Journal of Computational Physics 268 (2014) 134–153

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

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

Yingda Cheng *, Zixuan Wang

Department of Mathematics, Michigan State University, East Lansing, MI 48824, USA

ARTICLE INFO

Article history:
Received 8 November 2013
Received in revised form 7 February 2014
Accepted 21 February 2014
Available online 17 March 2014

Keywords:
Hamilton–Jacobi equation
Discontinuous Galerkin methods
Entropy fix
Unstructured mesh

ABSTRACT

In this paper, we improve upon the discontinuous Galerkin (DG) method for Hamilton–Jacobi (HJ) equation with convex Hamiltonians in [5] and develop a new DG method for directly solving the general HJ equations. The new method avoids the reconstruction of the solution across elements by utilizing the Roe speed at the cell interface. Besides, we propose an entropy fix by adding penalty terms proportional to the jump of the normal derivative of the numerical solution. The particular form of the entropy fix was inspired by Harten and Hyman's entropy fix [12] for Roe scheme for the conservation laws. The resulting scheme is compact, simple to implement even on unstructured meshes, and is demonstrated to work for nonconvex Hamiltonians. Benchmark numerical experiments in one dimension and two dimensions are provided to validate the performance of the method.

1. Introduction

In this paper, we consider the numerical solution of the time-dependent Hamilton–Jacobi (HJ) equation

$$\varphi_t + H(\nabla_{\mathbf{x}}\varphi, \mathbf{x}) = 0, \qquad \varphi(\mathbf{x}, 0) = \varphi^0(\mathbf{x}), \quad \mathbf{x} \in \Omega \in \mathbb{R}^d \tag{1.1}$$

with suitable boundary conditions on $\partial\Omega$. The HJ equation arises in many applications, e.g., optimal control, differential games, crystal growth, image processing and calculus of variations. The solution of such equation may develop discontinuous derivatives in finite time even when the initial data is smooth. The viscosity solution [7,8] was introduced as the unique physically relevant solution, and has been the focus of many numerical methods. Starting from [9,24], finite difference methods such as essentially non-oscillatory (ENO) [19,20] or weighted ENO (WENO) methods [14,27] have been developed to solve the HJ equation. Those finite difference methods work quite efficiently for Cartesian meshes, however they lose the advantage of simplicity on unstructured meshes [1,27].

Alternatively, the Runge–Kutta discontinuous Galerkin (RKDG) method, originally devised to solve the conservation laws [6], is more flexible for arbitrarily unstructured meshes. The first work of DG methods for HJ equations [13,17] relies on solving the conservation law system satisfied by the derivatives of the solution. The methods work well numerically even on unstructured mesh, with provable stability results for certain special cases, and were later generalized in e.g. [4,11]. Unfortunately, the procedure of recovering φ from its derivatives has made the algorithm indirect and complicated. In contrast, the design of DG methods for directly solving the HJ equations is appealing but challenging, because the HJ equation

☆ Research supported by NSF grants DMS-1217563, DMS-1318186, AFOSR grant FA9550-12-1-0343 and the startup fund from Michigan State University.
* Corresponding author.
E-mail addresses: ycheng@math.msu.edu (Y. Cheng), wangzix1@msu.edu (Z. Wang).

http://dx.doi.org/10.1016/j.jcp.2014.02.041
0021-9991/

is not written in the conservative form, for which the framework of DG methods could easily apply. In [5], a DG method for directly solving the HJ equation was developed. This scheme has provable stability and error estimates for linear equations and demonstrates good convergence to the viscosity solutions for nonlinear equations. However, in entropy violating cells, a correction based on the schemes in [13,17] is necessary to guarantee stability of the method, and moreover, the method in [5] only works for equations with convex Hamiltonians. Later, this algorithm was applied to solve front propagation problems [3] and front propagation with obstacles [2], in which simplified implementations of the entropy fix procedure were proposed. Meanwhile, central DG [18] and local DG [26] methods were recently developed for the HJ equation. Numerical experiments demonstrate that both methods work for nonconvex Hamiltonians. In addition, the first order version of the local DG method [26] reduces to the monotone schemes and thus has provable convergence properties. However, the central DG methods based on overlapping meshes are difficult to implement on unstructured meshes, and the local DG methods still need to resort to the information about the derivatives of φ, making the method less direct in computation. L^2 error estimates for smooth solutions of the DG method [5] and local DG method [26] have been established in [25]. For recent developments of high order and DG methods for HJ equations, one can refer to the review papers [21,22].

In this paper, we improve upon the DG scheme in [5] and develop a new DG method for directly solving the general HJ equation. Based on the observation that the method in [5] is closely related to Roe's linearization, we use interfacial terms involving the Roe speed and develop a new entropy fix that was inspired by Harten and Hyman's entropy fix [12] for Roe scheme for the conservation laws. The new method has the following advantages. Firstly, the scheme is shown to work on unstructured meshes even for nonconvex Hamiltonian. Secondly, the method is simple to implement. The cumbersome L^2 reconstruction of the solutions' derivative at the cell interface in [5] is avoided, and the entropy fix is automatically incorporated by the added jump terms in the derivatives of the numerical solution. Finally, the scheme is direct and compact, and the computation only needs the information about the current cell and its immediate neighbors.

The rest of the paper is organized as follows: in Section 2, we introduce the numerical schemes for one-dimensional and multi-dimensional HJ equations. Section 3 is devoted to the discussion of the numerical results. We conclude and discuss about future work in Section 4.

2. Numerical methods

In this section, we will describe the numerical methods. We follow the method of lines approach, and below we will only describe the semi-discrete DG schemes. The resulting method of lines ODEs can be solved by the total variation diminishing (TVD) Runge–Kutta methods [23] or strong stability preserving Runge–Kutta methods [10].

2.1. Scheme in one dimension

In this subsection, we will start by designing the scheme for one-dimensional HJ equation. In this case, (1.1) becomes

$$\varphi_t + H(\varphi_x, x) = 0, \qquad \varphi(x,0) = \varphi^0(x). \tag{2.2}$$

Assume the computational domain is $[a,b]$, we will divide it into N cells as follows

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

Now the cells and their centers are defined as

$$I_j = (x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}), \quad x_j = \frac{1}{2}(x_{j-\frac{1}{2}} + x_{j+\frac{1}{2}}), \ j = 1, \ldots, N \tag{2.4}$$

and the mesh sizes are

$$\Delta x_j = x_{j+\frac{1}{2}} - x_{j-\frac{1}{2}}, \qquad h = \max_j \Delta x_j. \tag{2.5}$$

The DG approximation space is

$$V_h^k = \left\{ v\colon\ v|_{I_j} \in P^k(I_j),\ j = 1, \ldots, N \right\} \tag{2.6}$$

where $P^k(I_j)$ denotes all polynomials of degree at most k on I_j, and we let $H_1 = \frac{\partial H}{\partial \varphi_x}$ be the partial derivative of the Hamiltonian with respect to φ_x.

To introduce the scheme, we need to define several quantities at the cell interface where the DG solution is discontinuous. If x_* is a point located at the cell interface, then $\varphi_h \in V_h^k$ would be discontinuous at x_*. We can then define the Roe speed at x_* to be

$$\tilde{H}_{\varphi_h}(x_*) := \begin{cases} \frac{H((\varphi_h)_x(x_*^+), x_*^+) - H((\varphi_h)_x(x_*^-), x_*^-)}{(\varphi_h)_x(x_*^+) - (\varphi_h)_x(x_*^-)}, & \text{if } (\varphi_h)_x(x_*^+) \neq (\varphi_h)_x(x_*^-) \\ \frac{1}{2}(H_1((\varphi_h)_x(x_*^+), x_*^+) + H_1((\varphi_h)_x(x_*^-), x_*^-)), & \text{if } (\varphi_h)_x(x_*^+) = (\varphi_h)_x(x_*^-). \end{cases}$$