SIAM J. SCI. COMPUT. © 2007 Society for Industrial and Applied Mathematics
Vol. 29, No. 6, pp. 2258–2282

A DISCONTINUOUS GALERKIN MOVING MESH METHOD FOR HAMILTON–JACOBI EQUATIONS[*]

J. A. MACKENZIE[†] AND A. NICOLA[‡]

Abstract. In this paper we consider the numerical solution of first-order Hamilton–Jacobi equations using the combination of a discontinuous Galerkin finite element method and an adaptive r-refinement (mesh movement) strategy. Particular attention is given to the choice of an appropriate adaptivity criterion when the solution becomes discontinuous. Numerical examples in one and two dimensions are presented to demonstrate the effectiveness of the adaptive procedure.

Key words. adaptivity, moving meshes, discontinuous Galerkin finite elements, Hamilton–Jacobi equations

AMS subject classifications. 35F25, 35L45, 35L65, 65M50, 65M60

DOI. 10.1137/060656243

1. Introduction.

In this paper we consider the adaptive numerical solution of Hamilton–Jacobi (HJ) equations

$$\phi_t + H(\phi_{x_1}, \dots, \phi_{x_d}) = 0, \quad \phi(\boldsymbol{x}, 0) = \phi_0(\boldsymbol{x}), \tag{1.1}$$

where $\boldsymbol{x} = (x_1, \dots, x_d) \in \mathbb{R}^d$, $t > 0$. HJ equations arise in many practical areas such as differential games, mathematical finance, image enhancement, and front propagation. It is well known that solutions of (1.1) are Lipschitz continuous, but derivatives can become discontinuous even if the initial data is smooth. Since generalized solutions are not unique, a selection principle is required to pick out the physically relevant solution. For HJ equations the most commonly used condition is the vanishing viscosity condition, which requires that the correct solution should be the vanishing viscosity limit of smooth solutions of corresponding viscous problems. The notion of viscosity solutions was introduced by Crandall and Lions [8], where the questions of existence, uniqueness, and stability of solutions were addressed.

Crandall and Lions were also the first to study numerical approximations of (1.1) and introduced the important class of monotone finite difference methods which were shown to converge to the viscosity solution [7]. However, monotonic schemes are well known to be at most first-order accurate.

There is a close relation between HJ equations and hyperbolic conservation laws (see section 2.1 below). With this in mind, it is not surprising to find that many of the numerical methods used to solve HJ equations are motivated by conservative finite difference or finite volume methods for conservation laws. Methods that have been proposed include high-order essentially nonoscillatory (ENO) schemes [22], [23], weighted ENO schemes [15], and high-resolution central schemes [21].

An increasingly popular approach to solving hyperbolic conservation laws is the discontinuous Galerkin (DG) finite element method [5], [6]. Recently, Hu and Shu

[*]Received by the editors April 4, 2006; accepted for publication (in revised form) February 20, 2007; published electronically October 5, 2007.
http://www.siam.org/journals/sisc/29-6/65624.html

[†]Department of Mathematics, University of Strathclyde, Livingstone Tower, 26 Richmond Street, Glasgow, G1 1XH, UK (jam@maths.strath.ac.uk).
[‡]Faculty of Mathematics and Computer Science, University of Ovidius, Blvd. Mamaia 124, 900527, Constanta, Romania (anicola@univ-ovidius.ro).

[11] proposed a DG method to solve HJ equations by first rewriting (1.1) as a system of conservation laws

$$(w_i)_t + (H(\boldsymbol{w}))_{x_i} = 0, \quad i = 1, \dots, d, \quad \boldsymbol{w}(\boldsymbol{x}, 0) = \nabla \phi_0(\boldsymbol{x}), \tag{1.2}$$

where $\boldsymbol{w} = \nabla\phi$. The usual DG formulation would be obtained if $\boldsymbol{w}$ belonged to a space of piecewise polynomials. However, we note that w_i, $i = 1, \dots, d$, are not independent due to the restriction that $\boldsymbol{w} = \nabla\phi$. In [11] a least squares procedure was used to enforce this condition. More recently it was shown that it is possible to enforce the gradient condition using a smaller solution space [19]. Theoretical analysis of the accuracy and stability of the method was performed in [18].

One of the often cited advantages of DG methods is that since the numerical solution is not continuous across interelement boundaries, this, in theory, makes solution adaptive strategies much easier to implement. This has led to the development of a number of adaptive methods based on *hp*-refinement strategies for hyperbolic conservation laws [4], [10].

An alternative adaptive strategy that has worked well for time-dependent problems is to use moving meshes. A useful way to construct an adaptive moving mesh is to regard it as the image of a uniform mesh covering a computational domain under a time-dependent transformation that clusters mesh elements towards areas where improved spatial resolution is required. The transformation is often found through a variational formulation where the mapping is the minimizer of a functional involving properties of the mesh and the solution (see, e.g., [17], [29]). To improve the stability and smooth the evolution of the moving mesh, Russell, Huang, and coworkers [14], [12] found it useful to obtain the mapping as the solution of parabolic moving mesh PDEs (MMPDEs) which are modified gradient flow equations for the minimization of a suitable mesh functional.

For problems where solution discontinuities exist, the correct choice of an adaptivity criterion, or monitor function, is problematic [30]. It is not unusual to find in the literature that many grid adaptation criteria are singular at solution discontinuities. To prevent singularities from producing degenerate meshes, either some form of smoothing procedure is employed or a regularized functional is used in the variational formulation [1].

There are two main ways to solve PDEs using moving meshes. The first is to reformulate the governing equations with respect to the moving reference frame. If conservation is important, then this reformulation must be done carefully. Examples of this approach are arbitrary Lagrangian–Eulerian (ALE) methods which are commonly used for problem with moving domains. Alternatively one may evolve the numerical solution over one time step using a stationary mesh. Thereafter, the mesh is moved, and some form of interpolation procedure is used to transfer the solution values from one mesh to another. This approach has been used successfully in the work of Tang and coworkers [20], [31], [27] to solve hyperbolic problems where a suitable conservative form of interpolation was proposed.

The aim of this paper is to consider the use of the DG method of Hu and Shu [11] to solve HJ equations using a moving mesh method based on the solution of MMPDEs. The governing equation is transformed to include the effect of the movement of the mesh and this is done in such a way that the conservation properties of the original equation are not lost. The adaptive mesh is driven by a monitor function which is shown to be nonsingular in the presence of solution discontinuities. To produce an acceptable mesh we smooth the monitor function before it is used to drive the adaptive procedure.

The layout of the rest of this paper is as follows: in the next section we present the governing HJ equations and their discretization using a moving mesh and a DG formulation. In section 3 we present the MMPDEs and their discretization and we discuss the choice of the monitor function. Finally, in section 4 we conduct a number of numerical experiments in one and two dimensions.

2. DG discretization of HJ equations.

2.1. HJ equation in one dimension. Let us consider the HJ equation in one dimension:

$$\begin{cases} \phi_t + H(\phi_x) = 0, & (x,t) \in (a,b) \times (0,T], \\ \phi(x,0) = \phi_0(x), \end{cases} \tag{2.1}$$

with appropriate boundary conditions. If $u = \phi_x$ and we differentiate (2.1) with respect to x, then u satisfies the hyperbolic conservation law

$$\begin{cases} u_t + H(u)_x = 0, & (x,t) \in (a,b) \times (0,T], \\ u(x,0) = u_0(x). \end{cases} \tag{2.2}$$

We assume that the physical domain $\Omega_p = [a,b]$ is the image of a computational domain $\Omega_c = [0,1]$ which is obtained via the time-dependent mapping $x = x(\xi,t)$. If (2.2) is considered with respect to the moving coordinate frame, then

$$\dot{u} - \dot{x}u_x + H(u)_x = 0, \quad (x,t) \in (a,b) \times (0,T], \tag{2.3}$$

where $(\dot{\cdot})$ represents differentiation with respect to time keeping ξ fixed. Note that the transformed equation is not in conservation form. Therefore, a discretization of (2.3) will not be conservative in general, even if the $H(u)_x$ term is treated conservatively, and this will lead to problems in convergence towards discontinuous solutions of (2.2) (see [24]). Therefore, instead we will consider a discretization of the conservative form in computational coordinates:

$$\begin{cases} (\dot{x_\xi u}) + (H(u) - \dot{x}\,u)_\xi = 0, & (\xi,t) \in (0,1) \times (0,T], \\ u(\xi,0) = u_0(\xi). \end{cases} \tag{2.4}$$

The aim is to discretize (2.4) in space using a DG method.

2.2. The DG discretization. For each partition $\{\xi_{j+\frac{1}{2}}\}_{j=0}^{N}$ of Ω_c, we denote

$$I_j = \left(\xi_{j-\frac{1}{2}}, \xi_{j+\frac{1}{2}}\right), \quad \Delta_j = \xi_{j+\frac{1}{2}} - \xi_{j-\frac{1}{2}}, \quad j = 1, \dots, N.$$

We use a uniform mesh to cover the computational domain such that

$$h = \xi_{j+\frac{1}{2}} - \xi_{j-\frac{1}{2}} = \frac{1}{N}, \quad \xi_j = \frac{1}{2}\left(\xi_{j+\frac{1}{2}} + \xi_{j-\frac{1}{2}}\right), \quad j = 1, \dots, N. \tag{2.5}$$

If we multiply (2.4) by an arbitrary smooth function v, then integrating by parts over the interval I_j we obtain

$$\int_{I_j} (\dot{x_\xi u})\, v\, d\xi - \int_{I_j} \widetilde{H}(u,\dot{x})\, v_\xi\, d\xi + \widetilde{H}(u,\dot{x})\, v\Big|_{I_j} = 0, \tag{2.6}$$

where

$$\widetilde{H}(u,\dot{x}) = H(u) - \dot{x}\,u, \tag{2.7}$$

and

$$\widetilde{H}(u,\dot{x})v\Big|_{I_j} = \widetilde{H}(u(\xi_{j+\frac{1}{2}},t),\dot{x}(\xi_{j+\frac{1}{2}}))\,v(\xi_{j+\frac{1}{2}}) - \widetilde{H}(u(\xi_{j-\frac{1}{2}},t),\dot{x}(\xi_{j-\frac{1}{2}}))\,v(\xi_{j-\frac{1}{2}}). \tag{2.8}$$

In this paper we will approximate the exact solution u of (2.6) and the smooth function v locally by linear polynomials. The numerical approximations u_h and v_h belong to the finite dimensional space

$$V_h^1 = \left\{v : v|_{I_j} \in P^1(I_j), \quad j = 1,\dots,N\right\}, \tag{2.9}$$

where $P^1(I_j)$ denotes the space of polynomials on I_j of degree at most one. In the discontinuous Galerkin numerical method u_h and v_h are discontinuous at the points $\xi_{j+\frac{1}{2}}$, $j = 0,\dots,N$, and the question is how to evaluate their values in (2.8). We set

$$v^-_{h_{j+\frac{1}{2}}} = \lim_{\xi\to\xi^-_{j+\frac{1}{2}}} v_h(\xi) \quad \text{and} \quad v^+_{h_{j-\frac{1}{2}}} = \lim_{\xi\to\xi^+_{j-\frac{1}{2}}} v_h(\xi). \tag{2.10}$$

Furthermore, we replace the nonlinear flux function $\widetilde{H}(u_h,\dot{x})$ defined in (2.7) by a numerical flux function that depends on the values of u_h from the left and right at the point $\left(\xi_{j+\frac{1}{2}},t\right)$, that is,

$$\widetilde{H}(u_h,\dot{x})_{\xi_{j+\frac{1}{2}}}(t) \approx \overline{H}_{j+\frac{1}{2}}\left(u_h\left(\xi^-_{j+\frac{1}{2}},t\right), u_h\left(\xi^+_{j+\frac{1}{2}},t\right), \dot{x}_{\xi_{j+\frac{1}{2}}}(t)\right). \tag{2.11}$$

In this paper we have used the local Lax–Friedrichs flux

$$\overline{H}(p,q,\dot{x}) = \frac{1}{2}\left(\widetilde{H}(p,\dot{x}) + \widetilde{H}(q,\dot{x}) - c(q-p)\right), \tag{2.12}$$

where

$$c = \max_{\min(p,q)\le u\le\max(p,q)} \left|\frac{\partial \widetilde{H}}{\partial u}\right|.$$

It is well known that some form of slope or flux limiting procedure is necessary to prevent oscillations at solution discontinuities when high-order methods are used to solve nonlinear hyperbolic conservation laws [25]. For example, if u_j^0 denotes the cell average of u_h in I_j, then we have

$$u_h(\xi) = u_j^0 + u_j^1(\xi - \xi_j), \quad j = 1,\dots,N, \tag{2.13}$$

and

$$\frac{du_h}{d\xi} = u_j^1, \quad j = 1,\dots,N,$$

is the slope of the numerical solution. One formula for limiting the slope in the numerical approximation is the minmod function

$$m(q,r,s) = \begin{cases} \operatorname{sgn}(q)\,\min(|q|,|r|,|s|) & \text{if } \operatorname{sgn}(q) = \operatorname{sgn}(r) = \operatorname{sgn}(s), \\ 0 & \text{otherwise,} \end{cases} \tag{2.14}$$

where sgn is the usual sign function. The initial linear expression for u_h is then replaced by

$$u_h(\xi) = u_j^0 + (u_j^1)^{\text{mod}}(\xi - \xi_j), \quad j = 1, \dots, N, \tag{2.15}$$

where $(u_j^1)^{\text{mod}}$ is the limited slope computed as

$$(u_j^1)^{\text{mod}} = m(u_j^1, \Delta_+ u_j^0/\Delta_j, \Delta_- u_j^0/\Delta_j),$$

in which Δ_+ and Δ_- are the usual forward and backward difference operators.

The discontinuous Galerkin space approximation is given as the solution of the weak formulation

$$\int_{I_j} (x_\xi \dot{u}_h) v_h \, d\xi - \int_{I_j} \widetilde{H}(u_h, \dot{x})(v_h)_\xi \, d\xi + \overline{H}_{j+\frac{1}{2}} v^-_{h_{j+\frac{1}{2}}} - \overline{H}_{j-\frac{1}{2}} v^+_{h_{j-\frac{1}{2}}} = 0, \tag{2.16}$$

and the initial condition is given by the projection

$$\int_{I_j} u_h(\xi, 0) v_h(\xi) \, d\xi = \int_{I_j} u_0(\xi) v_h(\xi) \, d\xi, \quad j = 1, \dots, N, \tag{2.17}$$

for all $v_h \in V_h^1$.

It remains to describe how the gradient of the mesh mapping x_ξ is approximated. We will assume that the map $x = x(\xi, t)$ is piecewise linear so that

$$x(\xi, t) = x_{j-\frac{1}{2}}(t) + N\left(\xi - \xi_{j-\frac{1}{2}}\right)\left(x_{j+\frac{1}{2}}(t) - x_{j-\frac{1}{2}}(t)\right) \tag{2.18}$$

for

$$\xi_{j-\frac{1}{2}} \le \xi \le \xi_{j+\frac{1}{2}}, \quad j = 1, \dots, N.$$

It then follows that x_ξ used in the weak formulation (2.16) is given by

$$x_\xi(\xi, t) = N(x_{j+\frac{1}{2}}(t) - x_{j-\frac{1}{2}}(t)), \quad j = 1, \dots, N. \tag{2.19}$$

2.2.1. Temporal integration. The discontinuous Galerkin discretization gives rise to a system of ODEs which can be written as

$$\begin{cases} \dfrac{d}{dt}(x_\xi \mathbf{u}_h) = L_h(\dot{x}, x_\xi, \mathbf{u}_h) & \text{in } (0, T], \\ \mathbf{u}_h(0) = \mathbf{u}_{0h}, \end{cases} \tag{2.20}$$

where $\mathbf{u}_{0h}$ is the initial approximation for the vector of the degrees of freedom $\mathbf{u}_h$. To integrate (2.20) numerically we use the second-order TVD Runge–Kutta method

$$\begin{cases} x_\xi^{n+1}\mathbf{u}_h^{(1)} = x_\xi^n \mathbf{u}_h^n + \Delta t \, L(\dot{x}^n, x_\xi^n, \mathbf{u}_h^n), \\ x_\xi^{n+1}\mathbf{u}_h^{n+1} = \dfrac{1}{2} x_\xi^n \mathbf{u}_h^n + \dfrac{1}{2} x_\xi^{n+1}\mathbf{u}_h^{(1)} + \dfrac{1}{2}\Delta t \, L(\dot{x}^{n+1}, x_\xi^{n+1}, \mathbf{u}_h^{(1)}). \end{cases} \tag{2.21}$$

For the temporal approximation of $x(\xi, t)$ we will assume that we have a set of grid points at time level t^n and at t^{n+1} which have been obtained using the MMPDE described in section 3. The map will be assumed to be piecewise linear in time so that

$$x_{j+\frac{1}{2}}(t) = x^n_{j+\frac{1}{2}} + (t - t^n)\left(\frac{x^{n+1}_{j+\frac{1}{2}} - x^n_{j+\frac{1}{2}}}{t^{n+1} - t^n}\right), \quad t^n \le t \le t^{n+1}, \tag{2.22}$$

and hence

$$\dot{x}_{j+\frac{1}{2}}(t) = \frac{x^{n+1}_{j+\frac{1}{2}} - x^n_{j+\frac{1}{2}}}{t^{n+1} - t^n}, \quad j = 0, \dots, N. \tag{2.23}$$

Therefore, $\dot{x}(\xi, t)$ will be piecewise linear in space and piecewise constant in time.

As the time integrator is explicit, the time step Δt has to be chosen to satisfy an appropriate CFL condition $\nu \le 1$, where the CFL number is

$$\nu = \Delta t \max_j \left\{ \frac{|(H_u)_{j+\frac{1}{2}} - \dot{x}_{j+\frac{1}{2}}|}{x_{j+1} - x_j} \right\}. \tag{2.24}$$

Note that the time step restriction on a moving mesh is different from that arising on a fixed nonuniform mesh due to the use of a modified wave speed. If the mesh velocity is close to the same speed as a solution discontinuity, then the shifted wave speed will be small where the mesh spacing is small, leading to global relaxation of the CFL condition. On the other hand, if the mesh is stationary and is highly graded towards large solution gradients, then stability requires the use of a globally small time step. The use of local time-stepping procedures can mitigate against this problem, but this certainly makes the overall algorithm more complicated (see, e.g., [26]).

2.2.2. Recovery of $\boldsymbol{\phi_h}$. If $(\phi_h)_x = u_h$, then ϕ_h will be determined on each interval I_j up to a constant. Following [11], the constant can be retrieved in two ways:

1. Require that

$$\int_{I_j} (\dot{\phi}_h + \tilde{H}(u_h, \dot{x}))\, \mathrm{d}\xi = 0, \quad j = 1, \dots, N. \tag{2.25}$$

   The ODE (2.25) is again solved using the second-order TVD Runge–Kutta method (2.21).
2. Use (2.25) to update only the leftmost element I_1, and then since $(\phi_h)_x = u_h$ we have

$$\phi_h(x_j, t) = \phi_h(x_1, t) + \int_{x_1}^{x_j} u_h(x, t)\, \mathrm{d}x. \tag{2.26}$$

In section 4 we compare the performance of both recovery procedures.

2.3. HJ equations in two dimensions. In two dimensions we have

$$\phi_t + H(\phi_x, \phi_y) = 0, \quad (x, y) \in \Omega_p, \quad t > 0, \tag{2.27}$$

where $\Omega_p \subseteq \mathbb{R}^2$ is the physical domain. To generate an adaptive mesh we have seen earlier that it is useful to regard the physical domain Ω_p as the image of a computational domain Ω_c under the invertible maps

$$x = x(\xi, \eta, t), \quad y = y(\xi, \eta, t), \quad \text{and} \quad \xi = \xi(x, y, t), \quad \eta = \eta(x, y, t), \tag{2.28}$$

where $\boldsymbol{x} = (x, y)^T$ and $\boldsymbol{\xi} = (\xi, \eta)^T$ are the physical and computational coordinates, respectively. A mesh covering Ω_p is obtained by applying the mapping given by (2.28) to a partitioning of Ω_c.

Given that the mesh can move, it is necessary to express the Eulerian ($\boldsymbol{x}$-fixed) temporal derivative in (2.27) in terms of the Lagrangian derivative along the trajectory of the moving mesh. If a dot denotes differentiation with respect to time with (ξ, η) fixed, then (2.27) becomes

$$\dot{\phi} - \dot{x}\phi_x - \dot{y}\phi_y + H(\phi_x, \phi_y) = 0, \tag{2.29}$$

where $(\dot{x}, \dot{y})$ represents the mesh velocity. If we let $u = \phi_\xi$ and $v = \phi_\eta$, then by differentiating (2.29) with respect to ξ and η we arrive at the system

$$\dot{u} + \bar{H}_\xi(u, v, \dot{x}, \dot{y}) = 0, \tag{2.30}$$

$$\dot{v} + \bar{H}_\eta(u, v, \dot{x}, \dot{y}) = 0, \tag{2.31}$$

where

$$\bar{H}(u, v, \dot{x}, \dot{y}) = H(\xi_x u + \eta_x v, \xi_y u + \eta_y v) - \dot{x}(\xi_x u + \eta_x v) - \dot{y}(\xi_y u + \eta_y v). \tag{2.32}$$

Using the vectorial notation

$$\boldsymbol{u} = \begin{pmatrix} u \\ v \end{pmatrix}, \quad \boldsymbol{F}_1 = \begin{pmatrix} \bar{H} \\ 0 \end{pmatrix}, \quad \boldsymbol{F}_2 = \begin{pmatrix} 0 \\ \bar{H} \end{pmatrix},$$

we can rewrite (2.30) and (2.31) as the system of conservation laws

$$\dot{\boldsymbol{u}} + [\boldsymbol{F}_1(\boldsymbol{u})]_\xi + [\boldsymbol{F}_2(\boldsymbol{u})]_\eta = 0. \tag{2.33}$$

We apply a discontinuous Galerkin method to solve this coupled system (2.33) for $(u, v) = \nabla_{\boldsymbol{\xi}}\phi$, where the discretization takes place in the computational domain Ω_c. A recovery procedure will then be used to obtain an approximation of ϕ.

2.4. The DG discretization in two dimensions. Let $\mathcal{T}_h$ denote a partition of the computational domain Ω_c and K an arbitrary element in this partition. We consider the space of all polynomials of degree k restricted on the element K to be

$$V_h(K) = \{p \in P^k(K) : p \text{ is a polynomial of degree at most } k \text{ on } K\}. \tag{2.34}$$

To obtain a weak formulation we take the inner product of (2.33) with a test function $\boldsymbol{v}_h \in V_h$, integrate over $K \in \mathcal{T}_h$, and replace $\boldsymbol{u}$ by its approximation $\boldsymbol{u}_h \in V_h$. That is,

$$\frac{d}{dt}\int_K \boldsymbol{u}_h(\boldsymbol{\xi}, t) \cdot \boldsymbol{v}_h(\boldsymbol{\xi})\, d\boldsymbol{\xi} + \int_K \left[(\boldsymbol{F}_1)_\xi \cdot \boldsymbol{v}_h + (\boldsymbol{F}_2)_\eta \cdot \boldsymbol{v}_h\right]\, d\boldsymbol{\xi} = 0, \tag{2.35}$$

and using integration by parts we obtain

$$\begin{aligned} &\frac{d}{dt}\int_K \boldsymbol{u}_h(\boldsymbol{\xi}, t) \cdot \boldsymbol{v}_h(\boldsymbol{\xi})\, d\boldsymbol{\xi} \\ &+ \sum_{e \in \partial K} \int_e \left\{(n_{e,K})_1 \boldsymbol{F}_1 \cdot \boldsymbol{v}_h(\boldsymbol{\xi}^{\text{int(K)}}) + (n_{e,K})_2 \boldsymbol{F}_2 \cdot \boldsymbol{v}_h(\boldsymbol{\xi}^{\text{int(K)}})\right\} ds \\ &- \int_K \left[\boldsymbol{F}_1 \cdot (\boldsymbol{v}_h)_\xi + \boldsymbol{F}_2 \cdot (\boldsymbol{v}_h)_\eta\right]\, d\boldsymbol{\xi} = 0, \end{aligned} \tag{2.36}$$

where $n_{e,K} = [(n_{e,K})_1, (n_{e,K})_2]$ is the unit normal to the edge e of the element K, $\text{int}(K)$ denotes the value taken from the interior of the element K, and $\text{ext}(K)$ denotes the value taken from the exterior of the element K. Next we define the normal flux

$$\boldsymbol{F}_{e,K}(\boldsymbol{u}, \dot{x}, \dot{y}) = (n_{e,K})_1 \boldsymbol{F}_1 + (n_{e,K})_2 \boldsymbol{F}_2 = \begin{bmatrix} (n_{e,K})_1 \bar{H} \\ (n_{e,K})_2 \bar{H} \end{bmatrix}. \tag{2.37}$$

Note that $\boldsymbol{F}_{e,K}$ is not well defined on the element edge as $\boldsymbol{u}_h$ is discontinuous there. Therefore, we replace it by the numerical flux function

$$\widehat{\boldsymbol{F}}_{e,K}\left(\boldsymbol{u}_h(\boldsymbol{\xi}^{\text{int(K)}}), \boldsymbol{u}_h(\boldsymbol{\xi}^{\text{ext(K)}}), \dot{x}, \dot{y}\right). \tag{2.38}$$

Again we will use the Lax–Friedrichs flux

$$\widehat{\boldsymbol{F}}_{e,K}(\boldsymbol{a}, \boldsymbol{b}, \dot{x}, \dot{y}) = \frac{1}{2}\left[\boldsymbol{F}_{e,K}(\boldsymbol{a}) + \boldsymbol{F}_{e,K}(\boldsymbol{b}) + \alpha_{e,K}(\boldsymbol{a} - \boldsymbol{b})\right], \tag{2.39}$$

where

$$\alpha_{e,K} = \rho\left((n_{e,K})_1 \frac{\partial \boldsymbol{F}_1}{\partial \boldsymbol{u}} + (n_{e,K})_2 \frac{\partial \boldsymbol{F}_2}{\partial \boldsymbol{u}}\right) \tag{2.40}$$

and $\rho(\cdot)$ denotes the spectral radius of $(\cdot)$.

In this paper we will consider only the case that $\boldsymbol{u}_h \in (P^1)^2$. In particular we now discuss the basis used when $\boldsymbol{u}_h$ is a discontinuous bilinear function defined on a topologically rectangular mesh covering Ω_c. We will assume that $\Omega_c = [0,1]^2$ is covered by a uniform $N \times N$ element mesh. Following Hu and Shu [11], we assume that the approximate solution of the HJ equation ϕ_h is piecewise discontinuous and quadratic. Therefore, in the cell $K_{ij} = (\xi_{i-1/2}, \xi_{i+1/2}) \times (\eta_{j-1/2}, \eta_{j+1/2})$ we assume

$$\begin{aligned} \phi_h(t) &= \overline{\phi}(t) + \phi_\xi(t)\mu_i + \phi_\eta(t)\nu_j + \phi_{\xi\eta}(t)\mu_i\nu_j \\ &\quad + \phi_{\xi\xi}(t)\left(\mu_i^2 - \frac{1}{3}\right) + \phi_{\eta\eta}(t)\left(\nu_j^2 - \frac{1}{3}\right), \end{aligned} \tag{2.41}$$

where

$$\mu_i(\xi) = \frac{2(\xi - \xi_i)}{\Delta\xi} \quad \text{and} \quad \nu_j(\eta) = \frac{2(\eta - \eta_j)}{\Delta\eta}.$$

The gradient in computational space $\nabla_{\boldsymbol{\xi}}\phi_h$ therefore takes the form

$$\nabla_{\boldsymbol{\xi}}\phi_h = \begin{bmatrix} \frac{2}{\Delta\xi}\phi_\xi(t) + \frac{2}{\Delta\xi}\phi_{\xi\eta}(t)\nu_j + \frac{4}{\Delta\xi}\phi_{\xi\xi}(t)\mu_i \\ \frac{2}{\Delta\eta}\phi_\eta(t) + \frac{2}{\Delta\eta}\phi_{\xi\eta}(t)\mu_i + \frac{4}{\Delta\eta}\phi_{\eta\eta}(t)\nu_j \end{bmatrix}, \tag{2.42}$$

and hence there are five unknowns that determine $\nabla_{\boldsymbol{\xi}}\phi_h$. A suitable basis for

$$V_2^1 = \{(v_1, v_2, v_3, v_4, v_5) : \boldsymbol{v}|_K = \nabla_{\boldsymbol{\xi}}\phi, \quad \phi \in P^2(K) \quad \forall K \in \mathcal{T}_h\}$$

is given by

(2.43)
$$\boldsymbol{b}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \boldsymbol{b}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \boldsymbol{b}_3 = \begin{bmatrix} \mu_i \\ 0 \end{bmatrix}, \quad \boldsymbol{b}_4 = \begin{bmatrix} 0 \\ \nu_j \end{bmatrix}, \quad \boldsymbol{b}_5 = \begin{bmatrix} \frac{\nu_j}{\Delta\xi} \\ \frac{\mu_i}{\Delta\eta} \end{bmatrix}.$$

The local basis functions (2.43) can easily be shown to be orthogonal, and hence the mass matrix is diagonal.

The evaluation of the edge and element integrals in (2.36) is achieved using quadrature. For the edges,

$$\int_e \widehat{\boldsymbol{F}}_{e,K} \cdot \boldsymbol{v}_h(\boldsymbol{\xi}^{\mathrm{int}(K)})$$
$$\approx \sum_{l=1}^{q} \omega_l \widehat{\boldsymbol{F}}_{e,K} \left(\boldsymbol{u}_h(\boldsymbol{\xi}_l^{\mathrm{int(K)}}), \boldsymbol{u}_h(\boldsymbol{\xi}_l^{\mathrm{ext(K)}}), \dot{x}_l, \dot{y}_l \right) \cdot \boldsymbol{v}_h(\boldsymbol{\xi}_l^{\mathrm{int(K)}})|e|,$$

where ω_l are the weights and $\boldsymbol{\xi}_l$ are the quadrature points along the edge. To estimate these integrals we use the two-point Gauss quadrature rule. Similarly, for the element integrals appearing in (2.36) we use the 2×2 Gauss product rule.

The mesh mapping $\boldsymbol{x}(\boldsymbol{\xi}, t)$ is assumed to vary bilinearly in space based on the position of the four grid nodes defining each element.

After carrying out the integrations and inverting the mass matrix, we finally arrive at the system of ODEs

$$\dot{\boldsymbol{u}} = L(\dot{\boldsymbol{x}}, \boldsymbol{u}), \quad \boldsymbol{u} = (u_1, u_2, u_3, u_4, u_5)^T, \tag{2.44}$$

which will be solved using the explicit two-stage explicit TVD Runge–Kutta scheme (2.21) to evolve $\boldsymbol{u}$, where $\dot{\boldsymbol{x}}$ is assumed to be piecewise constant in time.

The solution of (2.44) allows us to evolve $\nabla_{\boldsymbol{\xi}} \phi_h$. To calculate ϕ_h, we note from (2.41) that we require a procedure to calculate the cell average ϕ_{av}. Following Hu and Shu [11], we project the original HJ equation onto piecewise constants. That is, for any element K we set

$$\int_K (\dot{\phi}_h + \bar{H}(u_h, v_h, \dot{x}, \dot{y}))\, d\xi d\eta = 0. \tag{2.45}$$

We note from (2.41) that

$$\phi_{av}(t) = \frac{1}{|K|} \int_K \phi_h \, d\xi \, d\eta,$$

and hence

$$\dot{\phi}_{av} = -\frac{1}{|K|} \int_K \bar{H}(u_h, v_h, \dot{x}, \dot{y})\, d\xi d\eta. \tag{2.46}$$

We approximate the integral on the right-hand side of (2.46) using a 2×2 Gauss quadrature rule so that

$$\dot{\phi}_{av} = -\frac{1}{|K|} \sum_{l=1}^{4} \frac{\omega_l}{4} \, \bar{H}((u_h)_l, (v_h)_l, \dot{x}_l, \dot{y}_l)\, |K|. \tag{2.47}$$

The ODE (2.47) is again solved using the second-order TVD Runge–Kutta method (2.21). One possibility is to solve (2.47) for each element of the mesh. However, numerical experience reported in [11] indicates that this approach does not work well when there are singularities in the derivatives and the integral path in time passes through the singularities at earlier times. Alternatively, one can solve (2.47) for only one element, where there are no derivative singularities, and use the formula

$$\phi(B,t) = \phi(A,t) + \int_A^B (\phi_x \, dx + \phi_y \, dy) \tag{2.48}$$

to update all of the remaining elements.

3. Moving mesh equations.

3.1. MMPDEs in one and two dimensions.

Let us consider $\boldsymbol{x} = (x_1, x_2)^T$ to be a point in the physical domain Ω_p and $\boldsymbol{\xi} = (\xi_1, \xi_2)^T$ to be a point in the computational domain Ω_c. The mapping from the computational domain to the physical domain is assumed to minimize the energy functional

$$I[\xi_1, \xi_2] = \frac{1}{2} \int_{\Omega_p} \left((\nabla \xi_1)^T G^{-1} \nabla \xi_1 + (\nabla \xi_2)^T G^{-1} \nabla \xi_2 \right) \, dx \, dy, \tag{3.1}$$

where G is a symmetric positive-definite monitor matrix which involves various properties of the mesh and physical solution, and ∇ is the gradient operator with respect to $\boldsymbol{x} = (x_1, x_2)^T$.

A two-dimensional MMPDE is given by the gradient flow equations

$$\frac{\partial \boldsymbol{\xi}}{\partial t} = -\frac{P}{\tau} \frac{\delta I}{\delta \boldsymbol{\xi}} = \frac{P}{\tau} \nabla \cdot \left(G^{-1} \nabla \boldsymbol{\xi} \right), \tag{3.2}$$

where $\tau > 0$ is a temporal smoothing parameter and P is an operator with positive spectrum used to regularize the evolution of the mesh. The parameter τ determines how quickly the mesh adapts in time in an attempt to minimize (3.1). The value of τ should be chosen to be small in relation to the time scale of the physical PDE to ensure that the mesh adapts quickly enough to follow important solution features. On the other hand, it is important to make sure that τ is not too small; otherwise the mesh can evolve in an erratic manner, and this can have an adverse effect on accuracy. Further discussion on the choice of τ can be found in [13].

In practice we work directly with the map $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t)$ because it defines explicitly the location of the physical mesh points. Following [13] and [12], we switch the roles of the dependent and independent variables in (3.2) to obtain

$$\tau \frac{\partial \boldsymbol{x}}{\partial t} = P \left(\sum_{i,j=1}^{2} (\boldsymbol{a}^i \cdot G^{-1} \boldsymbol{a}^j) \frac{\partial^2 \boldsymbol{x}}{\partial \xi_i \partial \xi_j} - \sum_{i,j=1}^{2} \left(\boldsymbol{a}^i \cdot \frac{\partial G^{-1}}{\partial \xi_j} \boldsymbol{a}^j \right) \frac{\partial \boldsymbol{x}}{\partial \xi_i} \right), \tag{3.3}$$

where $\boldsymbol{a}^i = \nabla \xi_i$. In [12], Huang proposed choosing P in order to limit the variation over the domain of the coefficients in (3.3). To do this we choose

$$P = \left(\sum_{i=1}^{2} a_{i,i}^2 + b_i^2 \right)^{-\frac{1}{2}}, \tag{3.4}$$

where

$$a_{i,j} = \boldsymbol{a}^i \cdot G^{-1} \boldsymbol{a}^j, \quad b_i = -\sum_{i,j=1}^{2} \left(\boldsymbol{a}^i \cdot \frac{\partial G^{-1}}{\partial \xi_j} \boldsymbol{a}^j \right). \tag{3.5}$$

Dirichlet boundary conditions for the above system are obtained by solving a one-dimensional MMPDE. If $\partial_p \in \partial\Omega_p$ and $\partial_c \in \partial\Omega_c$ denote the physical and computational boundary segments with arc-lengths l and l_c, respectively, then the mesh on ∂_p is the solution of

$$\tau \frac{\partial s}{\partial t} = \frac{1}{\sqrt{(M^2 + (M_\zeta)^2}} \frac{\partial}{\partial \zeta}\left(M \frac{\partial s}{\partial \zeta}\right), \qquad \zeta \in (0, l_c), \tag{3.6}$$

with $s(0) = 0$ and $s(l_c) = l$. Here M is the one-dimensional projection of the two-dimensional monitor function along the boundary. That is, if $\boldsymbol{t}$ is a unit tangent vector along the boundary, then $M(s,t) = \boldsymbol{t}^T G \boldsymbol{t}$.

3.2. Discretization and boundary conditions. The temporal discretization of (3.3) is achieved using a semi-implicit method where

$$(\boldsymbol{x}^{n+1} - \boldsymbol{x}^n) = \frac{\Delta t P^n}{\tau}(a_{11}^n \boldsymbol{x}_{\xi\xi}^{n+1} + 2a_{12}^n \boldsymbol{x}_{\xi\eta}^{n+1} + a_{22}^n \boldsymbol{x}_{\eta\eta}^{n+1} + b_1^n \boldsymbol{x}_{\xi}^{n+1} + b_2^n \boldsymbol{x}_{\eta}^{n+1}). \tag{3.7}$$

Freezing the coefficients a_{ij}, b_i, and P has two very important effects. The first is that it linearizes the equations that define the mesh $\boldsymbol{x}^{n+1}$. Moreover, the system of PDEs represented by (3.7) for $x^{n+1}(\xi,\eta)$ and $y^{n+1}(\xi,\eta)$ decouples into two scalar equations which have the same spatial derivatives. For simplicity, the spatial discretization of (3.7) is performed using second-order central finite differences on an $N \times N$ uniform partition of $\Omega_c = (0,1) \times (0,1)$. Since the spatial derivatives are identical, the same coefficient matrix needs to be inverted to find the x and y coordinates. We therefore solve first (3.7) for the x coordinates using an ILU-preconditioned BiCGStab method until the mean-squared-root residual is less than 10^{-8}. The same preconditioner is then used to solve (3.7) for the y coordinates. In general, we have found that convergence of the iterative solver is achieved in around 3–4 iterations per time step. Additional details about the spatial discretization of the MMPDEs and the performance of the ILU-preconditioned BiCGStab routine can be found in [3].

3.3. Choice of monitor function. The choice of a suitable adaptivity criterion for problems which develop shocks is highly nontrivial. It is important that the mesh points move smoothly towards regions where discontinuities exist so that the $O(1)$ error at the shock is localized within one or two mesh elements. For robustness, it is also important that the rate of convergence of the method is not severely impaired by the nonuniformity of the moving mesh.

A popular choice of monitor function is the scaled solution arc-length

$$M = \sqrt{1 + \frac{1}{\alpha}|u_x|^2}, \tag{3.8}$$

where $\alpha > 0$ is a user-chosen parameter. The role of this parameter when $\alpha > 1$ is to reduce the local value of the monitor function when $|u_x|$ is large. This then leaves the issue of the proper choice of α. The simplest approach is to set $\alpha =$ constant. However, this procedure is unsatisfactory as the ideal value of α will be problem dependent and any a priori choice of α will never prevent $M \to \infty$ as $|u_x| \to \infty$.

A more sophisticated approach is to make α solution dependent. For example, in [9], $\alpha \approx \max |u|^2$. However, for problems where u becomes discontinuous, this choice of α will be ineffective in again stopping M from becoming unbounded. Another popular choice is

$$M(x,t) = \left(1 + \frac{|u_x|^2}{\alpha}\right)^{1/2}, \qquad \alpha = \frac{1}{\beta(b-a)} \int_a^b |u_x|^2 dx. \tag{3.9}$$

If $\beta = 1$, then (3.9) is exactly a regularization of the monitor function used in [24]. For this monitor function the local derivative term is scaled by its average value over the domain. It is not obvious that this monitor function is bounded above in the presence of a solution discontinuity. Let us assume an idealized situation where

$$|u_x| \simeq \mathcal{O}\left(\frac{1}{\Delta x_{\min}}\right) \tag{3.10}$$

in the cell where $|u_x|$ is large and is constant $|u_x| \simeq C$ everywhere else. If the mesh is obtained from an equidistribution principle, then

$$\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} M(x,t)\,dx = \frac{1}{N}\int_a^b M(x,t)\,dx, \quad j = 1,\dots,N. \tag{3.11}$$

If we expand the integral in (3.9), we get

(3.12)
$$\alpha = \frac{1}{\beta(b-a)}\left(\int_a^{x^*-\frac{1}{2}\Delta x_{\min}} C^2\,dx + \int_{x^*-\frac{1}{2}\Delta x_{\min}}^{x^*+\frac{1}{2}\Delta x_{\min}} \frac{1}{\Delta x_{\min}^2}\,dx + \int_{x^*+\frac{1}{2}\Delta x_{\min}}^{b} C^2\,dx\right),$$

and hence

$$\alpha \simeq \frac{1}{\beta}\left((1-\Delta x_{\min})\,C^2 + \frac{1}{\Delta x_{\min}(b-a)}\right) \approx (\Delta x_{\min})^{-1}. \tag{3.13}$$

It is clear from (3.13) that $\alpha \to \infty$ as $\Delta x_{\min} \to 0$. If we examine the behavior of M, we observe that the monitor function is approximately 1 away from the shock as $|u_x|$ is bounded and α tends to infinity. Close to the shock,

$$M \simeq \mathcal{O}\left(1 + \frac{\frac{1}{\Delta x_{\min}^2}}{\frac{1}{\Delta x_{\min}}}\right)^{\frac{1}{2}} \simeq \mathcal{O}\left(\sqrt{\frac{1}{\Delta x_{\min}}}\right),$$

which is not bounded as $\Delta x_{\min}$ tends to zero. The problem with this monitor function is that the maximum value of $|u_x|^2$ and its average value do not scale in the same way as N increases.

To obtain a bounded monitor function we consider

$$M(x,t) = \left(1 + \frac{|u_x|^2}{\alpha}\right)^{\frac{1}{2}}, \qquad \alpha = \frac{1}{\beta^2}\max_x |u_x|^2, \tag{3.14}$$

and β is a user-chosen parameter. It is clear that this monitor function satisfies the bounds

$$1 \le M(x,t) \le \sqrt{1+\beta^2}.$$

If we have a location x^* where $|u_x|$ becomes large (such as at a shock) and the solution elsewhere is smooth, then the monitor function will have a localized peak at x^*. It is clear that the transition from the smooth to the nonsmooth region of the domain will be almost discontinuous. This leads to an extremely rapid change in the mesh size which might be far from ideal in practice and can lead to a loss in accuracy. Again if the mesh is found by equidistribution of the monitor function, then if $M \simeq 1$ except in the cell containing x^*, we have

$$\sqrt{1+\beta^2}\Delta x_{\min} \simeq \frac{1}{N}\left((b-a)+\sqrt{1+\beta^2}\,\Delta x_{\min}\right). \tag{3.15}$$

Therefore, the minimum mesh spacing is

$$\Delta x_{\min} \simeq \frac{(b-a)}{(N-1)\sqrt{1+\beta^2}}, \tag{3.16}$$

which is approximately a factor of $\sqrt{1+\beta^2}$ smaller than the spacing that occurs when using a uniform mesh. This is the monitor function that we will use in the following one-dimensional examples.

For two-dimensional problems the monitor function used is of Winslow type, where

$$G = \begin{bmatrix} w & 0 \\ 0 & w \end{bmatrix}$$

and

$$w = \sqrt{1 + \frac{|\nabla u|^2}{\alpha}}, \quad \text{where} \quad \alpha = \frac{1}{\beta^2}\max_{\Omega_P} |\nabla u|^2.$$

This monitor function behaves in a similar fashion to the one-dimensional monitor in the presence of solution discontinuities since the maximum of the solution gradient is used rather than its average.

3.3.1. Smoothing of the monitor function. If the underlying problem involves large solution variations, then the monitor function will be nonsmooth in space, and this will affect the smoothness of the coordinate transformation. In practice, the monitor function is smoothed before it is used in the numerical approximation of the MMPDE.

A smooth transformation can be achieved using a boundary value problem (see [13]) which involves an artificial diffusion term for smoothing the monitor function. The smoothed monitor function $\widetilde{M}$ satisfies the boundary value problem

$$\widetilde{M} - \frac{1}{\lambda^2}\widetilde{M}_{\xi\xi} = M, \tag{3.17}$$

with the boundary conditions

$$\widetilde{M}_\xi(0,t) = \widetilde{M}_\xi(1,t) = 0, \tag{3.18}$$

where λ is a positive parameter. Equation (3.17) is discretized using central differences as

$$\widetilde{M}^n_{j+\frac{1}{2}} - \frac{1}{\lambda^2 h^2}\left(\widetilde{M}_{j+\frac{3}{2}} - 2\widetilde{M}_{j+\frac{1}{2}} + \widetilde{M}_{j-\frac{1}{2}}\right) = M_{j+\frac{1}{2}}, \quad j = 1, \ldots, N-1, \tag{3.19}$$

where $h = 1/N$. The boundary conditions are discretized as

$$\widetilde{M}_{-\frac{1}{2}} = \widetilde{M}_{\frac{1}{2}}, \quad \widetilde{M}_{N+\frac{1}{2}} = \widetilde{M}_{N-\frac{1}{2}}. \tag{3.20}$$

It can be shown that the smoothed monitor function so obtained is bounded and equidistribution leads to a locally quasi-uniform mesh [13]. In particular it can be shown that

$$\nu < \frac{x_{j+\frac{3}{2}}(t) - x_{j+\frac{1}{2}}(t)}{x_{j+\frac{1}{2}}(t) - x_{j-\frac{1}{2}}(t)} < \nu^{-1}, \qquad j = 1, \dots, N-1,$$

where

$$\nu = \frac{\sqrt{1+4\gamma} - 1}{\sqrt{1+4\gamma} + 1} \quad \text{and} \quad \gamma = \frac{1}{\lambda h^2}.$$

We can therefore see that the mesh becomes less smooth as we increase the parameter λ. A similar procedure is used to smooth the monitor function in two dimensions.

3.3.2. Complete algorithm. If we assume that at time $t = t^n$ we have an approximation of the physical solution u_h^n and a mesh $\boldsymbol{x}^n$, then we integrate forward in time using the algorithm below.

1. Compute the monitor matrix $G^n(\boldsymbol{x}) = G(\boldsymbol{x}^n, t^n)$ using u_h^n and $\boldsymbol{x}^n$.
2. Integrate the discretized MMPDE to get the mesh $\boldsymbol{x}^{n+1}$.
3. Integrate the physical PDEs to obtain u_h^{n+1} using the meshes $\boldsymbol{x}^n$ and $\boldsymbol{x}^{n+1}$ to form $\dot{\boldsymbol{x}}$.
4. Goto the next time step.

In theory this algorithm could be modified to return to step 1 after u_h^{n+1} has been obtained in step 3 and the coefficients a_{ij}, b_i in (3.5) have been evaluated at t^{n+1} and $\boldsymbol{x}^{n+1}$. A new estimate of the mesh $\boldsymbol{x}^{n+1}$ could then be found by resolving the MMPDEs. This process could be repeated a fixed number of times or until some measure of grid convergence is reached. A similar approach was investigated in [2] and [3], where it was found that larger time steps could be used without affecting accuracy and that the gain in efficiency outweighed the additional cost of performing the additional steps. In the numerical experiments in the next section we have, however, considered only the use of one pass of the algorithm above.

4. Numerical examples.

4.1. One-dimensional problems.

4.1.1. Example 1. The first example considered is the Burgers equation

$$\begin{cases} \phi_t + \dfrac{(\phi_x + 1)^2}{2} = 0, & -1 < x < 1, \ t > 0, \\ \phi(x,0) = -\cos(\pi(x - 0.85)) & \end{cases} \tag{4.1}$$

with periodic boundary conditions. This problem has also been considered in [28] and [15]. As the solution evolves from the smooth initial condition, a discontinuity in the derivative appears and propagates across the domain.

We first test the rate of convergence of the moving mesh DG method when the solution is still smooth. In all of the following results we set CFL $= 0.1$, $\beta = 10$, $\lambda = 25$, and $\tau = 0.001$. Table 1 shows that the DG solution maintains its second-order accuracy. Note that slope limiting was not needed for this example.

Figure 1 shows the computed DG solutions on a fixed and moving mesh at $t = 0.99/\pi^2$, which is just before the solution becomes discontinuous. We can see that the moving mesh results are a considerable improvement over the fixed mesh results

TABLE 1
Convergence of moving mesh DG solution at $t = 0.5/\pi^2$.

N	$\|\|e_u\|\|_{L_\infty}$	Rate	$\|\|e_\phi\|\|_{L_\infty}$	Rate	$\|\|e_u\|\|_{L_1}$	Rate	$\|\|e_\phi\|\|_{L_1}$	Rate
20	1.92E − 01	–	4.26E − 03	–	4.13E − 02	–	3.21E − 03	–
40	4.24E − 02	2.18	9.75E − 04	2.13	8.53E − 03	2.28	8.03E − 04	2.00
80	1.10E − 02	1.95	2.61E − 04	1.90	2.04E − 03	2.07	2.17E − 04	1.89
160	2.78E − 03	1.98	6.71E − 05	1.96	5.05E − 04	2.01	5.75E − 05	1.92
320	7.02E − 04	1.99	1.71E − 05	1.97	1.26E − 04	2.00	1.48E − 05	1.96

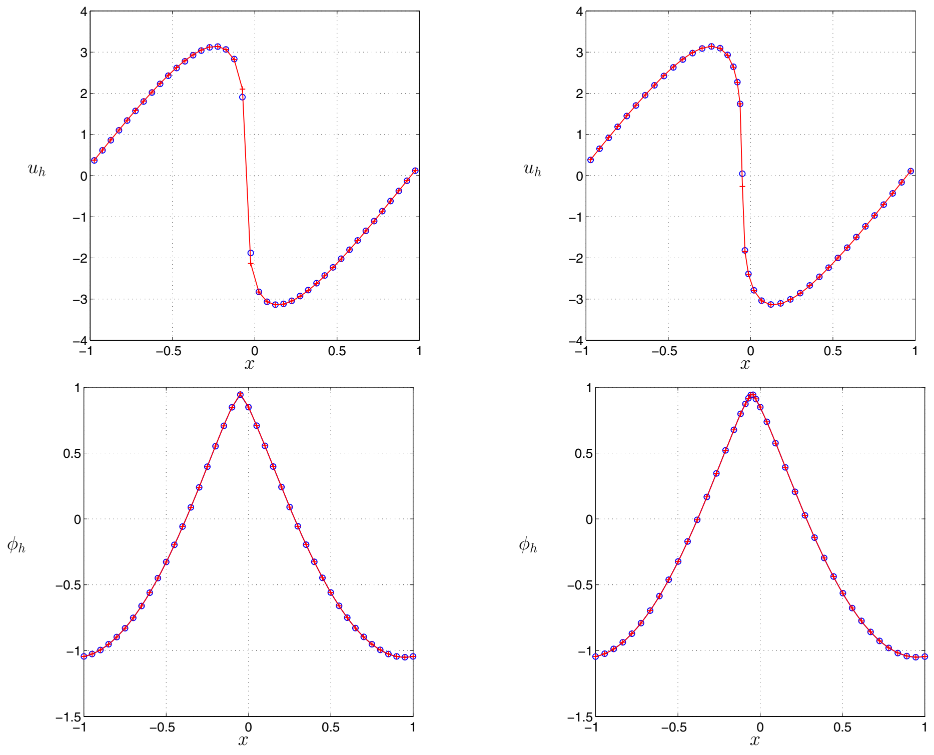

FIG. 1. *DG numerical solution (○) and exact solution (+) obtained on a uniform stationary mesh (left) and on a moving mesh (right) at time $t = 0.99/\pi^2$.*

as the shock in u and the corner in ϕ are much better resolved. Figure 2 shows the solutions at $t = 3/\pi^2$, where we can see that the shock in u is resolved over one or two cell widths that are much smaller around the shock than those used with the uniform mesh.

The computed mesh trajectories shown in Figure 3 are quite smooth, which is the result of the smoothing of the monitor function. We observe that once the discontinuity forms, the mesh points cluster around it to resolve it more accurately. Numerical experience shows that as long as the numerical solution is smooth, both recovery procedures (2.25) and (2.26) give similar results. In Figure 3 we also compare the two recovery procedures at $t = 1.5/\pi^2$ when the solution is nonsmooth. We can see that the recovery procedure (2.25) does not perform well, whereas (2.26) appears to perform much better and hence is the procedure we use for all further examples.

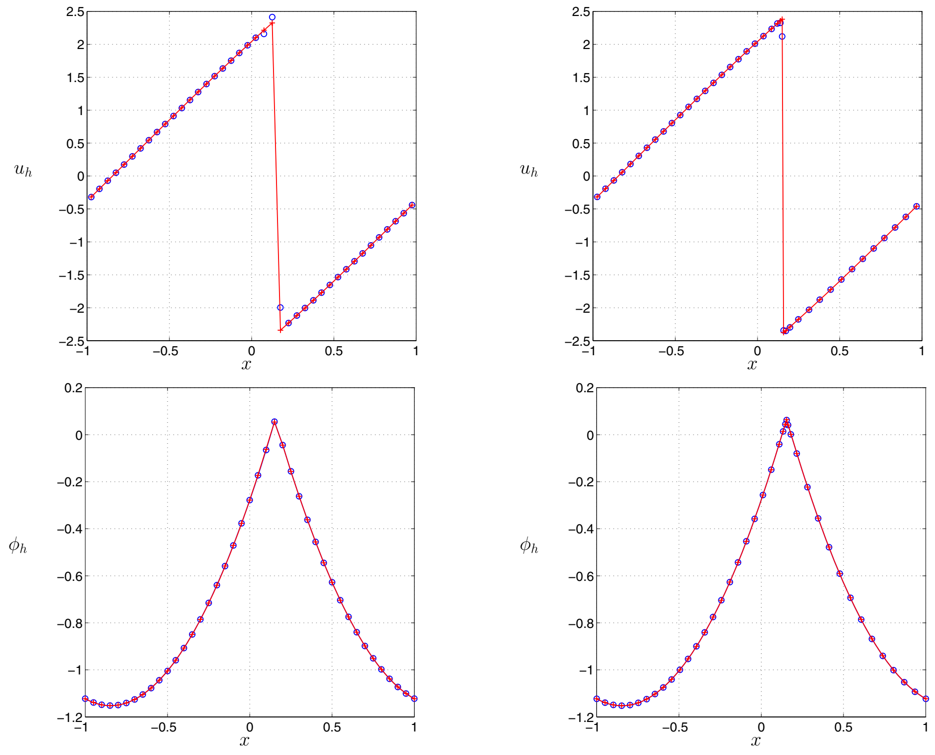

FIG. 2. *DG solution (○) and exact solution (+) obtained on a uniform stationary mesh (left) and on a moving mesh (right) at time $t = 3/\pi^2$.*

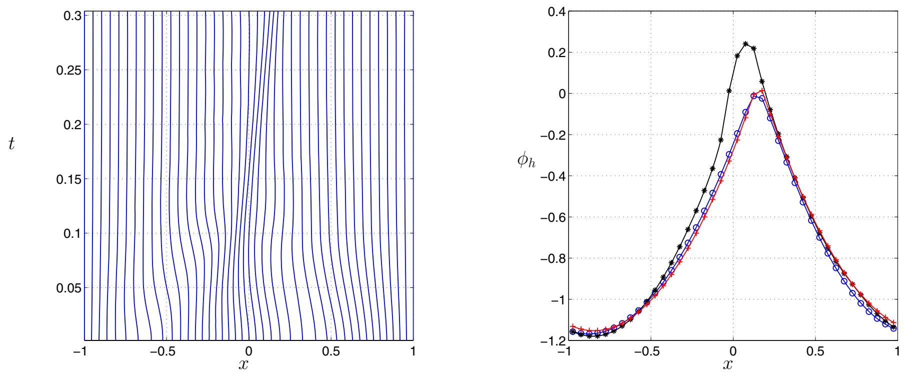

FIG. 3. *Mesh trajectories and comparison between recovery procedures at $t = 1.5/\pi^2$, (2.25) (*), (2.26) (○), and exact solution (+), with $N = 40$.*

In Figure 4 we plot the evolution of the time step using a uniform stationary mesh and the adaptive moving mesh. We can see that the time step used with the adaptive mesh is not considerably smaller than that used with the uniform mesh even though the mesh cells close to the shock region are much smaller than those occurring with the uniform mesh.

4.1.2. Example 2. The second example considered is the following Riemann problem with a nonconvex flux:

$$\begin{cases} \phi_t + \frac{1}{4}(\phi_x^2 - 1)(\phi_x^2 - 4) = 0, & -1 < x < 1,\ t > 0, \\ \phi(x,0) = -2|x|. \end{cases} \tag{4.2}$$

This problem has been considered in [11] and [28]. This test problem has an initial discontinuity in the derivative of the solution u, and in time this develops into two discontinuities moving in opposite directions. In [11] the authors point out that the DG method fails to "open up" the initial single discontinuity in the derivative to generate the correct entropy solution. Therefore, for this example we used the slope limiting procedure as outlined earlier.

A comparison of the DG solutions using a uniform fixed mesh and a moving mesh is shown in Figure 5. Again there is considerable improvement in the resolution of u_h and ϕ_h using the moving mesh. In Figure 6 we show the computed mesh trajectories and we can see that the monitor function is able to both track discontinuities and concentrate points in regions where more accuracy is needed.

4.1.3. Example 3. The third example considered is

$$\begin{cases} \phi_t - \left(1 + \phi_x^2\right)^{1/2} = 0, & -1 < x < 1,\ t > 0, \\ \phi(x,0) = \dfrac{1}{2}\cos(6\pi x), \end{cases} \tag{4.3}$$

with periodic boundary conditions. This is an example of front propagation in one dimension and has also been considered in [22]. As the numerical solution evolves in time, the solution develops corners since the graph of the initial solution moves in its normal direction with unit speed. For this example we found it necessary to use the slope limiting procedure.

In Figure 7 we show a comparison between ϕ_h obtained with DG using a stationary mesh and a moving mesh. The differences between the numerical solutions can be seen more clearly in Figure 8. It can be observed that the adaptive mesh does a much better job of resolving the corner which has developed in the numerical solution ϕ_h. Furthermore, the smooth part of the solution also appears to be much better resolved using the moving mesh. In Figure 9 we plot mesh trajectories of the DG moving mesh solution with $N = 81$ mesh points. We can see that the monitor function drives mesh points towards the discontinuities in the first derivative of the solution. In Figure 9 we also plot the evolution of the numerical solution ϕ_h at time intervals of $\Delta t = 0.01$, up to the final time $t = 0.1$. We can see clearly the initial function evolving with unit speed in its normal direction and that the moving mesh DG method resolves well the evolution of the corner singularities.

4.2. Two-dimensional problems.

4.2.1. Example 4. We consider the two-dimensional Burgers equation

$$\phi_t + \frac{1}{2}(1 + \phi_x + \phi_y)^2 = 0, \quad (x,y) \in (-2,2)^2, \tag{4.4}$$

$$\phi(x,y,0) = -\cos(\pi(x+y)/2), \tag{4.5}$$

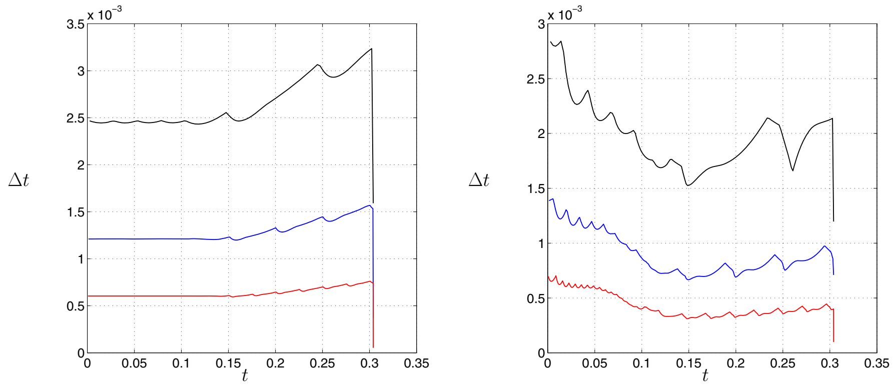

FIG. 4. *Time step evolution for Example* 1: *Uniform mesh (left), moving mesh (right),* $N = 20$ *(top line),* $N = 40$ *(middle line),* $N = 80$ *(bottom line).*

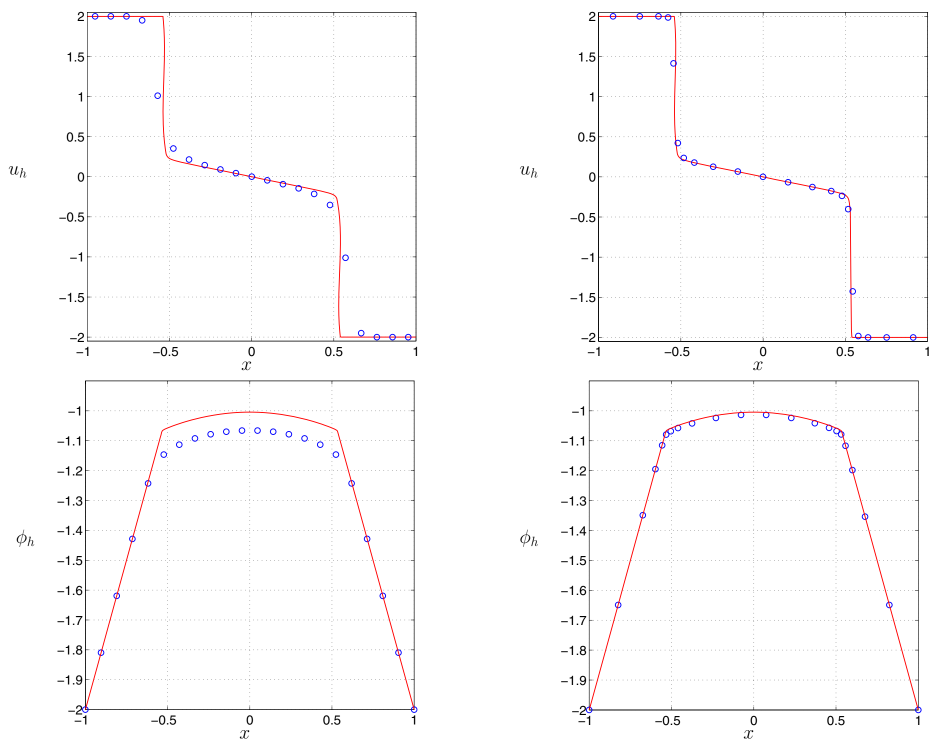

FIG. 5. *Comparison of results using DG uniform mesh (left), DG moving mesh method (right) at* $t = 1$ *with* $N = 21$, $CFL = 0.33$, $\beta = 10$, $\tau = 0.001$ *for Example* 2.

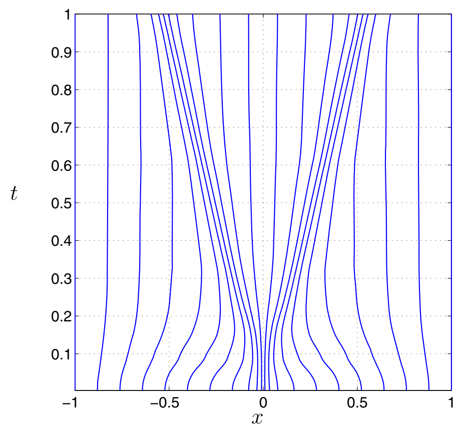

FIG. 6. *Mesh trajectories with $N = 21$ for Example 2.*

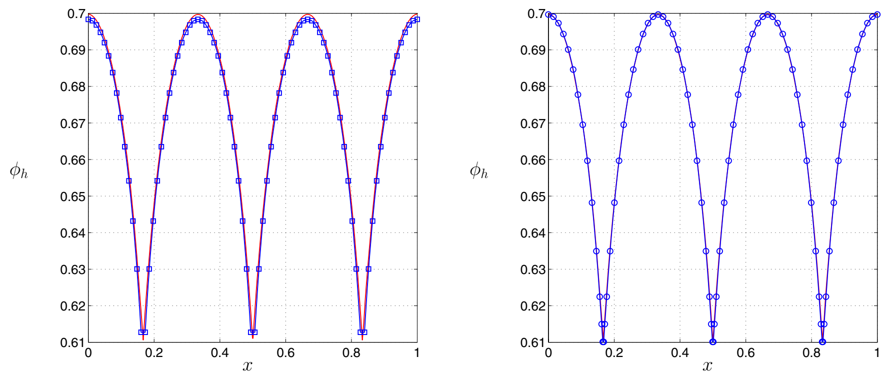

FIG. 7. *Numerical solution ϕ_h obtained with DG on a stationary mesh (□, left), on a moving mesh (○, right), and the reference solution (−) with $N = 81$ mesh points at $t = 0.2$.*

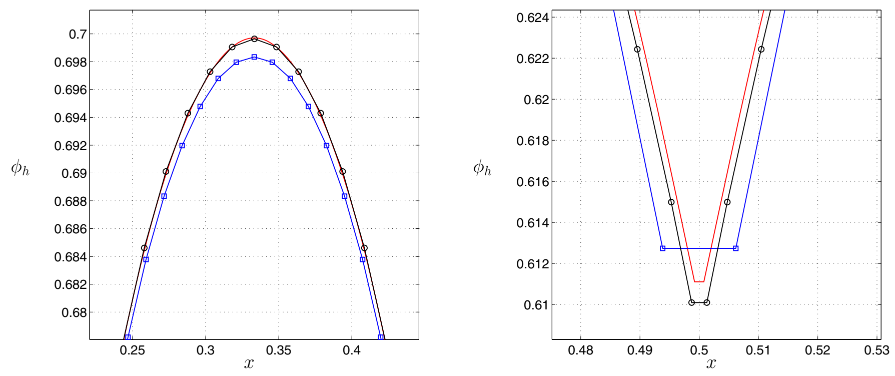

FIG. 8. *Zoom of the numerical solution ϕ_h obtained with DG on a stationary mesh (□, left), on a moving mesh (○, right), and the reference solution (−) with $N = 81$ at $t = 0.2$, in the smooth region (left), and nonsmooth region (right).*

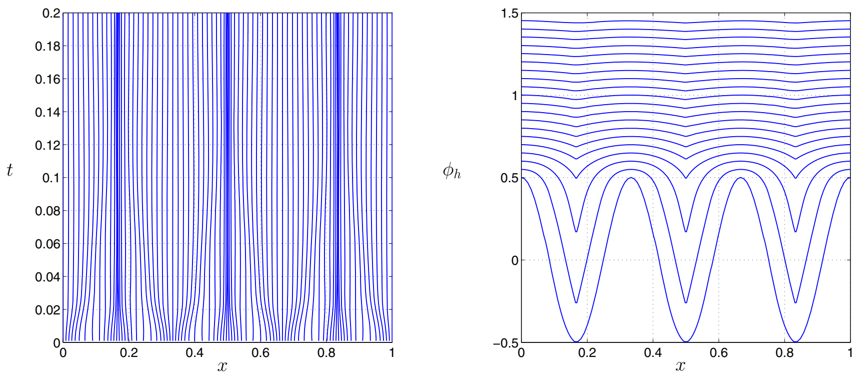

FIG. 9. *Mesh trajectories and evolution of the numerical solution at time intervals* $\Delta t = 0.01$, *up to* $t = 1$ *for Example* 3.

with periodic boundary conditions. This problem was proposed in [23] and has also been used as a test case in [28], [11], and [16]. From the smooth initial condition the solution develops a discontinuity in $\nabla\phi$ which propagates across the domain from the bottom left to the top right. Figure 10 shows the results calculated at $t = 1.5/\pi^2$ when $\nabla\phi$ has become discontinuous. Note that flux limiting was not necessary for this example, highlighting the additional stability properties of the DG formulation. We can see that while there seem to be some slight oscillations in the moving mesh calculations of ϕ_x, these seem to be damped out by the recovery procedure used to calculate ϕ_h. The computed adaptive meshes at $t = 0.5/\pi^2$ and $t = 1.5/\pi^2$ are shown in Figure 11, where we can see that the meshes have indeed clustered towards the areas where the gradient is largest. Finally, Figure 12 shows a comparison of the moving and fixed mesh results compared with a fine grid reference solution. We can see that the adaptive mesh results are a considerable improvement on the fixed grid results both in the corner region of the solution and in regions where the solution is smooth.

4.2.2. Example 5. We next consider a problem with a nonconvex flux

$$\phi_t - \cos(1 + \phi_x + \phi_y) = 0, \quad (x, y) \in (-2, 2)^2, \tag{4.6}$$

$$\phi(x, y, 0) = -\cos(\pi(x + y)/2), \tag{4.7}$$

with periodic boundary conditions. The numerical results for this case are shown in Figures 13 and 14. We again can see that the adaptive moving mesh algorithm does a good job of clustering elements towards the discontinuities in the solution gradient. The numerical results are comparable to the adaptive calculations presented in [28].

4.2.3. Example 6. Our final example is a prototype model in geometric optics [16], [22]. The governing equation is

$$\phi_t + \sqrt{1 + \phi_x^2 + \phi_y^2} = 0, \quad (x, y) \in (0, 1)^2, \tag{4.8}$$

$$\phi(x, y, 0) = 0.25(\cos(2\pi x) - 1)(\cos(2\pi y) - 1) - 1, \tag{4.9}$$

with periodic boundary conditions. The viscosity solution for this problem is a surface with its maximum focusing into a peak which opens up. The surface moves downward

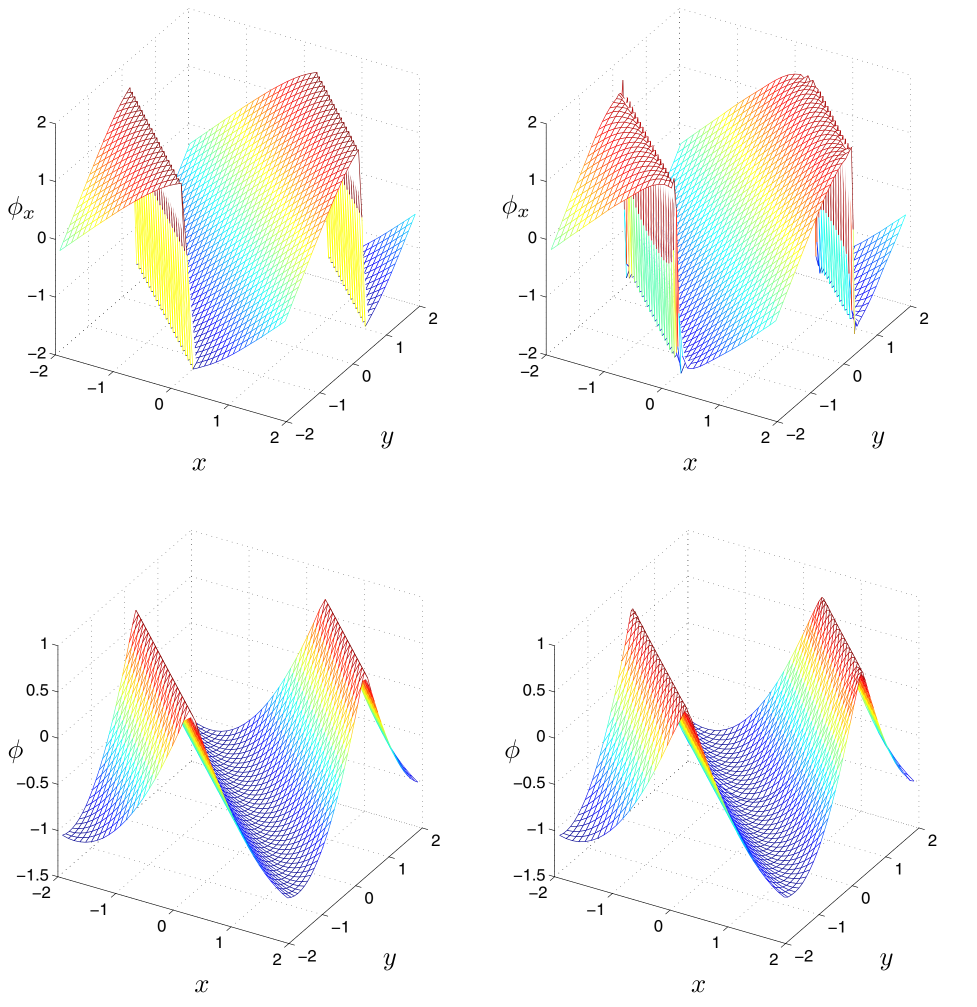

FIG. 10. *Approximations of ϕ_x (top) and ϕ (bottom) at $t = 1.5/\pi^2$ using uniform mesh (left) and moving mesh (right), with $N = 40$ for Example 4.*

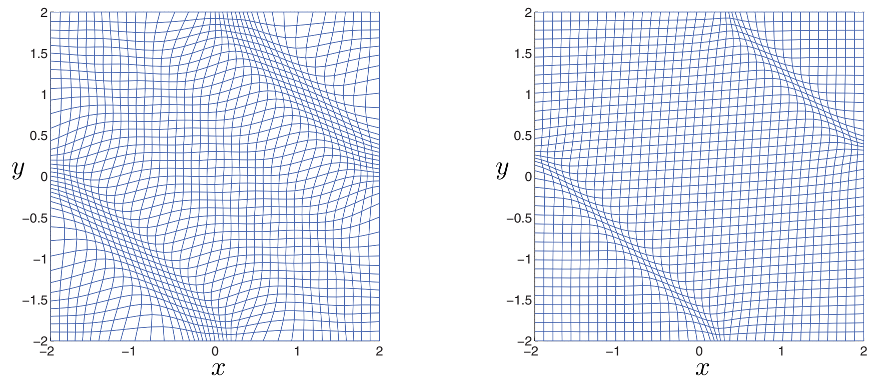

FIG. 11. *Adapted meshes at $t = 0.5/\pi^2$ (left) and $t = 1.5/\pi^2$ (right) obtained with $N = 40$ for Example 4.*

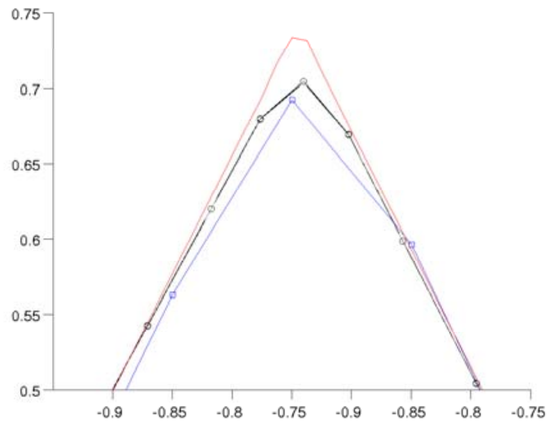

FIG. 12. *Comparison of numerical approximation of ϕ obtained using a uniform mesh with $N = 40$ ($\square$), moving mesh with $N = 40$ ($\circ$), and uniform mesh $N = 320$ (—) for Example* 4.

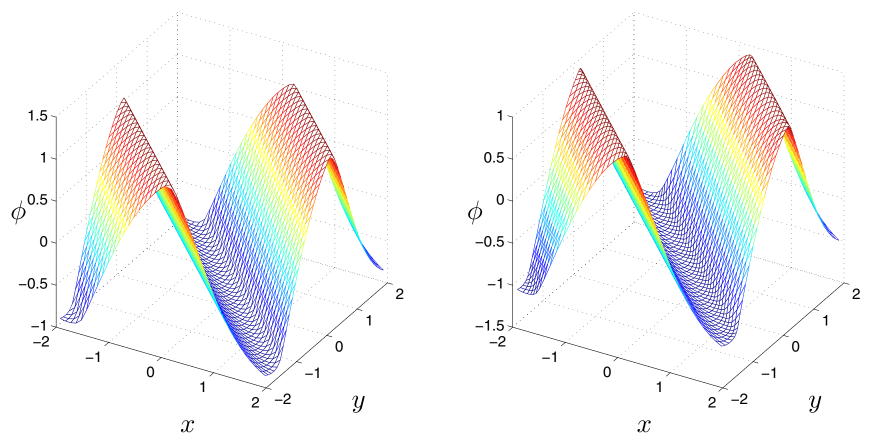

FIG. 13. *Approximations of ϕ at $t = 1.5/\pi^2$ using a uniform mesh (left) and adaptive moving mesh (right), where $CFL = 0.05$ and $N = 40$, for Example* 5.

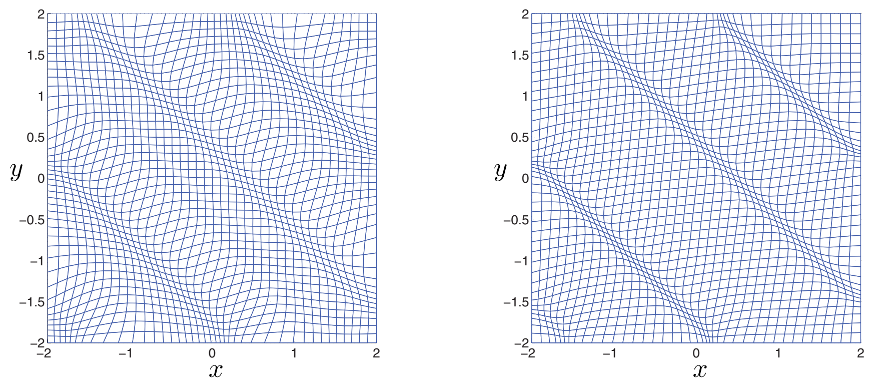

FIG. 14. 40×40 *adaptive meshes at $t = 0.5/\pi^2$ (left) and $t = 1.5/\pi^2$ (right) for Example* 5.

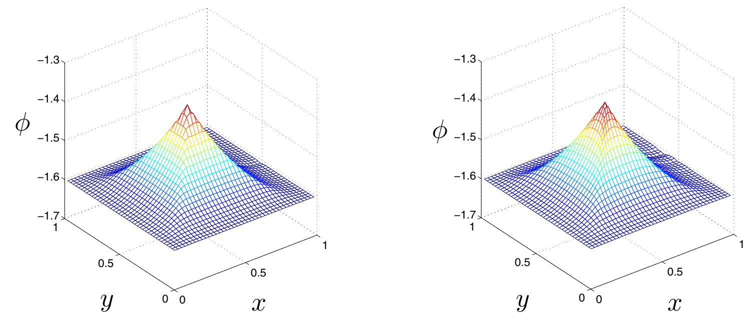

FIG. 15. *Approximations of* ϕ *at* $t = 0.6$ *using a uniform mesh (left) and moving mesh (right) for Example* 6.

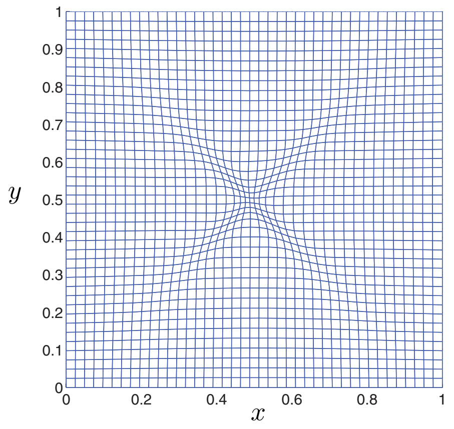

FIG. 16. 40×40 *adapted mesh at* $t = 0.6$ *for Example* 6.

with a unit speed, asymptotically approaching a flat sheet. Figure 15 shows the computed approximations of ϕ using a uniform and adaptive mesh with $N = 40$. We can see that the adaptive moving mesh does a better job of resolving the focusing of the solution. The mesh at the final time is shown in Figure 16, where we can see that the mesh has been concentrated towards the focusing of the solution at the center of the domain.

5. Conclusions.

We have presented a DG method for the solution of nonlinear HJ equations using an adaptive moving mesh refinement strategy. A suitable mesh refinement criterion was used which does not become singular when the solution become nonsmooth. Numerical experiments in one and two dimensions show that this method works well and efficiently delivers high-resolution solutions using relatively coarse meshes.

Although we have focused here on HJ equations, the moving mesh DG method can also be applied to other hyperbolic problems such as the compressible Euler equations, and work is progressing in this direction. Finally, solution adaptivity was achieved in this paper using mesh movement. Future work will include a study of the combination of mesh movement along with additional refinement strategies such as

h and p refinement. The DG framework should hopefully make this combination of strategies easier to implement.

REFERENCES

[1] B. N. Azarenok, *Variational barrier method of adaptive grid generation in hyperbolic problems of gas dynamics*. SIAM J. Numer. Anal., 40 (2002), pp. 651–682.

[2] G. Beckett, J. A. Mackenzie, A. Ramage, and D. M. Sloan, *On the numerical solution of one-dimensional PDEs using adaptive methods based on equidistribution*, J. Comput. Phys., 167 (2001), pp. 372–392.

[3] G. Beckett, J. A. Mackenzie, A. Ramage, and D. M. Sloan, *Computational solution of two-dimensional unsteady PDEs using moving mesh methods*, J. Comput. Phys., 182 (2002), pp. 478–495.

[4] K. S. Bey and J. T. Oden, *hp-version discontinuous Galerkin methods for hyperbolic conservation laws*, Comput. Methods Appl. Mech. Engrg., 133 (1996), pp. 259–286.

[5] B. Cockburn, G. E. Karniadakis, and C. W. Shu, *Discontinuous Galerkin Methods: Theory, Computation and Applications*, Springer, Berlin, 2000.

[6] 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.

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

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

[9] E. A. Dorfi and L. O'C. Drury, *Simple adaptive grids for* 1-*D initial value problems*, J. Comput. Phys., 69 (1987), pp. 175–195.

[10] P. Houston, B. Senior, and E. Suli, *hp-discontinuous Galerkin finite element methods for hyperbolic problems: Analysis and adaptivity*, Internat. J. Numer. Methods Fluids, 40 (2002), pp. 153–169.

[11] 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.

[12] W. Huang, *Practical aspects of formulation and solution of moving mesh partial differential equations*, J. Comput. Phys., 171 (2001), pp. 753–775.

[13] W. Huang and R. D. Russell, *Analysis of moving mesh partial differential equations with spatial smoothing*, SIAM J. Numer. Anal., 34 (1997), pp. 1106–1126.

[14] W. Huang and R. D. Russell, *Moving mesh strategy based on a gradient flow equation for two-dimensional problems*, SIAM J. Sci. Comput., 20 (1999), pp. 998–1015.

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

[16] S. Jin and Z. Xin, *Numerical passage from systems of conservation laws to Hamilton–Jacobi equations, and relaxation schemes*, SIAM J. Numer. Anal., 35 (1998), pp. 2385–2404.

[17] P. Knupp and S. Steinberg, *Fundamentals of Grid Generation*, CRC Press, Boca Raton, FL, 1994.

[18] O. Lepsky, C. Hu, and C. W. Shu, *Analysis of the discontinuous Galerkin method for Hamilton-Jacobi equations*, Appl. Numer. Math., 33 (2000), pp. 423–434.

[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] R. Li and T. Tang, *Moving mesh discontinuous Galerkin method for hyperbolic conservation laws*, J. Sci. Comp., 27 (2006), pp. 347–363.

[21] C.-T. Lin and E. Tadmor, *High-resolution nonoscillatory central schemes for Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 21 (2000), pp. 2163–2186.

[22] S. Osher and J. A. 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. M. Stockie, J. A. Mackenzie, and R. D. Russell, *A moving mesh method for one-dimensional hyperbolic conservation laws*, SIAM J. Sci. Comput., 22 (2001), pp. 1791–1813.

[25] P. K. Sweby, *High resolution schemes using flux limiters for hyperbolic conservation laws*, SIAM J. Numer. Anal., 21 (1984), pp. 995–1011.

[26] Z. Tan, Z. Zhang, Y. Huang, and T. Tang, *Moving mesh methods with locally varying time steps*, J. Comput. Phys., 200 (2004), pp. 347–367.

[27] H. Tang and T. Tang, *Moving mesh methods for one- and two-dimensional hyperbolic conservation laws*, SIAM J. Numer. Anal., 41 (2003), pp. 487–515.

[28] 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.

[29] J. F. Thompson, Z. A. Warsi, and C. W. Mastin, *Numerical Grid Generation: Foundations and Applications*, North–Holland, New York, 1985.

[30] N. K. Yamaleev and M. H. Carpenter, *On accuracy of adaptive grid methods for captured shocks*, J. Comput. Phys., 181 (2002), pp. 280–316.

[31] P. A. Zegeling, W. D. de Boer, and H. Tang, *Robust and efficient adaptive moving mesh solution of the* 2*-D Euler equations*, in Recent Advances in Adaptive Computation, Contemp. Math. 383, AMS, Providence, RI, 2005, pp. 375–386.