_{x\in J_{i+1/2}} H_1(\partial_x \varphi_h, \overline{\partial_y \varphi_h}, x_{i+1/2}, y) \right| \right) [\varphi_h](x_{i+\frac{1}{2}}, y) v_h(x^-_{i+\frac{1}{2}}, y) \mathrm{d}y\\ &+ \frac{1}{2}\int_{K_j} \left(\max_{x\in J_{i-1/2}} H_1(\partial_x \varphi_h, \overline{\partial_y \varphi_h}, x_{i-1/2}, y) + \left| \max_{x\in J_{i-1/2}} H_1(\partial_x \varphi_h, \overline{\partial_y \varphi_h}, x_{i-1/2}, y) \right| \right) [\varphi_h](x_{i-\frac{1}{2}}, y) v_h(x^+_{i-\frac{1}{2}}, y) \mathrm{d}y\\ &+ \frac{1}{2}\int_{J_i} \left(\min_{y\in K_{j+1/2}} H_2(\overline{\partial_x \varphi_h}, \partial_y \varphi_h, x, y_{j+1/2}) - \left| \min_{y\in K_{j+1/2}} H_2(\overline{\partial_x \varphi_h}, \partial_y \varphi_h, x, y_{j+1/2}) \right| \right) [\varphi_h](x, y_{j+\frac{1}{2}}) v_h(x, y^-_{j+\frac{1}{2}}) \mathrm{d}x\\ &+ \frac{1}{2}\int_{J_i} \left(\max_{y\in K_{j-1/2}} H_2(\overline{\partial_x \varphi_h}, \partial_y \varphi_h, x, y_{j-1/2}) + \left| \max_{y\in K_{j-1/2}} H_2(\overline{\partial_x \varphi_h}, \partial_y \varphi_h, x, y_{j-1/2}) \right| \right) [\varphi_h](x, y_{j-\frac{1}{2}}) v_h(x, y^+_{j-\frac{1}{2}}) \mathrm{d}x = 0\end{aligned} \tag{3.5}$$

holds for any $v_h \in V_h^k$. In the above formula, we define

$$\overline{\partial_x \varphi_h} = \frac{1}{2}((\partial_x \varphi_h)^+ + (\partial_x \varphi_h)^-), \qquad \overline{\partial_y \varphi_h} = \frac{1}{2}((\partial_y \varphi_h)^+ + (\partial_y \varphi_h)^-)$$

The main idea is that, on the interfaces of cells, along the normal direction we would use the reconstructed information of the partial derivatives as in the one-dimensional case. Tangential to the interface, the average of the partial derivatives from the two neighboring cells is used. The reconstruction process is the same as that

in the one-dimensional case, except that we need to fix x or y, then perform the reconstruction on the other spatial variable. If $H_1 > 0$ or $H_2 > 0$, we can apply the corresponding upwind scheme in that direction.

A similar entropy correction procedure as in the one-dimensional case is adopted here as well for the cases with linear Hamiltonians with discontinuous coefficients or nonlinear Hamiltonians:

1. For each cell $I_{i,j}$, determine if it is a potentially entropy violating cell. We will again provide the criteria for this determination in the numerical Section 4.
   If the cell $I_{i,j}$ is marked as a potentially entropy violating cell, then use Step 2 below to update φ_h in this cell; otherwise, update φ_h by (3.5).
2. Update φ_h by the DG method of Hu and Shu [9] as reinterpreted by Li and Shu [14], namely, recover $\partial_x\varphi_h$ and $\partial_y\varphi_h$ by taking the derivatives of φ_h, then compute $\partial_t(\partial_x\varphi_h)$ and $\partial_t(\partial_y\varphi_h)$ by the usual DG method for the conservation laws satisfied by $\partial_x\varphi_h$ and $\partial_y\varphi_h$ in a locally curl-free discontinuous Galerkin space. This will determine φ_h up to a constant. The missing constant is obtained by requiring

$$\int_{I_{i,j}} (\partial_t\varphi_h + H(\partial_x\varphi_h, \partial_y\varphi_h, x, y))\mathrm{d}x\mathrm{d}y = 0. \tag{3.6}$$

4. Numerical results

In this section, we provide numerical experimental results to demonstrate the behavior of our schemes.

4.1. Linear smooth problems

In this subsection, linear smooth problems are computed using our scheme. In this case, our scheme is equivalent to the standard DG scheme for conservation laws with source terms.

Example 4.1.1. We solve the one-dimensional problem

$$\begin{cases} \varphi_t + \sin(x)\varphi_x = 0 \\ \varphi(x,0) = \sin(x) \\ \varphi(0,t) = \varphi(2\pi,t) \end{cases} \tag{4.1}$$

The exact solution is

$$\varphi(x,t) = \sin\left(2\tan^{-1}\left(\mathrm{e}^{-t}\tan\left(\frac{x}{2}\right)\right)\right) \tag{4.2}$$

We use the general scheme (2.16) and list the results in Tables 4.1–4.4 for P^0, P^1, P^2 and P^3, respectively. We clearly observe $(k+1)$th order of accuracy for P^k polynomials.

Example 4.1.2. We solve the two-dimensional linear Hamilton–Jacobi equation with variable coefficients

$$\varphi_t - y\varphi_x + x\varphi_y = 0. \tag{4.3}$$

Table 4.1
Errors and numerical orders of accuracy for Example 4.1.1 when using P^0 polynomials and Runge–Kutta third order time discretization on a uniform mesh of N cells

N	L^1 error	Order	L^2 error	Order	L^∞ error	Order
40	0.49E−01		0.62E−01		0.29E+00	
80	0.25E−01	0.95	0.32E−01	0.93	0.16E+00	0.86
160	0.13E−01	0.97	0.17E−01	0.96	0.83E−01	0.96
320	0.65E−02	0.98	0.84E−02	0.98	0.42E−01	0.99
640	0.33E−02	0.99	0.42E−02	0.99	0.21E−01	1.00

Final time $t = 1$. CFL = 0.9.

Table 4.2
Errors and numerical orders of accuracy for Example 4.1.1 when using P^1 polynomials and Runge–Kutta third order time discretization on a uniform mesh of N cells

N	L^1 error	Order	L^2 error	Order	L^∞ error	Order
40	0.12E−02		0.25E−02		0.15E−01	
80	0.31E−03	1.96	0.68E−03	1.90	0.43E−02	1.81
160	0.78E−04	1.97	0.18E−03	1.94	0.11E−02	1.92
320	0.20E−04	1.98	0.46E−04	1.97	0.29E−03	1.96
640	0.50E−05	1.99	0.12E−04	1.98	0.74E−04	1.98

Final time $t = 1$. CFL = 0.3.

Table 4.3
Errors and numerical orders of accuracy for Example 4.1.1 when using P^2 polynomials and Runge–Kutta third order time discretization on a uniform mesh of N cells

N	L^1 error	Order	L^2 error	Order	L^∞ error	Order
40	0.48E−04		0.10E−03		0.52E−03	
80	0.60E−05	2.99	0.14E−04	2.88	0.88E−04	2.58
160	0.75E−06	3.00	0.18E−05	2.90	0.14E−04	2.70
320	0.94E−07	2.99	0.24E−06	2.93	0.20E−05	2.78
640	0.12E−07	2.99	0.31E−07	2.95	0.27E−06	2.85

Final time $t = 1$. CFL = 0.1.

Table 4.4
Errors and numerical orders of accuracy for Example 4.1.1 when using P^3 polynomials and Runge–Kutta third order time discretization on a uniform mesh of N cells

N	L^1 error	Order	L^2 error	Order	L^∞ error	Order
40	0.21E−05		0.51E−05		0.29E−04	
80	0.14E−06	3.96	0.35E−06	3.88	0.22E−05	3.75
160	0.87E−08	3.97	0.23E−07	3.93	0.16E−06	3.78
320	0.55E−09	3.97	0.15E−08	3.96	0.10E−07	3.91
640	0.35E−10	3.98	0.94E−10	3.98	0.68E−09	3.95

Final time $t = 1$. CFL = 0.05.

The computational domain is $[-1,1]^2$. The initial condition is given by

$$\varphi_0(x,y) = \begin{cases} 0 & 0.3 \leqslant r \\ 0.3 - r & 0.1 < r < 0.3 \\ 0.2 & r \leqslant 0.1 \end{cases} \tag{4.4}$$

where $r = \sqrt{(x-0.4)^2 + (y-0.4)^2}$. We also impose periodic boundary condition on the domain. This is a solid body rotation around the origin. The exact solution can be expressed as

$$\varphi(x,y,t) = \varphi_0(x\cos(t) + y\sin(t), -x\sin(t) + y\cos(t)) \tag{4.5}$$

For this problem, the derivatives of φ are not continuous. Therefore, we do not expect to obtain $(k+1)$th order of accuracy for P^k polynomials, see Table 4.5.

At $t = 2\pi$, i.e. the period of rotation, we take a snapshot at the line $x = y$ in Fig. 4.1. We can see that a higher order scheme can yield better results for this nonsmooth initial condition.

Example 4.1.3. We solve the same Eq. (4.3) as that in Example 4.1.2, but with a different initial condition as

$$\varphi_0(x,y) = \exp\left(-\frac{(x-0.4)^2 + (y-0.4)^2}{2\sigma^2}\right) \tag{4.6}$$

Table 4.5
Errors and numerical orders of accuracy for Example 4.1.2 when using P^2 polynomials and Runge–Kutta third order time discretization on a uniform mesh of $N \times N$ cells

$N \times N$	L^1 error	Order	L^2 error	Order	L^∞ error	Order
20×20	0.41E−03		0.13E−02		0.11E−01	
40×40	0.14E−03	1.58	0.55E−03	1.26	0.65E−02	0.82
80×80	0.47E−04	1.54	0.24E−03	1.22	0.36E−02	0.84
160×160	0.15E−04	1.62	0.10E−03	1.23	0.21E−02	0.81

Final time $t = 1$. CFL = 0.1.

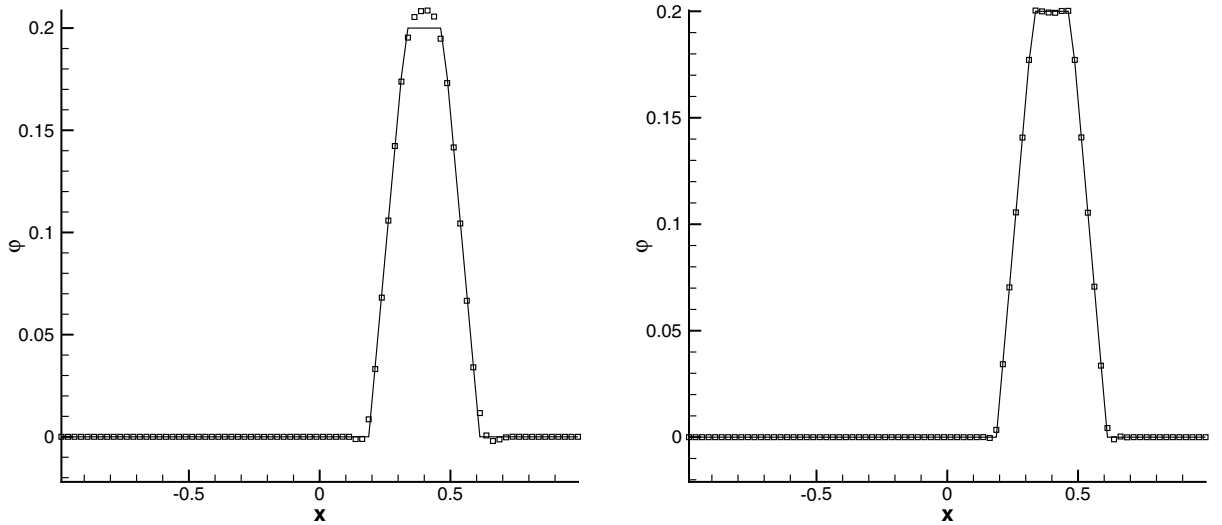

Fig. 4.1. Example 4.1.2. 80×80 uniform mesh. $t = 2\pi$. Solid line: the exact solution; rectangles: the numerical solution. One-dimensional cut of 45° with the x axis. Left: P^1 polynomial; right: P^2 polynomial.

We take $\sigma = 0.05$ such that at the domain boundary, φ is very small, hence imposing a periodic boundary condition will lead to small non-smoothness errors. We then observe the desired order of accuracy in Table 4.6.

Example 4.1.4. We solve the two-dimensional linear Hamilton–Jacobi equation with variable coefficients

$$\varphi_t + f(x,y,t)\varphi_x + g(x,y,t)\varphi_y = 0 \tag{4.7}$$

The computational domain is still $[-1,1]^2$, and the advection coefficients are

$$f(x,y,t) = \sin^2(\pi x)\sin(2\pi y)\cos\left(\frac{t}{T}\pi\right), \quad g(x,y,t) = -\sin^2(\pi y)\sin(2\pi x)\cos\left(\frac{t}{T}\pi\right)$$

where T is the period of deformation. The initial condition is given by

Table 4.6
Errors and numerical orders of accuracy for Example 4.1.3 when using P^2 polynomials and Runge–Kutta third order time discretization on a uniform mesh of $N \times N$ cells

$N \times N$	L^1 error	Order	L^2 error	Order	L^∞ error	Order
20×20	0.14E−02		0.10E−01		0.28E+00	
40×40	0.15E−03	3.21	0.15E−02	2.81	0.53E−01	2.41
80×80	0.11E−04	3.82	0.11E−03	3.73	0.58E−02	3.19
160×160	0.11E−05	3.30	0.12E−04	3.26	0.90E−03	2.69

Final time $t = 1$. CFL = 0.1.

$$\varphi_0(x,y) = \begin{cases} 0 & 0.3 \leqslant r \\ 0.3 - r & 0.1 < r < 0.3 \\ 0.2 & r \leqslant 0.1 \end{cases} \tag{4.8}$$

where $r = \sqrt{(x-0.4)^2 + (y-0.4)^2}$. This is a numerical test for incompressible flow first introduced by LeVeque in [13]. During the evolution, the initial data are severely deformed, then it returns to the original shape after one period. At $t = 1.5$, i.e. the period of rotation, we take a snapshot at the line $x = y$ in Fig. 4.2. We can clearly observe that a higher order scheme yields better results for this nonsmooth initial condition.

4.2. Linear nonsmooth problems

In this subsection, the Hamiltonian H is a linear function of $\nabla\varphi$ with nonsmooth coefficients.

Example 4.2.1. We solve the model problem

$$\begin{cases} \varphi_t + \text{sign}(\cos(x))\,\varphi_x = 0 \\ \varphi(x,0) = \sin(x) \\ \varphi(0,t) = \varphi(2\pi,t) \end{cases} \tag{4.9}$$

The exact solution is given by

- if $0 \leqslant t \leqslant \pi/2$

$$\varphi(x,t) = \begin{cases} \sin(x-t) & \text{if } 0 \leqslant x \leqslant \frac{\pi}{2} \\ \sin(x+t) & \text{if } \frac{\pi}{2} < x \leqslant \frac{3\pi}{2} - t \\ -1 & \text{if } \frac{3\pi}{2} - t < x \leqslant \frac{3\pi}{2} + t \\ \sin(x-t) & \text{if } \frac{3\pi}{2} + t < x \leqslant 2\pi \end{cases} \tag{4.10}$$

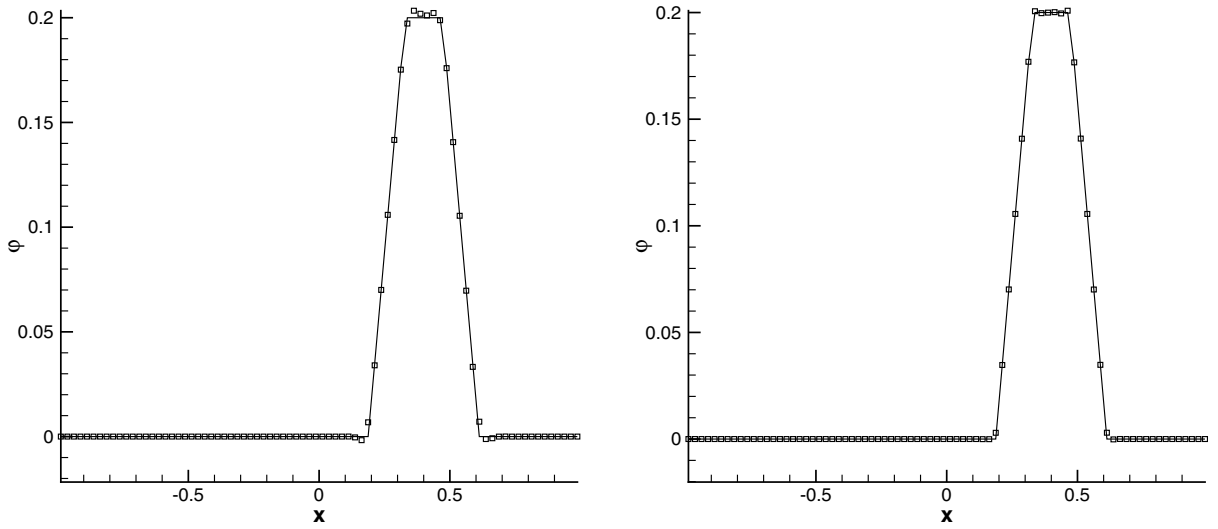

Fig. 4.2. Example 4.1.4. 80×80 uniform mesh. $t = 1.5$. Solid line: the exact solution; rectangles: the numerical solution. One-dimensional cut of 45° with the x axis. Left: P^1 polynomial; right: P^2 polynomial.

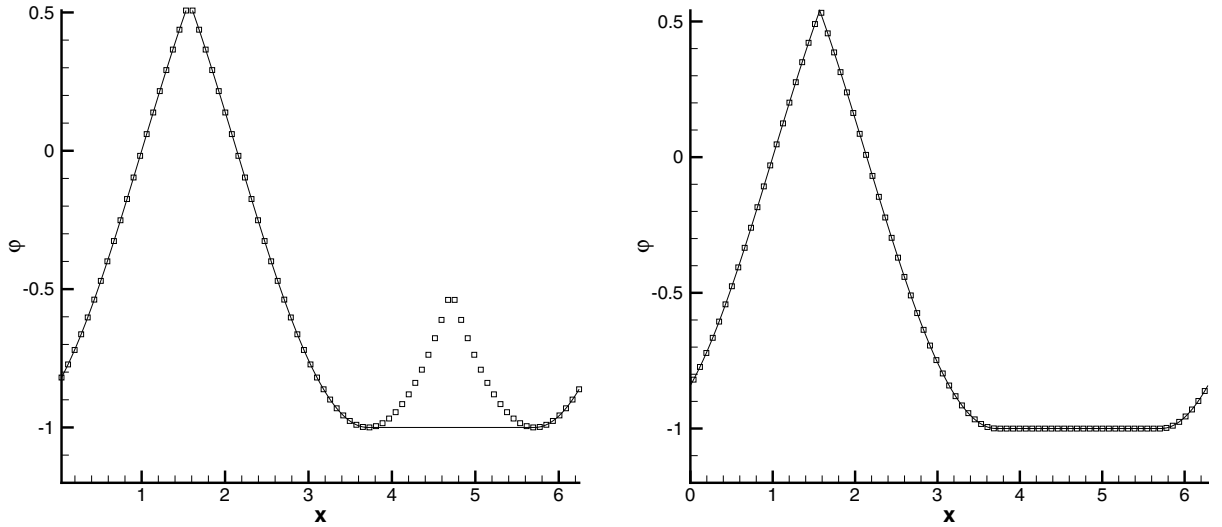

Fig. 4.3. Example 4.2.1. $t = 1$, CFL $= 0.1$, using P^2 polynomials. Solid line: the exact solution; rectangles: the numerical solution. Left: $N = 80$; right: $N = 81$.

- if $\pi/2 \leqslant t \leqslant \pi$

$$\varphi(x,t) = \begin{cases} -1 & \text{if } 0 \leqslant x \leqslant t - \frac{\pi}{2} \\ \sin(x-t) & \text{if } t - \frac{\pi}{2} < x \leqslant \frac{\pi}{2} \\ \sin(x+t) & \text{if } \frac{\pi}{2} < x \leqslant \frac{3\pi}{2} - t \\ -1 & \text{if } \frac{3\pi}{2} - t < x \leqslant 2\pi \end{cases} \tag{4.11}$$

- if $t \geqslant \pi$

$$\varphi(x,t) = -1. \tag{4.12}$$

For the viscosity solution, at $x = \frac{\pi}{2}$, there will be a shock forming in φ_x, and at $x = \frac{3\pi}{2}$, there is a rarefaction wave.

We first test the scheme without any entropy correction. If we take N to be a multiple of 4, then the discontinuity of $a(x)$ is exactly located at a cell interface. In this case, the entropy condition is violated by our scheme at the two cells neighboring $\frac{3\pi}{2}$, and the numerical solution obtained is not close to the viscosity solution, see Fig. 4.3, left. If instead, we take other values of N such that the discontinuity of $a(x)$ is not at the cell interface, then the entropy condition is not violated and the numerical solution obtained approximates the viscosity solution very well, see Fig. 4.3, right.

The test above indicates the necessity of an entropy correction in this case. The criteria for the entropy correction is as follows. For the cell $I_j = (x_{j\text{-}1/2}, x_{j+1/2})$, if

$$a^-(x_{j-1/2}) < 0 < a^+(x_{j-1/2}) \tag{4.13}$$

or

$$a^-(x_{j+1/2}) < 0 < a^+(x_{j+1/2}) \tag{4.14}$$

is satisfied, we will compute $(\varphi_x)_t$ on the cell I_j by solving the conservation law for $\varphi_x = u$ as

$$u_t + (\text{sign}(\cos(x))u)_x = 0 \tag{4.15}$$

using the standard DG method with polynomials in $P^{k\text{-}1}$, and then recover φ by requiring

$$\int_{I_j} (\varphi_t + \text{sign}(\cos(x))\varphi_x)\mathrm{d}x = 0. \tag{4.16}$$

For this one-dimensional example it does not increase the computational cost. We can see in Fig. 4.4 that, after this entropy correction, the numerical solution approximates the viscosity solution very well.

The numerical errors and order of accuracy are shown in Table 4.7. Since the exact solution is not smooth, we do not expect the full $(k+1)$th order accuracy.

4.3. Nonlinear smooth problems

In this subsection, the Hamiltonian H is a nonlinear smooth function of $\nabla\varphi$.

Example 4.3.1. One-dimensional Burgers' equation

$$\begin{cases} \varphi_t + \frac{\varphi_x^2}{2} = 0 \\ \varphi(x,0) = \sin(x) \\ \varphi(0,t) = \varphi(2\pi,t) \end{cases} \tag{4.17}$$

The exact solution when φ is still smooth is obtained by the characteristics methods. First solve x_0 from

$$x = x_0 + \cos(x_0)t \tag{4.18}$$

then get φ as

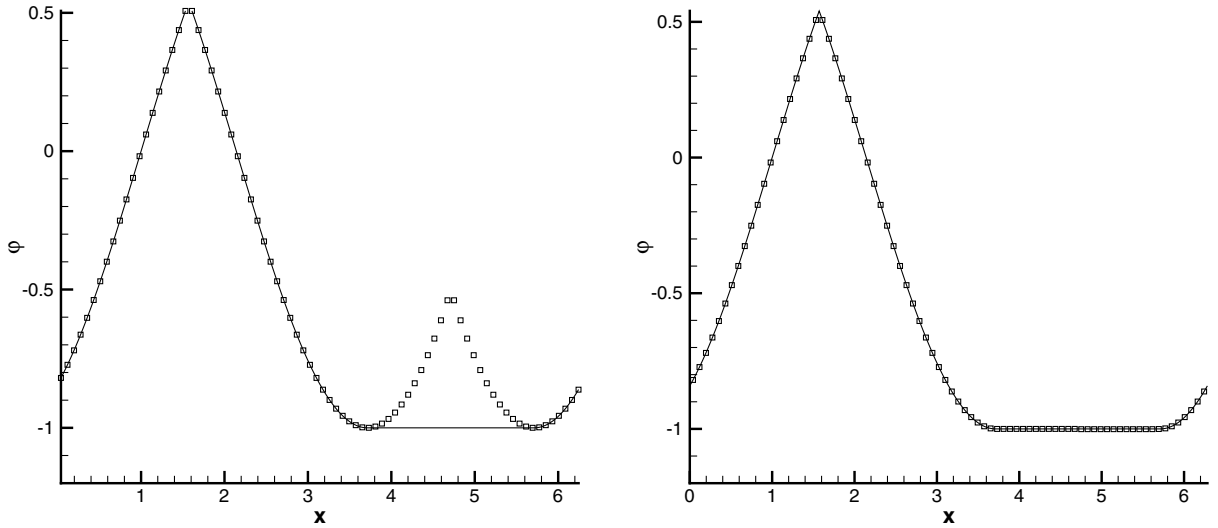

Fig. 4.4. Example 4.2.1. $t = 1$, CFL $= 0.1$, $N = 80$, using P^2 polynomials. Solid line: the exact solution; rectangles: the numerical solution. Left: without entropy correction; right: with entropy correction.

Table 4.7
Errors and numerical orders of accuracy for Example 4.2.1 when using P^2 polynomials and Runge–Kutta third order time discretization on a uniform mesh of N cells

N	L^1 error	Order	L^2 error	Order	L^∞ error	Order
40	0.64E−03		0.15E−02		0.41E−02	
80	0.16E−03	1.97	0.40E−03	1.94	0.10E−02	2.00
160	0.41E−04	1.99	0.10E−03	1.97	0.26E−03	2.00
320	0.10E−04	2.00	0.25E−04	1.99	0.64E−04	2.00
640	0.26E−05	2.00	0.64E−05	2.00	0.16E−04	2.00

Final time $t = 1$. CFL $= 0.1$.

$$\varphi(x,t) = \sin(x_0) + \frac{\cos(x_0)^2}{2}t \tag{4.19}$$

When $t = 0.5$, the solution is still smooth, and the expected third order accuracy is obtained for P^2 polynomials, see Table 4.8. After $t = 1$, a shock will form in φ_x, our scheme can resolve the derivative singularity sharply, see Fig. 4.5.

Example 4.3.2. One-dimensional Burgers' equation with a nonsmooth initial condition

$$\begin{cases} \varphi_t + \frac{\varphi_x^2}{2} = 0 \\ \varphi(x,0) = \begin{cases} \pi - x & \text{if } 0 \leqslant x \leqslant \pi \\ x - \pi & \text{else where in } [0, 2\pi], \end{cases} \\ \varphi(0,t) = \varphi(2\pi,t) \end{cases} \tag{4.20}$$

For the viscosity solution, the sharp corner at π will be smoothed out, and a rarefaction wave will form in the derivative. Since the entropy condition is violated by our Roe type scheme, we need to apply the entropy correction procedure. Fig. 4.6 shows the comparison of the numerical solution with and without the entropy correction. Clearly the entropy correction is needed to obtain a good approximation to the entropy solution. The criteria for the entropy correction is as follows. For the cell $I_j = (x_{j\text{-}1/2}, x_{j+1/2})$, if either

$$\varphi_x^-(x_{j-1/2}) < 0 < \varphi_x^+(x_{j-1/2}) \tag{4.21}$$

Table 4.8
Errors and numerical orders of accuracy for Example 4.3.1 when using P^2 polynomials and Runge–Kutta third order time discretization on a uniform mesh of N cells

N	L^1 error	Order	L^2 error	Order	L^∞ error	Order
40	0.13E−04		0.22E−04		0.84E−04	
80	0.17E−05	2.97	0.29E−05	2.93	0.12E−04	2.86
160	0.22E−06	2.98	0.37E−06	2.96	0.15E−05	2.92
320	0.27E−07	2.98	0.47E−07	2.97	0.20E−06	2.95
640	0.34E−08	2.99	0.59E−08	2.99	0.25E−07	2.97

Final time $t = 0.5$. CFL = 0.1.

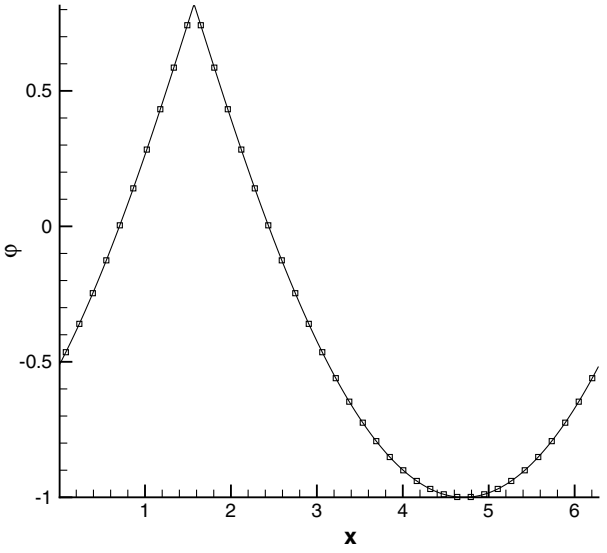

Fig. 4.5. Example 4.3.1. Numerical solution. Solid line: $N = 500$; rectangles: $N = 40$. Final time $t = 1.5$, CFL = 0.05, P^2 polynomials.

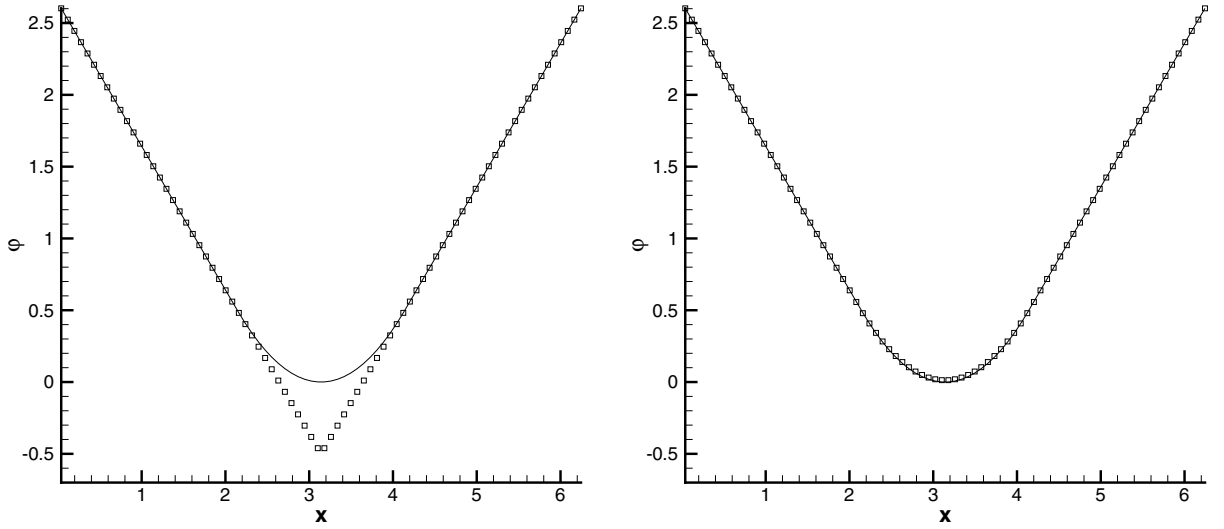

Fig. 4.6. Example 4.3.2. $t = 1$, CFL $= 0.05$, $N = 80$, using P^2 polynomials. Solid line: the exact solution; rectangles: the numerical solution. Left: without entropy correction; right: with entropy correction.

or

$$\varphi_x^-(x_{j+1/2}) < 0 < \varphi_x^+(x_{j+1/2}) \tag{4.22}$$

is satisfied, then the entropy correction is needed.

Example 4.3.3. Two-dimensional Burgers' equation.

$$\begin{cases} \varphi_t + \frac{(\varphi_x+\varphi_y+1)^2}{2} = 0 \\ \varphi(x,y,0) = -\cos(x+y) \end{cases} \tag{4.23}$$

with periodic boundary condition on the domain $[0,2\pi]^2$.

We use a uniform rectangular mesh. At $t = 0.1$, the solution is still smooth. Numerical errors and order of accuracy are listed in Table 4.9, demonstrating the expected order of accuracy. At $t = 1$, the solution is no longer smooth. We plot the numerical solution in Fig. 4.7. We observe good resolution of the kinks in the solution.

4.4. Nonlinear nonsmooth problems

In this subsection, the Hamiltonian H is a nonlinear nonsmooth function of $\nabla\varphi$.

Table 4.9
Errors and numerical orders of accuracy for Example 4.3.3 when using P^2 polynomials and Runge–Kutta third order time discretization on a uniform mesh of $N \times N$ cells

$N \times N$	L^1 error	Order	L^2 error	Order	L^∞ error	Order
10×10	0.30E−02		0.43E−02		0.35E−01	
20×20	0.38E−03	2.98	0.58E−03	2.90	0.56E−02	2.64
40×40	0.48E−04	2.97	0.77E−04	2.91	0.80E−03	2.81
80×80	0.66E−05	2.87	0.11E−04	2.83	0.14E−03	2.55

Final time $t = 0.1$. CFL $= 0.1$.

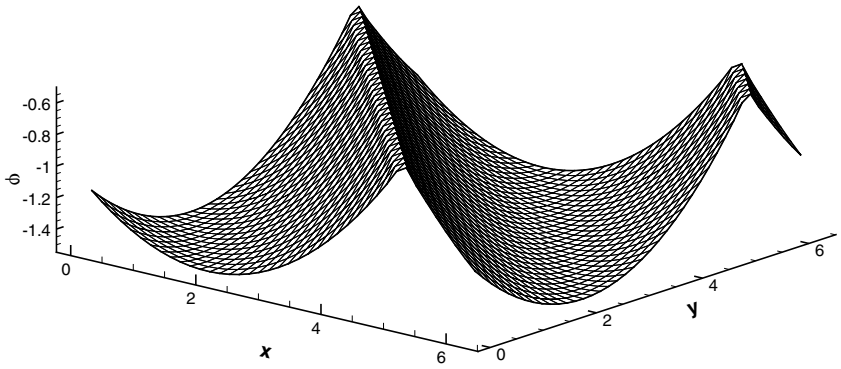

Fig. 4.7. Example 4.3.3. Numerical solution when $t = 1$, CFL $= 0.1$, 40×40 uniform mesh, using P^2 polynomials.

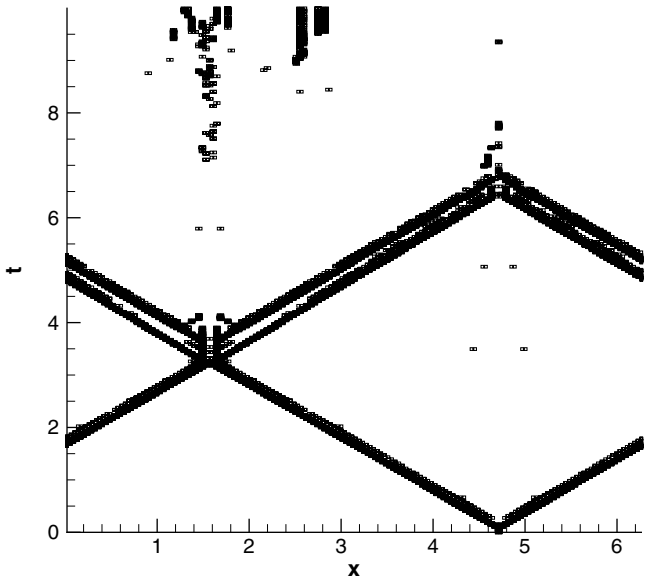

Fig. 4.8. Example 4.4.1. $t = 10$, CFL $= 0.1$, $N = 160$ uniform mesh, using P^2 polynomials. $\varepsilon = 10^{-10}$. Rectangular symbols mark the cells in which the entropy correction is performed. Those cells are plotted every five time steps.

Example 4.4.1. We solve the Eikonal equation given by

$$\begin{cases} \varphi_t + |\varphi_x| = 0 \\ \varphi(x,0) = \sin(x) \\ \varphi(0,t) = \varphi(2\pi,t) \end{cases} \tag{4.24}$$

The exact solution is the same as the exact solution of Example 4.2.1, given by (4.10)–(4.12). Because the entropy condition is violated by our scheme in some cells, we need to apply the entropy correction technique. The criteria are as follows. If we denote $u = \varphi_x$, then for the cell $I_j = (x_{j\text{-}1/2}, x_{j+1/2})$, if

$$u^-(x_{j-1/2}) < -\varepsilon \quad \text{and} \quad \varepsilon < u^+(x_{j-1/2}) \tag{4.25}$$

or

$$u^-(x_{j+1/2}) < -\varepsilon \quad \text{and} \quad \varepsilon < u^+(x_{j+1/2}) \tag{4.26}$$

are satisfied, we use the entropy correction on I_j. We take the parameter $\varepsilon = 10^{-10}$ in the calculation, which is introduced to avoid unnecessary entropy corrections due to small numerical errors in the computation.

Table 4.10
Errors and numerical orders of accuracy for Example 4.4.1 when using P^2 polynomials and Runge–Kutta third order time discretization on a uniform mesh of N cells

N	L^1 error	Order	L^2 error	Order	L^∞ error	Order
40	0.87E−03		0.15E−02		0.28E−02	
80	0.23E−03	1.89	0.41E−03	1.88	0.76E−03	1.89
160	0.64E−04	1.86	0.11E−03	1.86	0.21E−03	1.85
320	0.18E−04	1.85	0.31E−04	1.84	0.59E−04	1.85

Final time $t = 1$. CFL = 0.1.

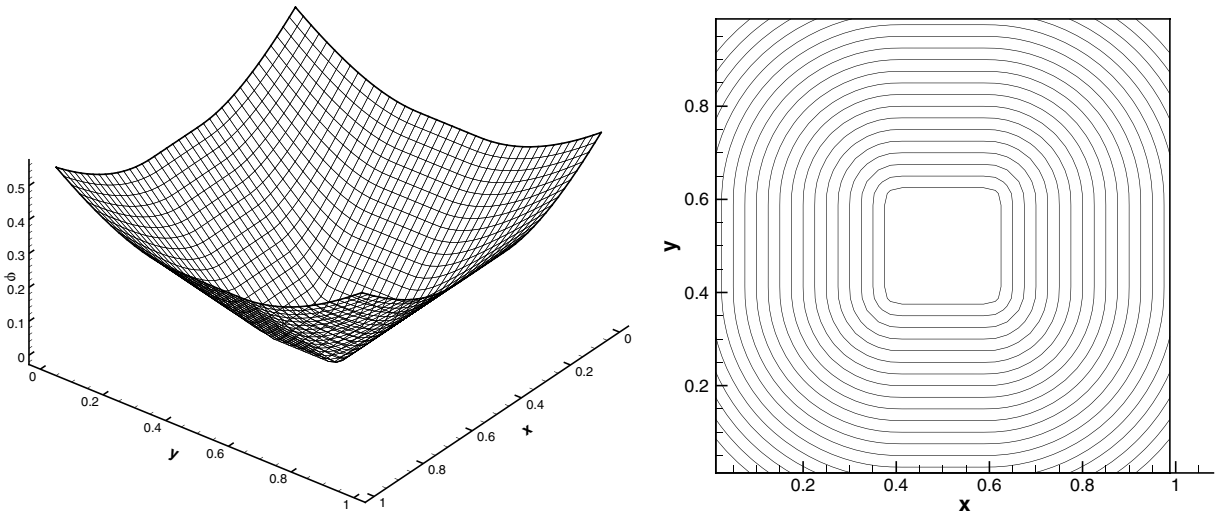

Fig. 4.9. Example 4.4.2. Steady state solution with 40×40 uniform mesh, using P^2 polynomials. Left: three-dimensional plot; right: contour plot.

Fig. 4.8 shows the space–time location where the entropy correction is applied. We observe that the correction is mostly applied at a few cells neighboring the boundary of the rarefaction wave. The number of cells in which the correction is performed is relatively small compared to the total number of cells.

The numerical errors and the order of accuracy are listed in Table 4.10. Since the solution is not smooth, we do not expect the full $(k+1)$th order accuracy.

Example 4.4.2. We solve the two-dimensional Eikonal equation

$$\varphi_t + \sqrt{\varphi_x^2 + \varphi_y^2} = 1 \tag{4.27}$$

First of all, we consider the case of the computational domain being $[0,1]^2 \backslash [0.4,0.6]^2$. For the inner boundary along $[0.4,0.6]^2$, we impose the boundary condition $\varphi = 0$. On the other hand, we impose free outflow boundary conditions on the outer boundary. The initial condition is taken as $\varphi_0(x,y) = \max\{|x-0.5|, |y-0.5|\} - 0.1$. The steady state solution should give us a function that is equal to the distance of the point to the inner boundary. For the outer boundary cells, we use the upwind version of our scheme according to the direction of the local wind. For all other cells, the general scheme (3.5) is used. We plot the numerical steady state solution in Fig. 4.9.

Next, we consider this example with a point source condition; namely, we take the inner boundary to be the center point $(0.5,0.5)$. In this case, we would need φ in the center cell to be the L^2 projection of the exact distance function. For all other cells, the computation is the same as for the previous case. The initial

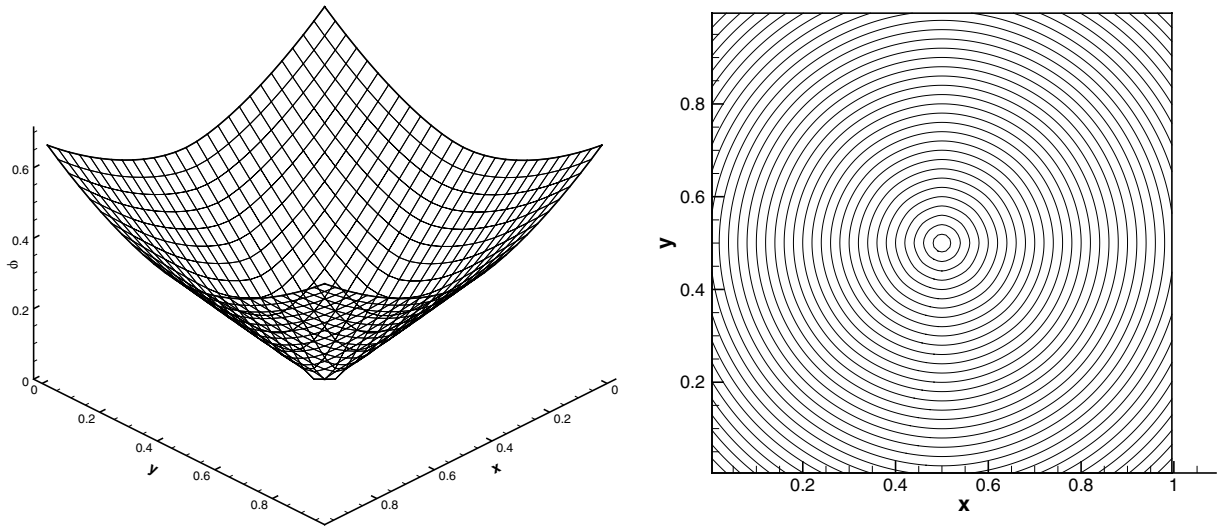

Fig. 4.10. Example 4.4.2. Steady state solution with 39×39 uniform mesh, using P^2 polynomials. Left: three-dimensional plot; right: contour plot.

condition is taken as $\varphi_0(x,y) = \max\{|x - 0.5|,\ |y - 0.5|\}$. We plot the numerical steady state solution in Fig. 4.10. We can see that in both cases we obtain very good resolution to the viscosity solution.

5. Concluding remarks

We have developed a discontinuous Galerkin finite element method for solving Hamilton–Jacobi equations approximating directly the solution variable rather than its derivatives as in the earlier work in [9,14]. Both linear and convex nonlinear Hamiltonians are considered in this paper, while the case for non-convex Hamiltonians is left for future study. One and two-dimensional numerical results demonstrate that the method approximates the viscosity solutions very well. In the future, we will also explore more direct entropy correction techniques without resorting to the techniques in [9,14]. We remark that Osher and Yan [15] has recently developed another class of discontinuous Galerkin type scheme for solving Hamilton–Jacobi equations, which also approximates directly the solution variable rather than its derivatives. A comparison of these two different approaches would be interesting.

Acknowledgments

Research supported by ARO Grant W911NF-04-1-0291, NSF Grant DMS-0510345 and AFOSR Grant FA9550-05-1-0123.

References

[1] B. Cockburn, S. Hou, 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.
[2] B. Cockburn, S.-Y. Lin, 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.
[3] B. Cockburn, C.-W. Shu, The Runge–Kutta local projection P1-discontinuous Galerkin finite element method for scalar conservation laws, Mathematical Modelling and Numerical Analysis 25 (1991) 337–361.
[4] B. Cockburn, 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.
[5] B. Cockburn, C.-W. Shu, The Runge–Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems, Journal of Computational Physics 141 (1998) 199–224.

[6] M. Crandall, P.L. Lions, Viscosity solutions of Hamilton–Jacobi equations, Transactions of the American Mathematical Society 277 (1983) 1–42.
[7] S. Gottlieb, C.-W. Shu, Total variation diminishing Runge–Kutta schemes, Mathematics of Computation 67 (1998) 73–85.
[8] S. Gottlieb, C.-W. Shu, E. Tadmor, Strong stability preserving high order time discretization methods, SIAM Review 43 (2001) 89–112.
[9] C. Hu, C.-W. Shu, A discontinuous Galerkin finite element method for Hamilton–Jacobi equations, SIAM Journal on Scientific Computing 21 (1999) 666–690.
[10] G. Jiang, D. Peng, Weighted ENO schemes for Hamilton–Jacobi equations, SIAM Journal on Scientific Computing 21 (1999) 2126–2143.
[11] G. Jiang, C.-W. Shu, On cell entropy inequality for discontinuous Galerkin methods, Mathematics of Computation 62 (1994) 531–538.
[12] O. Lepsky, C. Hu, C.-W. Shu, The Analysis of the discontinuous Galerkin method for Hamilton–Jacobi equations, Applied Numerical Mathematics 33 (2000) 423–434.
[13] R. LeVeque, High-resolution conservative algorithms for advection in incompressible flow, SIAM Journal on Numerical Analysis 33 (1996) 627–665.
[14] F. Li, C.-W. Shu, Reinterpretation and simplified implementation of a discontinuous Galerkin method for Hamilton–Jacobi equations, Applied Mathematics Letters 18 (2005) 1204–1209.
[15] S. Osher, J. Yan, A new discontinuous Galerkin method for the Hamilton–Jacobi equation, in preparation.
[16] S. Osher, C.-W. Shu, High order essentially non-oscillatory schemes for Hamilton–Jacobi equations, SIAM Journal on Numerical Analysis 28 (1991) 907–922.
[17] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, Journal of Computational Physics 77 (1988) 439–471.
[18] J. Yan, C.-W. Shu, A local discontinuous Galerkin method for KdV type equations, SIAM Journal on Numerical Analysis 40 (2002) 769–791.
[19] Y.-T. Zhang, C.-W. Shu, High order WENO schemes for Hamilton–Jacobi equations on triangular meshes, SIAM Journal on Scientific Computing 24 (2003) 1005–1030.

Journal of Computational Mathematics, Vol.26, No.3, 2008, 1–11.

Numerical Boundary Conditions for the Fast Sweeping High Order WENO Methods for Solving the Eikonal Equation[*] [1)]

Ling Huang
Department of Mathematics, University of Science and Technology of China
Hefei, Anhui 230026, P.R. China
E-mail: huangl0@mail.ustc.edu.cn
Chi-Wang Shu
Division of Applied Mathematics, Brown University, Providence, RI 02912, USA
E-mail: shu@dam.brown.edu
Mengping Zhang
Department of Mathematics, University of Science and Technology of China
Hefei, Anhui 230026, P.R. China
E-mail: mpzhang@ustc.edu.cn

Abstract

High order fast sweeping methods have been developed recently in the literature to solve static Hamilton-Jacobi equations efficiently. Comparing with the first order fast sweeping methods, the high order fast sweeping methods are more accurate, but they often require additional numerical boundary treatment for several grid points near the boundary because of the wider numerical stencil. It is particularly important to treat the points near the inflow boundary accurately, as the information would flow into the computational domain and would affect global accuracy. In the literature, the numerical solution at these boundary points are either fixed with the exact solution, which is not always feasible, or computed with a first order discretization, which could reduce the global accuracy. In this paper, we discuss two strategies to handle the inflow boundary conditions. One is based on the numerical solutions of a first order fast sweeping method with several different mesh sizes near the boundary and a Richardson extrapolation, the other is based on a Lax-Wendroff type procedure to repeatedly utilizing the PDE to write the normal spatial derivatives to the inflow boundary in terms of the tangential derivatives, thereby obtaining high order solution values at the grid points near the inflow boundary. We explore these two approaches using the fast sweeping high order WENO scheme in [18] for solving the static Eikonal equation as a representative example. Numerical examples are given to demonstrate the performance of these two approaches.

Mathematics subject classification: 65N06, 65N22.
Key words: fast sweeping method, WENO scheme, boundary condition.

1. Introduction

In this paper we are interested in the numerical solution of two dimensional static Hamilton-Jacobi equations

$$H(\phi_x, \phi_y) = f(x, y) \tag{1.1}$$

[*] Received ?? / Revised version received ?? / Accepted ?? /
[1)] Research supported by NSFC grant 10671190. Additional support for the second author is provided by NSF grant DMS-0510345.

which is defined on a domain Ω with suitable boundary conditions. Typically, the boundary condition for the solution ϕ is provided in the inflow part Γ of the boundary. In particular, we will only study the so-called Eikonal equation in this paper as an example, that is, the Hamiltonian H in (1.1) is given by

$$H(u, v) = \sqrt{u^2 + v^2}. \tag{1.2}$$

Notice that the solution to (1.1) may not always be differentiable or unique, and we are interested in the viscosity solution [4] which is unique, Lipschitz continuous, but may not be everywhere differentiable.

Applications in which the Hamilton-Jacobi equation (1.1), in particular the Eikonal equation (1.1)-(1.2), appears are abundant, for example the level set method, image processing and computer vision, and control theory. Some of the recently developed pedestrian flow models [7, 16] also involve the static Eikonal equation.

Numerical discretization for (1.1) includes first order monotone schemes on structured meshes [5] and on unstructured meshes [1], high order essentially non-oscillatory (ENO) schemes on structured meshes [11, 12], high order weighted ENO (WENO) schemes on structured meshes [8], high order WENO schemes on unstructured meshes [17], and high order discontinuous Galerkin methods on unstructured meshes [6, 3], among many others. A review of the discretization techniques for the Hamilton-Jacobi equations can be found in [14].

For a time dependent Hamilton-Jacobi equation

$$\phi_t + H(\phi_x, \phi_y) = f(x, y), \tag{1.3}$$

an explicit time discretization, such as the total variation diminishing (TVD) time discretization in [15], is often used. Such discretization can also be used to obtain the steady state solution of (1.1), by marching in time until the difference of the numerical solution between successive time steps becomes negligibly small. This however may not be the most efficient approach to obtain the solution of (1.1). In recent years, the fast sweeping method has been developed as one of the efficient techniques for obtaining the steady state solution of (1.1). The original fast sweeping method [2, 19] is only for first order monotone schemes on structured meshes. For such first order schemes, there is no issue for numerical boundary conditions, since the first order upwind discretization will only need values from the physically given boundary condition on the inflow part of the domain boundary. Later, the fast sweeping method is generalized to some of the high order spatial discretizations. For example, in [18], the fast sweeping method is generalized to the high order WENO scheme of [8]; and in [10], it is generalized to the high order discontinuous Galerkin method of [3]. These high order fast sweeping methods are also used in the pedestrian flow simulations in [7, 16], which require repeated solution of a static Eikonal equation. The high order fast sweeping methods produce much more accurate solutions on coarser meshes when compared with the first order fast sweeping method. However, they do involve an additional difficulty associated with high order spatial discretizations, namely the necessity to treat numerical boundary conditions near the boundary. Our numerical experiments indicate that the main difficulty is near the inflow boundary, as simple extrapolation could take care of the outflow boundary since the information there would flow out of the computational domain. We will use the high order WENO scheme in [18] as a representative example to explain this difficulty. Because of the wider numerical stencil required for the high order WENO interpolation, the high order fast sweeping WENO method needs a suitable numerical boundary treatment for several grid points near the inflow boundary. In [18] and also several other papers

on similar methods [10, 7, 16], the values of the numerical solution at these boundary points are either fixed with the exact solution, which is not always feasible, or computed with a first order discretization, which would reduce the global accuracy.

In this paper, we use the fast sweeping high order WENO scheme in [18] for solving the static Eikonal equation (1.1)-(1.2), as a representative example, to discuss two strategies to handle these numerical boundary conditions. In the first approach we use a first order fast sweeping method to produce numerical solutions with several different mesh sizes near the boundary, then we form a Richardson extrapolation to obtain suitable high order solution values at the grid points near the inflow boundary. This approach usually involves only a small additional computational cost because the numerical solution at the grid points near the inflow boundary can often be obtained with only local sweeping in the first order fast sweeping method. In the second approach we use a Lax-Wendroff type procedure, to repeatedly utilizing the PDE to write the normal spatial derivatives to the inflow boundary in terms of the tangential derivatives, which would then be readily available by the physical inflow boundary condition. With these normal spatial derivatives we can then obtain high order solution values at the grid points near the inflow boundary. This approach, when applicable, involves a negligibly small additional computational cost.

The rest of the paper is organized as follows. We describe the two approaches for the numerical boundary conditions in Section 2. In Section 3 we provide several numerical examples to demonstrate the performance of these two approaches. Concluding remarks are given in Section 4.

2. The numerical scheme and the treatment of boundary conditions

In this section we first give a brief description of the third order fast sweeping WENO scheme [18] for solving the static Eikonal equation (1.1)-(1.2). We then describe two approaches for the numerical boundary conditions.

2.1. The third order fast sweeping WENO scheme for the Eikonal equation

We give a very brief description of the third order fast sweeping WENO scheme [18] for solving the static Eikonal equation (1.1)-(1.2). For more details, we refer to [18, 8].

For simplicity, we assume that the computational domain Ω is a box $[0,1]^2$ which is covered by a tensor product mesh (x_i, y_j) with $0 \le i \le I$ and $0 \le j \le J$. We assume without loss of generality that the mesh is uniform, $x_i = i\Delta x$, $y_j = j\Delta y$ and $\Delta x = \Delta y = h$. The approximation of the solution to the static Eikonal equation (1.1)-(1.2) at the location (x_i, y_j) is denoted by $\phi_{i,j}$, which is obtained by a fast sweeping iterative procedure. For the convenience of the algorithm description below, we divide the set of mesh points (x_i, y_j) into the following four categories:

- *Category I* contains the points at the inflow part of the domain boundary. The numerical solution $\phi_{i,j}$ in Category I is fixed at the prescribed physical boundary condition and does not change during the fast sweeping iteration.

- *Category II* contains the points at the outflow part of the domain boundary, where no physical boundary condition is given, and the ghost points outside the computational

domain near the outflow boundary which are necessary for the wide stencil WENO interpolation. The numerical solution $\phi_{i,j}$ in Category II is obtained by extrapolation of suitable accuracy, based on the numerical solution inside the computational domain.

- *Category III* contains the few points inside the computational domain and near the inflow boundary. These points cannot be updated by the WENO scheme because of its wide stencil. For the third order WENO scheme under consideration, any point which has a horizontal or vertical distance less than $3h$ from the inflow boundary belongs to this category. The strategy in [18, 10] to treat points in Category III is to fix the numerical solution $\phi_{i,j}$ as the exact solution of the PDE (1.1)-(1.2) and it does not change during the fast sweeping iteration. This of course is not always feasible. The strategy in [7, 16] is to fix the numerical solution $\phi_{i,j}$ as that of a first order fast sweeping solution and it does not change during the fast sweeping iteration. This could of course lead to a loss of local and hence global accuracy, since information will flow from this part of the boundary into the computational domain. The main purpose of this paper is to study two strategies to handle the points in Category III.

- *Category IV* contains all the remaining points, which are updated during the fast sweeping iterations until convergence.

The solution from the first-order Godunov fast sweeping method [19] is used as the initial guess for all the grid points in Category IV. Grid values in Categories I and III are fixed as appropriate, and before each iteration, grid values in Category II are obtained by suitable extrapolation.

The following Gauss-Seidel iterations with four alternating direction sweepings are then performed:

$$\begin{array}{ll} (1) \quad i=0:I, j=0:J; & (2) \quad i=I:0, j=0:J; \\ (3) \quad i=I:0, j=J:0; & (4) \quad i=0:I, j=J:0. \end{array} \tag{2.1}$$

When we loop to a point (i,j), if it belongs to Category IV, the solution is updated as follows,

$$\phi_{i,j}^{new} = \begin{cases} \min(\phi_{i,j}^{xmin}, \phi_{i,j}^{ymin}) + f_{i,j}\, h, & \text{if } |\phi_{i,j}^{xmin} - \phi_{i,j}^{ymin}| \le f_{i,j}\, h, \\ \frac{\phi_{i,j}^{xmin} + \phi_{i,j}^{ymin} + \left(2 f_{i,j}^2 h^2 - (\phi_{i,j}^{xmin} - \phi_{i,j}^{ymin})^2\right)^{\frac{1}{2}}}{2}, & \text{otherwise} \end{cases} \tag{2.2}$$

where $f_{i,j} = f(x_i, y_j)$, and

$$\begin{cases} \phi_{i,j}^{xmin} = & \min\left(\phi_{i,j} - h\,(\phi_x)_{i,j}^{-},\ \phi_{i,j} + h\,(\phi_x)_{i,j}^{+}\right) \\ \phi_{i,j}^{ymin} = & \min\left(\phi_{i,j} - h\,(\phi_y)_{i,j}^{-},\ \phi_{i,j} + h\,(\phi_y)_{i,j}^{+}\right) \end{cases}, \tag{2.3}$$

with

$$(\phi_x)_{i,j}^{-} = (1-w_-)\left(\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2h}\right) + w_-\left(\frac{3\phi_{i,j} - 4\phi_{i-1,j} + \phi_{i-2,j}}{2h}\right), \tag{2.4}$$

$$(\phi_x)_{i,j}^{+} = (1-w_+)\left(\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2h}\right) + w_+\left(\frac{-3\phi_{i,j} + 4\phi_{i+1,j} - \phi_{i+2,j}}{2h}\right), \tag{2.5}$$

$$w_- = \frac{1}{1+2r_-^2}, \qquad r_- = \frac{\varepsilon + (\phi_{i,j} - 2\phi_{i-1,j} + \phi_{i-2,j})^2}{\varepsilon + (\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j})^2}, \tag{2.6}$$

$$w_+ = \frac{1}{1+2r_+^2}, \qquad r_+ = \frac{\varepsilon + (\phi_{i,j} - 2\phi_{i+1,j} + \phi_{i+2,j})^2}{\varepsilon + (\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j})^2}. \tag{2.7}$$

Here ε is a small number in the WENO nonlinear weights. The definitions for $(\phi_y)^-_{i,j}$ and $(\phi_y)^+_{i,j}$ are of course analogous.

Convergence is declared if

$$\left\|\phi^{new} - \phi^{old}\right\| \le \delta, \tag{2.8}$$

where δ is a given convergence threshold value.

2.2. Boundary treatment strategy I: Richardson extrapolation

The first strategy that we propose to treat points in Category III is to obtain several first order accurate solutions with different mesh sizes, then use Richardson extrapolation to obtain accurate point values for those points in Category III. This is feasible without excessive computational cost because points in Category III are close to the inflow boundary, hence the first order fast sweeping iterations can be performed locally, greatly reducing the computational cost.

Richardson extrapolation is a well-known idea so we will only describe our application of this idea briefly. Assume I_h is the numerical solution of the first order fast sweeping scheme with mesh size h at the location (x^*, y^*), which is a grid point in Category III. If we further assume

$$I_h - I = \alpha h + \beta h^2 + O(h^3)$$

with constants α and β, where I is the exact solution at the location (x^*, y^*), which is reasonable when the exact solution is smooth, then clearly

$$\tilde{I}_h = \frac{1}{3} I_h - 2 I_{h/2} + \frac{8}{3} I_{h/4} \tag{2.9}$$

would be a third order approximation to I:

$$\tilde{I}_h - I = O(h^3).$$

This boundary treatment strategy is suitable for most types of inflow boundaries, including the source boundary consisting of a single point. The efficiency of this strategy however depends on how fast we can compute the first order approximations I_h, $I_{h/2}$ and $I_{h/4}$ for all grid points inside Category III. When the characteristics from the inflow boundary do not intersect with each other, such first order fast sweeping computation can be performed locally and is very fast. When the characteristics from the inflow boundary do intersect with each other, the efficiency of this strategy would decrease. Fortunately, in this case the inflow boundary would not be a single point, hence the second strategy described in next subsection would usually be applicable.

2.3. Boundary treatment strategy II: a Lax-Wendroff type procedure

The original Lax-Wendroff scheme [9] uses an important idea of converting the time derivatives to spatial derivatives, by repeatedly using the PDE. We propose to use the same idea to obtain high order approximations to the solution values for the points in Category III.

To fix the ideas, let us assume that the left boundary

$$\Gamma = \{(x, y) : x = 0,\ 0 \le y \le 1\} \tag{2.10}$$

of the computational domain $[0, 1]^2$ is the inflow boundary, on which the solution is given as

$$\phi(0, y) = g(y), \qquad 0 \le y \le 1.$$

We would like to obtain a high order approximation to the solution value $\phi_{i,j} \approx \phi(x_i, y_j)$ for $i = 1, 2$ and a fixed j, which corresponds to a point (x_i, y_j) in Category III. A simple Taylor expansion gives, for $i = 1, 2$,

$$\phi(x_i, y_j) = \phi(0, y_j) + ih\,\phi_x(0, y_j) + \frac{(ih)^2}{2}\phi_{xx}(0, y_j) + O(h^3)$$

hence our desired approximation for the third order WENO scheme is

$$\phi_{i,j} = \phi(0, y_j) + ih\,\phi_x(0, y_j) + \frac{(ih)^2}{2}\phi_{xx}(0, y_j).$$

We already have $\phi(0, y_j) = g(y_j)$. The PDE (1.1), evaluated at the point $(0, y_j)$, becomes

$$H(\phi_x(0, y_j), g'(y_j)) = f(0, y_j) \tag{2.11}$$

in which the only unknown quantity is $\phi_x(0, y_j)$. Solving this (usually nonlinear) equation should give us $\phi_x(0, y_j)$. There might be more than one root, in which case we should choose the root so that

$$\partial_u H(\phi_x(0, y_j), g'(y_j)) > 0 \tag{2.12}$$

where ∂_u refers to the partial derivative with respect to the first argument in $H(u, v)$. The condition (2.12) guarantees that the boundary Γ in (2.10) is an inflow boundary. If the condition (2.12) still cannot pin down a root, then we would choose the root which is closest to the value from the first order fast sweeping solution at the same grid point. To obtain $\phi_{xx}(0, y_j)$, we first take the derivative with respect to y on the original PDE (1.1), and then evaluate it at the the point $(0, y_j)$, which yields

$$\partial_u H(\phi_x(0, y_j), g'(y_j))\phi_{xy}(0, y_j) + \partial_v H(\phi_x(0, y_j), g'(y_j))g''(y_j) = f_y(0, y_j), \tag{2.13}$$

where ∂_u and ∂_v refer to the partial derivatives with respect to the first and second arguments in $H(u, v)$, respectively. In this equation the only unknown quantity is $\phi_{xy}(0, y_j)$, hence we obtain easily its value, thanks to (2.12). We then take the derivative with respect to x on the original PDE (1.1), and evaluate it at the the point $(0, y_j)$ to obtain

$$\partial_u H(\phi_x(0, y_j), g'(y_j))\phi_{xx}(0, y_j) + \partial_v H(\phi_x(0, y_j), g'(y_j))\phi_{xy}(0, y_j) = f_x(0, y_j),$$

This time, the only unknown quantity is $\phi_{xx}(0, y_j)$, which we can obtain readily from this equality.

It is clear that this procedure can be carried out to any desired order of accuracy. Also, the inflow boundary Γ in (2.10) can be any piece of a smooth curve: we only need to change the x and y partial derivatives to normal and tangential derivatives with respect to Γ. However, for this approach to work, Γ can not consist of a single point.

3. Numerical examples

In this section, we demonstrate the performance of the two approaches for treating the inflow boundary conditions described in Section 2 through a few two dimensional numerical examples. The third order fast sweeping WENO method [18], outlined in Section 2.1, is used. In our computation, the threshold value at which iteration stops is taken to be $\delta = 10^{-11}$. The small number in the WENO nonlinear weights ε is taken as 10^{-6} unless otherwise stated.

Table 3.1: Example 1. Richardson extrapolation for the inflow boundary. N is the number of mesh points in each direction.

N	L^1 error	order	L^∞	order	iteration number
40	8.00E-04		6.36E-04		40
80	5.92E-05	3.76	3.33E-05	4.25	30
160	3.64E-06	4.02	1.54E-06	4.44	38
320	4.00E-07	3.19	1.58E-07	3.28	50
640	4.98E-08	3.00	2.02E-08	2.97	81

Example 1.

We solve the Eikonal equation (1.1)-(1.2) with

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

The inflow boundary Γ is the single point (0,0). The computational domain is $[-1,1]^2$. The exact solution for this problem is

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

Since the inflow boundary Γ consists of a single point, the second strategy described in Section 2.3 does not apply. We use the first strategy described in Section 2.2 to handle the inflow boundary condition. Namely, in the small box $[-2h,\, 2h]^2$, we apply the first order fast sweeping method [19] with three different mesh sizes h, $h/2$ and $h/4$, and then use the Richardson extrapolation (2.9) to obtain a third order approximation to the grid values $\phi_{i,j}$ in this small box, which are then fixed as the initial values during the third order fast sweeping WENO process. Notice that this process has very little computational cost since the box $[-2h,\, 2h]^2$ is very small. For the outflow boundary, which is the boundary of the box $[-1,1]^2$, we take a simple third order extrapolation to provide solution values in the ghost points outside the computational domain. The results are given in Table 3.1. We can see clearly that the scheme with the numerical boundary treatment gives the correct order of accuracy.

Example 2 (shape-from-shading).

We solve the Eikonal equation (1.1)-(1.2) with

$$\text{case (a):} \quad f(x,y) = \sqrt{(1-|x|)^2 + (1-|y|)^2}; \tag{3.1}$$

$$\text{case (b):} \quad f(x,y) = 2\sqrt{y^2(1-x^2)^2 + x^2(1-y^2)^2}. \tag{3.2}$$

The computational domain $\Omega = [-1,1]^2$. The inflow boundary for this example is the whole boundary of the box $[-1,1]^2$, namely $\Gamma = \{(x,y) :\ |x| = 1 \text{ or } |y| = 1\}$. The boundary condition $\phi(x,y) = 0$ is prescribed on Γ. For case (b), an additional boundary condition $\phi(0,0) = 1$ is also prescribed at the center, see [13]. The exact solutions for these two cases are given by

$$\text{case (a):} \quad \phi(x,y) = (1-|x|)(1-|y|); \tag{3.3}$$

$$\text{case (b):} \quad \phi(x,y) = (1-x^2)(1-y^2). \tag{3.4}$$

Table 3.2: Example 2. Richardson extrapolation for the inflow boundary. N is the number of mesh points in each direction.

N	L^1 error	order	iter	L^1 error	order	iter
	case (a)			case (b)		
80	3.45E-06		21	3.38E-05		35
160	3.04E-07	3.51	28	3.44E-06	3.30	50
320	2.58E-08	3.56	46	2.23E-07	3.95	81
640	2.14E-09	3.59	81	9.25E-09	4.59	149

For this example only, we set the parameter ε in the nonlinear WENO weights (2.6)-(2.7) as $\varepsilon = 10^{-6}h^2$. This smaller choice of ε seems to make the adjustment of the nonlinear weights better near the center singularity of the solution, especially for case (b).

For this example, we can apply both of the strategies in Sections 2.2 and 2.3. We first apply the Richardson extrapolation strategy of Section 2.2 for the points inside the computational domain which are of distance at most $2h$ away from the inflow boundary Γ, using the results of the first order fast sweeping method with three different mesh sizes h, $h/2$ and $h/4$. For case (b), even though an additional point $\phi(0,0) = 1$ is prescribed at the center, we do not take any special treatment for points near the center. The results are given in Table 3.2. We can again see clearly that the scheme with this numerical boundary treatment gives the correct order of accuracy.

Next, we apply the Lax-Wendroff type procedure of Section 2.2 to obtain third order approximations to the values of the numerical solution corresponding to the points inside the computational domain which are of distance at most $2h$ away from the inflow boundary Γ. Again, for case (b), no special treatment has been done for points near the center. The results are given in Table 3.3. This time, since the solution is a polynomial of degree lower than the order of the scheme, we are able to obtain the exact solution with only round-off errors, as the Lax-Wendroff type procedure of Section 2.2 is able to prescribe the values to the points inside the computational domain which are of distance at most $2h$ away from the inflow boundary Γ *exactly* by Taylor expansion[1)] . We remark that the first strategy of using the Richardson extrapolation in Section 2.2 is *not* able to provide the solution values to the points inside the computational domain which are of distance at most $2h$ away from the inflow boundary Γ exactly, but only to the designed third order accuracy, hence the final fast sweeping WENO results in Table 3.2 are also only high order accurate but not exact to round-off errors.

Example 3.

We solve the Eikonal equation (1.1)-(1.2) with $f(x,y) = 1$. The computational domain is $[-1,1]^2$, and the inflow boundary Γ is the unit circle of center (0,0) and radius 0.5, that is

$$\Gamma = \left\{ (x,y) : \ x^2 + y^2 = \frac{1}{4} \right\}.$$

[1)] We do note that for the coarser meshes in case (b) the solution has not settled to round-off errors. Our experiments show that this is related to the treatment of the center point: if we fix the numerical solution in the small box $[-2h, 2h]^2$ rather than just at the center point (0,0) by the exact solution, round-off error can be achieved. We will not further explore this difficulty as it is not related to the theme of this paper.

Table 3.3: Example 2. Lax-Wendroff type procedure for the inflow boundary. N is the number of mesh points in each direction.

N	L^1 error	L^∞	iter	L^1 error	L^∞	iter
	case (a)			case (b)		
80	2.06E-14	9.26E-13	1	8.64E-07	9.99E-04	35
160	1.58E-14	2.34E-12	1	5.25E-08	2.34E-04	47
320	1.09E-14	5.49E-12	1	4.43E-15	4.26E-12	62
640	1.01E-14	1.08E-11	2	1.41E-15	1.14E-12	90

Table 3.4: Example 3. Lax-Wendroff type procedure for the inflow boundary. N is the number of mesh points in each direction. The errors are measured in the computational domain but outside the box $[-0.15, 0.15]^2$.

N	L^1 error	order	L^∞	order	iteration number
80	0.573E-05		0.129E-03		25
160	0.122E-05	2.23	0.407E-05	4.98	32
320	0.191E-06	2.68	0.122E-05	1.74	46
640	0.246E-07	2.95	0.161E-06	2.92	62

The boundary condition $\phi(x, y) = 0$ is prescribed on Γ. The exact solution for this problem is the distance function to the circle Γ. This exact solution has a singularity at the center of the circle to which the characteristics converge, hence we exclude the box $[-0.15, 0.15]^2$ when measuring the errors. For this problem, it is not easy to apply the Richardson extrapolation strategy in Section 2.2, since we use rectangular meshes and the inflow boundary Γ is not on grid points. However, the Lax-Wendroff type procedure in Section 2.3 can be easily used to obtain the values of the numerical solution corresponding to the points inside the computational domain which have a horizontal or vertical distance less than $3h$ from the inflow boundary Γ. For the outflow boundary, which is the boundary of the box $[-1, 1]^2$, we take a simple third order extrapolation to provide solution values in the ghost points outside the computational domain. The results are given in Table 3.4. We can see happily again that the scheme with this numerical boundary treatment gives the correct order of accuracy away from the singularity at the center.

Example 4.

We solve the Eikonal equation (1.1)-(1.2) with $f(x, y) = 1$. The computational domain is $[-3, 3]^2$, and the inflow boundary Γ consists of two circles of equal radius 0.5 with centers located at $(-1, 0)$ and $(\sqrt{1.5}, 0)$, respectively, that is

$$\Gamma = \left\{ (x, y) : \ (x+1)^2 + y^2 = \frac{1}{4} \text{ or } (x - \sqrt{1.5})^2 + y^2 = \frac{1}{4} \right\}.$$

The boundary condition $\phi(x, y) = 0$ is prescribed on Γ. The exact solution for this problem is the distance function to Γ. The exact solution for this problem is the distance function to the circle Γ. This exact solution has singularities at the centers of the circles and on the line that has the same distance to the two circles, on which the characteristics converge, hence we exclude the boxes $[-1.15, -0.85] \times [-0.15, 0.15]$, $[\sqrt{1.5} - 0.15, \sqrt{1.5} + 0.15] \times [-0.15, 0.15]$ and

Table 3.5: Example 4. Lax-Wendroff type procedure for the inflow boundary. N is the number of mesh points in each direction. The errors are measured in the computational domain but outside the boxes $[-1.15,\ -0.85]\times[-0.15,\ 0.15]$, $[\sqrt{1.5}-0.15,\ \sqrt{1.5}+0.15]\times[-0.15, 0.15]$ and $[\sqrt{0.375}-0.65,\ \sqrt{0.375}-0.35]\times[-3,\ 3]$.

N	L^1 error	order	L^∞	order	iteration number
80	0.569E-02		0.274E-02		38
160	0.346E-03	4.04	0.766E-03	1.84	47
320	0.240E-04	3.85	0.294E-04	4.71	47
640	0.470E-05	2.35	0.336E-05	3.13	67

$[\sqrt{0.375}-0.65,\ \sqrt{0.375}-0.35]\times[-3,\ 3]$ when measuring the errors. Again, for this problem, it is not easy to apply the Richardson extrapolation strategy in Section 2.2, since we use rectangular meshes and the inflow boundary Γ is not on grid points. However, the Lax-Wendroff type procedure in Section 2.3 can again be easily used to obtain the values of the numerical solution corresponding to the points inside the computational domain which have a horizontal or vertical distance less than $3h$ from the inflow boundary Γ. For the outflow boundary, which is the boundary of the box $[-3,3]^2$, we take a simple third order extrapolation to provide solution values in the ghost points outside the computational domain. The results are given in Table 3.5. We can see happily again that the scheme with this numerical boundary treatment gives the correct order of accuracy away from the singularities.

4. Concluding remarks

In this paper we have discussed two strategies to handle the inflow boundary conditions for high order fast sweeping methods for solving static Hamilton-Jacobi equations. The first method is based on Richardson extrapolation and a local application of a first order fast sweeping method with several different mesh sizes. The second method is based on a Lax-Wendroff type procedure to use repeatedly the PDE to obtain a high order Taylor expansion solution for grid points near the inflow boundary. Numerical examples are provided to demonstrate that both strategies work well and can provide the designed high order accuracy. The second method involves smaller extra computational cost and usually gives more accurate results, hence it is preferred if the inflow boundary allows (i.e. if the inflow boundary is not a single point or a set of isolated points). Even though the boundary treatments are only discussed in the context of the fast sweeping method, they (especially the second approach relying on the Lax-Wendroff-type procedure) are actually quite general, and can be applied to various high order numerical schemes for solving both static and time dependent PDEs. We will explore these issues in more detail in future work.

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] 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.

[3] 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.

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

[5] M. Crandall and P.L. Lions, Monotone difference approximations for scalar conservation laws, *Mathematics of Computation,* **34** (1984), 1-19.

[6] C. Hu and C.-W. Shu, A discontinuous Galerkin finite element method for Hamilton-Jacobi equations, *SIAM Journal on Scientific Computing,* **21** (1999), 666-690.

[7] L. Huang, S.C. Wong, M. Zhang, C.-W. Shu and W.H.K. Lam, Revisiting Hughes' dynamic continuum model for pedestrian flow and the development of an efficient solution algorithm, *Transportation Research Part B,* submitted.

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

[9] P.D. Lax and B. Wendroff, Systems of conservation laws, *Communications in Pure and Applied Mathematics,* **13** (1960), 217-237.

[10] F. Li, C.-W. Shu, Y.-T. Zhang and H. Zhao, A second order discontinuous Galerkin fast sweeping method for Eikonal equations, *Journal of Computational Physics,* submitted.

[11] 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.

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

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

[14] C.-W. Shu, High order numerical methods for time dependent Hamilton-Jacobi equations, in *Mathematics and Computation in Imaging Science and Information Processing,* S.S. Goh, A. Ron and Z. Shen, Editors, Lecture Notes Series, Institute for Mathematical Sciences, National University of Singapore, volume 11, World Scientific Press, Singapore, 2007, pp.47-91.

[15] C.-W. Shu and S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, *Journal of Computational Physics,* **77** (1988), 439-471.

[16] Y. Xia, S.C. Wong, M. Zhang, C.-W. Shu and W.H.K. Lam, An efficient discontinuous Galerkin method on triangular meshes for a pedestrian flow model, *International Journal for Numerical Methods in Engineering,* to appear.

[17] 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.

[18] 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.

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