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.

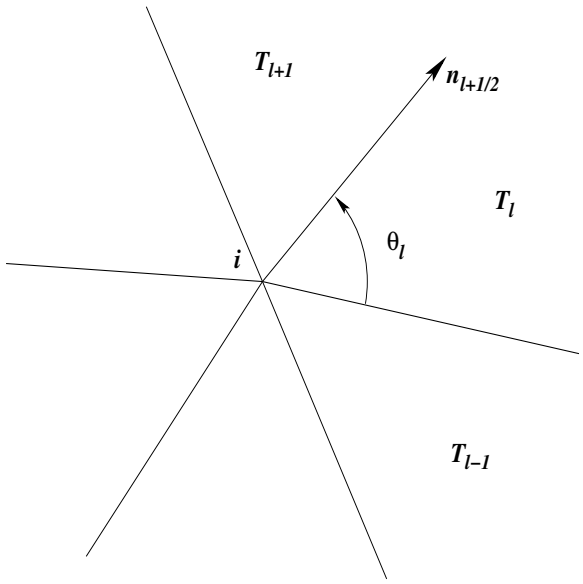

Fig. 1. Node i and its angular sectors.

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

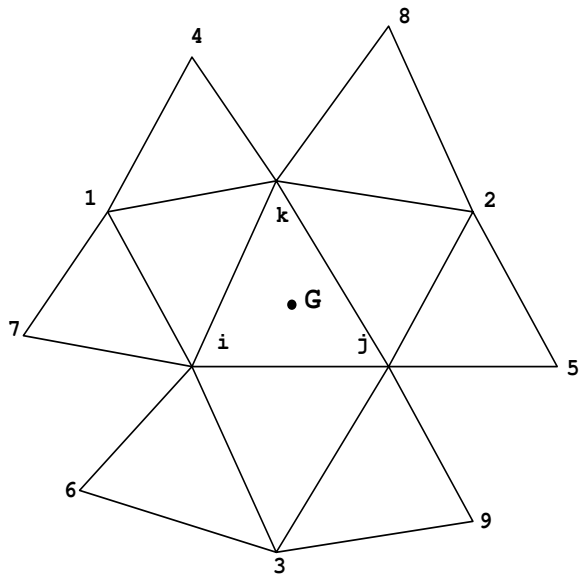

Fig. 2. The nodes used for the big stencil of the third-order scheme.

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

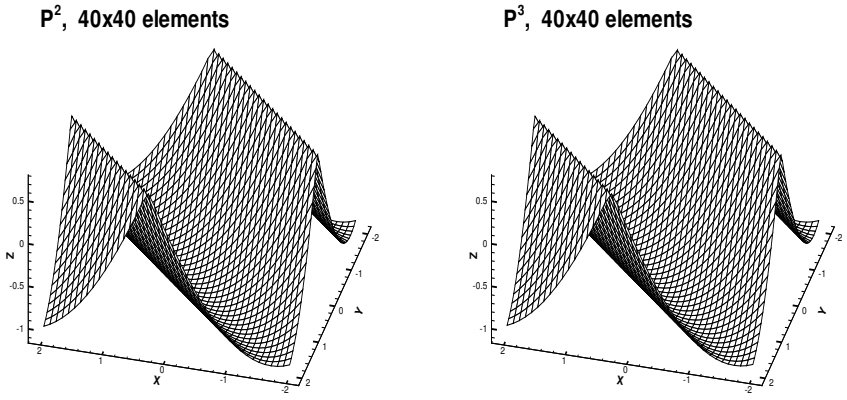

Fig. 3. Two dimension Burgers' equation, rectangular mesh, t=$1.5/\pi^2$.

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

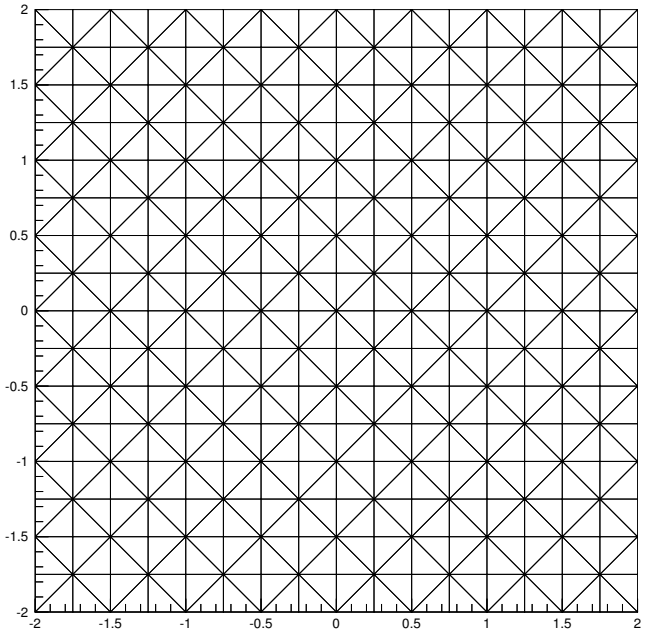

Fig. 4. Triangulation for two dimensional Burgers equation, $h = \frac{1}{4}$.

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

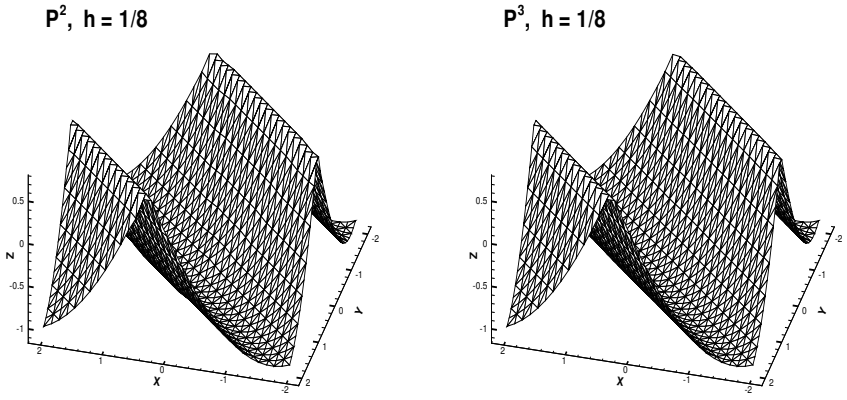

Fig. 5. Two dimension Burgers' equation, triangular mesh, t=$1.5/\pi^2$.

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

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

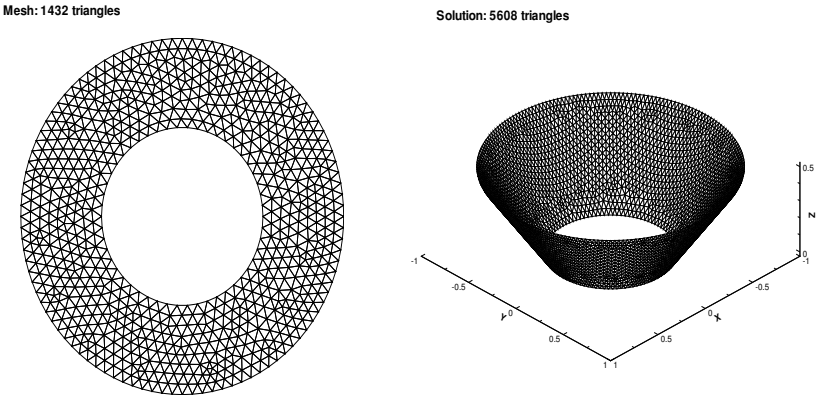

Fig. 6. The level set equation, P^2.

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

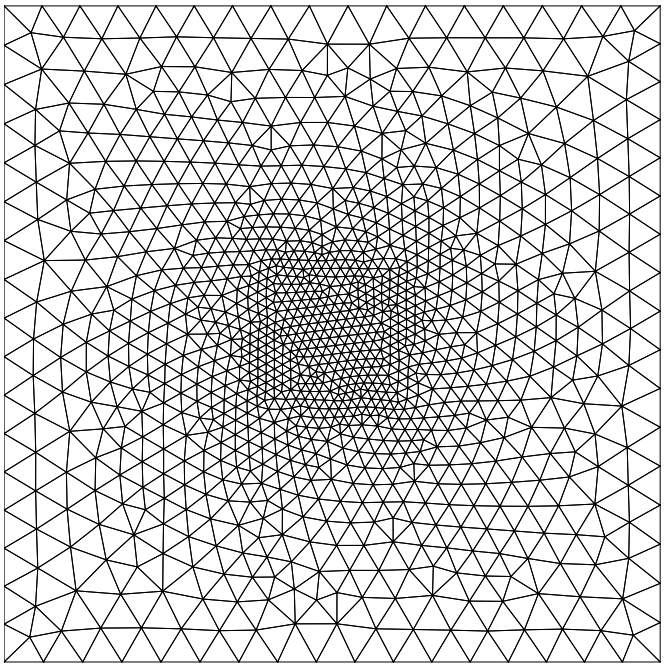

Fig. 7. Triangulation used for the propagating surfaces.

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.

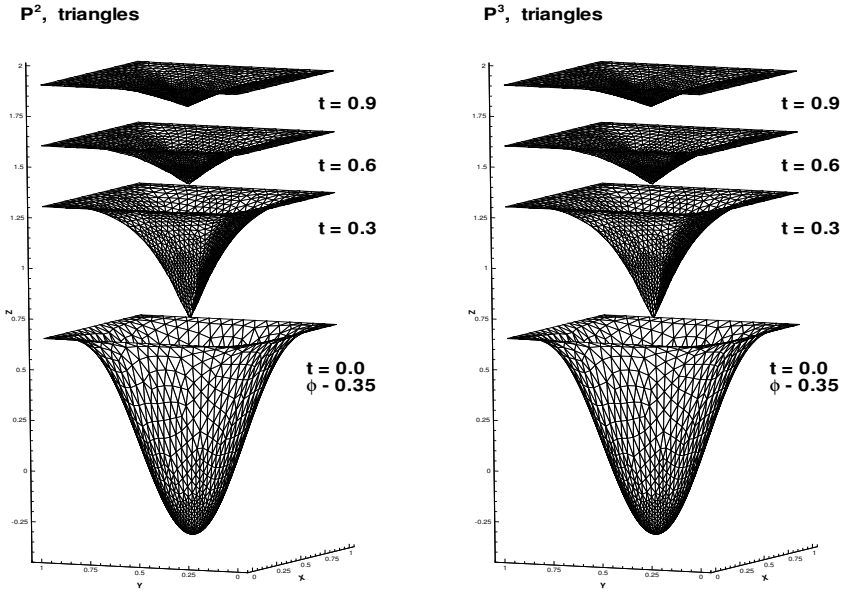

Fig. 8. Propagating surfaces, triangular mesh, $\varepsilon = 0$.

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]:

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

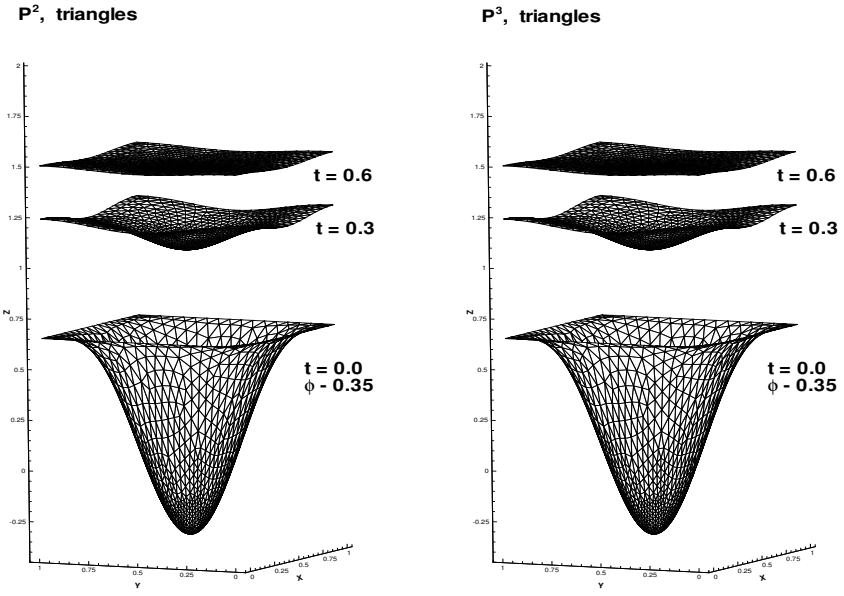

Fig. 9. Propagating surfaces, triangular mesh, $\varepsilon = 0.1$.

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

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

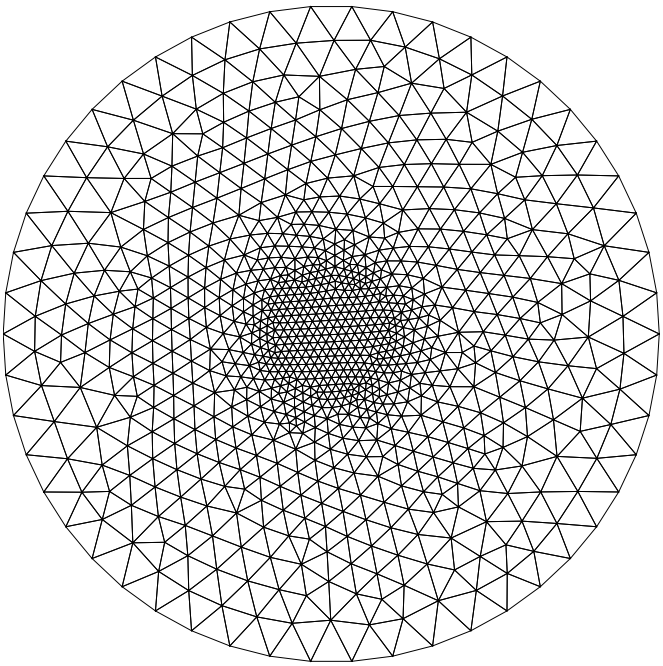

Fig. 10. Triangulation for the propagating surfaces on a disk.

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.

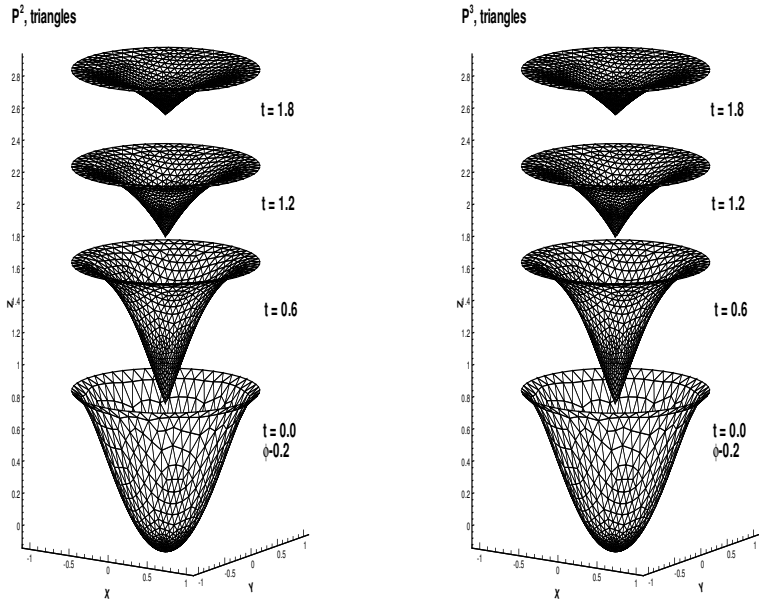

Fig. 11. Propagating surfaces on a disk, triangular mesh, $\varepsilon = 0$.

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.

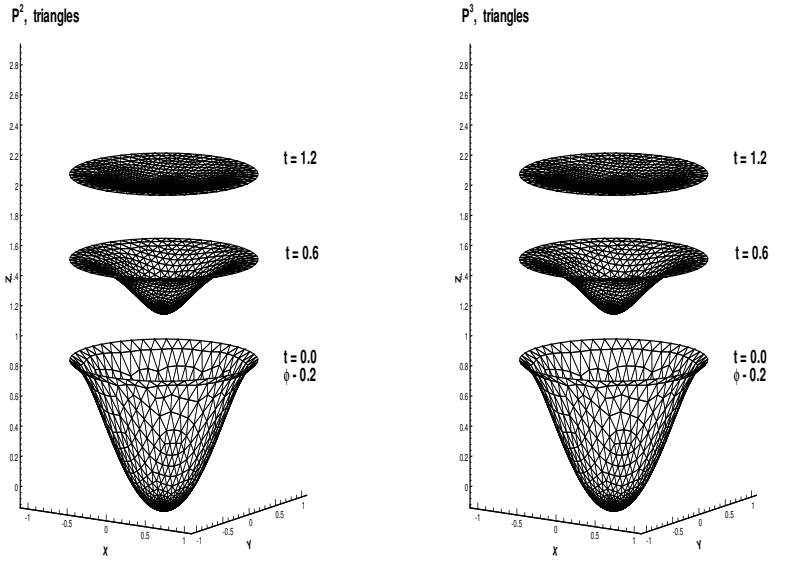

Fig. 12. Propagating surfaces on a disk, triangular mesh, $\varepsilon = 0.1$.

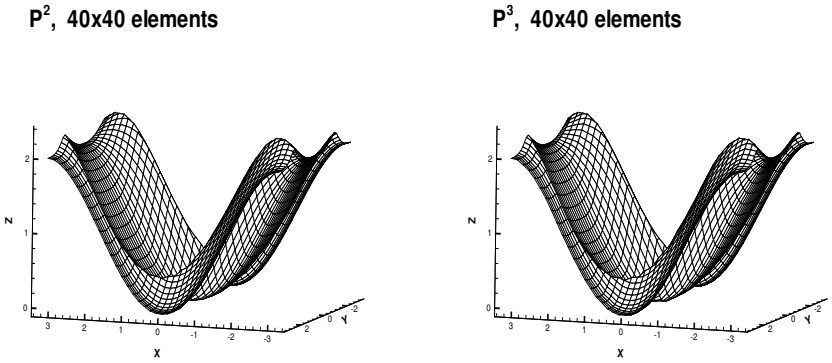

Fig. 13. Control problem, $t = 1$.

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.

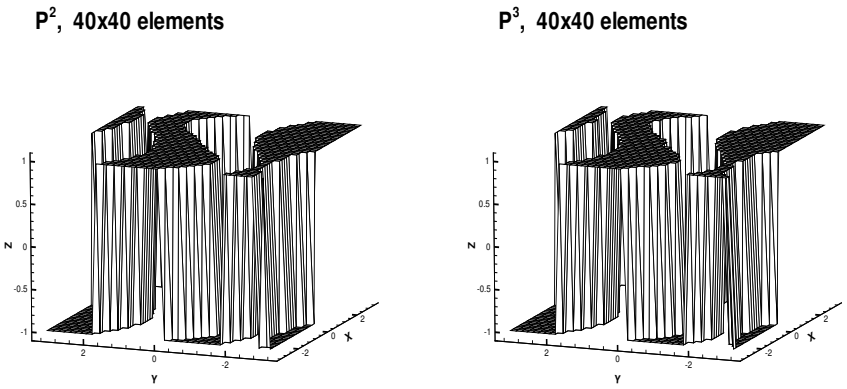

Fig. 14. Control problem, $t = 1, w = \text{sign}(\varphi_y)$.

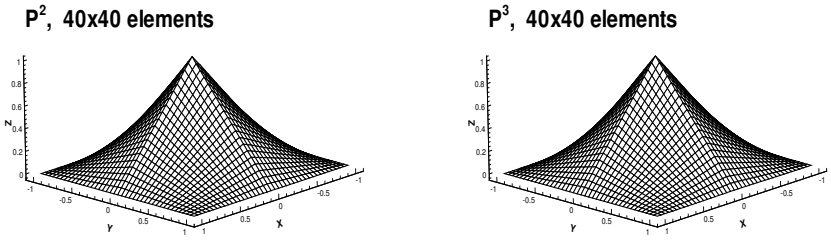

Fig. 15. Computer vision problem, $\varphi(x, y, \infty) = (1 - |x|)(1 - |y|)$.

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.

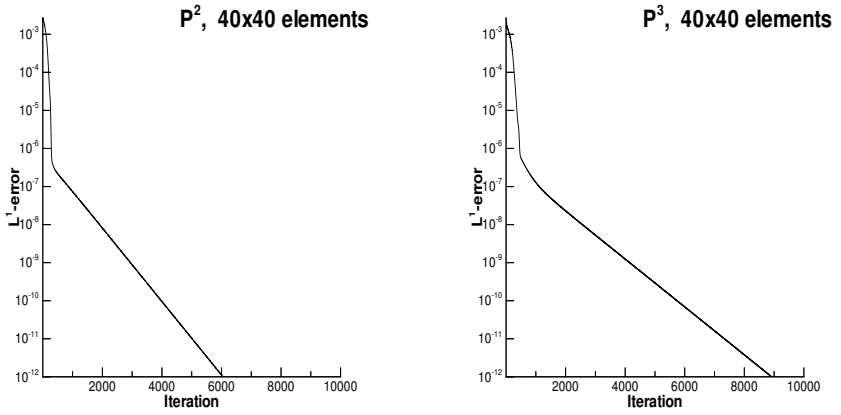

Fig. 16. Computer vision problem, history of iterations.

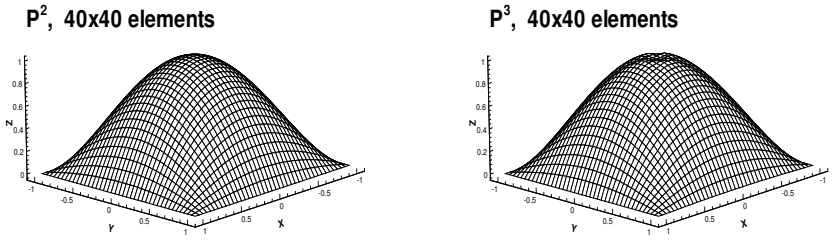

Fig. 17. Computer vision problem, $\varphi(x, y, \infty) = (1 - x^2)(1 - y^2)$.

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.