J Sci Comput (2017) 73:1316–1337
DOI 10.1007/s10915-017-0440-9

An *h*-Adaptive RKDG Method for the Two-Dimensional Incompressible Euler Equations and the Guiding Center Vlasov Model

Hongqiang Zhu[1] · **Jianxian Qiu**[2] · **Jing-Mei Qiu**[3]

Received: 22 December 2016 / Revised: 7 April 2017 / Accepted: 19 April 2017 /
Published online: 6 May 2017

Abstract In this paper, we generalize an *h*-adaptive Runge–Kutta discontinuous Galerkin scheme developed earlier in Zhu et al. (J Sci Comput 69:1346–1365, 2016) for the 1D Vlasov–Poisson system to the guiding center Vlasov model and the 2D time dependent incompressible Euler equations in the vorticity-stream function formulation. The main difficulty of this generalization lies in solving the 2D Poisson equation due to the irregular adaptive mesh with hanging nodes. We adopt a local discontinuous Galerkin method to solve the Poisson equation. The full adaptive algorithm and the related numerical implementation details are included. Extensive numerical tests have been performed to showcase the effectiveness of the adaptive scheme and its advantage over the fixed-mesh scheme in saving computational cost and improving solution quality.

Keywords Runge–Kutta discontinuous Galerkin · Incompressible Euler equations · *h*-Adaptive · Guiding center Vlasov model

Dedicated to Prof. Chi-Wang Shu on the occasion of his 60th birthday.

The research is partially supported by NSFC Grants 11201242 and 11571290, NSAF Grant U1630247, NSF Grant NSF-DMS-1522777, and Air Force Office of Scientific Computing FA9550-16-1-0179.

✉ Jing-Mei Qiu
jingqiu@math.uh.edu

Hongqiang Zhu
zhuhq@njupt.edu.cn

Jianxian Qiu
jxqiu@xmu.edu.cn

1 School of Natural Sciences, Nanjing University of Posts and Telecommunications, Nanjing 210023, Jiangsu, People's Republic of China

2 School of Mathematical Sciences and Fujian Provincial Key Laboratory of Mathematical Modeling and High-Performance Scientific Computing, Xiamen University, Xiamen 361005, Fujian, People's Republic of China

3 Department of Mathematics, University of Houston, Houston, TX 77204-3008, USA

1 Introduction

We are interested in solving the two-dimensional (2D) time-dependent incompressible Euler equation in the vorticity-stream function formulation. They consist of a convective transport equation for the scalar vorticity and a Poisson equation for the stream function

$$\omega_t + \nabla \cdot (\mathbf{u}\omega) = 0 \qquad \text{on } \Omega \times (0, T), \tag{1.1a}$$

$$\Delta\psi = \omega \qquad \text{on } \Omega \times (0, T), \tag{1.1b}$$

where $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)$, $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ and the velocity vector $\mathbf{u} = \left(-\frac{\partial \psi}{\partial y}, \frac{\partial \psi}{\partial x}\right)$. Another model of our interest is the guiding center approximation of the 2D Vlasov–Poisson system

$$\frac{\partial \rho}{\partial t} + \mathbf{E}^{\perp} \cdot \nabla \rho = 0, \quad -\Delta \psi = \rho, \quad \mathbf{E} = -\nabla \psi. \tag{1.2}$$

where ρ is the charge density, $\mathbf{E}^{\perp} = (-\frac{\partial \psi}{\partial y}, \frac{\partial \psi}{\partial x})$ with the electric field $\mathbf{E}$ determined by the 2D Poisson equation for the potential. Note that due to the incompressible condition for the vector field $\mathbf{E}^{\perp}$, the guiding center model can also be written in a conservative form $\frac{\partial \rho}{\partial t} + \nabla \cdot (\mathbf{E}^{\perp}\rho) = 0$. Despite different application backgrounds, the above two models are in an equivalent mathematical formulation up to a sign difference in the Poisson equation. The algorithm developed for one model can be directly applied to the other without any difficulty. We focus on algorithm development for the system (1.1), while presenting results from numerical tests for both models.

Adaptive methods are widely used to increase spatial and temporal resolution of numerical simulations beyond the limits imposed by the available hardware and to save the computational cost. The discontinuous Galerkin (DG) framework offers great convenience in data projection and data prolongation among different levels of mesh refinement in an h-adaptive scheme, while preserving the mass conservation. Moreover, compared with finite volume and finite difference methods, finite element DG methods are advantageous for their compactness, handling boundary conditions in a complex geometry, as well as ease for parallelization. In [30], we develop an h-adaptive Runge–Kutta discontinuous Galerkin (RKDG) scheme for simulating the 1D Vlasov–Poisson (VP) system, where an adaptive RKDG scheme is used for updating the Vlasov solution, while a local DG (LDG) scheme is used for solving the Poisson equation. This paper is a natural extension of our previous work. The major difference between our work in [30] and the current one is the Poisson solver. For the 1D VP system, the Poisson equation is 1D, while here we have a 2D Poisson system. Such difference, on one hand, makes the h-adaptive scheme more attractive since the cell number of a mesh is essential to the computational cost (both storage and CPU time) especially for solving elliptic equations. On the other hand, the need to solve the 2D Poisson system on an adaptive mesh brings up additional complications in implementation due to the appearance of hanging nodes. In our paper, we will provide a comprehensive description on our implementation strategy for the LDG scheme on the h-adaptive mesh.

Existing work for model problem (1.1) include the second-order Godunov-type upwinding methods [1], central finite difference methods [14], essentially non-oscillatory (ENO) methods [24], finite element DG methods [15,25] and semi-Lagrangian methods [4,7,20,23]. There have been many existing work on developing semi-Lagrangian methods for the guiding center Vlasov model [8,18,23]. Research work on the DG method with mesh adaptation for these model problems is rare. There are plenty of h-adaptive RKDG schemes for other model problems, for example, the ones developed by Flaherty et al. [2,10,11,21], Zhu et al.'[26–30] and Hartmann and Houston [12].

This paper is organized as follows. We first review the RKDG scheme for the 2D convective transport equation and the LDG scheme for the 2D Poisson equation in Sect. 2. Then in Sect. 3 we provide details in implementing the proposed h-adaptive RKDG–LDG scheme, especially for solving the 2D Poisson equation with adaptive meshes. Numerical results are presented in Sect. 4 and concluding remarks are followed in Sect. 5.

2 Review of RKDG–LDG Scheme

When fixed mesh is adopted, the RKDG–LDG method solves the model problem (1.1) in the following way. At each time level, the stream function ψ, hence the velocity vector $\mathbf{u}$, is firstly computed by numerically solving Eq. (1.1b) using the LDG scheme. Then, the vorticity ω is updated by applying the DG scheme to Eq. (1.1a). Such procedure can be repeated for every RK stage of one time step evolution, until the final time is reached. The h-adaptive approach is obtained by inserting a mesh-adaptation step at the beginning of each time level.

In this section we first briefly review the RKDG scheme for Eq. (1.1a). Detailed description of this method can be found in the review paper [6]. Then the LDG scheme [3] for Eq. (1.1b) is presented.

In our description below, we use the subscript h to denote the numerical discretization of the corresponding continuous function. For example, ω_h and $\mathbf{u}_h$ are the numerical approximations to the unknown functions ω and $\mathbf{u}$ respectively.

2.1 RKDG Scheme

Given a mesh discretization $\mathcal{M}_h$ of the domain Ω which only includes rectangular elements, we seek the approximate solution $\omega_h(x, y, t)$ in the finite element space of discontinuous piecewise polynomials

$$V_h^k = \left\{ \phi \in L^2(\Omega) : \phi|_K \in \mathbb{P}^k(K), \forall K \in \mathcal{M}_h \right\}, \tag{2.1}$$

where $\mathbb{P}^k(K)$ denotes the set of polynomials of total degree at most k on the element K. It is known that the dimension of such space is $Q_k + 1$ with $Q_k = k(k+3)/2$. We adopt the following orthogonal basis on the reference cell $[-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{2}, \frac{1}{2}]$ for V_h^k,

$$k = 1: \quad \phi_0 = 1, \quad \phi_1 = x, \quad \phi_2 = y \tag{2.2}$$

$$k = 2: \quad \phi_0,\ \phi_1,\ \phi_2,\ \phi_3 = x^2 - \frac{1}{12}, \quad \phi_4 = xy, \quad \phi_5 = y^2 - \frac{1}{12}. \tag{2.3}$$

Then the local orthogonal basis over cell K is given by

$$\phi_l^{(K)}(x, y) = \phi_l\left(\frac{x - x_K}{\Delta x_K}, \frac{y - y_K}{\Delta y_K} \right), \quad l = 0, \ldots, Q_k, \tag{2.4}$$

in which (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 $\omega_h(x, y, t)$ in the space V_h^k can be expressed as

$$\omega_h(x, y, t)|_K = \sum_{l=0}^{Q_k} \omega_K^{(l)}(t) \phi_l^{(K)}(x, y) \tag{2.5}$$

where $\omega_K^{(l)}(t), l = 0, \ldots, Q_k$ are the degrees of freedom. Particularly, $\omega_K^{(0)}(t)$ is the cell average of ω_h over K.

To obtain the RKDG scheme, we multiply Eq. (1.1a) by each of the basis, integrate over each computational cell and perform integration by parts to formulate the following weak form for the approximate solution ω_h: for all test functions $\phi_l^{(K)}$, with $l = 0, \ldots, Q_k$ and all $K \in \mathcal{M}_h$,

$$\frac{d}{dt}\int_K \omega_h \phi_l^{(K)}\, dxdy - \int_K F(\omega_h)\cdot\nabla\phi_l^{(K)}\, dxdy + \sum_{e\in\partial K}\int_e F(\omega_h)\cdot n_{e,K}\phi_l^{(K)}\, ds = 0 \tag{2.6}$$

where $F(\omega_h) = \mathbf{u}_h\omega_h$ and $n_{e,K}$ is the outward unit normal to the edge e. The volume integral term $\int_K F(\omega_h)\cdot\nabla\phi_l^{(K)}\, dxdy$ can be computed either exactly or by a numerical quadrature of sufficiently high order of accuracy. The line integral in Eq. (2.6) is typically discretized by a Gaussian quadrature with sufficient accuracy

$$\int_e F(\omega_h)\cdot n_{e,K}\phi_l^{(K)}\, ds \approx |e|\sum_{l=1}^{m}\omega_l F(\omega_h(G_l,t))\cdot n_{e,K}\phi_l^{(K)}(G_l,t) \tag{2.7}$$

where $F(\omega_h(G_l,t))\cdot n_{e,K}$ is replaced by a monotone numerical flux. In this paper, we use the simple Lax–Friedrichs flux

$$\begin{aligned} F(\omega_h(G_l,t))\cdot n_{e,K} \approx \frac{1}{2}\Big[&\left(F\left(\omega_h^-(G_l,t)\right) + F\left(\omega_h^+(G_l,t)\right)\right)\cdot n_{e,K} \\ &-\alpha\left(\omega_h^+(G_l,t) - \omega_h^-(G_l,t)\right)\Big] \end{aligned} \tag{2.8}$$

where α is taken as an upper bound for the eigenvalues of the Jacobian in the direction of $n_{e,K}$, and ω_h^- and ω_h^+ are the traces of ω_h at the Gaussian point G_l inside and outside the cell K respectively.

Using Eq. (2.5), the first term in Eq. (2.6) can be rewritten by $c_l\Delta x_K\Delta y_K\frac{d}{dt}\omega_K^{(l)}(t)$ where $c_l = \int_{-1/2}^{1/2}\int_{-1/2}^{1/2}\phi_l^2(x,y)\, dxdy, l = 0, \ldots, Q_k$ are constants. As a result, the semi-discrete scheme (2.6–2.8) is an ODE system for $\omega_K^{(l)}(t), l = 0, \ldots, Q_k, K \in \mathcal{M}_h$. This system coupled with a suitable time discretization scheme, such as the TVD (total variation diminishing) RK method [22], completes the RKDG scheme. In this paper for $k = 1$, we use the second order RK time stepping

$$\begin{aligned} \Phi^{(1)} &= \Phi^n + \Delta t L(\Phi^n), \\ \Phi^{n+1} &= \frac{1}{2}\Phi^n + \frac{1}{2}\Phi^{(1)} + \frac{1}{2}\Delta t L(\Phi^{(1)}) \end{aligned} \tag{2.9}$$

for the ODE system $\Phi_t = L(\Phi)$. For $k = 2$, we adopt the following third order version

$$\begin{aligned} \Phi^{(1)} &= \Phi^n + \Delta t L(\Phi^n), \\ \Phi^{(2)} &= \frac{3}{4}\Phi^n + \frac{1}{4}\Phi^{(1)} + \frac{1}{4}\Delta t L(\Phi^{(1)}), \\ \Phi^{n+1} &= \frac{1}{3}\Phi^n + \frac{2}{3}\Phi^{(2)} + \frac{2}{3}\Delta t L(\Phi^{(2)}). \end{aligned} \tag{2.10}$$

2.2 LDG Scheme

To define the LDG scheme for the 2D Poisson Eq. (1.1b), we introduce the auxiliary variable $\mathbf{g} = \nabla\psi$ and rewrite the equation as a system of first-order equations:

$$\nabla\psi = \mathbf{g} \quad \text{in } \Omega, \tag{2.11a}$$

$$\nabla \cdot \mathbf{g} = \omega \quad \text{in } \Omega. \tag{2.11b}$$

Periodic boundary conditions are assumed. Beyond periodicity, we need to enforce some additional conditions to uniquely determine ψ. In current work we set $\int_\Omega \psi(x, y, t)\, dxdy = 0$.

The LDG scheme is established on the same mesh $\mathcal{M}_h$ as the RKDG scheme. We make use of the approximation space V_h^k in (2.1) and

$$\mathbf{Z}_h^k = \left\{ \mathbf{r} \in (L^2(\Omega))^2 : \mathbf{r}|_K \in (P^k(K))^2, \forall K \in \mathcal{M}_h \right\},$$

then the approximate solution $(\psi_h, \mathbf{g}_h)$ is defined by the following weak formulation: for all $K \in \mathcal{M}_h$ and all test functions $(\mathbf{r}, v) \in \mathbf{Z}_h^k \times V_h^k$,

$$\sum_{e\in\partial K} \int_e \hat{\psi}_h \mathbf{r} \cdot n_{e,K}\, ds - \int_K \psi_h \nabla \cdot \mathbf{r}\, dxdy = \int_K \mathbf{g}_h \cdot \mathbf{r}\, dxdy, \tag{2.12a}$$

$$\sum_{e\in\partial K} \int_e v \hat{\mathbf{g}}_h \cdot n_{e,K}\, ds - \int_K \mathbf{g}_h \cdot \nabla v\, dxdy = \int_K \omega v\, dxdy. \tag{2.12b}$$

The functions $\hat{\psi}_h$ and $\hat{\mathbf{g}}_h$ in Eq. (2.12) are the so called numerical fluxes. To define them, we need to introduce some notations. We define the averages $\{\{\cdot\}\}$ and jumps $[\![\cdot]\!]$ at $(x, y) \in e \subset K$:

$$\{\{\psi\}\} = (\psi^+ + \psi^-)/2, \quad \{\{\mathbf{g}\}\} = \left(\mathbf{g}^+ + \mathbf{g}^-\right)/2, \tag{2.13}$$

$$[\![\psi]\!] = \left(\psi^- - \psi^+\right) n_{e,K}, \quad [\![\mathbf{g}]\!] = \left(\mathbf{g}^- - \mathbf{g}^+\right) \cdot n_{e,K} \tag{2.14}$$

where the superscripts "−" and "+" again denotes the traces of corresponding functions inside and outside the cell K. Then the numerical fluxes in Eq. (2.12) are defined as follows:

$$\hat{\mathbf{g}} = \{\{\mathbf{g}\}\} - C_{11}[\![\psi]\!] - \mathbf{C}_{12}[\![\mathbf{g}]\!], \tag{2.15a}$$

$$\hat{\psi} = \{\{\psi\}\} + \mathbf{C}_{12} \cdot [\![\psi]\!]. \tag{2.15b}$$

As suggested in [3,5], we set the stabilization parameter $C_{11}(e) = 1$ and the auxiliary parameter $\mathbf{C}_{12}(e)$ so that

$$\mathbf{C}_{12}(e) \cdot n_{e,K} = 0.5 \cdot \text{sign}(\mathbf{v}_0 \cdot n_{e,K}) \tag{2.16}$$

where $\mathbf{v}_0$ is an arbitrary but fixed vector with nonzero components.

The function $\mathbf{g}_h$ can be locally solved in terms of ψ_h through Eq. (2.12a) since $\hat{\psi}_h$ does not depend on $\mathbf{g}_h$. Hence it can be eliminated from Eq. (2.12b), leaving ψ_h as the only unknown. The resulting system is a linear system for ψ_h which is solved in our numerical tests by the generalized minimum residual method equipped with the incomplete LU decomposition for the preconditioning.

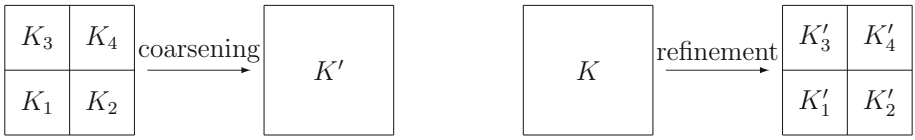

Fig. 1 Sketches of local mesh coarsening (*left*) and refinement (*right*)

3 Algorithm and Implementation Details

In this section, we propose the h-adaptive RKDG–LDG scheme for the model problem and discuss the related implementation issues. This scheme is a generalization of the scheme in [30], so we would like to focus this section on "what the scheme is" and "how it is implemented", and omit the reasoning if it remains the same as in [30]. From the standpoint of numerical simulations, the main difference of the scheme in this paper lies in the 2D Poisson solver, which will receive great emphasis in this section.

We will first present the grid and data structure, followed by the flow chart of our proposed adaptive algorithm. Specific issues, such as the adaptation criteria, data prolongation/projection and the coding of the LDG scheme will be discussed afterwards.

3.1 Grid and Data Structure

Each cell in the initial partition of the computational domain is considered as the *root* of a tree. We use a simple way to perform local mesh refinement and coarsening. More specifically, mesh refinement is achieved by dividing a cell into four new cells (*children*) of equal size (see the sketch on the right in Fig. 1). The four new cells, which are generated simultaneously in a single division, are called a 'GROUP'. To coarsen the mesh, we can only merge the cells that form a GROUP (see the left sketch in Fig. 1). A cell is called a *leaf* cell if it does not have any children. All the leaf cells constitute the computational mesh. Each leaf cell has a variable L denoting its mesh-level which is defined by the number of divisions needed to obtain this cell. L is set to zero for roots, and is increased by one after a division and decreased by one after a merger. We also need a maximal mesh-level, denoted by LEV, to restrict fineness of the adaptive mesh.

We describe the data structure associated with each leaf cell and non-leaf cell. For the non-leaf cells, the associated data include cell coordinates, mesh-level and the pointers to its father and children in the tree. For the leaf cells, additional data such as degrees of freedom for solutions, pointers for neighboring cells in four directions, indicator for mesh refinement and coarsening (value $-1/1$ for mesh coarsening/refinement respectively), and indicator for boundary cells (1 for boundary cells and 0 otherwise) are stored.

Since we need to visit all leaf cells and perform some operations for quite a few times in each time step, we propose to order all leaf cells and store them as a vector of pointers to facilitate the revisit of leaf cells. In particular, all leaf cells are naturally ordered by traversing the tree one by one. Each time the mesh is adaptively updated, the vector of pointers must be updated. Such order is also used in the formation of a large linear system for the LDG scheme.

3.2 Algorithm Flow Chart

Below, we first describe the flow chart of the algorithm. In our descriptions, the superscript n stands for the current time level t^n unless otherwise specified. For each of the computational

cell K (a leaf on the tree), associated information include the DG solution with $\{\omega_K^{(l)} : l = 0, \ldots, Q_k\}$ as its degrees of freedom and its mesh level L_K. Here and below, argument t in $\omega_K^{(l)}(t)$ is omitted for simplicity.

Algorithm 3.1 (*h-adaptive RKDG–LDG scheme*)

- *The initial set up* The algorithm starts from an uniform rectangular mesh $\mathcal{M}_h^0$ as the root grid. We perform the L^2 projection of the initial data $\omega(x, y, t = 0)$ on $\mathcal{M}_h^0$. The numerical solution is a piecewise polynomial of degree k living on $\mathcal{M}_h^0$. Associated with each root cell $K \in \mathcal{M}_h^0$, we have the following

$$\left\{\omega_K^{(l),0} : l = 0, \ldots, Q_k\right\}, \quad L_K^0 = 0.$$

- *Solution evolution from t^n to t^{n+1}, for $n = 0, 1, \ldots$*
  1. *Mesh refinement and coarsening* Each cell in the current mesh will be marked to be refined, coarsened, or kept unchanged via the "adaptive indicator" discussed in Sect. 3.3. We take a cell $K \in \mathcal{M}_h^n$ as an example to demonstrate the idea.
     - The cell will be quartered if it is marked to be refined and its mesh-level $L_K < LEV$. There are four newly generated cells $K_1, \ldots, K_4$ (children), each of which will have its mesh-level increased by one from that of its parent cell. The corresponding polynomial on new cells will be obtained from data prolongation mechanism discussed in Sect. 3.4.
     - Each GROUP of cells (four children) will be removed if all of them are marked to be merged. Mesh-level is decreased by one for the new cell. The corresponding polynomial on the new leaf cell will be obtained from data projection discussed in Sect. 3.4.
  2. *Poisson solver in physical space* Solve Eq. (1.1b) by LDG scheme to get ψ_h and $\mathbf{u}_h$.
  3. *Evolve solution* Evolve the solution on the current mesh from t_n to t_{n+1} by applying the RKDG procedure to Eq. (1.1a) to update $\{\omega_K^{(l),n+1} : l = 0, \ldots, Q_k, \forall K \in \mathcal{M}_h^{n+1}\}$.

3.3 Adaptation Criteria

One key component in an h-adaptive method is the adaptation criteria for mesh refinement and coarsening. For computational efficiency and efficacy, the mesh is to be refined around the region where local errors are relatively large, and is to be coarsened otherwise. A cell is marked to be refined, coarsened or kept unchanged according to some 'adaptive indicator'. In this work, we propose to use the solution variations measured by functions' gradient as described below, as an effective adaptive indicator. In particular, in the context of a DG scheme, the solution variations on a cell K in x and y directions can be approximated by $|\omega_K^{(1)}|$ and $|\omega_K^{(2)}|$ in Eq. (2.5) respectively. A cell will be marked to be refined (or coarsened) if its variations are much larger (or smaller) than the average values. Notice that there are many other types of adaptive indicator in literature. For example, in [19], a list of trouble-cell indicators were compared for solving hyperbolic problems. A posteriori error estimates are often used for an adaptive indicator. However, to our knowledge, an a posterior error analysis is not yet available for the model problems we considered in this paper.

Algorithm 3.2 (*Adaptive indicator*) For every cell $K \in \mathcal{M}_h$, let $\theta_x^K = |\omega_K^{(1)}|$, $\theta_y^K = |\omega_K^{(2)}|$. We compute their average values in the corresponding directions

$$\bar{\theta}_x = \frac{\Sigma_K \theta_x^K}{N}, \quad \bar{\theta}_y = \frac{\Sigma_K \theta_y^K}{N}, \tag{3.1}$$

with N being the total number of cells. Let $\bar{\theta}_{max} = \max(\bar{\theta}_x, \bar{\theta}_y)$, then a cell K is marked to be

$$\begin{cases} \text{coarsened}, & \text{if } \theta_x^K < \gamma_1 \bar{\theta}_{max} \text{ and } \theta_y^K < \gamma_1 \bar{\theta}_{max}, \\ \text{refined}, & \text{if } \theta_x^K > \gamma_2 \bar{\theta}_{max} \text{ or } \theta_y^K > \gamma_2 \bar{\theta}_{max}, \\ \text{kept unchanged}, & \text{otherwise.} \end{cases} \tag{3.2}$$

In our numerical tests, we set $\gamma_1 = 1/2$ and $\gamma_2 = 2$.

3.4 Data Prolongation and Data Projection

The L^2 projection is performed for data prolongation and projection between different levels of meshes. The formulas are shown below.

- *Data 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 are as follows for the DG scheme with P^2 polynomial space,

$$\begin{aligned}
\omega_{K'}^{(0)} &= \frac{1}{4}\left(\omega_{K_1}^{(0)} + \omega_{K_2}^{(0)} + \omega_{K_3}^{(0)} + \omega_{K_4}^{(0)}\right), \\
\omega_{K'}^{(1)} &= \frac{3}{4}\left(-\omega_{K_1}^{(0)} + \omega_{K_2}^{(0)} - \omega_{K_3}^{(0)} + \omega_{K_4}^{(0)}\right) + \frac{1}{8}\left(\omega_{K_1}^{(1)} + \omega_{K_2}^{(1)} + \omega_{K_3}^{(1)} + \omega_{K_4}^{(1)}\right), \\
\omega_{K'}^{(2)} &= \frac{3}{4}\left(-\omega_{K_1}^{(0)} - \omega_{K_2}^{(0)} + \omega_{K_3}^{(0)} + \omega_{K_4}^{(0)}\right) + \frac{1}{8}\left(\omega_{K_1}^{(2)} + \omega_{K_2}^{(2)} + \omega_{K_3}^{(2)} + \omega_{K_4}^{(2)}\right), \\
\omega_{K'}^{(3)} &= \frac{15}{16}\left(-\omega_{K_1}^{(1)} + \omega_{K_2}^{(1)} - \omega_{K_3}^{(1)} + \omega_{K_4}^{(1)}\right) + \frac{1}{16}\left(\omega_{K_1}^{(3)} + \omega_{K_2}^{(3)} + \omega_{K_3}^{(3)} + \omega_{K_4}^{(3)}\right), \\
\omega_{K'}^{(4)} &= \frac{9}{4}\left(\omega_{K_1}^{(0)} - \omega_{K_2}^{(0)} - \omega_{K_3}^{(0)} + \omega_{K_4}^{(0)}\right) + \frac{3}{8}\left(-\omega_{K_1}^{(1)} - \omega_{K_2}^{(1)} + \omega_{K_3}^{(1)} + \omega_{K_4}^{(1)}\right) \\
&\quad + \frac{3}{8}\left(-\omega_{K_1}^{(2)} + \omega_{K_2}^{(2)} - \omega_{K_3}^{(2)} + \omega_{K_4}^{(2)}\right) + \frac{1}{16}\left(\omega_{K_1}^{(4)} + \omega_{K_2}^{(4)} + \omega_{K_3}^{(4)} + \omega_{K_4}^{(4)}\right), \\
\omega_{K'}^{(5)} &= \frac{15}{16}\left(-\omega_{K_1}^{(2)} - \omega_{K_2}^{(2)} + \omega_{K_3}^{(2)} + \omega_{K_4}^{(2)}\right) + \frac{1}{16}\left(\omega_{K_1}^{(5)} + \omega_{K_2}^{(5)} + \omega_{K_3}^{(5)} + \omega_{K_4}^{(5)}\right).
\end{aligned} \tag{3.3}$$

  For the DG scheme with P^1 polynomial space, only the first three formulas are needed.
- *Data prolongation* 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 $k = 2$ can be computed by the following formulas with $l = 1, 2, 3, 4$,

$$\begin{aligned}
\omega_{K_l'}^{(0)} &= \omega_K^{(0)} + \lambda_x^{(l)}\omega_K^{(1)} + \lambda_y^{(l)}\omega_K^{(2)} + \lambda_x^{(l)}\lambda_y^{(l)}\omega_K^{(4)}, \\
\omega_{K_l'}^{(1)} &= \frac{1}{2}\omega_K^{(1)} + \lambda_x^{(l)}\omega_K^{(3)} + \frac{1}{2}\lambda_y^{(l)}\omega_K^{(4)}, \quad \omega_{K_l'}^{(2)} = \frac{1}{2}\omega_K^{(2)} + \frac{1}{2}\lambda_x^{(l)}\omega_K^{(4)} + \lambda_y^{(l)}\omega_K^{(5)}, \\
\omega_{K_l'}^{(3)} &= \frac{1}{4}\omega_K^{(3)}, \quad \omega_{K_l'}^{(4)} = \frac{1}{4}\omega_K^{(4)}, \quad \omega_{K_l'}^{(5)} = \frac{1}{4}\omega_K^{(5)}
\end{aligned} \tag{3.4}$$

where $\lambda_x^{(l)} = \frac{(-1)^l}{4}$ for $l = 1, \ldots, 4$, $\lambda_y^{(1)} = \lambda_y^{(2)} = -\frac{1}{4}$ and $\lambda_y^{(3)} = \lambda_y^{(4)} = \frac{1}{4}$. For $k = 1$, one can use the same formulas, but dropping the higher moment terms $\omega_K^{(3)}$, $\omega_K^{(4)}$, and $\omega_K^{(5)}$.

3.5 Coding the LDG Scheme

Both the RKDG and LDG schemes are capable of handling irregular meshes, including the ones with hanging nodes. To implement both schemes with the h-adaptive meshes in this work, it is important that the interval $e \subset \partial K$ in Eqs. (2.6) and (2.12) must be an elementary edge that contains no sub-edge. In doing so, the mass conservation will be guaranteed.

With hanging nodes, coding of the RKDG scheme is still trivial, but not for the LDG scheme. This is because that for RKDG scheme (2.6), cell K only interacts with its direct neighboring cells and the scheme is explicit with no large linear system to solve, while for LDG scheme (2.12), besides its direct neighboring cells, cell K may interact with neighbors of its direct neighboring cells, forming a large system of linear equations for the elliptic equation. Such complication brings in extra complexity in numerical implementation. In what follows we will describe how we code the LDG scheme. We use the same notations as in Sect. 2 unless otherwise specified.

With the LDG scheme, we form the large linear system for all degrees of freedom of ψ_h, by firstly using Eq. (2.11a) to express the degrees of freedom of $\mathbf{g}_h$ in terms of those of ψ_h and then inserting the expression into (2.11b). The numerical fluxes are chosen according to (2.15). In particular, let p, q denote the two components of $\mathbf{g}$ in Eq. (2.11) and take $\mathbf{v}_0 = (1, 1)$ in Eq. (2.16), then Eq. (2.15) gives us

$$\text{on vertical edges: } \hat{p} = p^E + \psi^E - \psi^W, \quad \hat{\psi} = \psi^W, \tag{3.5a}$$

$$\text{on horizontal edges: } \hat{q} = q^N + \psi^N - \psi^S, \quad \hat{\psi} = \psi^S, \tag{3.5b}$$

where the superscripts indicate the directions from which the traces on edges are taken. The diagram in Fig. 2 shows the notations on cell K along with some examples of traces. For instance,

$$p^E_{x_{3,K}} = p\left(x^E_{3,K}, y\right) = \lim_{\delta \to 0^+} p(x_{3,K} + \delta, y),$$

$$\psi^S_{y_{3,K}} = \psi\left(x, y^S_{3,K}\right) = \lim_{\delta \to 0^-} \psi(x, y_{3,K} + \delta).$$

When being applied to the h-adaptive mesh introduced before, Eq. (2.12) becomes the following: for all $K \in \mathcal{M}_h$ and any $r_1, r_2, v \in V_h^k$,

$$\int_{y_{1,K}}^{y_{3,K}} \hat{\psi}_h(x_{3,K}, y) r_1(x^W_{3,K}, y)\, dy - \int_{y_{1,K}}^{y_{3,K}} \hat{\psi}_h(x_{1,K}, y) r_1(x^E_{1,K}, y)\, dy - \int_K \psi_h \frac{\partial r_1}{\partial x}\, dxdy$$
$$= \int_K p_h r_1\, dxdy, \tag{3.6a}$$

$$\int_{x_{1,K}}^{x_{3,K}} \hat{\psi}_h(x, y_{3,K}) r_2(x, y^S_{3,K})\, dx - \int_{x_{1,K}}^{x_{3,K}} \hat{\psi}_h(x, y_{1,K}) r_2(x, y^N_{1,K})\, dx - \int_K \psi_h \frac{\partial r_2}{\partial y}\, dxdy$$
$$= \int_K q_h r_2\, dxdy, \tag{3.6b}$$

$$\int_{y_{1,K}}^{y_{3,K}} \hat{p}_h(x_{3,K}, y) v(x^W_{3,K}, y)\, dy - \int_{y_{1,K}}^{y_{3,K}} \hat{p}_h(x_{1,K}, y) v(x^E_{1,K}, y)\, dy - \int_K p_h \frac{\partial v}{\partial x}\, dxdy$$

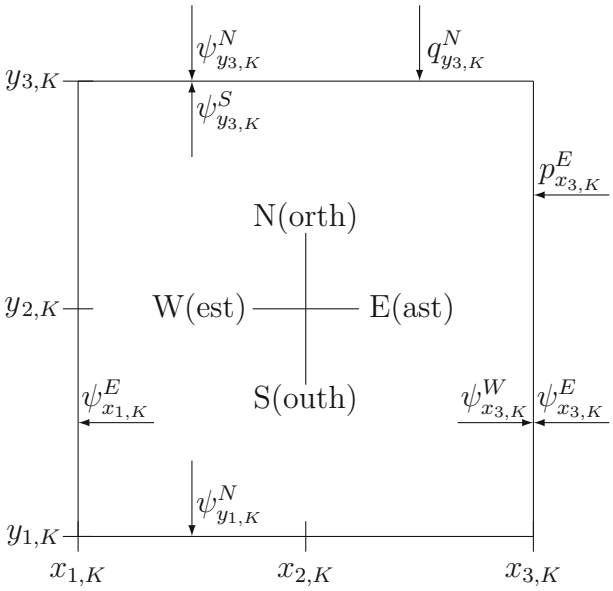

Fig. 2 Notations on cell K

$$+\int_{x_{1,K}}^{x_{3,K}} \hat{q}_h(x, y_{3,K}) v(x, y^S_{3,K})\, dx - \int_{x_{1,K}}^{x_{3,K}} \hat{q}_h(x, y_{1,K}) v(x, y^N_{1,K})\, dx$$
$$-\int_K q_h \frac{\partial v}{\partial y}\, dxdy = \int_K \omega v\, dxdy. \tag{3.6c}$$

To show the process, it is sufficient to deal with the p_h-related terms [Equation (3.6a) and the first three terms in Eq. (3.6c)] only and omit the q_h-related terms due to similarity. Introduce $\vec{\phi}_K = (\phi_0^{(K)}, \phi_1^{(K)}, \ldots, \phi_{Q_k}^{(K)})^T$ which is the vector of basis functions on cell K. Let $\vec{\psi}_K = (\psi_K^{(0)}, \psi_K^{(1)}, \ldots, \psi_K^{(Q_k)})^T$ be the vector of degrees of freedom of ψ_h on K. As a result, $\psi_h|_K = \vec{\psi}_K \cdot \vec{\phi}_K$. Similarly, let $\vec{p}_K$ and $\vec{q}_K$ be the vector of degrees of freedom of p_h and q_h, respectively. To obtain the system of equations for $\vec{\psi}_K$, we substitute $\vec{\phi}_K$ for r_1, r_2 and v in Eq. (3.6). In order to form the large linear system in a systematical way, we introduce a group of matrices that can be precomputed (so they need to be computed only once during the entire numerical simulation). These matrices depend only on the basis functions and LEV, and are independent of individual cell K.

$$C^{\phi\phi}: \quad C_{ij}^{\phi\phi} = \int_{-\frac{1}{2}}^{\frac{1}{2}} \int_{-\frac{1}{2}}^{\frac{1}{2}} \phi_j(x, y)\phi_i(x, y)\, dxdy, \tag{3.7a}$$

$$C^{\phi\phi_x}: \quad C_{ij}^{\phi\phi_x} = \int_{-\frac{1}{2}}^{\frac{1}{2}} \int_{-\frac{1}{2}}^{\frac{1}{2}} \phi_j(x, y) \frac{\partial \phi_i(x, y)}{\partial x}\, dxdy, \tag{3.7b}$$

$$D^{S_W T_W}: \quad D_{i,j}^{S_W T_W} = \int_{-\frac{1}{2}}^{\frac{1}{2}} \phi_j\left(\frac{1}{2}, y\right) \phi_i\left(\frac{1}{2}, y\right) dy, \tag{3.7c}$$

$$D^{S_E T_E}: \quad D_{i,j}^{S_E T_E} = \int_{-\frac{1}{2}}^{\frac{1}{2}} \phi_j\left(-\frac{1}{2}, y\right) \phi_i\left(-\frac{1}{2}, y\right) dy, \tag{3.7d}$$

$$A^{S_W T_E}(\alpha,\beta):\quad A_{i,j}^{S_W T_E} = \int_{-\frac{1}{2}}^{\frac{1}{2}} \phi_j\left(\frac{1}{2}, y\right)\phi_i\left(-\frac{1}{2}, \alpha y+\beta\right) dy, \tag{3.7e}$$

$$A^{S_E T_W}(\alpha,\beta):\quad A_{i,j}^{S_E T_W} = \int_{-\frac{1}{2}}^{\frac{1}{2}} \phi_j\left(-\frac{1}{2}, \alpha y+\beta\right)\phi_i\left(\frac{1}{2}, y\right) dy, \tag{3.7f}$$

$$B^{S_W T_E}(\alpha,\beta):\quad B_{i,j}^{S_W T_E} = \int_{-\frac{1}{2}}^{\frac{1}{2}} \phi_j\left(\frac{1}{2}, \alpha y+\beta\right)\phi_i\left(-\frac{1}{2}, y\right) dy, \tag{3.7g}$$

$$B^{S_E T_W}(\alpha,\beta):\quad B_{i,j}^{S_E T_W} = \int_{-\frac{1}{2}}^{\frac{1}{2}} \phi_j\left(-\frac{1}{2}, y\right)\phi_i\left(\frac{1}{2}, \alpha y+\beta\right) dy, \tag{3.7h}$$

where $i, j = 0, 1, \ldots, Q_k$. Parameters α and β have limited possible values for the h-adaptive mesh in this paper, i.e.,

$$\alpha = 2^{-\Delta L}, \quad \Delta L = 0, 1, \ldots, LEV,$$

and

$$\beta = \left(m-\frac{1}{2}\right)2^{-\Delta L} - \frac{1}{2}, \quad m = 1, 2, \ldots, 2^{\Delta L}$$

where ΔL is the difference of mesh levels between the two neighboring cells. The rationale behind this is given in Remark 3.1. The superscripts of matrices A, B, D indicate the directions where the traces are taken. For example, "$S_W T_E$" denotes the case that the basis ϕ_j is from the Solution function space with its trace taken from the West, while the basis ϕ_i corresponds to the Test function space with its trace taken from the East. Note that $A^{S_E T_W}$ and $B^{S_E T_W}$ are the transpose of $A^{S_W T_E}$ and $B^{S_W T_E}$ respectively.

Remark 3.1 The matrices in Eq. (3.7) are defined so that the integrals in Eq. (3.6) can be rewritten in a matrix form. We provide an example to show the mechanism here. Consider the case in Fig. 3 where we have cell K_s smaller than its east neighbor denoted by K_b. Integrals on an edge should be performed on the edge of the smaller cell. For the following integral, we have

$$\begin{aligned}
&\int_{y_{1,K_s}}^{y_{3,K_s}} \psi_h(x_{3,K_s}^W, y)\phi_i^{(K_b)}(x_{3,K_s}, y)\, dy \\
&= \int_{y_{1,K_s}}^{y_{3,K_s}} \left[\sum_{j=0}^{Q_k} \psi_{K_s}^{(j)}\phi_j^{(K_s)}(x_{3,K_s}, y)\right]\phi_i^{(K_b)}(x_{1,K_b}, y)\, dy \\
&= \sum_{j=0}^{Q_k}\psi_{K_s}^{(j)}\left[\int_{y_{1,K_s}}^{y_{3,K_s}} \phi_j\left(\frac{1}{2}, \frac{y-y_{2,K_s}}{\Delta y_{K_s}}\right)\phi_i\left(-\frac{1}{2}, \frac{y-y_{2,K_b}}{\Delta y_{K_b}}\right) dy\right] \\
&= \Delta y_{K_s}\sum_{j=0}^{Q_k}\psi_{K_s}^{(j)}\left[\int_{-\frac{1}{2}}^{\frac{1}{2}} \phi_j\left(\frac{1}{2}, z\right)\phi_i\left(-\frac{1}{2}, \frac{\Delta y_{K_s}}{\Delta y_{K_b}}z + \frac{y_{2,K_s}-y_{2,K_b}}{\Delta y_{K_b}}\right) dz\right].
\end{aligned} \tag{3.8}$$

The integral in the square brackets depends only on the size ratio $\alpha = \frac{\Delta y_{K_s}}{\Delta y_{K_b}}$ and the relative position $\beta = \frac{y_{2,K_s}-y_{2,K_b}}{\Delta y_{K_b}}$ of the two adjacent cells. If matrix $A^{S_W T_E}$ is defined as in Eq. (3.7), we have

$$\int_{y_{1,K_s}}^{y_{3,K_s}} \psi_h(x_{3,K_s}^W, y)\vec{\phi}_{K_b}(x_{3,K_s}, y)\, dy = \Delta y_{K_s} A^{S_W T_E}(\alpha,\beta)\vec{\psi}_{K_s}.$$

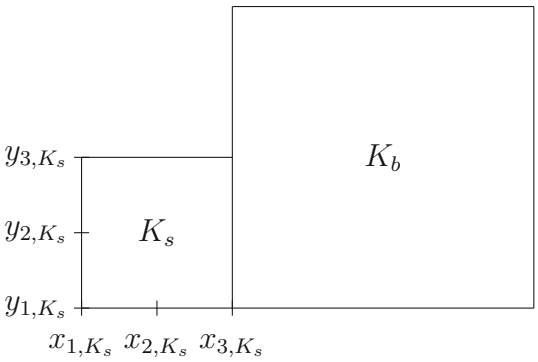

Fig. 3 Line integral on the common edge of two cells

With the preparation of precomputed matrices, we are ready to form the linear system. After some simple manipulations, the right hand side term in Eq. (3.6a) becomes

$$\int_K p_h \vec{\phi}_K \, dxdy = \Delta x_K \Delta y_K C^{\phi\phi} \vec{p}_K, \tag{3.9}$$

and the last term of the left hand side of Eq. (3.6a) becomes

$$\int_K \psi_h \frac{\partial \vec{\phi}_K}{\partial x} \, dxdy = \Delta y_K C^{\phi\phi_x} \vec{\psi}_K. \tag{3.10}$$

For the first term in Eq. (3.6a), inserting Eq. (3.5a) we have

$$\int_{y_{1,K}}^{y_{3,K}} \hat{\psi}_h(x_{3,K}, y) \vec{\phi}_K(x_{3,K}^W, y) \, dy = \Delta y_K D^{S_W T_W} \vec{\psi}_K. \tag{3.11}$$

As to the second term in Eq. (3.6a), there are two different cases. In the first case there is only one neighbor in the west of cell K. Let us denote it by K_W. With the matrix definitions and a simple manipulation, we can obtain

$$\int_{y_{1,K}}^{y_{3,K}} \hat{\psi}_h(x_{1,K}, y) \vec{\phi}_K \left(x_{1,K}^E, y\right) dy = \Delta y_K B^{S_W T_E}(\alpha_{K_W}, \beta_{K_W}) \vec{\psi}_{K_W}. \tag{3.12}$$

Inserting Eqs. (3.11), (3.12), (3.10) and (3.9) into Eq. (3.6a), we have

$$\Delta y_K D^{S_W T_W} \vec{\psi}_K - \Delta y_K B^{S_W T_E}(\alpha_{K_W}, \beta_{K_W}) \vec{\psi}_{K_W} - \Delta y_K C^{\phi\phi_x} \vec{\psi}_K = \Delta x_K \Delta y_K C^{\phi\phi} \vec{p}_K. \tag{3.13}$$

Then we can derive the formula for $\vec{p}_K$ in terms of the degrees of freedom of ψ_h:

$$\vec{p}_K = \frac{1}{\Delta x_K} (C^{\phi\phi})^{-1} \left[\left(D^{S_W T_W} - C^{\phi\phi_x} \right) \vec{\psi}_K - B^{S_W T_E}(\alpha_{K_W}, \beta_{K_W}) \vec{\psi}_{K_W} \right]. \tag{3.14}$$

The second case is for more than one west neighbor, see an example in the left panel of Fig. 4. Assume there are n_1 west neighbors which are denoted by $K_{W_m}, m = 1, \ldots, n_1$. Then

$$\begin{aligned} \int_{y_{1,K}}^{y_{3,K}} \hat{\psi}_h(x_{1,K}, y) \vec{\phi}_K \left(x_{1,K}^E, y\right) dy &= \sum_{m=1}^{n_1} \int_{y_{1,K_{W_m}}}^{y_{3,K_{W_m}}} \psi_h \left(x_{1,K}^W, y\right) \vec{\phi}_K \left(x_{1,K}^E, y\right) dy \\ &= \sum_{m=1}^{n_1} \Delta y_{K_{W_m}} A^{S_W T_E}(\alpha_{K_{W_m}}, \beta_{K_{W_m}}) \vec{\psi}_{K_{W_m}}. \end{aligned} \tag{3.15}$$

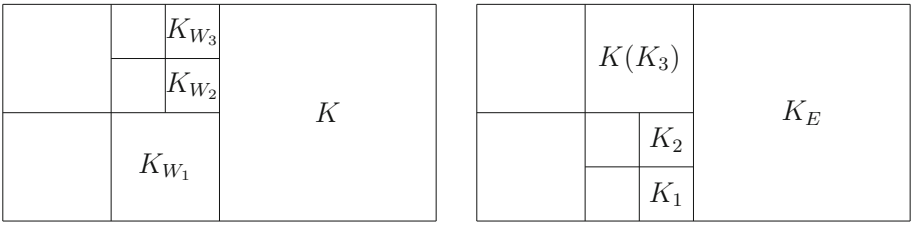

Fig. 4 Examples of K with different cases of neighbors. *Left* more than one west neighbor. *Right* only one east neighbor

Inserting Eqs. (3.11), (3.15), (3.10) and (3.9) into Eq. (3.6a), we can again obtain the formula for $\vec{p}_K$:

$$\vec{p}_K = \frac{1}{\Delta x_K}(C^{\phi\phi})^{-1}\left(\left(D^{S_W T_W} - C^{\phi\phi_x}\right)\vec{\psi}_K - \frac{1}{\Delta y_K}\sum_{m=1}^{n_1}\Delta y_{K_{W_m}} A^{S_W T_E}\left(\alpha_{K_{W_m}}, \beta_{K_{W_m}}\right)\vec{\psi}_{K_{W_m}}\right). \tag{3.16}$$

Now let us deal with Eq. (3.6c). We will take the first term of Eq. (3.6c) with the case of K having only one neighbor in the east as an example to show how to derive the formulas for coding. Other terms and cases can be handled similarly. To simplify the notations for this case, we denote the only east neighbor by K_E, and assume there are $n_2 \geq 1$ west neighbors (including K) for K_E, which are denoted by $K_1, \ldots, K_{n_2}$. The right panel in Fig. 4 gives an example of this particular case. It is worth mentioning for this example that K_1 is not a direct neighbor of cell K but is involved in the scheme for K. We have

$$\begin{aligned}
&\int_{y_{1,K}}^{y_{3,K}} \hat{p}_h(x_{3,K}, y)\vec{\psi}_K\left(x_{3,K}^W, y\right) dy \\
&= \int_{y_{1,K}}^{y_{3,K}} \left[p_h\left(x_{3,K}^E, y\right) + \psi_h\left(x_{3,K}^E, y\right) - \psi_h\left(x_{3,K}^W, y\right)\right]\vec{\psi}_K\left(x_{3,K}^W, y\right) dy \\
&= \Delta y_K A^{S_E T_W}(\alpha_{K_E}, \beta_{K_E})\left[\vec{p}_{K_E} + \vec{\psi}_{K_E}\right] - \Delta y_K D^{S_W T_W}\vec{\psi}_K.
\end{aligned} \tag{3.17}$$

From Eq. (3.16), we have

$$\vec{p}_{K_E} = \frac{1}{\Delta x_{K_E}}(C^{\phi\phi})^{-1}\left(\left(D^{S_W T_W} - C^{\phi\phi_x}\right)\vec{\psi}_{K_E} - \frac{1}{\Delta y_{K_E}}\sum_{m=1}^{n_2}\Delta y_{K_m} A^{S_W T_E}(\alpha_{K_m}, \beta_{K_m})\vec{\psi}_{K_m}\right).$$

Substituting it into Eq. (3.17), the first term of Eq. (3.6c) is finally expressed in terms of the degrees of freedom of ψ_h only. Now this term is ready for coding.

All the terms in Eq. (3.6c) can be expressed in terms of the degrees of freedom of ψ_h through a similar approach. When the LDG scheme is coded, for each cell K' involved in the scheme of the target cell K, the coefficient matrix of $\vec{\psi}_{K'}$ is firstly derived by the formulas like Eq. (3.17). Then it is immediately assembled into the final linear system.

4 Numerical Tests

This section provides numerical examples for the proposed h-adaptive RKDG–LDG scheme. Comparisons with the fixed-mesh RKDG–LDG scheme are made to demonstrate advantages

of the mesh adaptation. For convenience, we refer the fixed-mesh/h-adaptive RKDG–LDG scheme as nonadaptive/adaptive scheme respectively. Global time steps, which depend on the smallest cell size at each time-level, are used in the RK time discretization. Local time stepping treatments [9,13,16,17] may improve the time discretization efficiency, which are left to future investigation. Periodic boundary conditions are applied to all the test examples in this section. In order to make the scheme more robust during the numerical simulations, the maximum-principle-satisfying technique proposed in [25] is applied.

For clarity, the variance of the RKDG schemes will be denoted as `InitialResolution-LEVj-Order` where `InitialResolution` is expressed by the product of N_x (number of cells along the x-axis) and N_y (number of cells along the y-axis) at the initial time, `LEVj` indicates that $LEV = j$ is being used, and `Order` is either `P1` (P^1 case, i.e. $k = 1$) or `P2` (P^2 case, i.e. $k = 2$). Note that the special case `LEV0` stands for the nonadaptive RKDG schemes using fixed uniform meshes in this work. For example, `32*32-LEV4-P1` denotes the h-adaptive RKDG scheme with initial resolution $32 * 32$ and $k = 1$, and `64*64-LEV0-P2` denotes the nonadaptive RKDG scheme with (initial) resolution $64 * 64$ and $k = 2$.

Example 4.1 (Accuracy and convergence test) We solve the model problem (1.1) up to time $T = 1.0$ in the domain $[0, 2\pi] \times [0, 2\pi]$ with the initial condition

$$\omega(x, y, 0) = -2 \sin(x) \sin(y). \tag{4.1}$$

The exact solution is identical to the initial condition. In Fig. 5, we present a log–log plot of L^1 error versus CPU time (s) for both nonadaptive and adaptive schemes. The symbols along the curves for nonadaptive DG schemes (annotated as LEV0-P1 and LEV0-P2) are obtained from simulations with meshes 5×5, 10×10, ..., 320×320; and the symbols along curves for adaptive schemes (annotated as LEV4-P1 and LEV4-P2) are obtained from simulations with 5×5, 10×10, 20×20 initial meshes. The convergence of both nonadaptive and adaptive schemes is clearly shown. For this example, it is observed that adaptive schemes are not as efficient as nonadaptive ones, due to the fact that the solution is a smooth sin function and mesh refinement does not bring in advantages in solution resolution. The adaptive scheme is expected to be more effective and efficient for examples with multi-scale and fine-scale solution structures, e.g. in resolving shocks for hyperbolic systems [27,28] and in resolving filamentation structures for Vlasov equation [30] as well as examples presented later in this section.

For the rest of the test examples, since there are no analytical solutions, the results computed by scheme `256*256-LEV0-P2` are served as references. In addition, we only report the P^2 results with $LEV = 4$ to save space because they are sufficient to illustrate the capability of the adaptive scheme.

Example 4.2 (The shear flow problem taken from [1]) This is an example from the model problem (1.1). The initial condition is given by

$$\omega(x, y, 0) = \begin{cases} \delta \cos(x) - \frac{1}{\rho} \operatorname{sech}^2\left(\frac{y - \pi/2}{\rho}\right), & \text{if} \quad y \le \pi, \\ \delta \cos(x) + \frac{1}{\rho} \operatorname{sech}^2\left(\frac{3\pi/2 - y}{\rho}\right), & \text{if} \quad y > \pi, \end{cases} \tag{4.2}$$

where $\delta = 0.05$ and $\rho = \pi/15$. We solve this problem in the domain $[0, 2\pi] \times [0, 2\pi]$ until $T = 8$. The solution quickly develops into roll-ups with smaller and smaller scales,

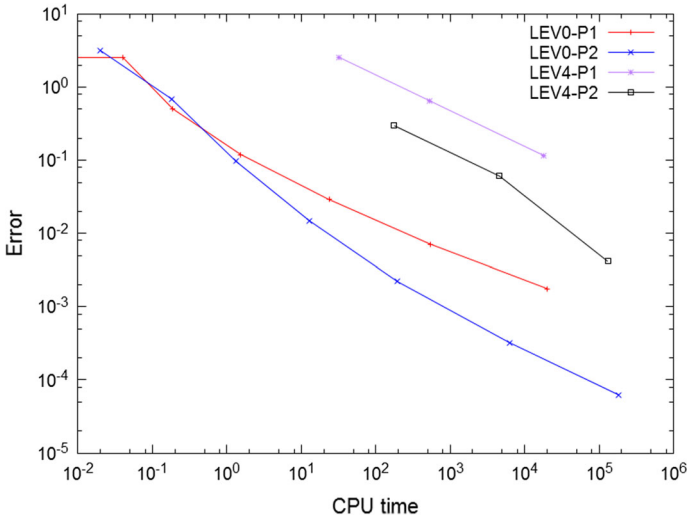

Fig. 5 Example 4.1 L^1 error versus CPU time (s)

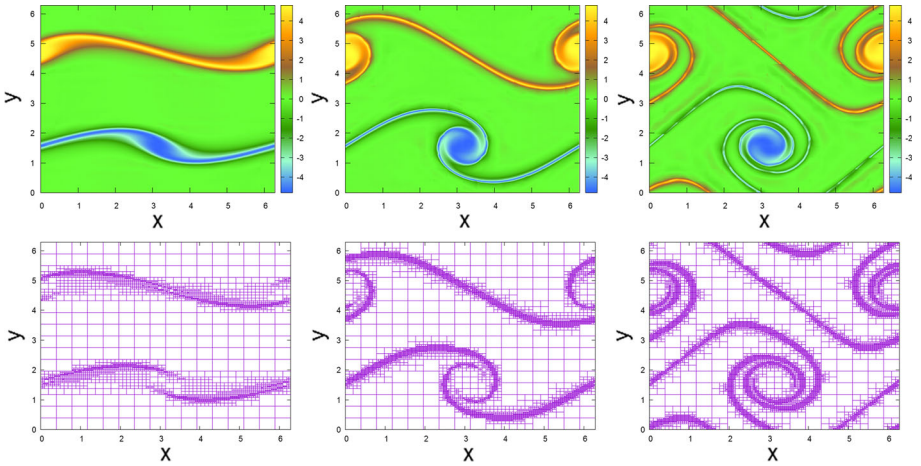

Fig. 6 Shear flow problem, time evolution of the vorticity and the adaptive meshes for scheme `16*16-LEV4-P2`. From *left* to *right* $t = 4, 6, 8$

so on any fixed mesh the full resolution is lost eventually. We plot the time evolution of the vorticity ω for scheme `16*16-LEV4-P2` in Fig. 6, along with the corresponding meshes. From these figures we can see that the adaptive meshes resolve the roll-ups very well. Fine meshes are generated at the regions with fine structures while coarse meshes are used elsewhere. The adaptive scheme works well in a dynamical way. In Fig. 7, 1D cuts of numerical solutions at $x = \pi$ are plotted. It is observed that the adaptive scheme `16*16-LEV4-P2` produces a comparable solution to that from the scheme `256*256-LEV0-P2` for this problem.

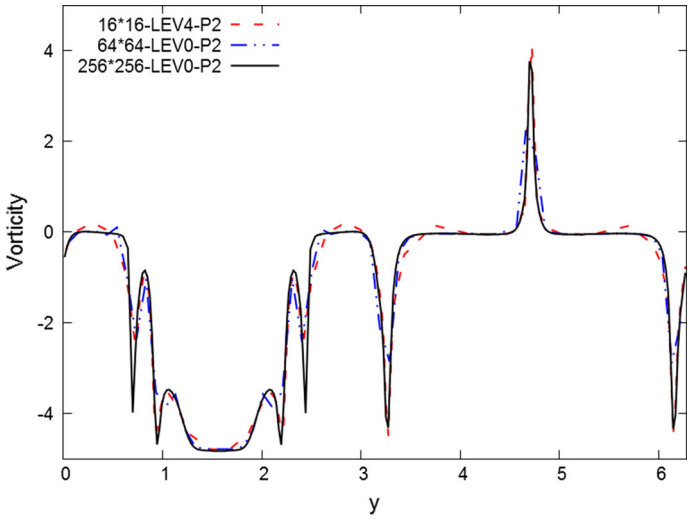

Fig. 7 1D cuts of the vorticity at $x = \pi$ for the shear flow problem at $T = 8$

Example 4.3 (The vortex patch problem) We solve the model problem (1.1) in the domain $[0, 2\pi] \times [0, 2\pi]$ with the initial condition

$$\omega(x, y, 0) = \begin{cases} -1, & \text{if} \quad (x, y) \in \left[\frac{\pi}{2}, \frac{3\pi}{2}\right] \times \left[\frac{\pi}{4}, \frac{3\pi}{4}\right], \\ 1, & \text{if} \quad (x, y) \in \left[\frac{\pi}{2}, \frac{3\pi}{2}\right] \times \left[\frac{5\pi}{4}, \frac{7\pi}{4}\right], \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

The final time is $T = 10$. Time evolution of the vorticity and adaptive meshes for scheme `16*16-LEV4-P2` are given in Fig. 8. It is clearly shown that the adaptive meshes are generated in accordance with the development of the solution. Fine meshes are generated in regions with large solution gradients, and dynamically follow these regions as they move. As a result, the adaptive scheme is able to capture the fine solution structures with a coarse initial mesh. In Fig. 9, 1D cuts of numerical solutions at $x = \pi$ are plotted. The solutions from scheme `16*16-LEV4-P2` and `256*256-LEV0-P2` are observed to be close to each other.

Example 4.4 (A Kelvin–Helmholtz (K–H) instability problem taken from [7]) This is an example from the 2D guiding-center model (1.2). The initial condition is

$$\omega(x, y, 0) = \sin(y) + 0.015 \cos(\eta x) \tag{4.4}$$

where we let $\eta = 0.5$ so that a K–H instability is created. This problem is solved until the final time $T = 40$ is reached. In Fig. 10 we show the time evolution of the density and the corresponding meshes for scheme `32*32-LEV4-P2`. We can see again that fine meshes are generated in regions where solutions have large gradients. The coarse mesh are used where solution has mild gradients. The dynamic mesh refinement and coarsening work well and the adaptive strategy is effective as expected. In Fig. 11, we show 1D cuts of solutions at $y = \pi$. The result from scheme `128*128-LEV0-P2` is used instead of the one from scheme `256*256-LEV0-P2` because based on our estimation, scheme `256*256-LEV0-P2` will cost more than 1400 CPU hours which is beyond the capability of our available computers.

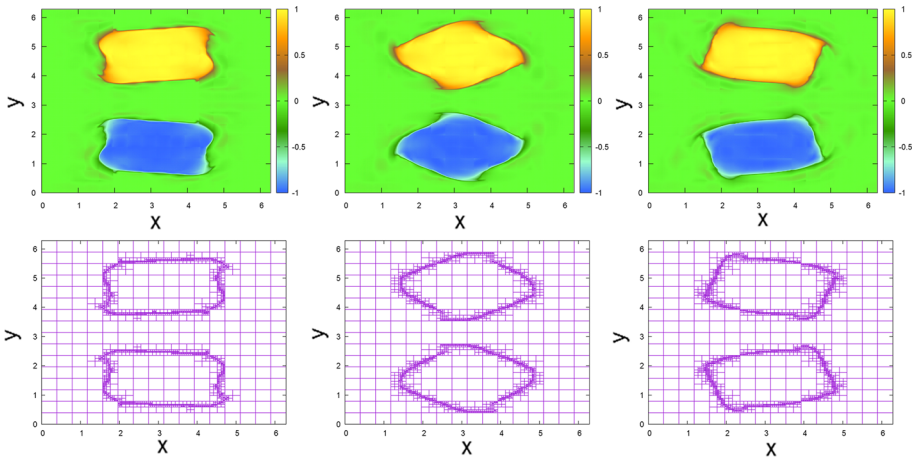

Fig. 8 Vortex patch problem, time evolution of the vorticity and the adaptive meshes for scheme `16*16-LEV4-P2`. From *left* to *right* $t = 6, 8, 10$

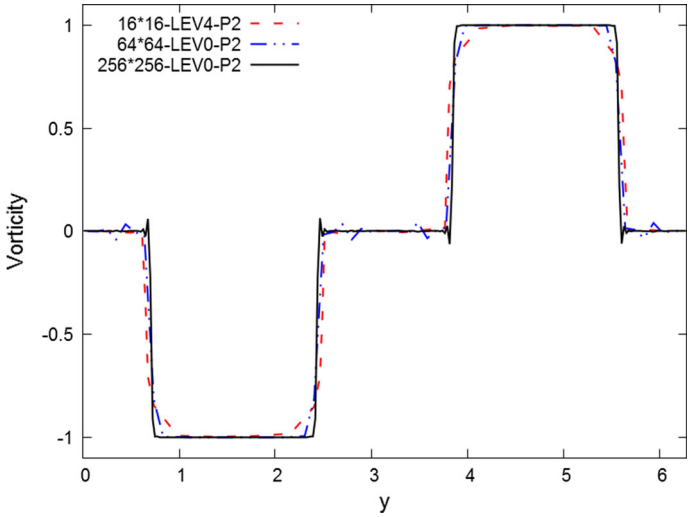

Fig. 9 1D cuts of the vorticity at $x = \pi$ for the vortex patch problem at $T = 10$

Again, comparable results between solutions from the adaptive mesh and the fine mesh are observed.

Next we make a detailed numerical comparison between the adaptive and nonadaptive schemes, in order to show the advantage of the mesh adaptation. Firstly, we provide numerical evidence of mass conservation in Fig. 12 for the adaptive schemes. Secondly, let us recall some analytically conserved quantities of the system which can be used as diagnostics in a numerical scheme.

1. Energy:

$$\|\mathbf{u}\|_{L^2}^2 = \int_\Omega \mathbf{u} \cdot \mathbf{u}\, dxdy. \tag{4.5}$$

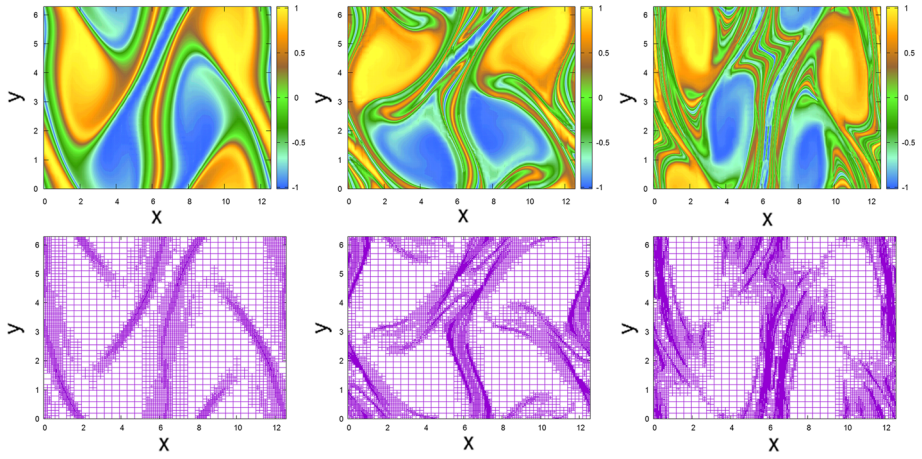

Fig. 10 K–H instability, time evolution of the vorticity and the adaptive meshes for scheme `32*32-LEV4-P2`. From *left* to *right* $t = 20, 30, 40$

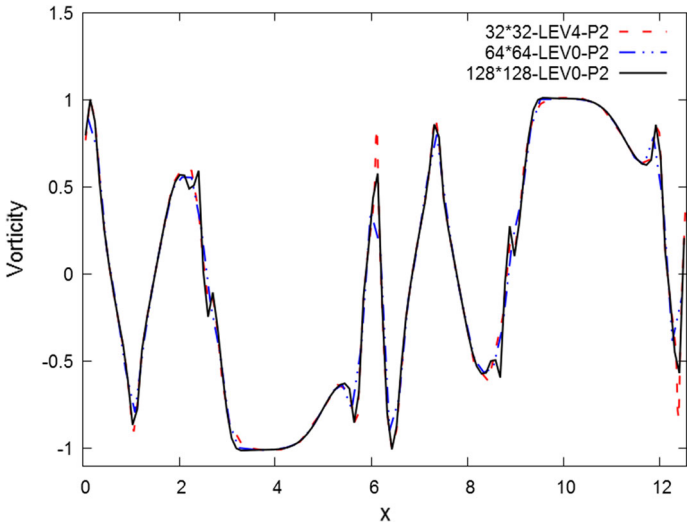

Fig. 11 1D cuts of the vorticity at $y = \pi$ for K–H instability problem at $T = 25$

2. Enstrophy:

$$\|\omega\|_{L^2}^2 = \int_\Omega \omega^2 \, dxdy. \tag{4.6}$$

Similarly, for the guiding center model, the energy and enstrophy are $\|\mathbf{E}\|_{L^2}^2$ and $\|\rho\|_{L^2}^2$ respectively. They are analytically conserved quantities. Tracking relative deviations of these invariants numerically provides a good measure on the quality of numerical schemes. The relative deviation is defined to be the deviation away from the corresponding initial value divided by the magnitude of the initial value. We show the time history of the relative deviations of energy and enstrophy in Fig. 13 for all the test examples. In general, the adaptive and nonadaptive schemes are comparable in conserving the energy and enstrophy.

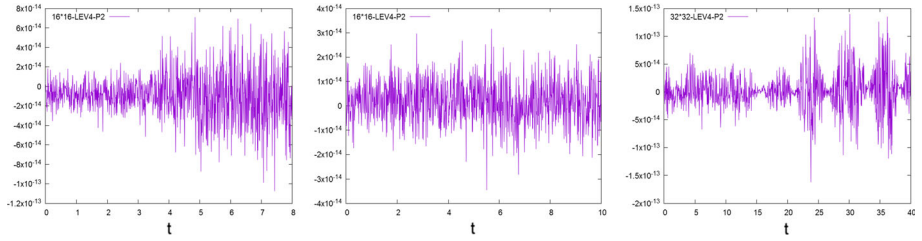

Fig. 12 Time history of $\int_\Omega \omega_h \, dxdy$ or $\int_\Omega \rho_h \, dxdy$ to demonstrate the mass conservation. From *left* to *right* shear flow, vortex patch, K–H instability

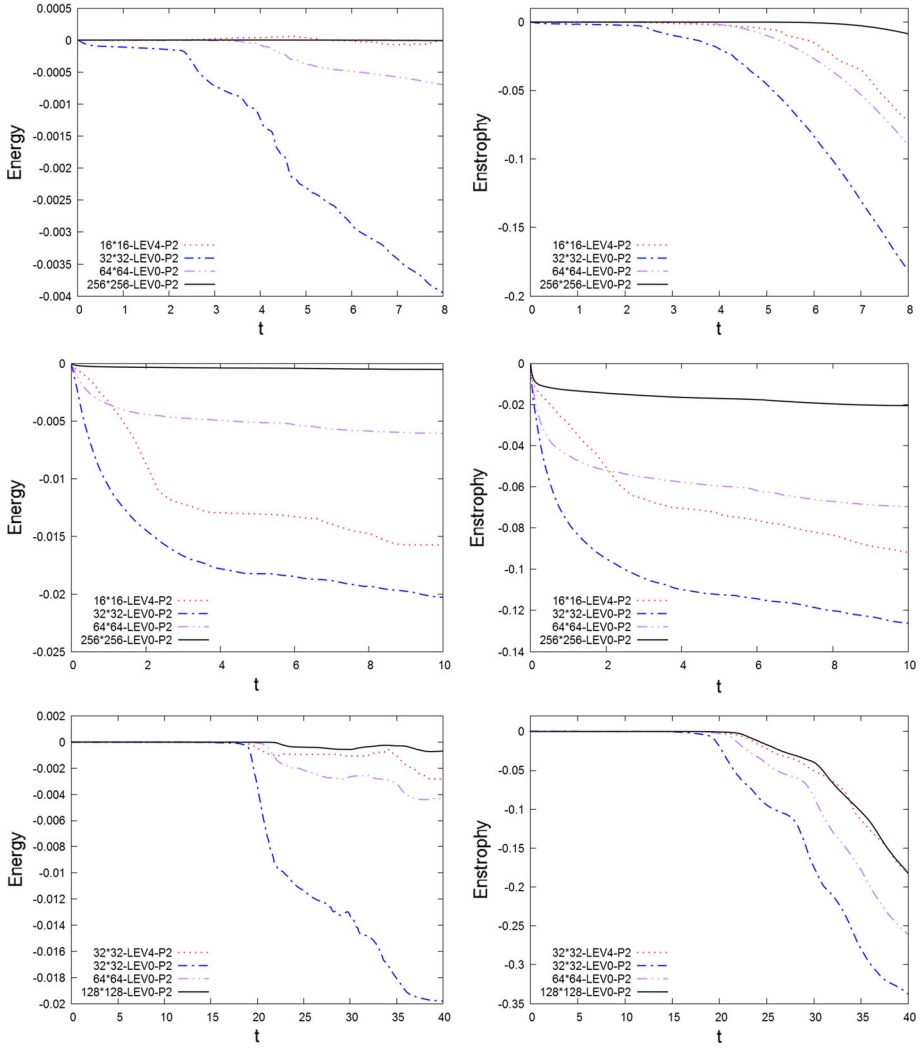

Fig. 13 Time history of the relative deviations of energy (*left column*) and enstrophy (*right column*). From *top* to *bottom row* shear flow, vortex patch, K–H instability

Table 1 Data on adaptive meshes

Example	Scheme	*TND*	N_T	$\bar{N}$	*PR*
Shear flow	16*16-LEV4-P2	1.4E+4	12,826	6096.2	9.30
Vortex patch	16*16-LEV4-P2	1.1E+4	3307	3064.3	4.68
K–H instability	32*32-LEV4-P2	2.2E+5	39,094	18,203.8	6.94

Table 2 Comparison of CPU time (h)

Example	Scheme	CPU time
Shear flow	16*16-LEV4-P2	7.12
	64*64-LEV0-P2	1.33
	128*128-LEV0-P2	21.58
	256*256-LEV0-P2	619.49
Vortex patch	16*16-LEV4-P2	0.92
	64*64-LEV0-P2	0.43
	128*128-LEV0-P2	11.00
	256*256-LEV0-P2	289.90
K–H instability	32*32-LEV4-P2	385.60
	64*64-LEV0-P2	1.47
	128*128-LEV0-P2	47.71
	256*256-LEV0-P2	>1400

To provide a quantitative understanding of adaptive meshes, for all the examples above we collect some important data on mesh and organize them as Table 1. The table includes (a) *TND*: total number of divisions; (b) N_T: number of cells at the final time; (c) $\bar{N}$: average number of cells, defined by $(\sum_{n=0}^{TOT} N_n)/TOT$ where N_n is the number of cells at the n-th time-level and *TOT* is the total number of time-levels; and (d) *PR*: the percentage ratio of $\bar{N}$ to the number of cells of a fully refined mesh, i.e. $PR = 100\bar{N}/(2^{LEV}N_0)$. It is shown in the table that all the values of PR are far less than 100, which indicates that the adaptive scheme uses much less cells than a nonadaptive one with a comparable resolution.

Finally, we compare the CPU time (h) in Table 2. The approximate number with a '>' symbol is derived by estimation. The data shows that all the adaptive schemes cost CPU time much less than the corresponding nonadaptive schemes with fully refined uniform meshes. For example, with $LEV = 4$, the 256*256 mesh is the fully refined mesh of a 16*16 mesh. The adaptive scheme could cost even less than the nonadaptive schemes using one-level coarser mesh than the fully refined one. Significant computational savings are observed when compared with the non-adaptive scheme with the finest level of mesh.

5 Conclusion

In this paper, we propose an h-adaptive RKDG–LDG scheme for solving the 2D time-dependent incompressible Euler equations in the vorticity-stream function formulation and for solving the guiding center Vlasov model. The scheme is a generalization of the adaptive scheme proposed in [30]. The main difficulty of this generalization lies in the 2D Poisson

solver due to the irregular adaptive mesh with hanging nodes. We provide an in-depth discussion on the adaptive LDG Poisson solver and provide a detailed description regarding the coding issue. Extensive numerical tests have been performed to illustrate the effectiveness of the adaptive scheme. Advantages of the h-adaptive scheme have been shown, especially in terms of computational savings, with comparable solution resolutions to those from a fine mesh simulation. Subsequent research directions include (1) developing anisotropic adaptive schemes to gain more benefits from mesh adaptation; (2) using triangular meshes to handle complicated geometries.

References

1. Bell, J., Colella, P., Glaz, H.: A second order projection method for the incompressible Navier–Stokes equations. J. Comput. Phys. **85**, 257–283 (1989)
2. Biswas, R., Devine, K., Flaherty, J.: Parallel, adaptive finite element methods for conservation laws. Appl. Numer. Math. **14**, 255–283 (1994)
3. Castillo, P., Cockburn, B., Perugia, I., Schötzau, D.: An a priori error analysis of the local discontinuous Galerkin method for elliptic problems. SIAM J. Numer. Anal. **38**(5), 1676–1706 (2000)
4. Christlieb, A., Guo, W., Morton, M., Qiu, J.-M.: A high order time splitting method based on integral deferred correction for semi-Lagrangian Vlasov simulations. J. Comput. Phys. **267**, 7–27 (2014)
5. Cockburn, B., Kanschat, G., Perugia, I., Schötzau, D.: Superconvergence of the local discontinuous Galerkin method for elliptic problems on Cartesian grids. SIAM J. Numer. Anal. **39**, 264–285 (2001)
6. Cockburn, B., Shu, C.-W.: Runge–Kutta discontinuous Galerkin methods for convection-dominated problems. J. Sci. Comput. **16**, 173–261 (2001)
7. Crouseilles, N., Mehrenberger, M., Sonnendrücker, E.: Conservative semi-Lagrangian schemes for Vlasov equations. J. Comput. Phys. **229**(6), 1927–1953 (2010)
8. Crouseilles, N., Respaud, T., Sonnendrücker, E.: A forward semi-lagrangian method for the numerical solution of the Vlasov equation. Comput. Phys. Commun. **180**, 1730–1745 (2009)
9. Dawson, C., Kirby, R.: High resolution schemes for conservation laws with locally varying time steps. SIAM J. Sci. Comput. **22**, 2256–2281 (2001)
10. Devine, K., Flaherty, J.: Parallel adaptive hp-refinement techniques for conservation laws. Appl. Numer. Math. **20**, 367–386 (1996)
11. Flaherty, J., Loy, R., Shephard, M., Szymanski, B., Teresco, J., Ziantz, L.: Adaptive local refinement with octree load-balancing for the parallel solution of three-dimensional conservation laws. J. Parallel Distrib. Comput. **47**, 139–152 (1997)
12. Hartmann, R., Houston, P.: Adaptive discontinuous Galerkin finite element methods for nonlinear hyperbolic conservation laws. SIAM J. Sci. Comput. **24**, 979–1004 (2002)
13. Krivodonova, L.: An efficient local time-stepping scheme for solution of nonlinear conservation laws. J. Comput. Phys. **229**, 8537–8551 (2010)
14. Levy, D., Tadmor, E.: Non-oscillatory central schemes for the incompressible 2-D Euler equations. Math. Res. Lett. **4**, 321–340 (1997)
15. Liu, J.-G., Shu, C.-W.: A high-order discontinuous Galerkin method for 2D incompressible flows. J. Comput. Phys. **160**, 577–596 (2000)
16. Liu, L., Li, X., Hu, F.Q.: Nonuniform time-step Runge–Kutta discontinuous Galerkin method for computational aeroacoustics. J. Comput. Phys. **229**, 6874–6897 (2010)
17. Maleki, F., Khan, A.: A novel local time stepping algorithm for shallow water flow simulation in the discontinuous Galerkin framework. Appl. Math. Model. **40**, 70–84 (2016)
18. Mehrenberger, M., Mendoza, L., Prouveur, C., Sonnendrücker, E.: Solving the guiding-center model on a regular hexagonal mesh. ESAIM Proc. Surv. **53**, 149–176 (2016)
19. Qiu, J., Shu, C.-W.: A comparison of troubled-cell indicators for Runge–Kutta discontinuous Galerkin mehtods using weighted essentially nonosillatory limiters. SIAM J. Sci. Comput. **27**, 995–1013 (2005)
20. Qiu, J.-M., Shu, C.-W.: Conservative high order semi-Lagrangian finite difference WENO methods for advection in incompressible flow. J. Comput. Phys. **230**, 863–889 (2011)
21. Remacle, J.-F., Flaherty, J., Shephard, M.: An adaptive discontinuous Galerkin technique with an orthogonal basis applied to compressible flow problems. SIAM Rev. **45**, 53–72 (2003)
22. Shu, C.-W., Osher, S.: Efficient implementation of essentially non-oscillatory shock-capturing schemes. J. Comput. Phys. **77**, 439–471 (1988)

23. Sonnendrücker, E., Roche, J., Bertrand, P., Ghizzo, A.: The semi-Lagrangian method for the numerical resolution of the Vlasov equation. J. Comput. Phys. **149**(2), 201–220 (1999)
24. Weinan, E., Shu, C.-W.: A numerical resolution study of high order essentially non-oscillatory schemes applied to incompressible flow. J. Comput. Phys. **110**, 39–46 (1994)
25. Zhang, X., Shu, C.-W.: On maximum-principle-satisfying high order schemes for scalar conservation laws. J. Comput. Phys. **229**, 3091–3120 (2010)
26. Zhu, H., Gao, Z.: An *h*-adaptive RKDG method with troubled-cell indicator for one-dimensional detonation wave simulations. Adv. Comput. Math. **42**, 1081–1102 (2016)
27. Zhu, H., Qiu, J.: Adaptive Runge–Kutta discontinuous Galerkin methods using different indicators: one-dimensional case. J. Comput. Phys. **228**, 6957–6976 (2009)
28. Zhu, H., Qiu, J.: An *h*-adaptive RKDG method with troubled-cell indicator for two-dimensional hyperbolic conservation laws. Adv. Comput. Math. **39**, 445–463 (2013)
29. Zhu, H., Qiu, J.: An *h*-adaptive Runge–Kutta discontinuous Galerkin method for Hamilton–Jacobi equations. Numer. Math. Theory Methods Appl. **6**, 617–636 (2013)
30. Zhu, H., Qiu, J., Qiu, J.-M.: An *h*-adaptive RKDG method for the Vlasov–Poisson system. J. Sci. Comput. **69**, 1346–1365 (2016)