Numer. Math. Theor. Meth. Appl.
doi: 10.4208/nmtma.2013.1235nm

Vol. **6**, No. 4, pp. 617-636
November 2013

An h-Adaptive Runge-Kutta Discontinuous Galerkin Method for Hamilton-Jacobi Equations

Hongqiang Zhu[1] and Jianxian Qiu[2,*]

[1] *School of Natural Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, Jiangsu, China.*
[2] *School of Mathematical Sciences, Xiamen University, Xiamen 361005, Fujian, China.*

Received 2 November 2012; Accepted (in revised version) 26 February 2013

Available online 2 October 2013

Abstract. In [35,36], we presented an h-adaptive Runge-Kutta discontinuous Galerkin method using troubled-cell indicators for solving hyperbolic conservation laws. A tree data structure (binary tree in one dimension and quadtree in two dimensions) is used to aid storage and neighbor finding. Mesh adaptation is achieved by refining the troubled cells and coarsening the untroubled "children". Extensive numerical tests indicate that the proposed h-adaptive method is capable of saving the computational cost and enhancing the resolution near the discontinuities. In this paper, we apply this h-adaptive method to solve Hamilton-Jacobi equations, with an objective of enhancing the resolution near the discontinuities of the solution derivatives. One- and two-dimensional numerical examples are shown to illustrate the capability of the method.

AMS subject classifications: 65M60, 65M99, 35L65

Key words: Runge-Kutta discontinuous Galerkin method, h-adaptive method, Hamilton-Jacobi equation.

1. Introduction

In this paper, we present an h-adaptive Runge-Kutta discontinuous Galerkin (RKDG) method for solving Hamilton-Jacobi (H-J) equations

$$\begin{cases} \phi_t + H(\nabla_x \phi) = 0, \\ \phi(x,0) = \phi_0(x), \end{cases} \tag{1.1}$$

where $x = (x_1, \cdots, x_d) \in \mathbf{R}^d$, $t > 0$. H-J equations are of practical importance with applications ranging from optimal control and differential games to geometric optics and image processing. Viscosity solutions of H-J equations are studied [11,12] to single out a unique,

*Corresponding author. *Email addresses:* zhuhq@njupt.edu.cn (H. Zhu), jxqiu@xmu.edu.cn (J. Qiu)

practically relevant solution. These viscosity solutions are Lipschitz continuous, and may develop discontinuous derivatives no matter how smooth the initial conditions are.

The study of numerical approximations to the viscosity solution of H-J equation (1.1) was started by Crandall and Lions. In [12] they proposed a monotone finite difference scheme and proved convergence to the viscosity solution. Unfortunately, monotone schemes can be at most first order accurate. Osher and Sethian [22] and Osher and Shu [23] constructed a class of high order ENO (essentially non-oscillatory) schemes which were adapted from ENO schemes in [13, 30, 31] for hyperbolic conservation laws. Their construction was based on the observation that there is a close relation between H-J equations and conservation laws, and as a result successful numerical methods for conservation laws can be adapted for solving H-J equations. For example, WENO (weighted ENO) schemes in [16, 21], Hermite WENO schemes in [24, 25] and RKDG method in [8, 9] were adapted for solving H-J equations by Jiang and Peng [15], Qiu and Shu [26] and Hu and Shu [14] (see also the reinterpretation work [19]), respectively. For other approaches, we mention the work of central high resolution schemes developed in [2, 3, 18, 20] and two different direct DG (discontinuous Galerkin) methods introduced respectively in [5] and [33].

In the traditional way numerical methods adopt fixed and pre-assigned meshes, so one has to develop higher-order (third, fourth or ever higher) numerical schemes in order to enhance the resolution of the numerical approximations. Lower-order schemes, which may be rather simple, can also produce high resolution with small number of mesh points if mesh adaptation is employed. So far a few works have been done on adaptive algorithms for H-J equation (1.1). We refer, for instance, to Tang et al. [32], where an adaptive mesh redistribution (r-adaptive) method was developed for solving two- and three-dimensional H-J equations. We also refer to Cockburn and his collaborators [1, 4].

The singularities in the solution derivatives cause great difficulties in obtaining numerical solutions of H-J equations. Local error at the discontinuities of the derivatives may be significantly larger and dominate the global error. Higher-order elements at discontinuities can not decrease the local error but may result in oscillatory solutions, so local mesh refinement is a good solution which can enhance the resolution by steepening the discontinuities. This motivates us to work on the h-adaptive method for H-J equations.

Our h-adaptive method for H-J equations proposed in this paper is based on the RKDG methods. The RKDG methods for solving hyperbolic conservation laws are high-order accurate and highly parallelizable methods which can easily handle complicated geometries, boundary conditions and $h-p$ adaptivity. These methods have made their way into the main stream of computational fluid dynamics and other areas of applications. The first DG method was introduced in 1973 by Reed and Hill [28] for the neutron transport problem. A major development of this method was carried out by Cockburn et al. in a series of papers [6–9], in which a framework to solve nonlinear time dependent hyperbolic conservation laws was established. They adopted explicit, nonlinearly stable high order Runge-Kutta time discretizations [30], DG space discretizations with exact or approximate Riemann solvers as interface fluxes and TVB (total variation bounded) nonlinear limiter [29] to achieve nonoscillatory properties, and the method was termed as RKDG method. Detailed description of the method as well as its implementation can be found in the review

paper [10].

In [35, 36], we presented an h-adaptive RKDG method using troubled-cell indicators for solving hyperbolic conservation laws. A tree data structure (binary tree in one dimension and quadtree in two dimensions) is used to aid storage and neighbor finding. A troubled cell is a cell that is believed to contain a discontinuity and need the limiting procedure. Troubled-cell indicators are used to identify the troubled cells and mesh adaptation is achieved by refining the troubled cells and coarsening the untroubled "children". Extensive numerical tests indicate that the proposed h-adaptive method is capable of saving the computational cost and enhancing the resolution near the discontinuities. In this paper, we apply this h-adaptive method to solve H-J equation (1.1), based on the RKDG method for H-J equations introduced by Hu and Shu in [14], with an objective of enhancing the resolution near the discontinuities of the solution derivatives.

The outline of the remainder of the paper is as follows. Firstly in Section 2, we briefly review the RKDG method for H-J equations in [14]. Then in Section 3, we show the h-adaptive RKDG method for H-J equations and the implementation details. After that in Section 4, we present a series of numerical results to validate our adaptive method. Finally, we give our concluding remarks in Section 5.

2. Review of RKDG method for H-J equations

Our h-adaptive RKDG method for H-J equations proposed in this paper is based on the RKDG method for H-J equations introduced by Hu and Shu [14], which is briefly reviewed in this section.

2.1. Space discretization in one dimension

For one-dimensional case, H-J equation (1.1) becomes

$$\begin{cases} \phi_t + H(\phi_x) = 0, \\ \phi(x,0) = \phi_0(x), \end{cases} \tag{2.1}$$

which is equivalent to the conservation law

$$\begin{cases} u_t + H(u)_x = 0, \\ u(x,0) = u_0(x) = \phi_0{}'(x), \end{cases} \tag{2.2}$$

if we identify $u = \phi_x$.

The key idea of designing RKDG scheme for H-J equation (2.1) is to solve the conservation law (2.2) by the RKDG method in [8]. Divide the computational domain $[a,b]$ into N cells with boundary points

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

Denote the center of cell $I_i = [x_{i-1/2}, x_{i+1/2}]$ by x_i, and the length of cell I_i by Δx_i. The solution as well as the test function space is given by $V_h^k = \{p : p|_{I_i} \in P_k(I_i), \forall i\}$,

where $P_k(I_i)$ is the space of polynomials of degree at most k on cell I_i. Then a k-th order discontinuous Galerkin scheme for (2.1) can be defined as follows: find $\phi \in V_h^k$ such that

$$\int_{I_i} \phi_{xt} v dx - \int_{I_i} H(\phi_x) v_x dx + \hat{H}_{i+\frac{1}{2}} v^-_{i+\frac{1}{2}} - \hat{H}_{i-\frac{1}{2}} v^+_{i-\frac{1}{2}} = 0,$$
$$i = 1, \cdots, N, \quad v \in V_h^{k-1}. \tag{2.3}$$

Here

$$\hat{H}_{i+\frac{1}{2}} = \hat{H}\left((\phi_x)^-_{i+\frac{1}{2}}, (\phi_x)^+_{i+\frac{1}{2}}\right) = \hat{H}\left(u^-_{i+\frac{1}{2}}, u^+_{i+\frac{1}{2}}\right)$$

is a consistent and monotone (nondecreasing in the first argument and nonincreasing in the second argument) flux and $u^{\pm}_{i+1/2} = u(x^{\pm}_{i+1/2}, t)$ are the left and right limits of the discontinuous solution u at the cell interface $x_{i+1/2}$. In this paper we use the simple Lax-Friedrichs flux

$$\hat{H}(u^-, u^+) = \frac{1}{2}[H(u^-) + H(u^+) - \alpha(u^+ - u^-)], \tag{2.4}$$

where $\alpha = \max|H'(u)|$ with the maximum taken over the whole region in which u^-, u^+ varies. The second term in (2.3) can be computed either exactly or by a numerical quadrature of sufficiently high order of accuracy.

The scheme described above is only for the derivative $u = \phi_x$. The missing constant can be obtained as follows: first determine the constant for cell I_1 by requiring that

$$\int_{I_1} \phi_t + H(u) dx = 0,$$

then use

$$\phi(x_i, t) = \phi(x_1, t) + \int_{x_1}^{x_i} u(x, t) dx$$

to determine the constant on cell I_i.

2.2. Space discretization in two dimensions

In two space dimensions, H-J equation (1.1) has the form

$$\begin{cases} \phi_t + H(\phi_x, \phi_y) = 0, \\ \phi(x, y, 0) = \phi_0(x, y), \end{cases} \tag{2.5}$$

which is in some sense equivalent to the following conservation law system

$$\begin{cases} u_t + H(u, v)_x = 0, \\ v_t + H(u, v)_y = 0, \\ (u, v)(x, y, 0) = (u, v)_0(x, y), \end{cases} \tag{2.6}$$

if we identify $(u,v)=(\phi_x,\phi_y)$.

This conservation law system is solved by the RKDG method in [9]. Given a triangulation $\mathcal{T}_h$ of the domain Ω with the approximation space $V_h^k=\{p:p|_K\in P_k(K),\ \forall K\in\mathcal{T}_h\}$, where $P_k(K)$ is the space of polynomials of degree at most k on cell K, firstly find $u\in V_h^{k-1}$ and $v\in V_h^{k-1}$ such that

$$\int_K u_t w dxdy-\int_K H(u,v)w_x dxdy+\sum_{e\in\partial K}\int_e \widehat{H(u,v)n_1}wd\Gamma=0,$$
$$\int_K v_t w dxdy-\int_K H(u,v)w_y dxdy+\sum_{e\in\partial K}\int_e \widehat{H(u,v)n_2}wd\Gamma=0,$$

for all $w\in V_h^{k-1}$ and all $K\in\mathcal{T}_h$. Here (n_1,n_2) is the unit outward normal to the edge e of cell K and

$$\widehat{H(u,v)n_i}=\hat{H}_{i,e,K}\big((u,v)^-,(u,v)^+\big),\quad i=1,2,$$

is again a consistent and monotone flux in which the superscript "$-$" implies that the value is taken from within the element K, and the superscript "+" implies that the value is taken from outside the element K and within the neighboring element K' sharing the edge e with K. We again use the simple Lax-Friedrichs flux

$$\hat{H}_{1,e,K}\big((u,v)^-,(u,v)^+\big)=\frac{1}{2}\big[H(u^-,v^-)+H(u^+,v^+)\big]n_1-\frac{1}{2}\alpha(u^+-u^-),$$
$$\hat{H}_{2,e,K}\big((u,v)^-,(u,v)^+\big)=\frac{1}{2}\big[H(u^-,v^-)+H(u^+,v^+)\big]n_2-\frac{1}{2}\beta(v^+-v^-),$$

where $\alpha=\max_{u,v}|\partial H(u,v)/\partial u|$, $\beta=\max_{u,v}|\partial H(u,v)/\partial v|$ with the maximum taken over the relevant (global) range. Then find $\phi\in V_h^k$ by least squares

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

For the missing constant, similar to the one-dimensional case, we first update the constant on one or a few cells, using

$$\int_K \phi_t+H(\phi_x,\phi_y)dxdy=0,$$

then determine the constant by

$$\phi(B,t)=\phi(A,t)+\int_A^B \phi_x dx+\phi_y dy.$$

2.3. Time discretization

Up to now, we have taken the method of lines approach and have left the time variable t continuous. For time discretization we use the TVD (total variation diminishing) high order Runge-Kutta methods introduced in [30]. For example, the third order version for solving the method of lines ODE

$$\phi_t = L(\phi)$$

is given by

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

3. Algorithm and implementation details

The RKDG method for H-J equations described in the previous section can be divided into two parts. The first part is to solve the conservation law(s) (2.2) and (2.6), which are related to the H-J equations (2.1) and (2.5) respectively, by the standard RKDG method. The second part is to recover the constant. To design h-adaptive algorithms for H-J equations, one can simply apply h-adaptive algorithms for conservation laws to the first part. In [35, 36], we developed an h-adaptive RKDG method for solving one- and two-dimensional hyperbolic conservation laws using troubled-cell indicators. Extensive numerical tests showed the effectiveness of this method. In this paper, our h-adaptive RKDG method for H-J equations is developed by applying this method to the first part of the RKDG method for H-J equations. This section gives the algorithm and its implementation details.

3.1. Data structure

As in [35, 36], a cell-based tree data structure (binary tree in one dimension and quadtree in two dimensions) is used to store the mesh so that mesh refinement and coarsening are trivial to accomplish.

We consider each of the initial cells as the *root* of a different tree. The root can be divided into 2^d new cells of equal sizes, and each of these new cells can be divided recursively until a stopping criterion is met. The new cells are called *children* of the divided cell (*father*). A cell is called a *leaf* cell if it does not have any children. All the leaf cells constitute the computational mesh. The *level* of a cell is the number of divisions needed to obtain this cell. The level of the root is zero.

3.2. Algorithm

The following flowchart illustrates the h-adaptive RKDG method in [35,36] for conservation laws

$$\begin{cases} u_t + \Sigma_{i=1}^{d}(f_i(u))_{x_i} = 0, \\ u(x,0) = u_0(x), \end{cases}$$

with which one can easily have the h-adaptive RKDG algorithm for H-J equations.

Algorithm 3.1: h-adaptive RKDG algorithm for conservation laws.

Given the maximum cell level LEV and the final time T,

Step 1 Partition the domain into uniform cells (intervals in one dimension and rectangles in two dimensions) and compute the degrees of freedom $\{u_K^{(l)}(t_0)\}$ by the initial condition.

Step 2 Suppose we have known the mesh $\mathcal{T}_h(t_n)$ and the degrees of freedom $\{u_K^{(l)}(t_n)\}$ at time level t_n. Follow the troubled-cell indicator procedure, mark each cell as either a troubled cell or an untroubled cell.

- For each troubled cell, if its cell level is equal to LEV, do nothing. If its cell level is less than LEV, refine this cell by dividing it into 2^d uniform cells (see Fig. 1).
- For a father's 2^d children that are all untroubled cells, coarsening these cells by merging them (see Fig. 1).

Now we get the new mesh $\mathcal{T}_h(t_{n+1})$.

Step 3 Using L_2 projection, project the degrees of freedom $\{u_K^{(l)}(t_n)\}$ on the mesh $\mathcal{T}_h(t_n)$ to the new mesh $\mathcal{T}_h(t_{n+1})$.

Step 4 Evolve the solution on the mesh $\mathcal{T}_h(t_{n+1})$ from t_n to t_{n+1} by the RKDG procedure which is described in Section 2 and get solution $\{u_K^{(l)}(t_{n+1})\}$ at t_{n+1}.

Step 5 If $t_{n+1} < T$, go to Step 2.

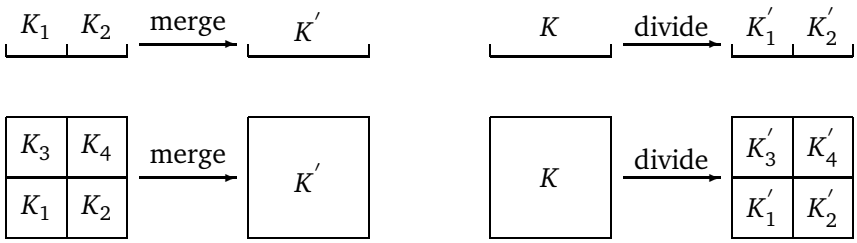

Figure 1: Sketches of merging (left) and dividing (right).

Attention has not been paid to the issue of time discretization efficiency in this paper, so global time steps are used in Runge-Kutta method, which are proportional to the smallest cell size at each time level. Study of local time-stepping scheme for the method is the subject of future work.

For two-dimensional problems we adopt rectangular meshes, as is indicated in the algorithm. So $\mathcal{T}_h$ is restricted to rectangular mesh with hanging nodes permitted, and the problem domain Ω is restricted to a domain that can be partitioned into uniform rectangles.

3.3. Basis functions

In order to simplify the implementation and calculation, we adopt the orthogonal basis of Legendre polynomials

$$\begin{cases} W_0(x) = 1, \\ W_l(x) = \dfrac{1}{2^l l!} \dfrac{d^l (x^2-1)^l}{dx^l}, \quad l > 0, \end{cases}$$

for the one-dimensional simulation. So the approximate solution $u_h(x,t)$ in the space V_h^{k-1} can be expressed as

$$u_h(x,t) = \sum_{l=0}^{k-1} u_i^{(l)}(t) W_l(2(x-x_i)/\Delta x_i), \quad \text{for } x \in I_i, \quad \forall i. \tag{3.1}$$

For the two-dimensional case, we form the following orthogonal basis

$$w_{\frac{i(i+1)}{2}+j}(x,y) = W_{i-j}(x) W_j(y), \quad i = 0, \cdots, k-1, \quad j = 0, \cdots, i,$$

which are tensor products of Legendre polynomials. Then the local orthogonal basis over cell K is given by

$$w_l^{(K)}(x,y) = w_l\Big(\frac{2(x-x_K)}{\Delta x_K}, \frac{2(y-y_K)}{\Delta y_K}\Big), \quad l = 0, \cdots, Q_{k-1},$$

in which $Q_k = k(k+3)/2$, (x_K, y_K) is the center of rectangle K, and Δx_K and Δy_K are lengths of K's sides in the direction of x and y respectively. Now the numerical solution u_h can be expressed as

$$u_h(x,y,t)|_K = \sum_{l=0}^{Q_{k-1}} u_K^{(l)}(t) w_l^{(K)}(x,y), \tag{3.2}$$

where $u_K^{(l)}(t)$ $(l = 0, \cdots, Q_{k-1})$ are the degrees of freedom. Particularly, $u_K^{(0)}(t)$ is the cell average of u_h over K.

3.4. L_2 projection

Let us now deduce the formulae of L_2 projection. To save space, only the formulae for two-dimensional case are shown. It is similar and easier to deduce the formulae for one-dimensional case.

Suppose we have already known u_h on mesh $\mathscr{T}_h(t_n)$, and we need to determine the degrees of freedom $u_{K'}^{(l)}(t_n)$ ($l=0,\cdots,Q_{k-1}$) in the new cell $K'\in\mathscr{T}_h(t_{n+1})$. Let $u_h^{'}$ denote the L_2 projection of u_h, it should satisfy the following equation

$$\int_{K'} u_h^{'}|_{K'} w_l^{(K')}(x,y)dxdy=\int_{K'} u_h w_l^{(K')}(x,y)dxdy,\quad l=0,\cdots,Q_{k-1}.$$

Represent $u_h^{'}|_{K'}$ using (3.2) and change the integral variables to get

$$u_{K'}^{(l)}(t_n)=\frac{4}{b_l\Delta x_{K'}\Delta y_{K'}}\int_{K'} u_h w_l^{(K')}(x,y)dxdy,\quad l=0,\cdots,Q_{k-1},\tag{3.3}$$

where

$$b_l=\int_{-1}^{1}\int_{-1}^{1}(w_l(x,y))^2dxdy,\quad l=0,\cdots,Q_{k-1},$$

are constants. For u_h is a piecewise polynomial, the integral in (3.3) can be computed exactly.

Now we are ready to give the formulae of L_2 projection. When four cells K_1, K_2, K_3, K_4 are merged to a new cell K' (see the left sketch in Fig. 1), the new degrees of freedom computed by (3.3) are as follows when $k-1=2$ (for simplicity we drop the time variable).

$$\begin{aligned}
u_{K'}^{(0)}&=\frac{1}{4}(u_{K_1}^{(0)}+u_{K_2}^{(0)}+u_{K_3}^{(0)}+u_{K_4}^{(0)}),\\
u_{K'}^{(1)}&=\frac{3}{8}(-u_{K_1}^{(0)}+u_{K_2}^{(0)}-u_{K_3}^{(0)}+u_{K_4}^{(0)})+\frac{1}{8}(u_{K_1}^{(1)}+u_{K_2}^{(1)}+u_{K_3}^{(1)}+u_{K_4}^{(1)}),\\
u_{K'}^{(2)}&=\frac{3}{8}(-u_{K_1}^{(0)}-u_{K_2}^{(0)}+u_{K_3}^{(0)}+u_{K_4}^{(0)})+\frac{1}{8}(u_{K_1}^{(2)}+u_{K_2}^{(2)}+u_{K_3}^{(2)}+u_{K_4}^{(2)}),\\
u_{K'}^{(3)}&=\frac{5}{16}(-u_{K_1}^{(1)}+u_{K_2}^{(1)}-u_{K_3}^{(1)}+u_{K_4}^{(1)})+\frac{1}{16}(u_{K_1}^{(3)}+u_{K_2}^{(3)}+u_{K_3}^{(3)}+u_{K_4}^{(3)}),\\
u_{K'}^{(4)}&=\frac{9}{16}(u_{K_1}^{(0)}-u_{K_2}^{(0)}-u_{K_3}^{(0)}+u_{K_4}^{(0)})+\frac{3}{16}(-u_{K_1}^{(1)}-u_{K_2}^{(1)}+u_{K_3}^{(1)}+u_{K_4}^{(1)})\\
&\quad+\frac{3}{16}(-u_{K_1}^{(2)}+u_{K_2}^{(2)}-u_{K_3}^{(2)}+u_{K_4}^{(2)})+\frac{1}{16}(u_{K_1}^{(4)}+u_{K_2}^{(4)}+u_{K_3}^{(4)}+u_{K_4}^{(4)}),\\
u_{K'}^{(5)}&=\frac{5}{16}(-u_{K_1}^{(2)}-u_{K_2}^{(2)}+u_{K_3}^{(2)}+u_{K_4}^{(2)})+\frac{1}{16}(u_{K_1}^{(5)}+u_{K_2}^{(5)}+u_{K_3}^{(5)}+u_{K_4}^{(5)}).
\end{aligned}$$

For $k-1=1$, only the first three formulae are needed.

When a cell K is divided into four subcells $K_1^{'},K_2^{'},K_3^{'},K_4^{'}$ (see the right sketch in Fig. 1), the new degrees of freedom for $k-1=2$ can be computed by setting

$$\begin{aligned}
&K_1^{'}:\ K'\to K_1^{'}, && \lambda_{1x}=-1/4, && \lambda_{1y}=-1/4, && \lambda_{2x}=1/2, && \lambda_{2y}=1/2,\\
&K_2^{'}:\ K'\to K_2^{'}, && \lambda_{1x}=1/4, && \lambda_{1y}=-1/4, && \lambda_{2x}=1/2, && \lambda_{2y}=1/2,\\
&K_3^{'}:\ K'\to K_3^{'}, && \lambda_{1x}=-1/4, && \lambda_{1y}=1/4, && \lambda_{2x}=1/2, && \lambda_{2y}=1/2,\\
&K_4^{'}:\ K'\to K_4^{'}, && \lambda_{1x}=1/4, && \lambda_{1y}=1/4, && \lambda_{2x}=1/2, && \lambda_{2y}=1/2,
\end{aligned}$$

in

$$u_{K'}^{(0)} = u_K^{(0)} + 2\lambda_{1x}u_K^{(1)} + 2\lambda_{1y}u_K^{(2)} + \Big(6\lambda_{1x}^2 + \frac{1}{2}\lambda_{2x}^2 - \frac{1}{2}\Big)u_K^{(3)} + 4\lambda_{1x}\lambda_{1y}u_K^{(4)} + \Big(6\lambda_{1y}^2 + \frac{1}{2}\lambda_{2y}^2 - \frac{1}{2}\Big)u_K^{(5)}, \tag{3.4a}$$

$$u_{K'}^{(1)} = \lambda_{2x}(u_K^{(1)} + 6\lambda_{1x}u_K^{(3)} + 2\lambda_{1y}u_K^{(4)}), \tag{3.4b}$$

$$u_{K'}^{(2)} = \lambda_{2y}(u_K^{(2)} + 2\lambda_{1x}u_K^{(4)} + 6\lambda_{1y}u_K^{(5)}), \tag{3.4c}$$

$$u_{K'}^{(3)} = \lambda_{2x}^2 u_K^{(3)}, \tag{3.4d}$$

$$u_{K'}^{(4)} = \lambda_{2x}\lambda_{2y}u_K^{(4)}, \tag{3.4e}$$

$$u_{K'}^{(5)} = \lambda_{2y}^2 u_K^{(5)}, \tag{3.4f}$$

where

$$\lambda_{1x} = \frac{x_{K'} - x_K}{\Delta x_K}, \qquad \lambda_{2x} = \frac{\Delta x_{K'}}{\Delta x_K},$$
$$\lambda_{1y} = \frac{y_{K'} - y_K}{\Delta y_K}, \qquad \lambda_{2y} = \frac{\Delta y_{K'}}{\Delta y_K}.$$

For $k-1=1$, one just sets $u_K^{(3)} = u_K^{(4)} = u_K^{(5)} = 0$ in (3.4).

3.5. Limiter and troubled-cell indicator

For some test problems of H-J equations, DG method needs a nonlinear limiter to generate the correct entropy solution [14]. Usually a limiter is composed of two parts: the first part is a troubled-cell indicator which is to detect the discontinuous regions and the second part is the solution reconstruction which is to control the oscillations. For the first part we use the so called KXRCF troubled-cell indicator which was based on a shock-detection technique introduced by Krivodonova et al. [17], and was numerically proved to be efficient and reliable in [35,36]. It works in the following way.

Partition the boundary of a cell K into two portions ∂K^- and ∂K^+, where the flow is into ($\vec{v}\cdot\vec{n} < 0$, $\vec{n}$ is the normal vector to ∂K) and out of ($\vec{v}\cdot\vec{n} > 0$) cell K, respectively. Cell K is identified as a troubled cell, if

$$\frac{\Big|\int_{\partial K^-}(u^h|_K - u^h|_{K_n})ds\Big|}{h_K^{\frac{k}{2}}\big|\partial K^-\big|\,\|u^h|_K\|} > 1,$$

where h_K is the radius of the circumscribed circle in cell K, K_n is the neighbor of K on the side of ∂K^- and the norm $\|\cdot\|$ is based on the maximum norm taken at the integration quadrature points in two dimensions and based on a cell average in one dimension.

Note that KXRCF troubled-cell indicator has two functions in the algorithm. One is to identify the troubled cells for mesh adaptation, the other is to determine the cells to be limited for the limiter.

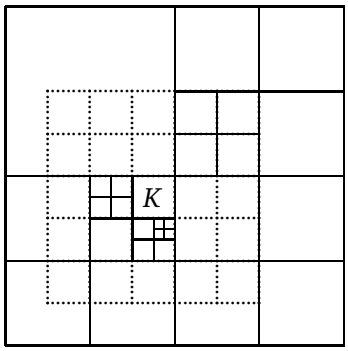

Figure 2: The imaginary local uniform mesh of K (dotted lines) when $k-1=2$.

For the second part of the limiter, we do not use the WENO type solution reconstruction method given in [35] for the one-dimensional case. We switch to use a simple WENO reconstruction method recently developed in [34], which is of higher order and is also easy to implement. For the two-dimensional case, if a cell K needs solution reconstruction, we build a local uniform mesh which consists of $(2k-1)\times(2k-1)$ cells with K in the center. See an example in Fig. 2. We use these imaginary cells instead of the real cells to reconstruct the solution in K. Since the imaginary local mesh is uniformly rectangular, the WENO solution reconstruction introduced by Qiu and Shu in [27] can be applied directly. The only problem left is to compute the cell averages in the imaginary local mesh. This can be done easily for we have already got the L_2 projection formula (3.3).

For more implementation details we refer to [35,36].

4. Numerical results

In this section we provide a series of numerical examples to illustrate the good behavior of our h-adaptive RKDG method for H-J equations. In all examples, we only plot the results obtained with a particular choice of initial mesh and with $LEV=3$ to save space. We have verified with the aid of successive refinements of initial mesh, that in all cases, the approximations are numerically convergent.

Example 4.1. In this first example we consider the one-dimensional Burgers' equation

$$\begin{cases} \phi_t+\dfrac{(\phi_x+1)^2}{2}=0, \quad -1<x<1, \\ \phi(x,0)=-\cos(\pi x), \end{cases} \tag{4.1}$$

with periodic boundary conditions.

Numerical solutions of ϕ at $t=10/\pi^2$ and the mesh changing figures which show when and where cell dividing and merging occur are presented in Fig. 3. Here and below, in a mesh changing figure each "□" represents a "dividing" and each "+" represents a

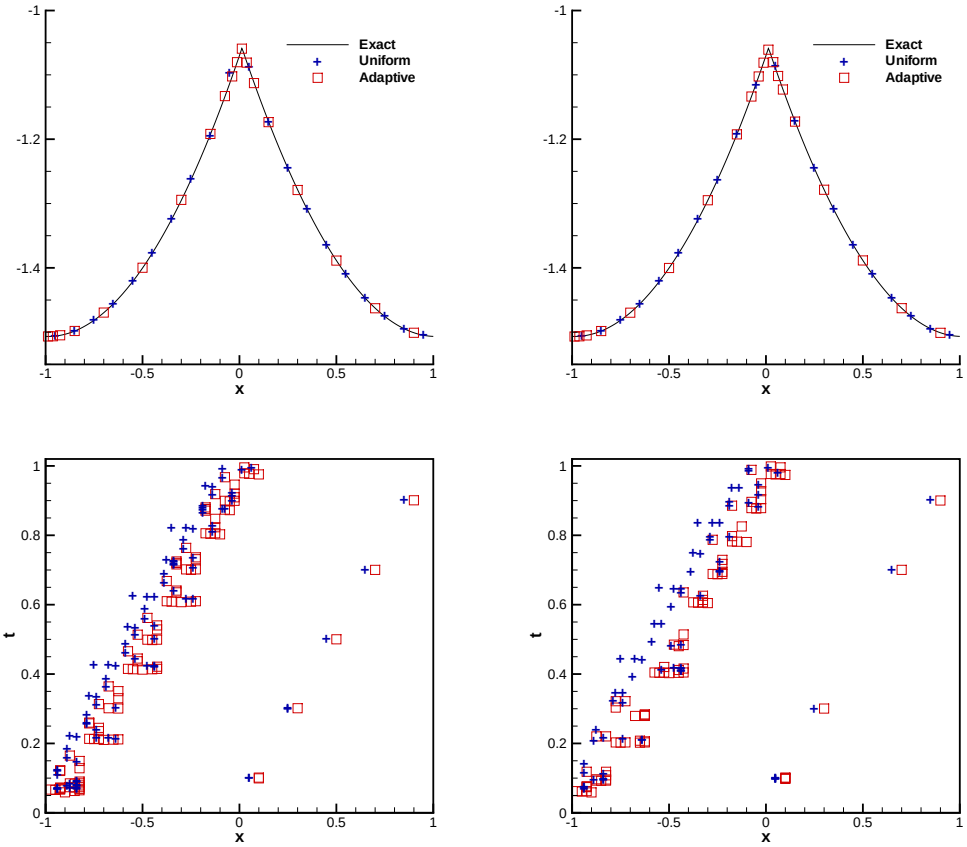

Figure 3: One-dimensional Burgers' equation (Example 4.1), solution ϕ (upper) and mesh changing (lower), $k=2$ (left) and $k=3$ (right); adaptive-mesh solution: $N_0=10$; uniform-mesh solution: $N=20$.

"merging". Note that only the information at part of the time levels is shown so that the figure size won't be too large. In a solution figure the solid line is the exact solution and the squares are numerical solutions obtained by our h-adaptive RKDG method. For comparison, in the solution figure we also show the numerical solution (pluses in the figure) obtained by Hu and Shu's RKDG method with a uniform mesh which uses more cells than the average cell number of the adaptive case. In the adaptive computation, we denote the initial cell number by N_0 and denote the average cell number by $\bar{N} = (\sum_{q=0}^{TOT} N_q)/TOT$ where N_q is the cell number at the q-th time level and TOT is the total number of time levels. These mesh data can be found at the end of this section in Table 1.

We can see from these figures that almost all the mesh refinement and coarsening are near the discontinuity of the derivative (the corner in the solution). As we expect, a cell is refined when a discontinuity comes close and is coarsened after the discontinuity leaves. In the final mesh the corner region is refined while the other regions use coarse mesh. As a result the adaptive algorithm produces much sharper corners than the algorithm using uniform mesh.

Example 4.2. We solve the one-dimensional linear equation

$$\begin{cases} \phi_t + \phi_x = 0, & -1 < x < 1, \\ \phi(x,0) = \phi_0(x-0.5), \end{cases} \tag{4.2}$$

with periodic boundary conditions. Here

$$\phi_0(x) = -\Big(\frac{\sqrt{3}}{2} + \frac{9}{2} + \frac{2\pi}{3}\Big)(x+1) + \begin{cases} 2\cos\Big(\dfrac{3\pi x^2}{2}\Big) - \sqrt{3}, & -1 \le x < -\dfrac{1}{3}, \\ \dfrac{3}{2} + 3\cos(2\pi x), & -\dfrac{1}{3} \le x < 0, \\ \dfrac{15}{2} - 3\cos(2\pi x), & 0 \le x < \dfrac{1}{3}, \\ \dfrac{28 + 4\pi + \cos(3\pi x)}{3} + 6\pi x(x-1), & \dfrac{1}{3} \le x < 1. \end{cases}$$

In Fig. 4 we show the computed solution at $t = 2$ and the mesh changing figure. We can see that finest meshes are used around all the discontinuities of ϕ_x. Our adaptive

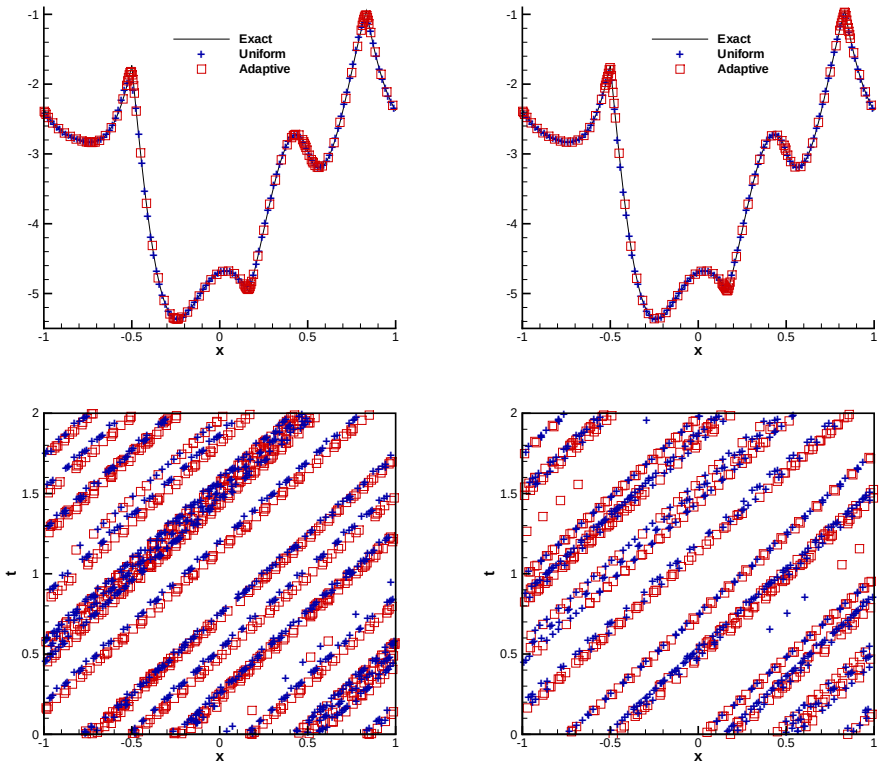

Figure 4: One-dimensional linear equation (Example 4.2), solution ϕ (upper) and mesh changing (lower), $k = 2$ (left) and $k = 3$ (right); adaptive-mesh solution: $N_0 = 60$; uniform-mesh solution: $N = 120$.

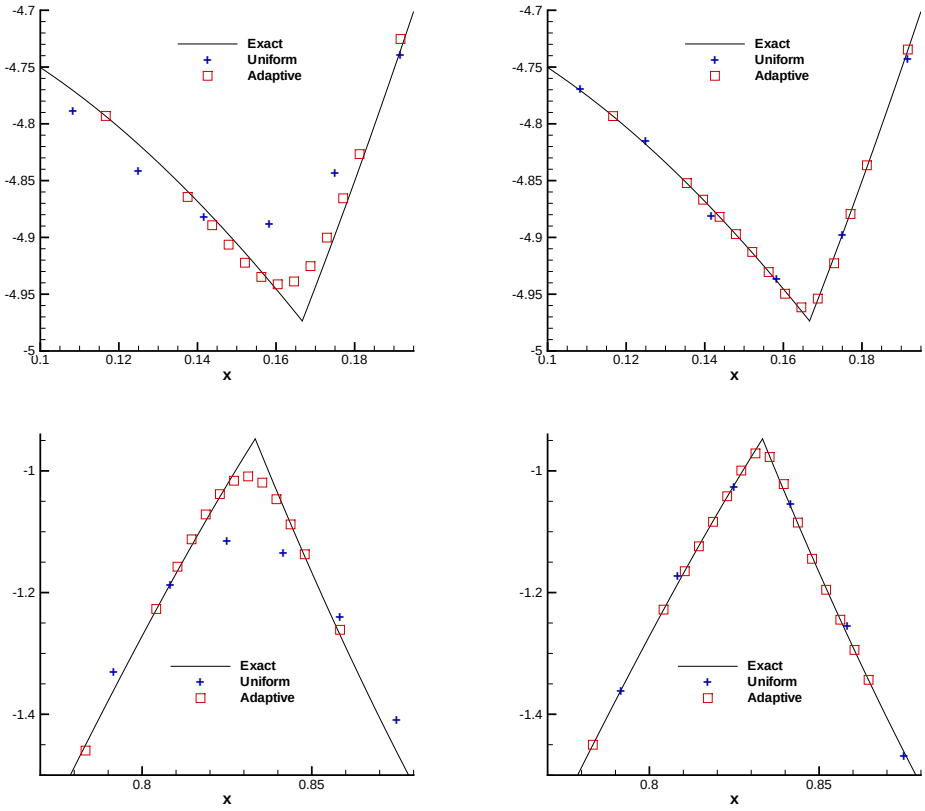

Figure 5: One-dimensional linear equation (Example 4.2), zoomed-in corners of the solution figures in Fig. 4, $k=2$ (left) and $k=3$ (right).

algorithm gives satisfactory numerical approximations with $N_0 = 60$ initial cells. For a better view of the details, we give the zoomed-in plots of two corners in Fig. 5. Higher resolution of the adaptive-mesh solutions over the uniform-mesh ones is clearly observed.

Example 4.3. One-dimensional Riemann problem with a non-convex flux:

$$\begin{cases} \phi_t + \dfrac{1}{4}(\phi_x^2 - 1)(\phi_x^2 - 4) = 0, \quad -1 < x < 1, \\ \phi(x,0) = -2|x|. \end{cases} \tag{4.3}$$

For this test case, the nonlinear limiter described in Section 3 is used in order to obtain the correct viscosity solution. We give the numerical results at $t = 1$ in Fig. 6, including the zoomed-in plots around the interesting features of the solution. It is found that the mesh refinement and coarsening are moving with the discontinuities of ϕ_x. The improvement of the adaptive solutions is clearly demonstrated by comparing the adaptive-mesh solutions and the uniform-mesh solutions.

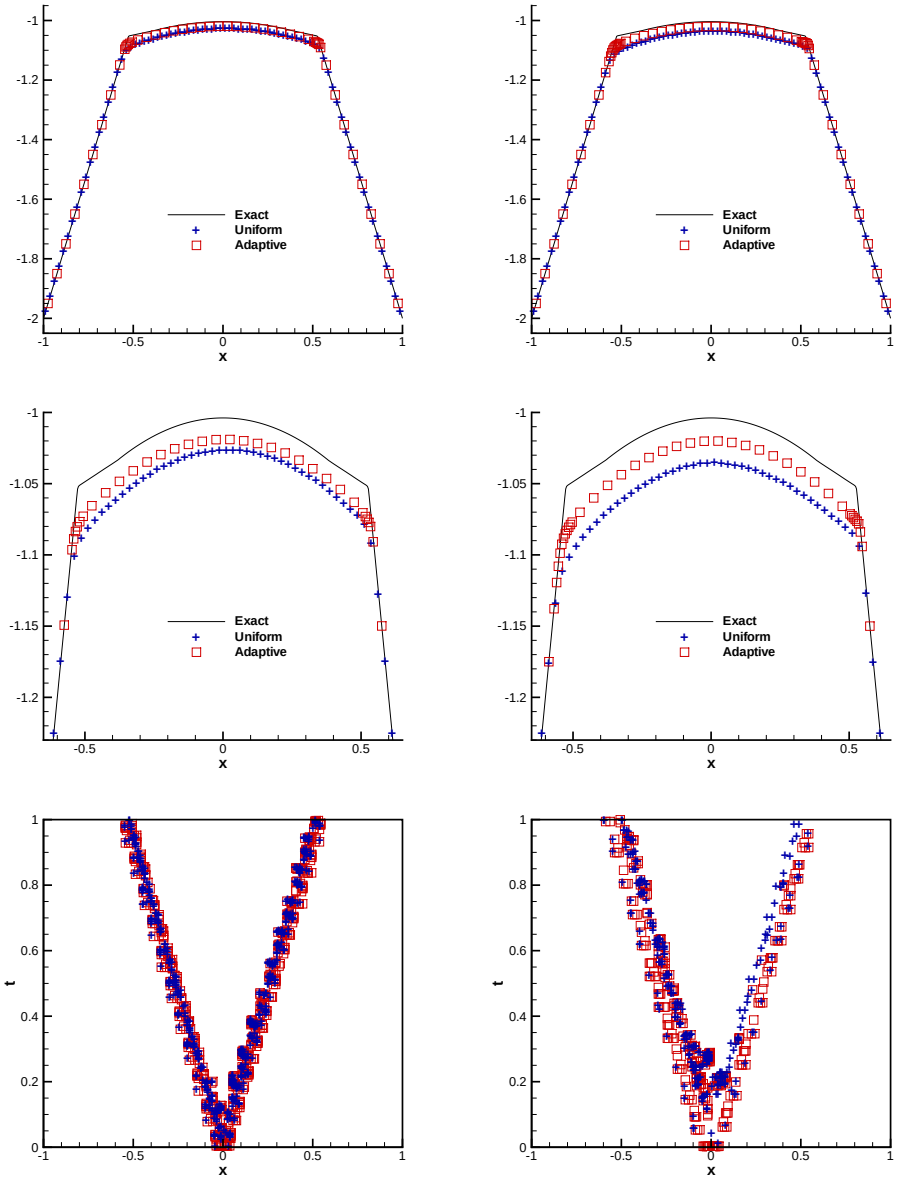

Figure 6: One-dimensional Riemann problem with a non-convex flux (Example 4.3); solution ϕ (upper), zoomed-in plots (middle) and mesh changing (lower); $k=2$ (left) and $k=3$ (right); adaptive-mesh solution: $N_0=40$; uniform-mesh solution: $N=80$.

Example 4.4. We consider the following two-dimensional Burgers' equation

$$\begin{cases} \phi_t + \dfrac{(\phi_x+\phi_y+1)^2}{2} = 0, & -2<x,y<2, \\ \phi(x,y,0) = -\cos\left(\dfrac{\pi(x+y)}{2}\right), \end{cases} \tag{4.4}$$

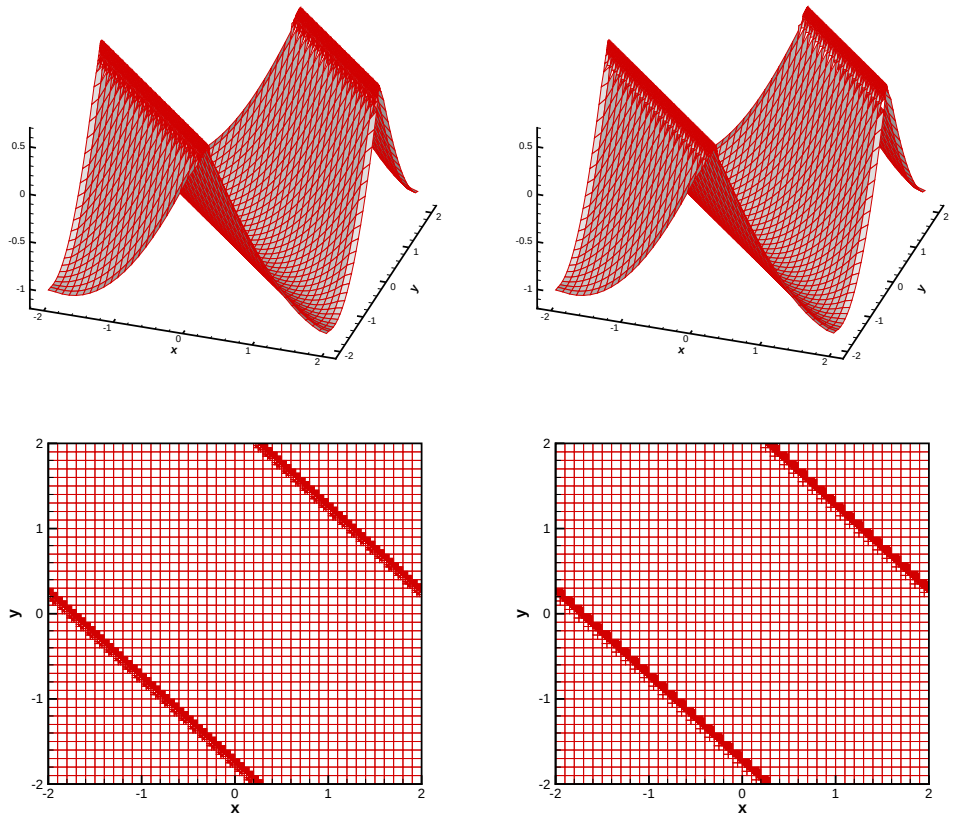

Figure 7: Two-dimensional Burgers' equation (Example 4.4), solution ϕ (upper) and adaptive mesh (lower) at $t=1.5/\pi^2$, 40×40 initial mesh, $k=2$ (left) and $k=3$ (right).

with periodic boundary conditions.

In Fig. 7 we plot the computed solution ϕ and the adaptive mesh at $t=1.5/\pi^2$. It is clearly seen that only the cells around the discontinuities of ϕ_x are refined. To illustrate the improvement of the adaptive solutions, in Fig. 8 we show the adaptive-mesh and uniform-mesh (also using more cells than the average cell number of the adaptive case) solutions along with the exact solution on the cut-line $y=x$. The corresponding zoomed-in plots around the left peak are also shown. We can see that the resolution at the peaks is higher for the adaptive solutions compared to the solutions obtained with uniform mesh.

Example 4.5. We solve the two-dimensional Riemann problem with a non-convex flux

$$\begin{cases} \phi_t+\sin(\phi_x+\phi_y)=0, & -1<x,y<1,\\ \phi(x,y,0)=\pi(|y|-|x|). \end{cases} \tag{4.5}$$

For this test case, we also need the nonlinear limiter described in Section 3 to get the viscosity solution. We give the numerical results at $t=1$ in Fig. 9. It is seen that satisfactory effects of the mesh adaptation are obtained.

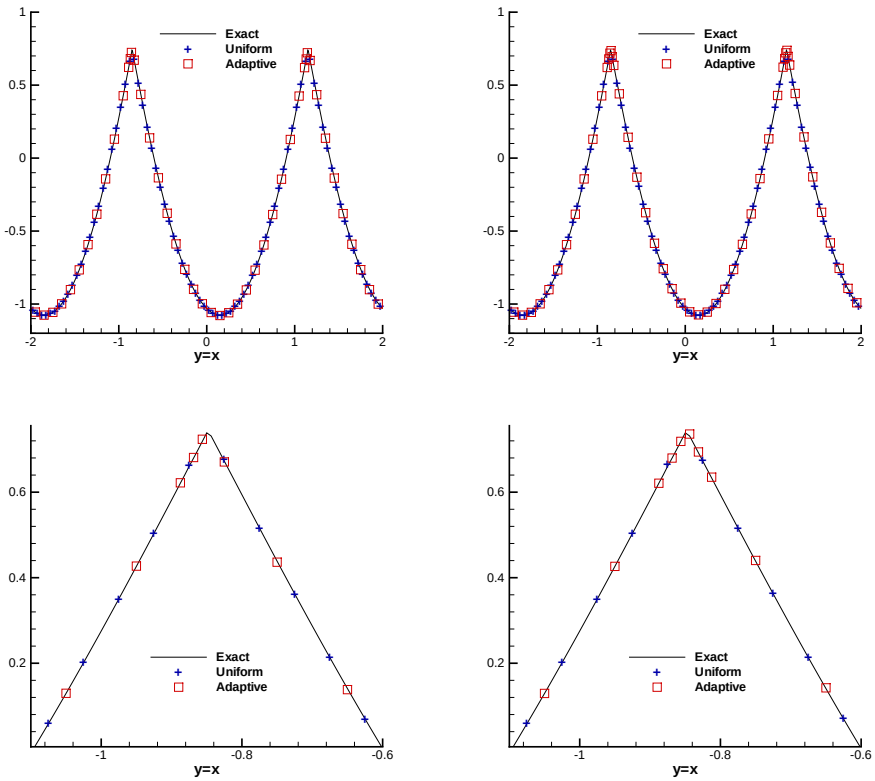

Figure 8: Two-dimensional Burgers' equation (Example 4.4), solution ϕ on the cut-line $y = x$ (upper) and zoomed-in plots around the left peak (lower); $k = 2$ (left) and $k = 3$ (right); adaptive-mesh solution: $N_0 = 40 \times 40$; uniform-mesh solution: $N = 80 \times 80$.

To gain a better understanding of the effectiveness of the h-adaptive RKDG method in this paper, for all the cases above we show the corresponding mesh data in Table 1, including (a) N_0: number of initial cells; (b) TDT: total dividing times; (c) N_T:

Table 1: Mesh data.

Case	N_0	k	TDT	N_T	$\bar{N}$	PR
Example 4.1	10	2	1.0E+2	19	16.7	20.92
		3	9.9E+1	20	17.7	22.07
Example 4.2	60	2	4.5E+3	120	116.1	24.19
		3	5.7E+3	103	103.8	21.64
Example 4.3	40	2	9.7E+2	50	50.8	15.87
		3	5.0E+2	57	62.0	19.36
Example 4.4	1600	2	2.9E+3	4480	3239.4	3.16
		3	2.8E+3	4480	3743.5	3.66
Example 4.5	400	2	7.9E+3	3868	2598.2	10.15
		3	8.4E+3	6637	4491.8	17.55

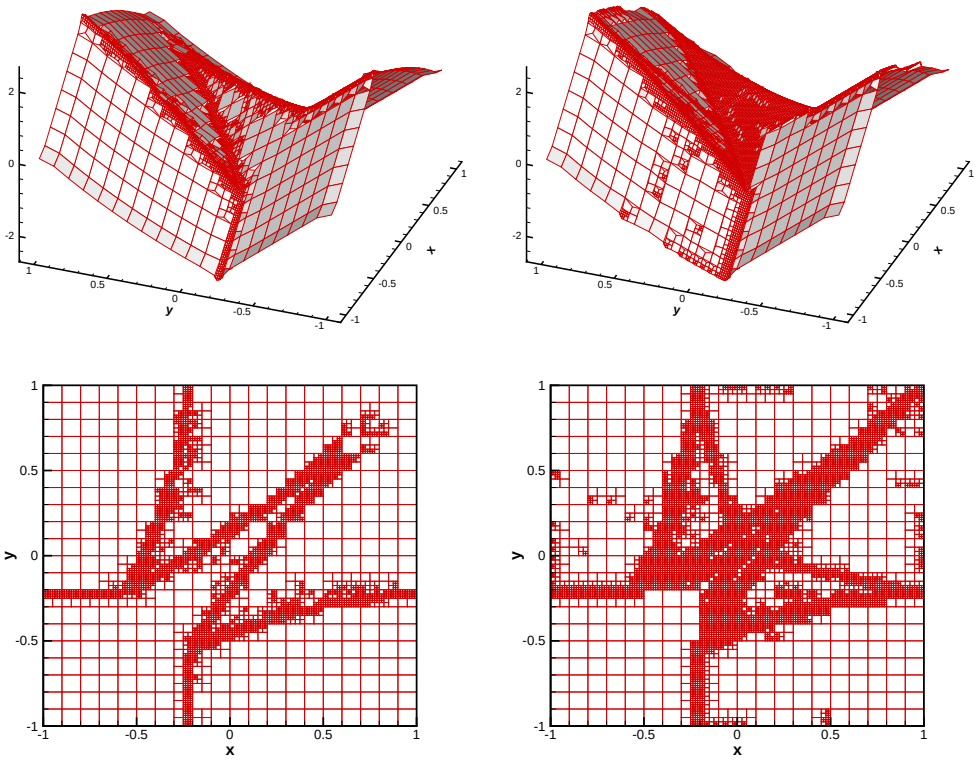

Figure 9: Two-dimensional Riemann problem with a non-convex flux (Example 4.5), solution ϕ (upper) and adaptive mesh (lower) at $t=1$, 20×20 initial mesh, $k=2$ (left) and $k=3$ (right).

number of cells at the final time level; (d) $\bar{N}$: average number of cells; and (e) PR: the percentage ratio of $\bar{N}$ to the number of cells if a fully refined mesh was used, i.e., $PR=100\bar{N}/(2^{d\cdot LEV}N_0)$. Total merging times TMT is not shown in the table as it can be calculated by $TMT=TDT-(N_T-N_0)/(2^d-1)$.

In the table we can see that all the values of PR are far less than 100, which means our adaptive algorithm needs much less cells than the one adopting fixed mesh provide that they produce comparable solutions at the discontinuities of the derivatives. As a result, our adaptive algorithm has the advantage of saving the storage space and improving the solution quality.

5. Concluding remarks

In this paper, an h-adaptive RKDG method for solving Hamilton-Jacobi equations is presented. A tree data structure (binary tree in one dimension and quadtree in two dimensions) is adopted to aid storage and neighbor finding. Troubled-cell indicator is used to identify the troubled cells and mesh adaptation is achieved by refining the troubled cells and coarsening the untroubled children. Numerical results of one- and two-dimensional

test problems show the capability of our adaptive method in improving the resolution at the discontinuities of the solution derivatives.

Acknowledgments The research was partially supported by NSFC grant 10931004, 11126287, 11201242, NJUPT grant NY211029 and ISTCP of China grant No. 2010DFR00700. The authors would like to thank the referees for the helpful suggestions.

References

[1] S. Albert, B. Cockburn, D. French, and T. Peterson, *A posteriori error estimates for general numerical methods for Hamilton-Jacobi equations, part I: the steady state case*, Math. Comput., 71 (2001), pp. 49–76.

[2] S. Bryson and D. Levy, *High-order central WENO schemes for multidimensional Hamilton-Jacobi equations*, SIAM J. Numer. Anal., 41 (2003), pp. 1339–1369.

[3] S. Bryson and D. Levy, *High-order semi-discrete central-upwind schemes for multi-dimensional Hamilton-Jacobi equations*, J. Comput. Phys., 189 (2003), pp. 63–87.

[4] Y. Chen and B. Cockburn, *An adaptive high-order discontinuous Galerkin method with error control for the Hamilton-Jacobi equations, part I: the one-dimensional steady state case*, J. Comput. Phys., 226 (2007), pp. 1027–1058.

[5] Y. Cheng and C.-W. Shu, *A discontinuous Galerkin finite element method for directly solving the Hamilton-Jacobi equations*, J. Comput. Phys., 223 (2007), pp. 398–415.

[6] 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*, Math. Comput., 54 (1990), pp. 545–581.

[7] 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.

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

[9] B. Cockburn and C.-W. Shu, *The Runge-Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems*, J. Comput. Phys., 141 (1998), pp. 199–224.

[10] B. Cockburn and C.-W. Shu, *Runge-Kutta discontinuous Galerkin methods for convection-dominated problems*, J. Sci. Comput., 16 (2001), pp. 173–261.

[11] M. G. Crandall and P. L. Lions, *Viscosity solutions of Hamilton-Jacobi equations*, Trans. Amer. Math. Soc., 277 (1983), pp. 1–42.

[12] M. G. Crandall and P. L. Lions, *Two approximations of solutions of Hamilton-Jacobi equations*, Math. Comput., 43 (1984), pp. 1–19.

[13] A. Harten, B. Engquist, S. Osher, and S. Chakravathy, *Uniformly high order accurate essentially non-oscillatory schemes, III*, J. Comput. Phys., 71 (1987), pp. 231–303.

[14] C. Hu and C.-W. Shu, *A discontinuous Galerkin finite element method for Hamilton-Jacobi equations*, SIAM J. Sci. Comput., 21 (1999), pp. 666–690.

[15] G. Jiang and D. Peng, *Weighted ENO schemes for Hamilton-Jacobi equations*, SIAM J. Sci. Comput., 21 (2000), pp. 2126–2143.

[16] G. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, J. Comput. Phys., 126 (1996), pp. 202–228.

[17] L. KRIVODONOVA, J. XIN, J.-F. REMACLE, N. CHEVAUGEON, AND J. FLAHERTY, *Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws*, Appl. Numer. Math., 48 (2004), pp. 323–338.
[18] A. KURGANOV AND E. TADMOR, *New high-resolution semi-discrete central schemes for Hamilton-Jacobi equations*, J. Comput. Phys., 160 (2000), pp. 720–742.
[19] F. LI AND C.-W. SHU, *Reinterpretation and simplified implementation of a discontinuous Galerkin method for Hamilton-Jacobi equations*, Appl. Math. Lett., 18 (2005), pp. 1204–1209.
[20] C.T. LIN AND E. TADMOR, *High-resolution non-oscillatory central schemes for Hamilton-Jacobi equations*, SIAM J. Sci. Comput., 21 (2000), pp. 2163–2186.
[21] X.-D. LIU, S. OSHER, AND T. CHAN, *Weighted essentially nonoscillatory schemes*, J. Comput. Phys., 115 (1994), pp. 200–212.
[22] 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.
[23] S. OSHER AND C.-W. SHU, *High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations*, SIAM J. Numer. Anal., 28 (1991), pp. 907–922.
[24] J. QIU AND C.-W. SHU, *Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method: one dimensional case*, J. Comput. Phys., 193 (2004), pp. 115–135.
[25] J. QIU AND C.-W. SHU, *Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method II: two dimensional case*, Comput. Fluid., 34 (2005), pp. 642–663.
[26] J. QIU AND C.-W. SHU, *Hermite WENO schemes for Hamilton-Jacobi equations*, J. Comput. Phys., 204 (2005), pp. 82–99.
[27] J. QIU AND C.-W. SHU, *Runge-Kutta discontinuous Galerkin method using WENO limiters*, SIAM J. Sci. Comput., 26 (2005), pp. 907–929.
[28] W. H. REED AND T. R. HILL, *Triangular mesh methods for neutron transport equation*, Technical report LA-UR-73-479, Los Alamos Scientific Laboratory, Los Alamos, NM, 1973.
[29] C.-W. SHU, *TVB uniformly high-order schemes for conservation laws*, Math. Comput., 49 (1987), pp. 105–121.
[30] C.-W. SHU AND S. OSHER, *Efficient implementation of essentially non-oscillatory shock-capturing schemes*, J. Comput. Phys., 77 (1988), pp. 439–471.
[31] C.-W. SHU AND S. OSHER, *Efficient implementation of essentially non-oscillatory shock-capturing schemes, II*, J. Comput. Phys., 83 (1989), pp. 32–78.
[32] H. Z. TANG, T. TANG, AND P. W. ZHANG, *An adaptive mesh redistribution method for nonlinear Hamilton-Jacobi equations in two- and three-dimensions*, J. Comput. Phys., 188 (2003), pp. 543–572.
[33] J. YAN AND S. OSHER, *A local discontinuous Galerkin method for directly solving Hamilton-Jacobi equations*, J. Comput. Phys., 230 (2011), pp. 232–244.
[34] X. ZHONG AND C.-W. SHU, *A simple weighted essentially nonoscillatory limiter for Runge-Kutta discontinuous Galerkin methods*, J. Comput. Phys., 232 (2013), pp. 397–415.
[35] H. ZHU AND J. QIU, *Adaptive Runge-Kutta discontinuous Galerkin methods using different indicators: one-dimensional case*, J. Comput. Phys., 228 (2009), pp. 6957–6976.
[36] H. ZHU AND J. QIU, *An h-adaptive RKDG method with troubled-cell indicator for two-dimensional hyperbolic conservation laws*, Adv. Comput. Math., DOI: 10.1007/s10444-012-9287-7.