Journal of Computational Physics 228 (2009) 6957–6976

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Adaptive Runge–Kutta discontinuous Galerkin methods using different indicators: One-dimensional case ☆

Hongqiang Zhu, Jianxian Qiu *

Department of Mathematics, Nanjing University, Nanjing, Jiangsu 210093, PR China

ARTICLE INFO

Article history:
Received 13 November 2008
Received in revised form 12 June 2009
Accepted 15 June 2009
Available online 26 June 2009

MSC:
65M60
65M99
35L65

Keywords:
Runge–Kutta discontinuous Galerkin method
Limiters
Troubled cell indicators
Adaptivity

ABSTRACT

In this paper, we systematically investigate adaptive Runge–Kutta discontinuous Galerkin (RKDG) methods for hyperbolic conservation laws with different indicators which were based on the troubled cell indicators studied by Qiu and Shu [J. Qiu, C.-W. Shu, A comparison of troubled-cell indicators for Runge–Kutta discontinuous Galerkin mehtods using weighted essentially non-osillatory limiters, SIAM J. Sci. Comput. 27 (2005) 995–1013]. The emphasis is on comparison of the performance of adaptive RKDG method using different indicators, with an objective of obtaining efficient and reliable indicators to obtain better performance for adaptive computation to save computational cost. Both *h*-version and *r*-version adaptive methods are considered in the paper. The idea is to first identify "troubled cells" by different troubled-cell indicators, namely those cells where limiting might be needed and discontinuities might appear, then adopt an adaptive approach in these cells. A detailed numerical study in one-dimensional case is performed, addressing the issues of efficiency (less CPU cost and more accurate), non-oscillatory property, and resolution of discontinuities.

1. Introduction

In this paper, we systematically investigate adaptive Runge–Kutta discontinuous Galerkin (RKDG) methods for hyperbolic conservation laws with different indicators which are based on the troubled-cell indicators studied by Qiu and Shu [25], with an objective of obtaining efficient and reliable indicators to obtain better performance for adaptive computation to save computational cost. The discontinuous Galerkin (DG) method is a locally conservative, stable, and high-order accurate method which can easily handle complex geometries, irregular meshes with hanging nodes, and approximations that may have polynomials of different degrees in different elements. DG method was originally introduced in 1973 by Reed and Hill [27] for the neutron transport problem which was controlled by steady state linear hyperbolic equations. A major development of DG method was carried out by Cockburn et al. in a series of papers [9,8,7,5,10], in which a framework to solve nonlinear time dependent hyperbolic conservation laws was established. They adopted explicit, nonlinearly stable high order Runge–Kutta time discretizations and DG space discretizations and the method was termed as RKDG method. Now DG method is being used not only for hyperbolic equations, but also for elliptic and parabolic partial differential equations and ordinary differential equations [6].

☆ The research was partially supported by NSFC Grant 10671091, the European project ADIGMA on the development of innovative solution algorithms for aerodynamic simulations and JSNSF BK2006511.

* Corresponding author. Tel.: +86 25 8359 6053; fax: +86 25 8359 7130.
E-mail addresses: hongqiang.zhu@gmail.com (H. Zhu), jxqiu@nju.edu.cn (J. Qiu).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.06.022

The solutions of nonlinear hyperbolic conservation laws often exhibit a wide range of localized structures, such as shock waves, contact discontinuities, and rarefaction waves. It is difficult for numerical methods based on a fixed mesh to capture these features accurately without a large number of mesh cells or points. So the use of locally refined adaptive meshes becomes necessary for both steady and non-steady problems. It is easy to implement adaptivity for DG method since there is no need of continuity on cell boundaries. There are three types of adaptive methods using finite element approach, namely the *h*-method (mesh refinement), the *p*-method (order enrichment), and the *r*-method (mesh motion). The *r*-method is also called moving mesh method. In the literature, there are a few papers which dealt with *h*-, *p*- and *hp*-adaptive DG methods. The first work was by Lesaint and Raviart [22] which contained a priori error estimates for *h*-methods for linear scalar hyperbolic problems, and Johnson and his collaborators [19,20] gave a detailed analysis for the same case and obtained quasi-optimal a priori estimates. Till now many refinement indicators have been proposed which are based on some a priori knowledge of the behavior of solution. Typically, these indicators are loosely based on interpolation error estimates applied to key variables. While this approach may provide some relative measure of the local error in the solution, it does not in general provide a reliable estimate of the actual error in the approximate solution. Bey and Oden [2] obtained both a priori and a posteriori error estimates for adaptive strategy and their numerical comparisons demonstrated the effectiveness of the a posteriori estimates in providing reliable estimates of the actual error in the numerical solution. There are many types of a posteriori error estimates designed to control the adaptive process. A lot of work in this approach has been done by Flaherty et al. [3,12,1,28], Houston et al. [16,17], and recently Dedner et al. [11]. For *r*-adaptive DG method there is little discussion in the literature. But some *r*-adaptive algorithms are easy to apply to DG methods, for instance the approach in [23,33,13] in which PDE time-evolution and mesh-redistribution are totally separated.

In this paper we will use a different class of adaptive strategies, that is using the 'troubled-cell' indicators. 'Troubled cells', whose name came from Qiu and Shu [25], are those cells which might need the limiting procedure. This also means that discontinuities might appear in these cells. So in these cells limiter must be used to control numerical spurious oscillations. The limiter is composed of two parts: one is the troubled-cell indicator (to detect the discontinuous regions), the other is the solution reconstruction (to control the oscillations). The troubled-cell indicators can come from any limiters or shock detecting techniques. In [25], a detailed review of limiters and shock detecting techniques and different troubled-cell indicators are presented. We will list them in Section 2.

The idea using troubled-cell indicators to control the adaptive approach was enlightened by Devine and Flaherty [12]. In the paper the authors used a limiter, which was the extension to two dimensions of the limiter in [3] (the BDF limiter in Section 2), to indicate a preference for *h*- or *p*-refinement. To be specific, they preferred to increase the degree of polynomial in the smooth regions and refine the mesh in which the discontinuous is. Their numerical tests verified the effectiveness of this strategy, but for Euler equations there was too much refinement. According to our numerical experiments, we believe that this is caused by the improper limiter they chose. This problem will be investigated in this paper.

We will design an *h*-method and an *r*-method using different troubled-cell indicators for RKDG method, and compare their performances. Troubled-cell indicators are used to detect troubled cells as the first part of limiter, and to control mesh adaptivity and movement. For the second part of limiter we choose a WENO type reconstruction described in Section 2.

The paper is organized as follows. In Section 2, we make a brief review of RKDG method, troubled-cell indicators and the WENO type reconstruction. In Sections 3 and 4, using the 'troubled-cell indicator' idea, we present an *h*-method and an *r*-method respectively, together with numerical tests. Concluding remarks are contained in Section 5.

2. Description of RKDG and troubled-cell indicators

In this section we will review RKDG method, troubled-cell indicators and the WENO type reconstruction.

2.1. RKDG method

We consider the one-dimensional scalar conservation law:

$$\begin{cases} u_t + f(u)_x = 0, \\ u(x,0) = u_0(x). \end{cases} \tag{2.1}$$

The computational domain is $[0, L]$. We divide it into N cells with boundary points

$$0 = x_{\frac{1}{2}} < x_{\frac{3}{2}} < \cdots < x_{N+\frac{1}{2}} = L.$$

Denote the center of cell $I_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$ by x_i, the length of cell I_i by Δx_i, and the maximum cell length by $h = \max_i \Delta x_i$. The solution as well as the test function space is given by $V_h^k = \{p : p|_{I_i} \in P_k(I_i)\}$, where $P_k(I_i)$ is the space of polynomials of degree $\leqslant k$ on the cell I_i. We adopt the Legendre polynomials

$$W_l(x) = \frac{1}{2^l l!} \frac{d^l (x^2 - 1)^l}{dx^l}, \quad l = 0, \ldots, k$$

after suitable scaling, as the local basis functions. However, we emphasize that the procedure described below does not depend on the specific basis chosen for the polynomials and works also in multi dimensions. The Legendre polynomials are L_2-orthogonal, namely,

$$\int_{-1}^{1} W_l(s)W_{l'}(s)ds = \frac{2}{2l+1}\delta_{ll'}.$$

And for $x \in I_i$, we can express our approximate solution u^h as follows:

$$u^h(x,t) = \sum_{l=0}^{k} u_i^{(l)}(t)\, v_l^{(i)}(x) \quad \text{for } x \in I_i, \tag{2.2}$$

where

$$v_l^{(i)}(x) = W_l(2(x-x_i)/\Delta x_i)$$

and $u_i^{(l)}(t),\ l = 0,\ldots,k$ are the degrees of freedom. We will omit the argument t and denote them as $u_i^{(l)}$ in this paper.

In order to determine the approximate solution, we evolve the degrees of freedom $u_i^{(l)}$:

$$\frac{\Delta x_i}{2l+1}\frac{d}{dt}u_i^{(l)} - \int_{I_i} f(u^h(x,t))\frac{d}{dx}v_l^{(i)}(x)dx + \hat{f}(u^-_{i+\frac{1}{2}}, u^+_{i+\frac{1}{2}}) - (-1)^l\hat{f}(u^-_{i-\frac{1}{2}}, u^+_{i-\frac{1}{2}}) = 0, \quad l = 0,\ldots,k, \tag{2.3}$$

where $u^{\pm}_{i+1/2} = u^h(x^{\pm}_{i+1/2}, t)$ are the left and right limits of the discontinuous solution u^h at the cell interface $x_{i+1/2}$, and $\hat{f}(u^-, u^+)$ is a consistent and monotone (nondecreasing in the first argument and nonincreasing in the second argument) flux for the scalar case and an exact or approximate Riemann solver for the system case. The ODE system (2.3) is discretized in time by a nonlinearly stable Runge–Kutta time discretization, e.g. the third order version in [30]. The integral term in (2.3) can be computed either exactly or by a suitable numerical quadrature accurate to at least $O(h^{k+l+2})$.

In order to maintain stability and non-oscillatory property of the RKDG method for solving conservation laws (2.1) with strong shocks, a nonlinear limiter must be applied. We adopt the following framework for limiter procedure in this paper: First we identify the "troubled cells", namely those cells which might need the limiting procedure; then we replace the solution polynomials in those troubled cells by reconstructed polynomials which maintain the original cell averages (conservation), have the same orders of accuracy as before, but are less oscillatory. For the first part of limiter procedure, we will identify troubled cells using different indicators which are also used as indicator for the adaptive strategy. These indicators will be described in Subsection 2.2. For the second part of limiter, we will reconstruct the polynomials by the WENO type reconstruction described in Subsection 2.3.

2.2. Review of troubled-cell indicators

Seven different types of indicators were described in [25], which we will use in this paper to identify the troubled-cells and also use as indicators for the adaptive strategy.

1. The minmod-based TVB limiter [8]. (TVB, we will use the same short name as in [25] for each indicator). Denote

$$u^-_{i+\frac{1}{2}} = u_i^{(0)} + \tilde{u}_i, \quad u^+_{i-\frac{1}{2}} = u_i^{(0)} - \tilde{\tilde{u}}_i.$$

From (2.2) we can derive

$$\tilde{u}_i = \sum_{l=1}^{k} u_i^{(l)} v_l^{(i)}(x_{i+\frac{1}{2}}), \quad \tilde{\tilde{u}}_i = -\sum_{l=1}^{k} u_i^{(l)} v_l^{(i)}(x_{i-\frac{1}{2}}).$$

These are modified by the TVB-modified minmod function

$$\begin{aligned} \tilde{u}_i^{(mod)} &= \tilde{m}(\tilde{u}_i, u_{i+1}^{(0)} - u_i^{(0)}, u_i^{(0)} - u_{i-1}^{(0)}), \\ \tilde{\tilde{u}}_i^{(mod)} &= \tilde{m}(\tilde{\tilde{u}}_i, u_{i+1}^{(0)} - u_i^{(0)}, u_i^{(0)} - u_{i-1}^{(0)}), \end{aligned} \tag{2.4}$$

where $\tilde{m}$ is given by

$$\tilde{m}(a_1, a_2, \ldots, a_n) = \begin{cases} a_1 & \text{if } |a_1| \leqslant Mh^2, \\ m(a_1, a_2, \ldots, a_n) & \text{otherwise} \end{cases} \tag{2.5}$$

and the minmod function m is given by

$$m(a_1, a_2, \ldots, a_n) = \begin{cases} s \cdot \min_{1\leqslant j\leqslant n}|a_j| & \text{if } \operatorname{sign}(a_1) = \operatorname{sign}(a_2) = \cdots = \operatorname{sign}(a_n) = s, \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

The parameter $M > 0$ is a constant. If one of the two functions in (2.4) returns other than the first argument, the cell is declared as a troubled cell. Unfortunately, the TVB limiter constant M is dependent on the problem, there is no automatic switching which works well for various situations. As it was pointed out in [7], the resolution of solution is dependent on the choice of the constant M; and sometimes, the case of $k = 1$ may give better resolution to shock or contact discontinuous than the case of $k = 2$ if an inappropriate M is chosen. Tuning M requires much experimental research. But once we get the appropriate M, the TVB limiter will make the solution much better. We have experimented and settled on the following: one choice is $M = 0.1$ which is small enough to avoid oscillations for all the test problems and the troubled-cell indicator is denoted as TVB-1. The other choice of M depends on a specific problem. We choose it neither too small nor too big for our comparison, and the troubled-cell indicator is denoted as TVB-2.

2. Moment limiter of Biswas et al. [3] (BDF).

$$u_i^{(l),mod} = \frac{1}{(2l-1)} m\Big((2l-1)u_i^{(l)}, u_{i+1}^{(l-1)} - u_i^{(l-1)}, u_i^{(l-1)} - u_{i-1}^{(l-1)}\Big), \quad 1 \leqslant l \leqslant k, \tag{2.7}$$

where m is again the minmod function. If the function returns other than the first argument for the highest order case $l = k$, the cell is identified as a troubled cell.

3. A modification of the moment limiter by Burbeau et al. [4] (BSB). If both (2.7) and

$$\hat{u}_i^{(l),mod} = \frac{1}{(2l-1)} m\Big((2l-1)u_i^{(l)}, u_{i+\frac{1}{2}}^{(l-1)+} - u_i^{(l-1)}, u_i^{(l-1)} - u_{i-\frac{1}{2}}^{(l-1)-}\Big), \quad 1 \leqslant l \leqslant k, \tag{2.8}$$

where

$$u_{i+\frac{1}{2}}^{(l-1)+} = u_{i+1}^{(l-1)} - (2l-1)u_{i+1}^{(l)}, \qquad u_{i-\frac{1}{2}}^{(l-1)-} = u_{i-1}^{(l-1)} + (2l-1)u_{i-1}^{(l)},$$

are enacted for the highest order moment $u_i^{(k)}$, then the cell is identified as a troubled cell.

4. The monotonicity-preserving limiter [32] (MP). Define

$$\text{median}(x, y, z) = x + m(y - x, z - x), \tag{2.9}$$

where m is the minmod function. If

$$u_{i+\frac{1}{2}}^{-} \neq \text{median}\Big(u_{i+\frac{1}{2}}^{-}, u_{i+\frac{1}{2}}^{min}, u_{i+\frac{1}{2}}^{max}\Big), \tag{2.10}$$

where

$$u_{i+\frac{1}{2}}^{min} = \max\Big[\min(u_i^{(0)}, u_{i+1}^{(0)}, u_{i+\frac{1}{2}}^{MD}),\ \min(u_i^{(0)}, u_{i+\frac{1}{2}}^{UL}, u_{i+\frac{1}{2}}^{LC})\Big],$$
$$u_{i+\frac{1}{2}}^{max} = \min\Big[\max(u_i^{(0)}, u_{i+1}^{(0)}, u_{i+\frac{1}{2}}^{MD}),\ \max(u_i^{(0)}, u_{i+\frac{1}{2}}^{UL}, u_{i+\frac{1}{2}}^{LC})\Big]$$

and

$$d_i = u_{i+1}^{(0)} - 2u_i^{(0)} + u_{i-1}^{(0)},$$
$$d_{i+\frac{1}{2}}^{M4X} = m(4d_i - d_{i+1}, 4d_{i+1} - d_i, d_i, d_{i+1}, d_{i-1}, d_{i+2}),$$
$$u_{i+\frac{1}{2}}^{\text{MD}} = \frac{1}{2}\Big(u_i^{(0)} + u_{i+1}^{(0)} - d_{i+\frac{1}{2}}^{M4X}\Big), \quad u_{i+\frac{1}{2}}^{\text{UL}} = u_i^{(0)} + \alpha\Big(u_i^{(0)} - u_{i-1}^{(0)}\Big),$$
$$u_{i+\frac{1}{2}}^{\text{LC}} = u_i^{(0)} + \frac{1}{2}\Big(u_i^{(0)} - u_{i-1}^{(0)}\Big) + \frac{\beta}{3} d_{i-\frac{1}{2}}^{M4X}$$

or if $u_{i-\frac{1}{2}}^{+}$ satisfies a similar (symmetric) condition, then the cell is identified as a troubled cell. We take the parameters $\alpha = 2$ and $\beta = 4$ in the numerical tests in the next section, as suggested in [32].

5. A modification of the MP limiter [29] (MMP).

$$\phi = \min\left(1, \Delta\bar{u}^{\min}/\Delta_{\min} u\right), \tag{2.11}$$

where

$$\Delta\bar{u}^{\min} = u_i^{(0)} - \min\Big(u_{i-1}^{(0)}, u_i^{(0)}, u_{i+1}^{(0)}\Big), \quad \Delta_{\min} u = u_i^{(0)} - \min\Big(u_{i-\frac{1}{2}}^{+}, u_{i+\frac{1}{2}}^{-}\Big).$$

When $\phi \neq 1$, the limiter enacts and the cell is identified as a troubled cell.

6. A shock-detection technique by Krivodonova et al. [21] (KXRCF). Partition the boundary of a cell I_i into two portions ∂I_i^- and ∂I_i^+, where the flow is into ($\vec{v} \cdot \vec{n} < 0, \vec{n}$ is the normal vector to ∂I_i) and out of ($\vec{v} \cdot \vec{n} > 0) I_i$, respectively. The cell I_i is identified as a troubled cell, if

$$\frac{\left|\int_{\partial I_i^-} (u^h|_{I_i} - u^h|_{I_{n_i}}) ds\right|}{h_i^{\frac{k+1}{2}} |\partial I_i^-| \|u^h|_{I_i}\|} > 1. \tag{2.12}$$

Here we choose h_i as the radius of the circumscribed circle in the element I_i. I_{n_i} is the neighbor of I_i on the side of ∂I_i^- and the norm is based on an element average in one dimension.

Remark 1. In [25], there was still one more troubled-cell indicator which was based on Harten's subcell resolution idea [14]. But it was unstable in some cases of our numerical experiments. So we will not use it in this paper.

For the case of hyperbolic systems, to identify the troubled cells, we could use either a componentwise indicator or a characteristic one. The former works on each component of the solution and identifies a troubled cell when any component of the solution is flagging this cell as a troubled cell. The latter works in the local characteristic direction to do this identification. Their advantages and disadvantages are compared in [25] and the former is the choice there. In this paper we have to use the characteristic one, because some of the componentwise indicators are unstable in our methods when nonuniform meshes are used.

2.3. WENO type reconstruction

For simplicity we describe the reconstruction in one-dimensional scaler case. If the cell I_i is identified as a troubled cell, we will reconstruct the degrees of freedom $u_i^{(l)}$ for $l = 1, \ldots, k$ and retain the cell average $u_i^{(0)}$. We denote $k+1$ stencils $S_j, j = 0, \ldots, k$, where $S_j = \bigcup_{l=0}^{k} I_{i+j-l}$. We can construct the kth degree polynomial $p_j(x)$ on stencil S_j such that:

$$\frac{1}{\Delta x_{i+j-l}} \int_{I_{i+j-l}} p_j(x)dx = u_{i+j-l}^{(0)}, \quad l = 0, \ldots, k.$$

Because the degree of polynomial $p_j(x)$ is k for $j = 0, \ldots, k$, for any linear weights $\eta_0, \ldots, \eta_k$ with $\sum_{j=0}^{k} \eta_j = 1$, the order of $\sum_{j=0}^{k} \eta_j p_j(x)$ approximating to the solution u is $k+1$. In this paper we choose:

$$\eta_0 = \frac{1}{2}, \quad \eta_1 = \frac{1}{2}, \quad \text{for } k = 1,$$
$$\eta_0 = \frac{1}{4}, \quad \eta_1 = \frac{1}{2}, \quad \eta_2 = \frac{1}{4}, \quad \text{for } k = 2.$$

Next, calculate the smoothness indicators:

$$\beta_j = \sum_{l=1}^{k} \int_{I_i} \Delta x_i^{2l-1} \left(\frac{\partial^l}{\partial x^l} p_j(x) \right)^2 dx$$

and then the nonlinear weights:

$$w_j = \frac{\bar{w}_j}{\sum_l \bar{w}_l}, \quad \bar{w}_j = \frac{\eta_j}{\sum_l (\varepsilon + \beta_l)^2},$$

where ε is a small number to avoid the denominator becoming zero. We use $\varepsilon = 10^{-6}$ in all computations in this paper. Finally the WENO type approximation is given by

$$u^{h,\mathrm{WENO}}|_{I_i} = \sum_{j=0}^{k} w_j p_j(x),$$

which will replace the current solution u^h on I_i. For the system case, the WENO type reconstructions are performed in local characteristic directions.

Remark 2. There are other reconstruction methods such as TVB reconstruction [8,7] and WENO reconstruction [26]. We neither choose TVB reconstruction for which turns to decay the order of the method; nor the WENO reconstruction described in [26], though the linear weights are optimal, they depend on the geometry of mesh, and have to be recomputed after geometry of meshes is modified. The procedure of computation for the optimal linear weights would cost much CPU time, and the computation results with these two WENO types reconstruction are similar.

3. The *h*-method for RKDG using troubled-cell indicators

For the h-method, the key point is to identify where the mesh should be refined and coarsened. This work can be done by troubled-cell indicators because they tell us where the discontinuities are. We can refine the troubled cells and coarsen cells which are not troubled. Following this idea, the h-method can be illustrated by the following flowchart:

Algorithm 1 (*h-method using troubled-cell indicators).* Given the maximum refined mesh level *LEV* and the solving time T,

- Step 1. Given a uniform partition of the domain, compute the degrees of freedom $\{u_i^{(l)}(t_0)\}$ from the initial data $u_0(x)$, and set all cells' initial mesh level $\{lev_i(0)\} = 0$.

- Step 2. Suppose we have known the mesh $\{I_i^n\}$, the mesh level $\{lev_i^n\}$ and the degrees of freedom $\{u_i^{(l)}(t_n)\}$ at time level t_n. Do the troubled-cell indicator procedure, mark every cell as a troubled cell or an untroubled cell.
  - For a troubled cell, if its mesh level $lev_i^n = LEV$, do nothing. If $lev_i^n < LEV$, divide it into two cells and set the two new cells' mesh level to be $lev_i^{n+1} = lev_i^n + 1$.
  - For a pair of untroubled cells which come from one dividing at some previous time step, merge them and set the new cell's mesh level to be $lev_i^{n+1} = lev_i^n - 1$.

  Now we get the new mesh $\{I_i^{n+1}\}$ and the corresponding new mesh level $\{lev_i^{n+1}\}$.
- Step 3. Using L_2 projection, project the degrees of freedom $\{u_i^{(l)}(t_n)\}$ on the mesh $\{I_i^n\}$ to the new mesh $\{I_i^{n+1}\}$ and denote the results by $\{\tilde{u}_i^{(l)}(t_n)\}$.
- Step 4. Evolve the solution from t_n to t_{n+1} by the RKDG procedure and get $\{u_i^{(l)}(t_{n+1})\}$ on the mesh $\{I_i^{n+1}\}$.
- Step 5. If $t_{n+1} < T$, go to Step 2.

We remark that each troubled-cell indicator in ALG 1 (we abbreviate Algorithm to ALG in this paper) has two functions. One is to control the mesh refinement, the other is to determine the limiting cells.

In what follows we will implement the *h*-method described in ALG 1 and make a comparison of its performances with different troubled-cell indicators. We will also choose another *h*-method to compare with ours. In [24], a general *h*-method is provided which can be followed for any choice of an error indicator. The use of solution gradient is popular for the error indicator [3,12]. Following these, we give a simple *h*-method, and compare its performance with ours.

Algorithm 2 (*A simple h-method).* The difference between ALG 1 and 2 is only in Step 2. For ALG 2 we switch Step 2 to:

- Step 2′. Suppose we have known the mesh $\{I_i^n\}$, the mesh level $\{lev_i^n\}$ and the degrees of freedom $\{u_i^{(l)}(t_n)\}$ at time level t_n. Calculate the differences of average between neighboring cells $g_i = |u_i^{(0)} - u_{i-1}^{(0)}|,\ i = 2, \ldots, N$ and let $G = max_{i=2}^N g_i$. For cell I_i, if its mesh level $lev_i < LEV$ and if its left or right difference of average, which is g_i or g_{i+1} respectively, is bigger than $\varphi_1 G$, we divide the cell into two cells and set the two cells' mesh level to be $lev_i + 1$. For a pair of cells I_{i-1} and I_i that come from one dividing at some previous time step, if $g_{i+m} < \varphi_2 G$ for $m = -1, 0, 1$, merge them and set the new cell's mesh level to be $lev_i - 1$. Now we get the new mesh $\{I_i^{n+1}\}$ and the corresponding new mesh level $\{lev_i^{n+1}\}$.

In this paper, we choose $\varphi_1 = 0.7, \varphi_2 = 0.4$. In one-dimensional system case, *Step* 2′ works in a componentwise way.

We consider the one-dimensional Euler equations of gas dynamics with three different initial data. The PDEs are

$$\begin{bmatrix} \rho \\ \rho v \\ E \end{bmatrix}_t + \begin{bmatrix} \rho v \\ \rho v^2 + p \\ v(E+p) \end{bmatrix}_x = 0.$$

Here ρ is the density, v is the velocity, E is the total energy, p is the pressure, related to the total energy by $E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho v^2$ with $\gamma = 1.4$.

Example 3.1. The Lax problem. The initial condition is

$$(\rho, v, p) = \begin{cases} (0.445, 0.698, 3.528) & \text{if } x \leqslant 0, \\ (0.5, 0, 0.571) & \text{if } x > 0. \end{cases}$$

We use both ALG 1 and 2 to solve this problem till $T = 1.3$. In the TVB-2 indicator, $M = 1$.

We first compare the performances of different troubled-cell indicators in ALG 1. All the computations in this paper are performed on a computer with CPU "AMD Opteron(tm) Processor 850" – 2.4 GHz and RAM 32 GB. In Table 3.1, we show the results of CPU time and mesh details, including total dividing times *TDT*, cell numbers at the last time level N_T, and average cell numbers $\overline{N}$ which is given by $\overline{N} = \left(\sum_{q=0}^{TOT} N_q\right)/TOT$, where N_q is the cell number at the qth time level, and *TOT* is the total number of time levels. Total merging times *TMT* will not be shown in the table as it can be calculated by $TMT = N_0 + TDT - N_T$.

From Table 3.1 we can see that for ALG 1 the three troubled-cell indicators BDF, BSB and MP work badly in this problem, for they result in too much mesh refinement (and coarsening) and too many cells, and cost much more CPU time than other indicators. For the other indicators we show the computed density ρ and mesh changing in Figs. 3.1 and 3.2. In the density figure, the solid line is the exact solution, and "□" represents average density of a cell. In the mesh changing figure, each "□" represents a "dividing" and each "+" represents a "merging".

From the figures we can see that the solution with MMP is not good. There are oscillations between the two discontinuities for both $k = 1$ and $k = 2$. For the TVB-1 indicator, the solution at the contact discontinuity has too many transition points. For KXRCF with $k = 1$, oscillations appear at the contact discontinuity. For KXRCF with $k = 2$, and for the TVB-2 indicator, the solutions are good. All the refinement is at or near the discontinuities and the final meshes are perfectly what we need. We use very few cells and get very sharp shocks.

Using more cells will give us better solutions, but will also cost more CPU time. So it is hard to tell which indicator is the best (most efficient) for ALG 1. We compute the L_1 error for density and CPU time for all the indicators using

Table 3.1
The Lax problem, ALG 1, CPU time and mesh details.

N_0	Troubled-cell indicators	$k=1$				$k=2$			
		TDT	N_T	$\overline{N}$	cpu_1	*TDT*	N_T	$\overline{N}$	cpu_1
50	TVB-1	7.0E+2	63	62.4	0.3	1.0E+4	155	119.2	2.7
	TVB-2	2.6E+2	50	54.1	0.2	2.3E+3	64	67.4	1.4
	BDF	1.1E+4	668	644.3	2.8	5.0E+4	773	781.4	17.6
	BSB	1.6E+4	682	631.4	2.9	6.7E+4	762	774.5	17.7
	MP	1.2E+4	502	341.9	2.2	2.0E+4	584	418.3	11.4
	MMP	7.3E+3	100	93.3	0.6	1.6E+4	118	106.8	2.8
	KXRCF	5.3E+2	56	56.1	0.3	5.9E+2	57	59.0	1.2
100	TVB-1	4.5E+3	210	150.5	1.5	1.9E+4	310	224.3	10.4
	TVB-2	8.1E+2	106	107.5	0.9	1.2E+4	161	150.8	6.6
	BDF	4.2E+4	1174	1105.7	9.8	1.6E+5	1372	1283.7	58.9
	BSB	6.4E+4	1210	1098.4	10.2	2.2E+5	1345	1285.5	59.6
	MP	3.3E+4	824	541.7	7.0	4.4E+4	1047	657.2	35.5
	MMP	2.1E+4	176	159.9	2.1	4.4E+4	202	176.5	9.3
	KXRCF	1.3E+3	106	107.9	1.0	2.0E+3	108	115.3	4.5

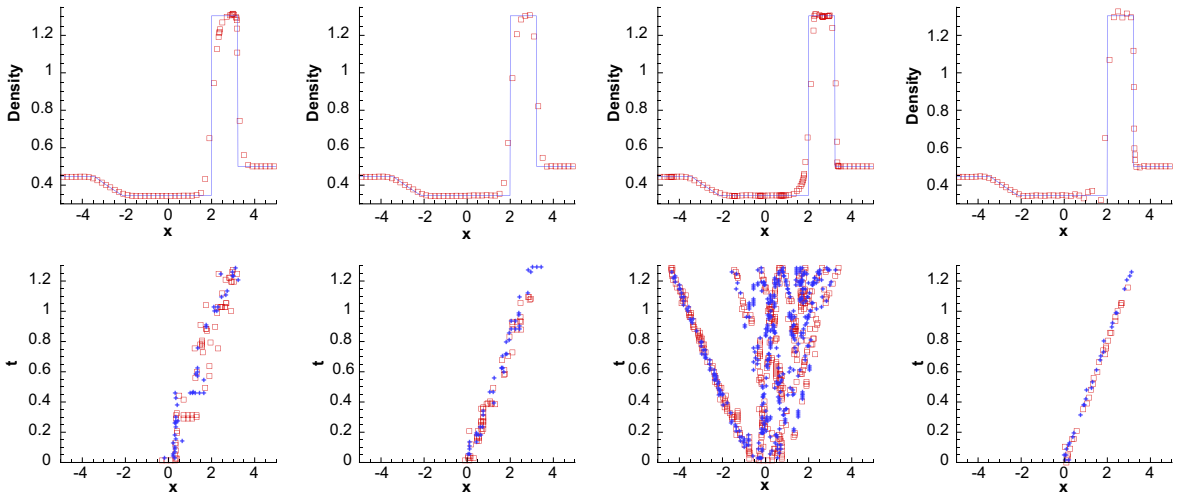

Fig. 3.1. Lax problem, density (top) and mesh changing (bottom), ALG 1 with $N_0 = 50, k = 1, LEV = 4$; from left to right: TVB-1, TVB-2, MMP, KXRCF.

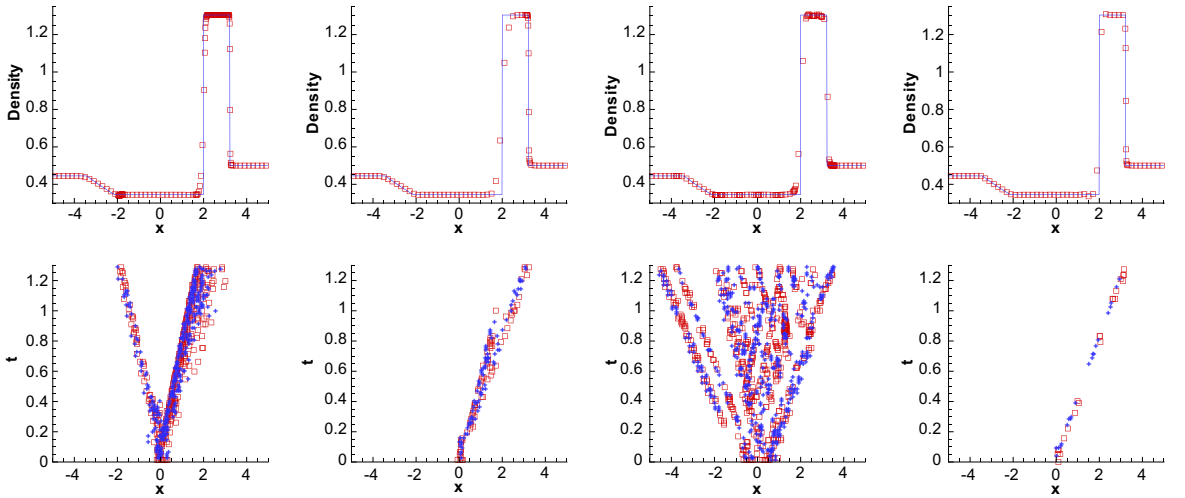

Fig. 3.2. Lax problem, density (top) and mesh changing (bottom), ALG 1 with $N_0 = 50, k = 2, LEV = 4$; from left to right: TVB-1, TVB-2, MMP, KXRCF.

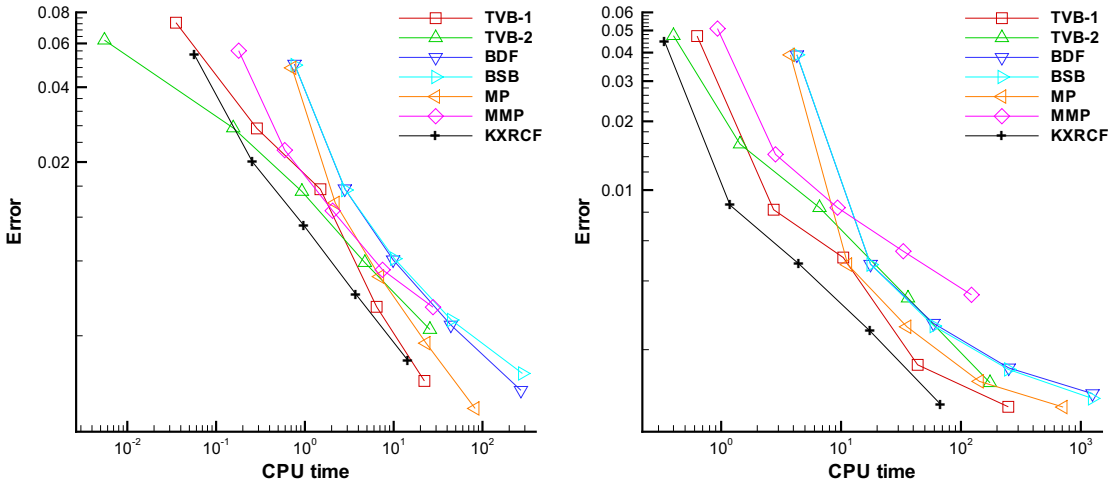

Fig. 3.3. Lax problem, ALG 1, CPU-L_1-error for density. Left, $k = 1$; right, $k = 2$.

$25 \times 2^n (n = 0, \ldots, 4)$ initial cells and plot the CPU-L_1-error curves in Fig. 3.3. These figures show the convergence of ALG 1, and with KXRCF ALG 1 has the highest efficiency.

For ALG 2 we find all the results are almost the same with respect to different troubled-cell indicators which only work in the limiter part. Also we find from the solutions that all the refinement is at the discontinuities, but the resolution is not good. The distribution of cells depends totally on the gradient of the solution, so there are always too many transition cells around discontinuities. We plot the solutions with TVB-1, TVB-2 and KXRCF indicators as examples in Fig. 3.4. As the results for ALG 2 are not satisfactory in both this test case and other test cases, we will show no more results for this algorithm.

After all the comparisons above we conclude that for the Lax problem, BDF, BSB, MP and MMP are bad indicators for ALG 1. This is also true for other test problems. BDF, BSB, MP troubled-cell indicators always result in too much refinement and coarsening and the solution with MMP indicator often has oscillations or overshoots. So we will show no more results for these indicators except CPU-L_1-error figure. We also conclude for Lax problem that ALG 1 with TVB and KXRCF indicators performs better than ALG 2 and KXRCF is the best troubled-cell indicator.

We also compared other cases with different values of N_0 and *LEV*. Their performances are similar (which is also true for other test problems below). So in this paper we will only show the results with $LEV = 4$.

Example 3.2. The shock density wave interaction problem. The solution of this problem contains both shocks and complex smooth regions. The initial condition is

$$(\rho, v, p) = \begin{cases} (3.857143, 2.629369, 10.333333) & \text{if } x < -4, \\ (1 + 0.2sin(5x), 0, 1) & \text{if } x \geqslant -4. \end{cases}$$

We will compute the solution up to $T = 1.8$. In the TVB-2 indicator, the parameter $M = 100$. The reference 'exact' solutions which will be used later are computed by a fifth-order WENO scheme in [18] using 5000 grid points.

Again, we will compare the CPU time and mesh first. From Table 3.2 we can see the results are similar to those in Table 3.1.

For the solutions, we show the results with TVB-1, TVB-2 and KXRCF indicators in Figs. 3.5 and 3.6. From these figures we can see that for $k = 1$, all the solutions are not satisfactory. But for $k = 2$, all the solutions are good.

At last we compare the efficiency. We plot the CPU-L_1-error curves for density in Fig. 3.7. Again in this case we see that KXRCF is the best.

Example 3.3. The blast wave problem. This problem involves interaction of blast waves and its initial condition is given by

$$(\rho, v, p) = \begin{cases} (1, 0, 1000) & \text{if } 0 \leqslant x < 0.1, \\ (1, 0, 0.01) & \text{if } 0.1 \leqslant x < 0.9, \\ (1, 0, 100) & \text{if } 0.9 \leqslant x \leqslant 1. \end{cases}$$

A reflecting boundary condition is applied to both ends, see [34,15]. We will compute the solution till $T = 0.038$. In the TVB-2 indicator, the parameter $M = 10$. The reference 'exact' solutions for this problem are also computed by a fifth-order WENO scheme in [18] using 5000 grid points.

For all the results are similar to the two above examples, we will just present the table (see Table 3.3) and the figures (see Figs. 3.8–3.10), and leave the conclusions to the end of this section.

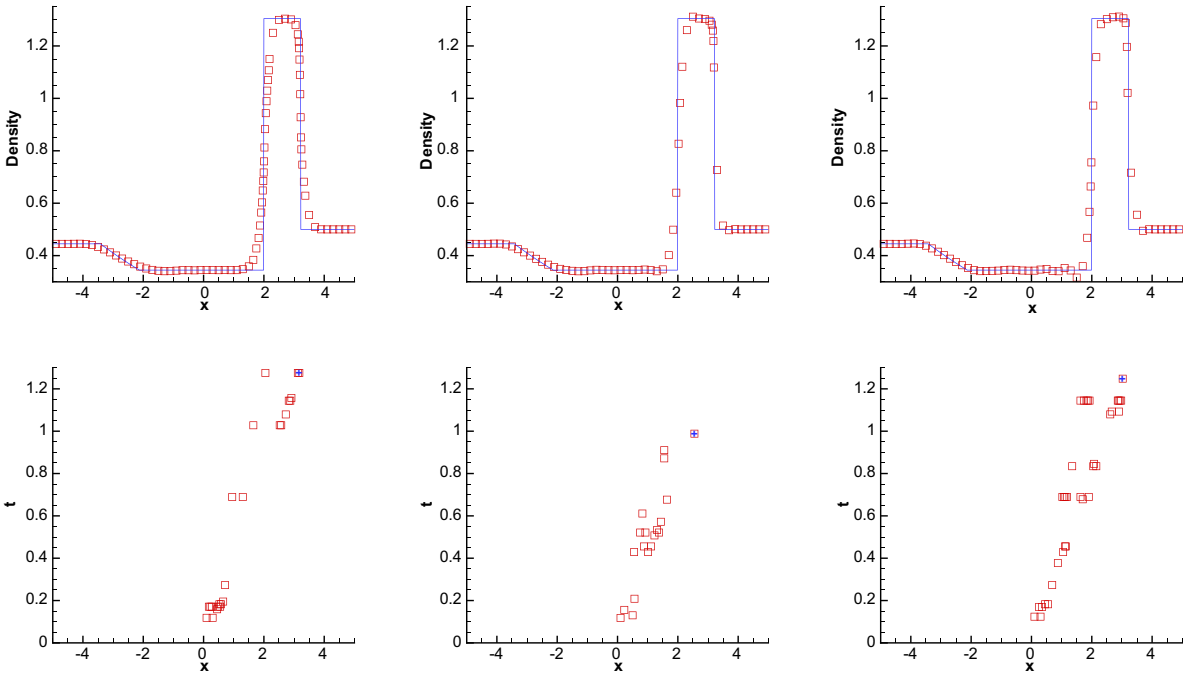

Fig. 3.4. Lax problem, density (top) and mesh changing (bottom); ALG 2 with TVB-1 (left), TVB-2 (middle) and KXRCF (right) indicators, $N_0 = 50, k = 1, LEV = 4$.

Table 3.2
The shock density wave interaction problem, ALG 1, CPU time and mesh details.

N_0	Troubled-cell indicators	$k = 1$				$k = 2$			
		TDT	N_T	$\overline{N}$	cpu_1	*TDT*	N_T	$\overline{N}$	cpu_1
50	TVB-1	4.6E+3	114	118.4	1.5	8.4E+4	276	235.9	15.9
	TVB-2	4.1E+2	56	54.2	0.2	9.6E+3	112	86.9	5.1
	KXRCF	6.4E+2	59	56.6	0.3	6.6E+2	60	59.4	2.8
100	TVB-1	1.4E+4	217	193.4	5.3	2.6E+5	415	335.0	45.7
	TVB-2	1.2E+3	107	104.8	1.9	3.6E+4	195	151.9	18.2
	KXRCF	1.3E+3	109	107.5	2.1	1.5E+3	109	109.9	11.5

From the above three examples we can conclude that: (1) BDF, BSB, MP and MMP are not proper troubled-cell indicators for ALG 1; (2) TVB and KXRCF are proper indicators for ALG 1. With them we can use very few cells and get very sharp shocks; (3) ALG 1 with TVB and KXRCF indicators performs better than ALG 2; (4) KXRCF is the best indicators for ALG 1.

Remark 3. The above facts also verify the conclusion in [25] that BDF, BSB MP and MMP troubled-cell indicators are not as good as TVB and KXRCF.

Remark 4. For shock problems, L_1 order of an h-method is at most second order with respect to the number of degrees of freedom. But ALG 1 only delivers linear convergence rate. This is because the optimal convergence rate is achieved only when the maximum mesh level *LEV* is depend on the initial mesh size. In ALG 1 *LEV* is fixed.

4. The *r*-method for RKDG using troubled-cell indicators

In this section, we will apply the troubled-cell indicator idea to the *r*-method. First, we will take a brief introduction to this *r*-method. We will use the framework in Tang and Tang [33] because in their method mesh redistribution and PDE time evolution are independent from each other, and the PDE time evolution part can be any suitable high-resolution method such as the wave-propagation algorithm, central schemes, and ENO methods. In this paper, we will use the RKDG method in this part.

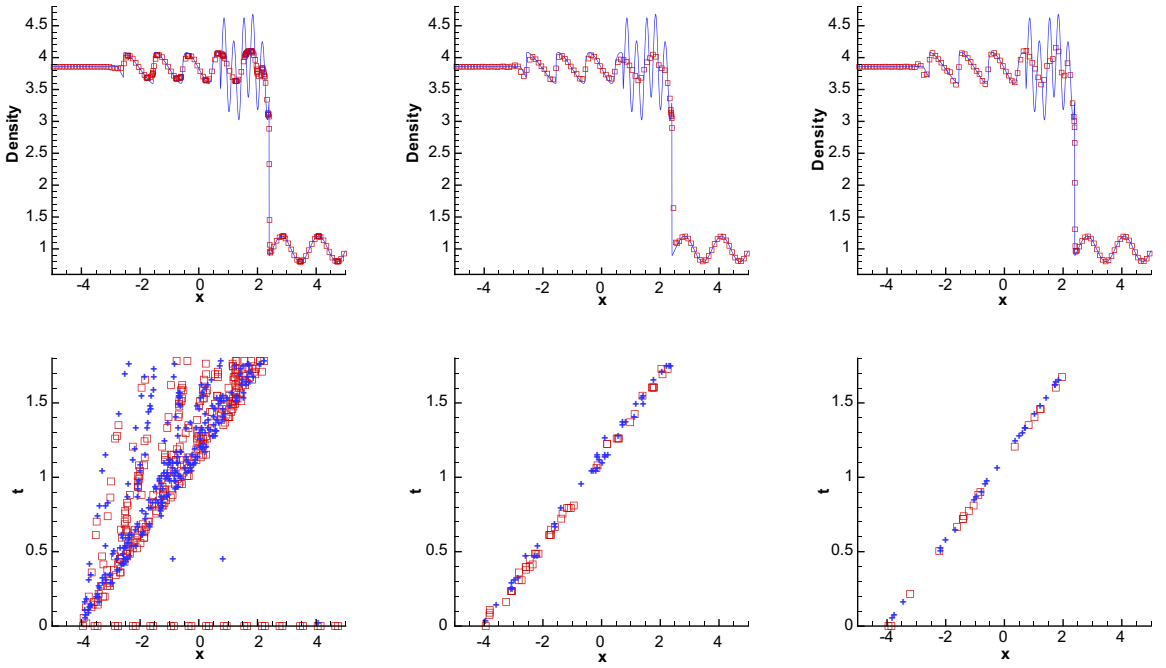

Fig. 3.5. The shock density wave interaction problem, density (top) and mesh changing (bottom), ALG 1 with TVB-1 (left), TVB-2 (middle) and KXRCF (right) indicators, $N_0 = 100, k = 1$.

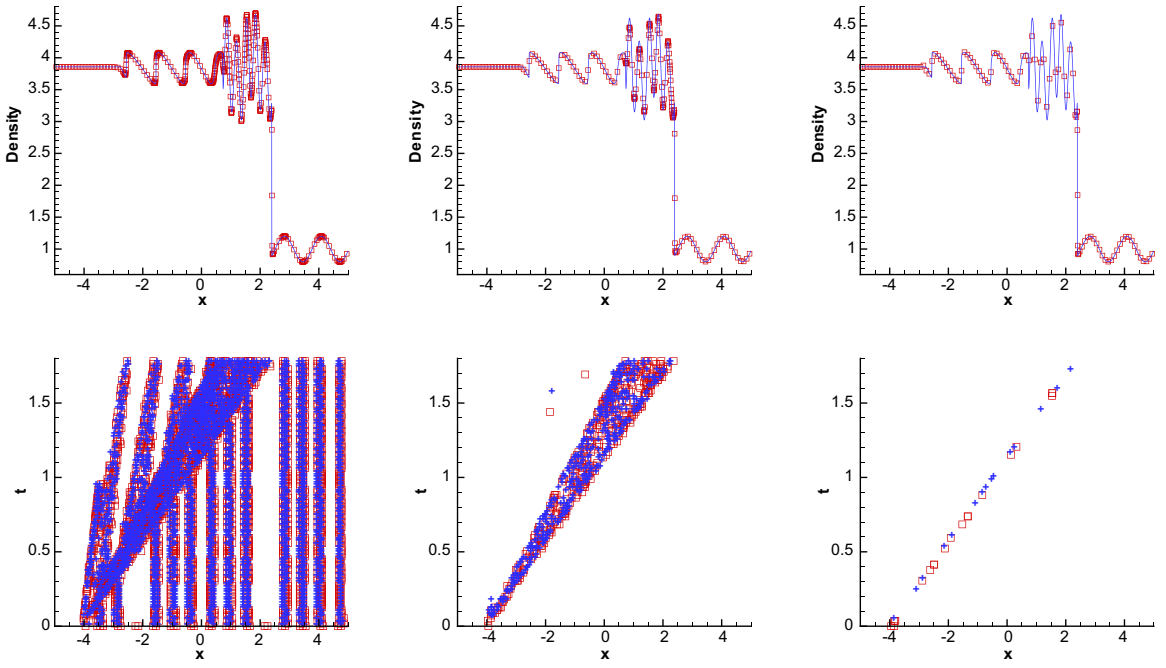

Fig. 3.6. The shock density wave interaction problem, density (top) and mesh changing (bottom), ALG 1 with TVB-1 (left), TVB-2 (middle) and KXRCF (right) indicators, $N_0 = 100, k = 2$.

Consider the one-dimensional scalar case. Let x and ξ be the physical and logical coordinates. Their corresponding domains, without loss of generality, are assumed to be $[a, b]$ and $[0, 1]$. A one-to-one coordinate transformation between the two domains is denoted by

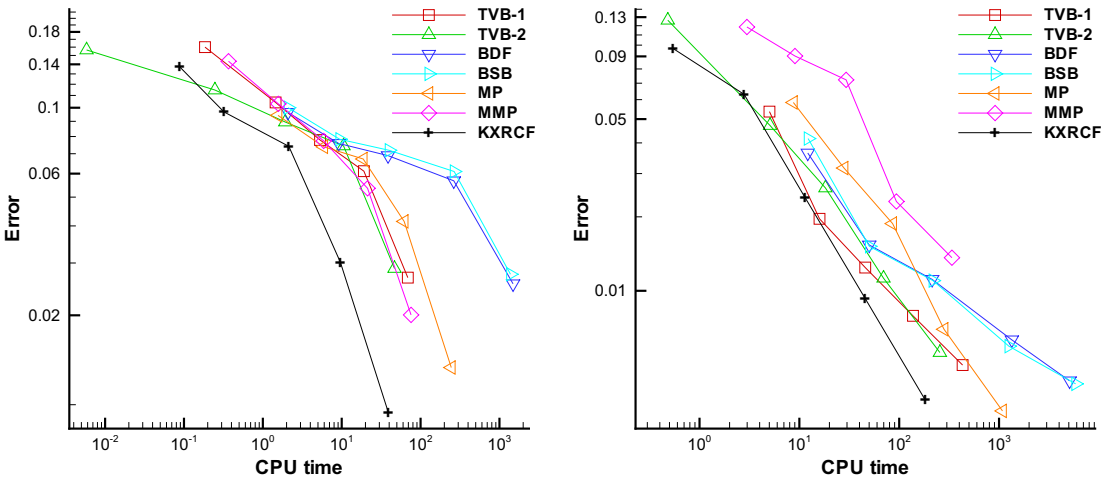

Fig. 3.7. The shock density wave interaction problem, ALG 1, CPU-L_1-error for density. Left, $k = 1$; right, $k = 2$.

Table 3.3
The blast wave problem, ALG 1, CPU time and mesh details.

N_0	Troubled-cell indicators	$k = 1$				$k = 2$			
		TDT	N_T	$\overline{N}$	cpu_1	*TDT*	N_T	$\overline{N}$	cpu_1
50	TVB-1	1.8E+4	104	183.9	2.4	1.2E+5	189	327.2	20.9
	TVB-2	3.1E+3	55	61.5	0.7	1.7E+4	101	96.3	5.3
	KXRCF	9.7E+3	91	114.5	1.3	7.5E+3	147	144.6	8.2
100	TVB-1	7.8E+4	172	388.3	10.6	3.7E+5	443	622.8	80.7
	TVB-2	7.5E+3	125	117.4	7.9	7.4E+4	172	165.3	54.5
	KXRCF	2.8E+4	169	179.6	4.1	1.7E+4	241	226.1	25.7

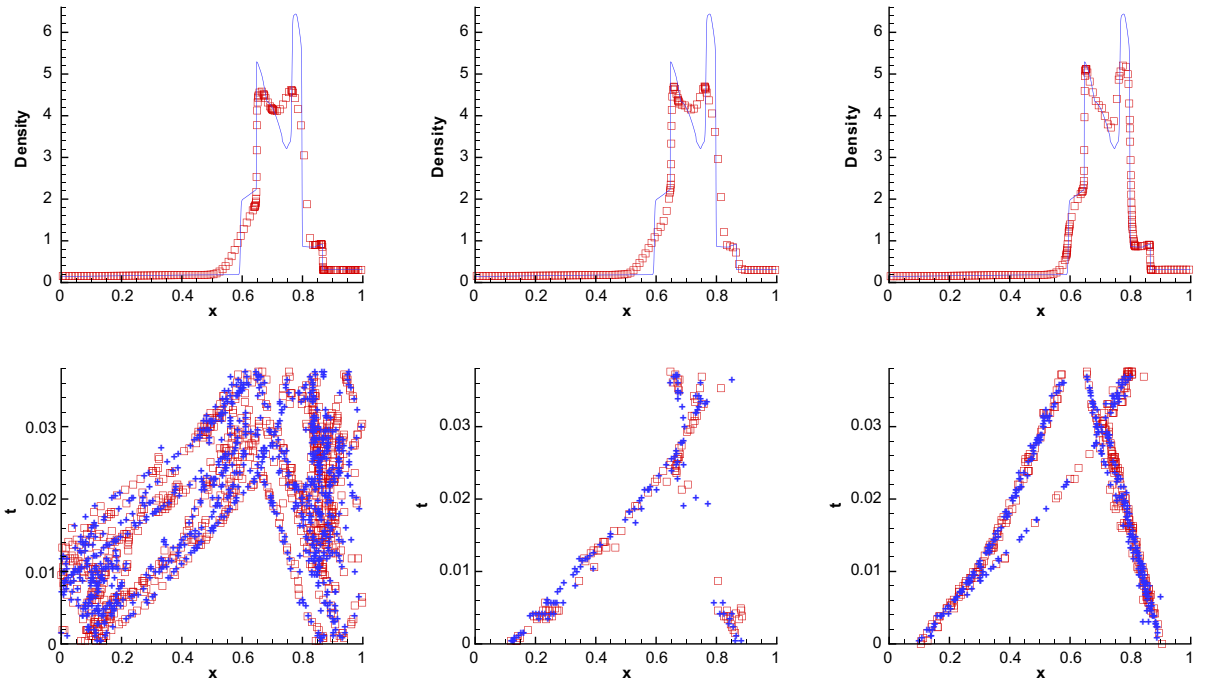

Fig. 3.8. The blast wave problem, density (top) and mesh changing (bottom), ALG 1 with TVB-1 (left), TVB-2 (middle) and KXRCF (right) indicators, $N_0 = 100, k = 1$.

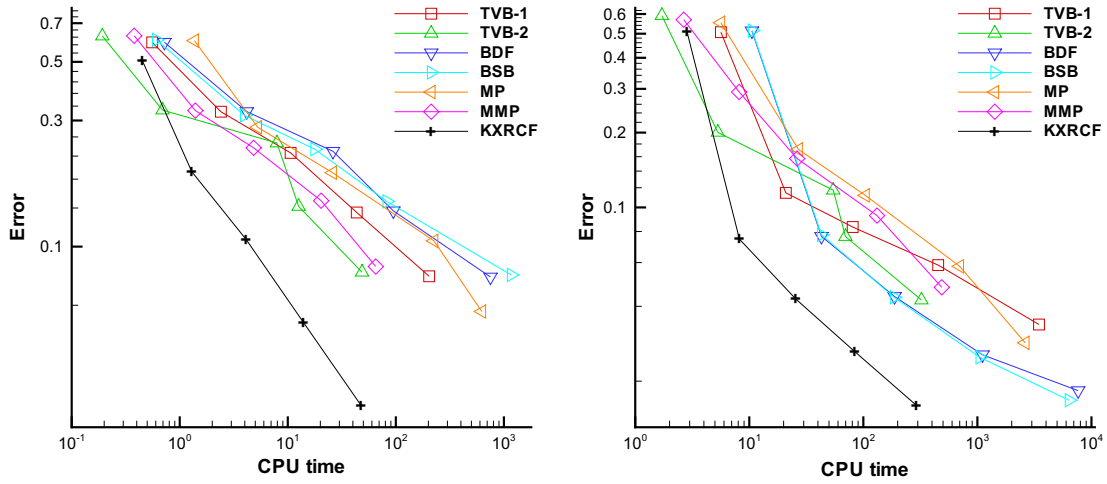

Fig. 3.10. The blast wave problem, ALG 1, CPU-L_1-error for density. Left, $k = 1$; right, $k = 2$.

$$x = x(\xi), \quad \xi \in [0, 1],$$
$$x(0) = a, \quad x(1) = b$$

and can be described by 1D Euler–Lagrange equation

$$(\omega^{-1}\xi_x)_x = 0,$$

or equivalently

$$(\omega x_\xi)_\xi = 0, \tag{4.1}$$

where $\omega = \omega(u)$ is a positive weight function and u is the solution of the underlying PDE. This map transforms a uniform mesh in the logical domain, clustering grid points in those regions of the physical domain where the solution has the largest gradients.

Eq. (4.1) is the mesh-redistribution equation. In order to solve this equation, we introduce an artificial time τ and solve

$$\begin{cases} x_\tau = (\omega x_\xi)_\xi, & 0 < \xi < 1, \\ x(0,\tau) = a, \quad x(1,\tau) = b. \end{cases} \tag{4.2}$$

to static state by iteration. A fixed uniform mesh on the logical domain is given by

$$0 = \xi_{\frac{1}{2}} < \xi_{\frac{3}{2}} < \cdots < \xi_{N+\frac{1}{2}} = 1.$$

Then we have the discretized equation

$$\begin{cases} \tilde{x}_{i+\frac{1}{2}} = x_{i+\frac{1}{2}} + \frac{\Delta\tau}{(\Delta\xi)^2}\left[\omega(u_{i+1})\left(x_{i+\frac{3}{2}} - x_{i+\frac{1}{2}}\right) - \omega(u_i)\left(x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}\right)\right], \quad 1 \leqslant i \leqslant N-1, \\ x_{\frac{1}{2}} = a, \quad x_{N+\frac{1}{2}} = b, \end{cases} \tag{4.3}$$

where $\Delta\xi = \frac{1}{N}$ and $u_i = u(x_i, t)$. Eq. (4.3) is equivalent to

$$\begin{cases} \tilde{x}_{i+\frac{1}{2}} = \alpha_{i+1} x_{i+\frac{3}{2}} + (1 - \alpha_{i+1} - \alpha_i) x_{i+\frac{1}{2}} + \alpha_i x_{i-\frac{1}{2}}, \quad 1 \leqslant i \leqslant N-1, \\ x_{\frac{1}{2}} = a, \quad x_{N+\frac{1}{2}} = b, \end{cases} \tag{4.4}$$

where

$$\alpha_i = \frac{\Delta\tau}{(\Delta\xi)^2}\omega(u_i).$$

Table 4.1
The Lax problem.

N	Troubled-cell indicators	$k=1$					$k=2$				
		err_4	IE	cpu_4	SC	MC	err_4	IE	cpu_4	SC	MC
100	TVB-1	1.74E−2	4.0	0.19	62.0	17.2	1.21E−2	1.4	0.84	52.4	19.6
	TVB-2	9.42E−3	4.1	0.23	63.2	16.6	8.39E−3	2.2	0.88	56.7	16.3
	KXRCF	1.00E−2	4.5	0.21	65.9	17.7	9.14E−3	3.0	0.78	58.3	17.1
200	TVB-1	9.23E−3	1.9	0.69	64.8	10.0	6.36E−3	3.3	3.33	55.5	13.8
	TVB-2	6.21E−3	4.0	0.78	66.7	8.5	5.16E−3	6.0	3.28	58.7	9.0
	KXRCF	5.69E−3	0.6	0.70	69.6	9.1	5.58E−3	8.6	2.94	60.3	9.7

Table 4.2
The shock density wave interaction problem.

N	Troubled-cell indicators	$k=1$					$k=2$				
		err_4	IE	cpu_4	SC	MC	err_4	IE	cpu_4	SC	MC
100	TVB-1	1.11E−1	−1.6	0.50	55.8	21.2	9.02E−2	−0.2	3.95	5.6	88.7
	TVB-2	8.01E−2	25.1	0.39	60.3	12.4	4.47E−2	20.1	1.29	60.2	14.4
	KXRCF	8.25E−2	16.3	0.32	64.4	12.5	5.25E−2	19.5	1.12	61.5	14.4
200	TVB-1	8.29E−2	0.4	1.79	59.9	15.5	6.67E−2	0.7	11.62	21.7	65.5
	TVB-2	5.13E−2	19.8	1.08	71.0	7.0	2.75E−2	8.0	4.89	62.3	10.4
	KXRCF	4.79E−2	24.5	0.90	73.1	7.2	3.15E−2	13.8	4.01	66.6	7.9

Table 4.3
The blast wave problem.

N	Troubled-cell indicators	$k=1$					$k=2$				
		err_4	IE	cpu_4	SC	MC	err_4	IE	cpu_4	SC	MC
100	TVB-1	3.23E−1	2.4	0.89	29.2	57.3	1.94E−1	0.0	4.62	17.5	71.5
	TVB-2	2.84E−1	5.4	0.59	55.9	26.0	1.86E−1	9.7	2.97	46.8	34.2
	KXRCF	2.92E−1	3.1	0.52	57.2	31.9	2.19E−1	7.7	2.97	43.7	40.5
200	TVB-1	2.29E−1	3.6	3.71	37.1	47.7	1.17E−1	3.2	18.78	24.4	62.8
	TVB-2	2.04E−1	6.9	2.30	61.9	16.3	1.04E−1	9.3	11.65	52.7	22.0
	KXRCF	2.26E−1	8.8	1.99	63.6	19.8	1.34E−1	5.9	12.04	49.0	29.0

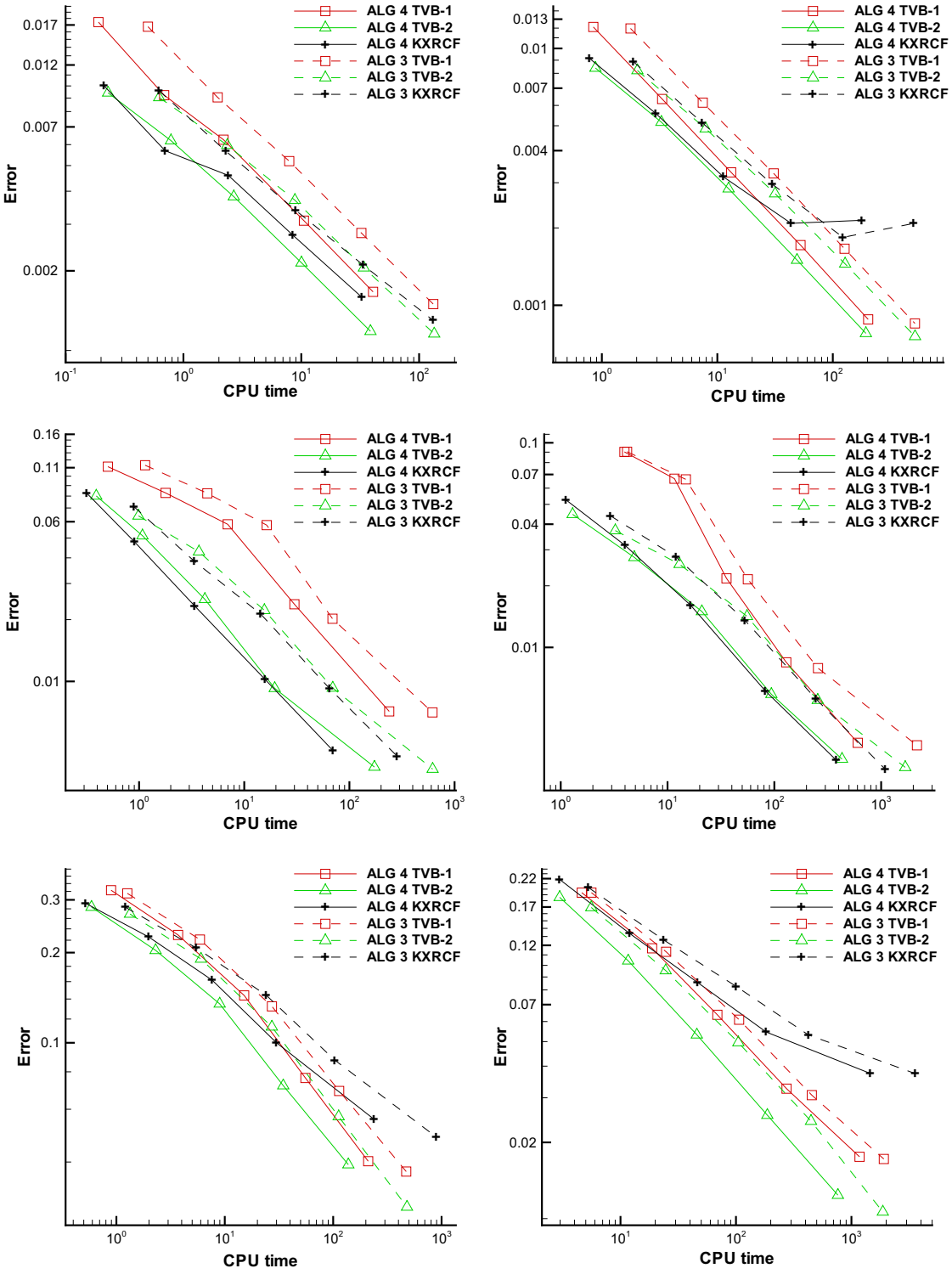

Fig. 4.1. Comparison of efficiency, CPU-L_1-error for density. Left, $k = 1$; right, $k = 2$. Top, Lax problem; middle, shock density wave interaction problem; bottom, blast wave problem.

For stability, we must ensure $\max_i \alpha_i \leqslant \frac{1}{2}$, and this can be controlled by the choice of $\Delta\tau$. In additional, like in [33], in order to smooth the monitor, we let

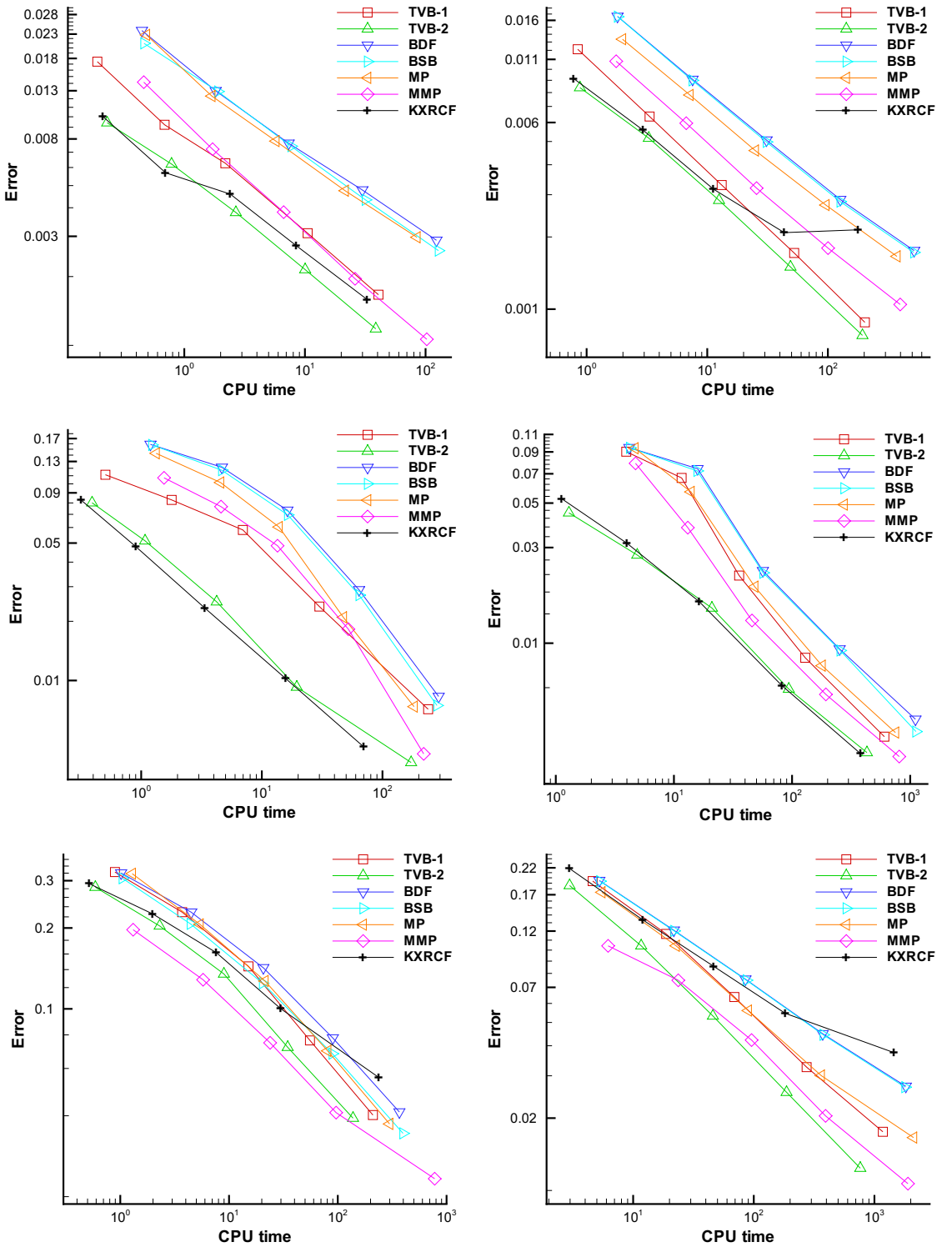

Fig. 4.2. ALG 4, CPU-L_1-error for density. Left, $k = 1$; right, $k = 2$. Top, Lax problem; middle, shock density wave interaction problem; bottom, blast wave problem.

$$\omega(u_i) \leftarrow \frac{1}{4}\omega(u_{i-1}) + \frac{1}{2}\omega(u_i) + \frac{1}{4}\omega(u_{i+1}).$$

The r-method procedure based on RKDG is similar to that in [33].

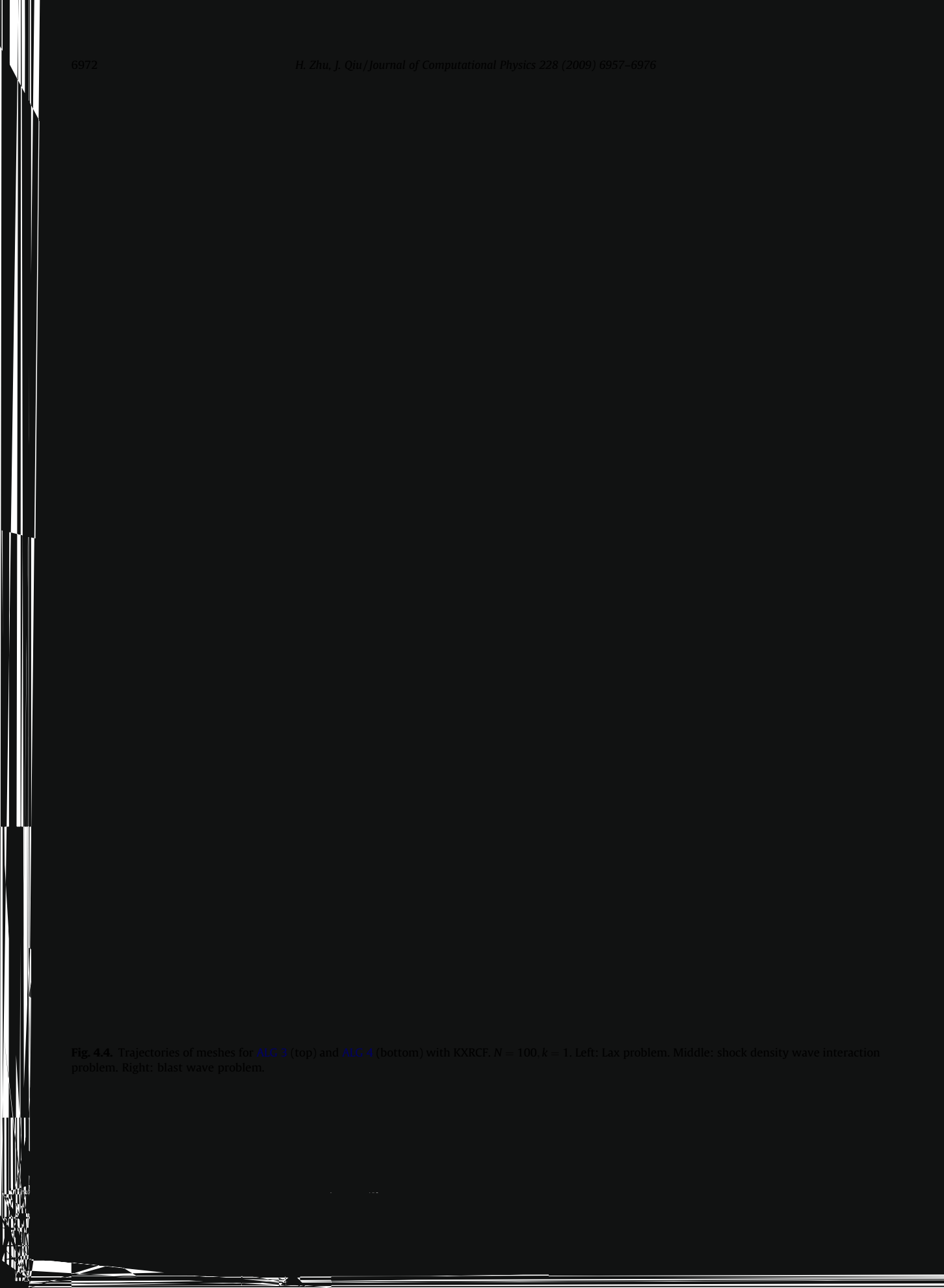

Fig. 4.4. Trajectories of meshes for ALG 3 (top) and ALG 4 (bottom) with KXRCF. $N = 100$, $k = 1$. Left: Lax problem. Middle: shock density wave interaction problem. Right: blast wave problem.

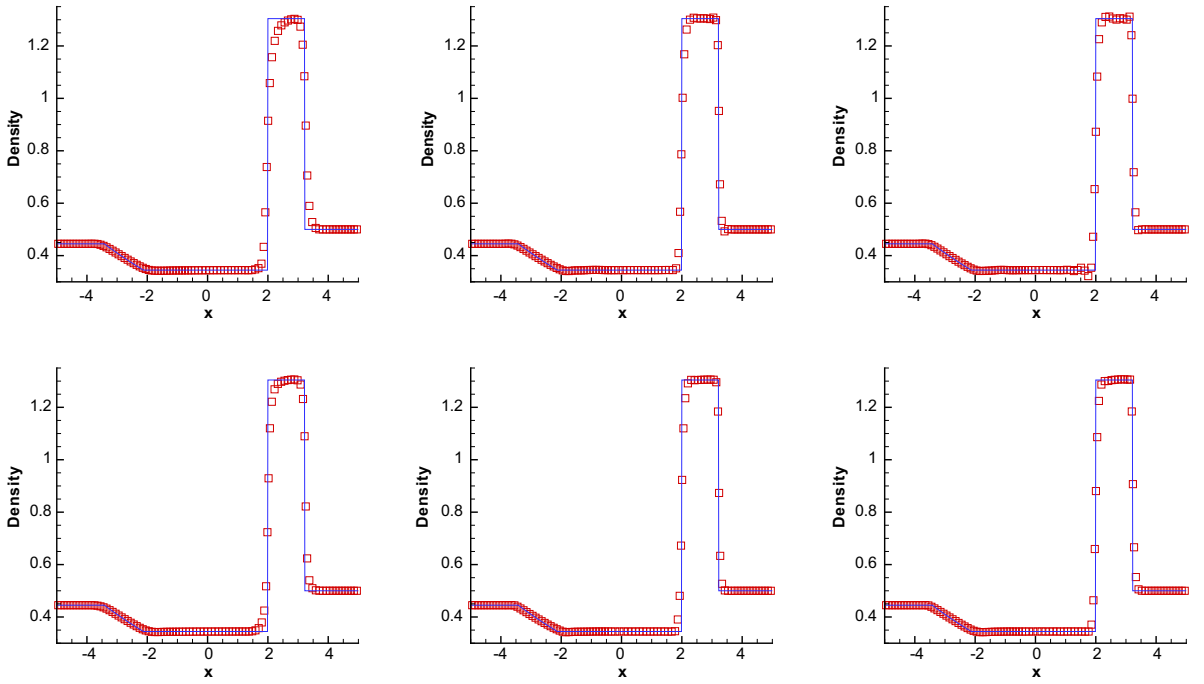

Fig. 4.5. The Lax problem, density; ALG 4 with TVB-1 (left), TVB-2 (middle) and KXRCF (right) indicators, $N = 100$; top: $k = 1$; bottom: $k = 2$.

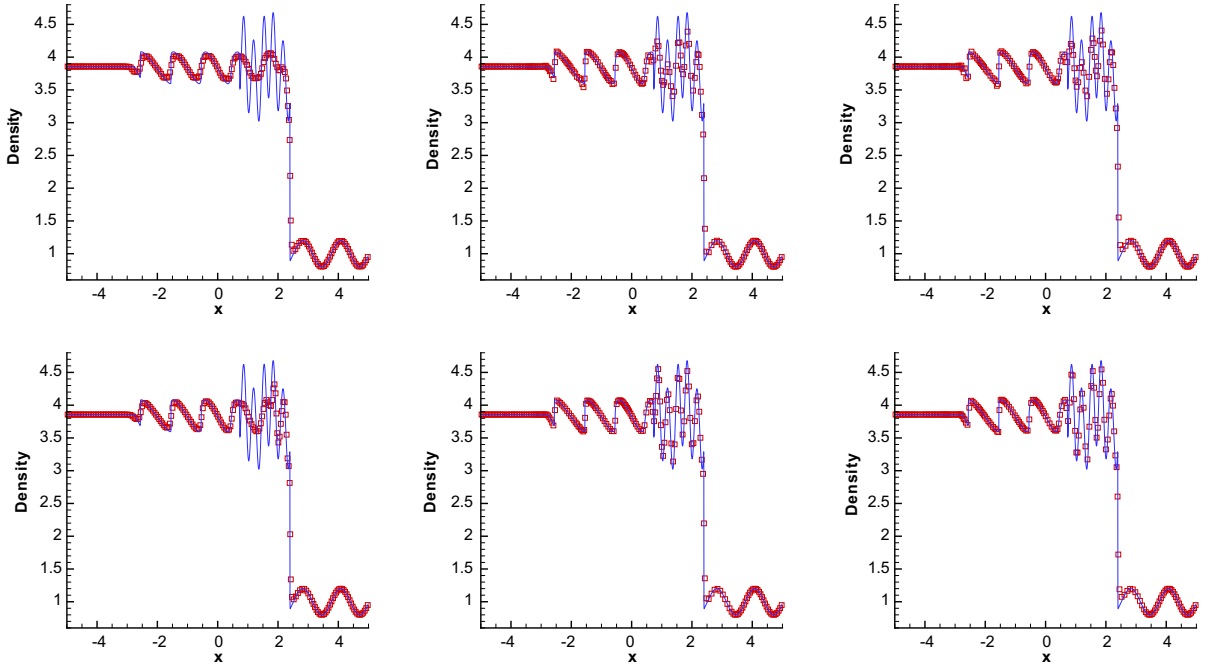

Fig. 4.6. The shock density wave interaction problem, density; ALG 4 with TVB-1 (left), TVB-2 (middle) and KXRCF (right) indicators, $N = 200$; top: $k = 1$; bottom: $k = 2$.

- Step 2. Suppose we have known the mesh $\{I_i^n\}$ and degrees of freedom $\{u_i^{(l)}(t_n)\}$ at the time level t_n. Move grid $\{I_i^{n,j-1}\}$ to $\{I_i^{n,j}\}$ based on the scheme (4.4) for $j = 1, \ldots, J$ with $\{I_i^{n,0}\} = \{I_i^n\}$. The degrees of freedom on the mesh $\{I_i^{n,j}\}$, denoted by $\{u_i^{(l)}(t_{n,j})\}$, will be determined by L_2 projection from $\{u_i^{(l)}(t_n)\}$.

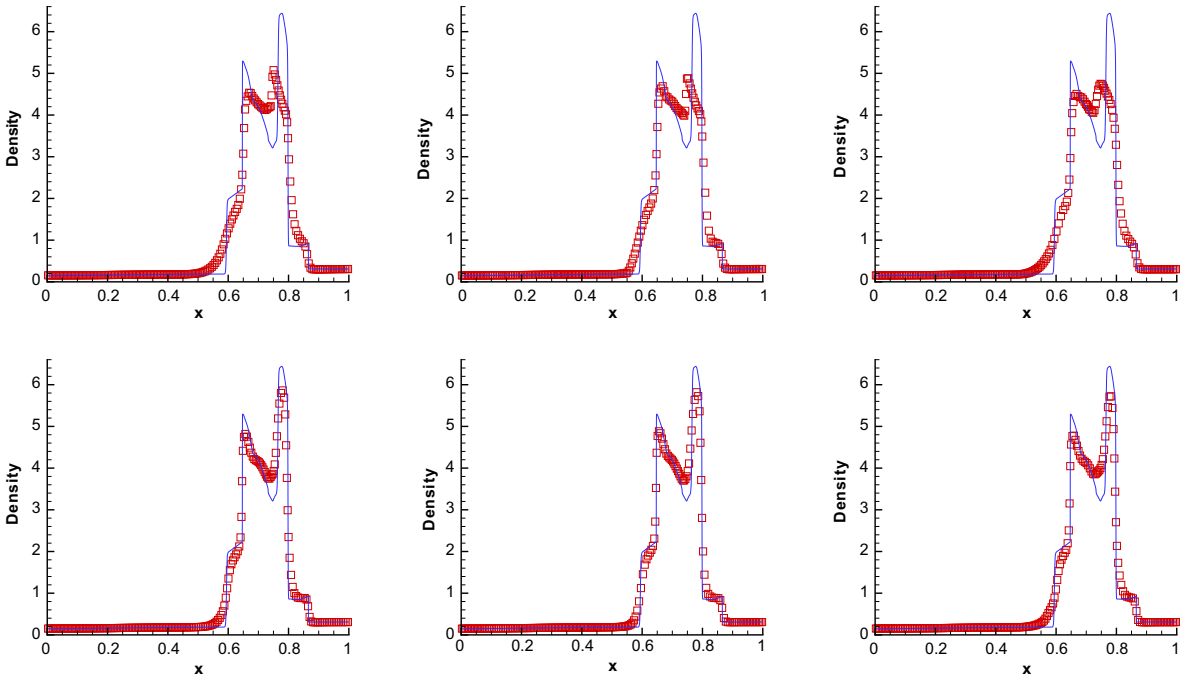

Fig. 4.7. The shock density wave interaction problem, density; ALG 4 with TVB-1 (left), TVB-2 (middle) and KXRCF (right) indicators, $N = 200$; top: $k = 1$; bottom: $k = 2$.

- Step 3. Evolve the solution of the underlying PDEs using the RKDG method on the mesh $\{I_i^{nJ}\}$ and obtain the degrees of freedom $\{u_i^{(l)}(t_{n+1})\}$.
- Step 4. If $t_{n+1} < T$, let $\{I_i^{n+1}\} = \{I_i^{nJ}\}$ and go to Step 2.

According to [33], we choose the iteration times $J = 6$ in Step 2. In Step 3, the limiter procedure is the same as in ALG 1.

For most problems, we can find from the trajectory of the mesh generated by the *r*-method that most of the grid points move slightly. Only the cells that are near the large solution variations move rapidly. It seems that we can make the mesh motion local to decrease the computation. Because the objective of the mesh motion is to 'cluster grid points in those regions where the solution has the largest gradients', and the troubled-cell indicators can show us where these regions are, we can apply the mesh motion to and near the troubled cells. But this trial does not work well in the numerical experiments. A few cells may get too large. So we tried some modifications and found that if we let the large cells also do the mesh motion, the problem will be solved. We now describe our *r*-method using troubled-cell indicators.

Algorithm 4 (*r-method using troubled-cell indicators).* The difference between ALG 3 and 4 is only in Step 2. For ALG 4 we switch Step 2 to:

- Step 2′. We first identify the troubled cells by troubled-cell indictors, and also identify the 'big' cells, whose length is larger than $1 + \theta$ times of the average cell length. Then we mark both the troubled cells and the 'big' cells together with their m left neighbor cells and m right neighbor cells, respectively. All of these marked cells are grouped into several sets with continuously marked cells, and the sets of marked cells are separated from each other. At last, we apply the mesh motion procedure like Step 2 of ALG 3 to each of these sets, respectively, to obtain the new mesh.

We choose $\theta = 0.5$ and $m = 3$ in the computation in this paper.

In what follows we will do some numerical tests for ALG 3 and 4. We will use the same test problems as in Section 3. The monitor function will be chosen to be $\omega(u_i) = \omega([\rho, \rho v, E]_i) = \sqrt{1 + \beta(\rho_i^{(1)})^2}$, where $\rho_i^{(1)}$ is the coefficient in Eq. (2.2), $\beta = 200$ for Lax problem, $\beta = 50$ for the shock density wave interaction problem and $\beta = 10$ for the blast wave problem. This choice of monitor function is not the best (see [31]), but is simple and easy to compute, and is enough for our tests. The value of M in the TVB-2 indicator is 10 for the Lax problem, and 100 for the other two test problems.

We will deal with the three test problems together for convenience. Before the tests, we must emphasize that our motivation of the new *r*-method using troubled-cell indicators is to improve the efficiency, and it is easy to see from the two algorithms that ALG 4 cannot get better solutions than ALG 3. We expect that ALG 4 spends much less CPU time than the other, and the local mesh moving of it does not decrease the accuracy too much.

We remark here that like the performance in ALG 1, BDF, BSB and MP troubled-cell indicators in ALG 4 always identify too much cells (which means almost all the cells are moving). And the solution with MMP indicator often has overshoots or oscillations (which cannot be reflected obviously in L_1 error). So we conclude here that these four indicators are bad indicators for ALG 4 and we will not show the results with them. For the other indicators we will compare the two algorithms and try to find the best one for our new algorithm.

We first compare the CPU time and the accuracy. Tables 4.1, 4.2 and 4.3 show the L_1 error for density of ALG 4 (err_4), percentage of increased error $IE = (err_4 - err_3)/err_3 \times 100\%$, CPU time of ALG 4 ($cpu_4$), percentage of saved CPU $SC = (cpu_3 - cpu_4)/cpu_3 \times 100\%$, and percentage of moving cells for ALG 4 $MC = (\sum_{q=0}^{TOT} mc_q)/N/TOT \times 100\%$, where TOT is the total number of time levels, and mc_q is the number of moving cells at the qth time level.

The data shows that in most of the cases, more than 50% of the CPU time is saved by ALG 4 while the increased error is below 10%. In the worst case, that is for the shock density wave interaction problem with TVB-2 and KXRCF indicators, the percentage of saved CPU is more than two times larger than the percentage of increased error. CPU-L_1-error figures are more convincing for the efficiency comparison. We plot the results with $N = 100 \times 2^n (n = 0, \ldots, 4)$ in Fig. 4.1. We can see clearly that for each troubled-cell indicator the solid line is below the dashed line which demonstrates the higher efficiency of ALG 4.

Now we will try to find the best indicator for ALG 4. Again we use the CPU-L_1-error figures. Fig. 4.2 tells us that TVB-2 and KXRCF are the best two indicators overall.

At last we show some resulting meshes and solutions. Figs. 4.3 and 4.4 show the typical mesh trajectories with TVB-2 and KXRCF indicators for the two algorithms. We can clearly see from the figures that there are much fewer moving cells in ALG 4 than in ALG 3. As we expected, we make the mesh moving local. The solutions of density with TVB-1, TVB-2 and KXRCF indicators are shown in Figs. 4.5, 4.6 and 4.7.

Now we can conclude that: (1) BDF, BSB, MP and MMP are not proper troubled-cell indicators for ALG 4; (2) With TVB-1, TVB-2 and KXRCF indicators ALG 4 is more efficient than ALG 3. (3) TVB-2 and KXRCF are the best indicators for ALG 4.

5. Concluding remarks

In this paper, we have systematically studied and compared adaptive RKDG methods using a few different troubled-cell indicators for one-dimensional conservation laws. The new r-version adaptive RKDG with local moving mesh instead of global moving mesh was presented. Extensive simulations on the hyperbolic systems of Euler equations indicate the indicator KXRCF by Krivodonova et al. [21] is the best indicator for h-method, and both the minmod-based TVB indicator (when the TVB constant M is suitably chosen) and the KXRCF indicator are better than other choices in all the test cases.

This technique of troubled-cell indicator can also be applied to p-adaptive DG methods for hyperbolic conservation laws. The idea is straight. One can use high order polynomials to approximate the solution for untroubled cells and use low order (e.g. constant) polynomials for troubled cells. Combining this with the h-method properly may lead to a more effective hp-method. The research on these problems and extending the adaptive methods to multi-dimensions are going on.

Acknowledgments

The research of the first author was partially supported by Oversea Education Plan of the China Scholarship Council when he was a joint trained student in Division of Applied Mathematics, Brown University. We would like to thank Professor Chi-Wang Shu in Brown University for his helpful suggestions, and we also wish to thank the unnamed reviewers for their careful review works and helpful suggestions which led to an improved paper.

References

[1] S. Adjerid, K.D. Devine, J.E. Flaherty, L. Krivodonova, A posteriori error estimation for discontinuous Galerkin solutions of hyperbolic problems, Comput. Methods Appl. Mech. Eng. 191 (2002) 1097–1112.
[2] Kim S. Bey, J. Tinsley Oden, hp-version discontinuous Galerkin methods for hyperbolic conservation laws, Comput. Methods Appl. Mech. Eng. 133 (1996) 259–286.
[3] R. Biswas, K. Devine, J. Flaherty, Parallel, adaptive finite element methods for conservation laws, Appl. Numer. Math. 14 (1994) 255–283.
[4] A. Burbeau, P. Sagaut, C.H. Bruneau, A problem-independent limiter for high-order Runge–Kutta discontinuous Galerkin methods, J. Comput. Phys. 169 (2001) 111–150.
[5] B. Cockburn, S. Hou, C.-W. Shu, The Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case, Math. Comput. 54 (1990) 545–581.
[6] B. Cockburn, G. Karniadakis, C.-W. Shu (Eds.), Discontinuous Galerkin Methods: Theory, Computation and Applications, Lecture Notes in Computational Science and Engineering, vol. 11, Springer, Berlin, 2000.
[7] B. Cockburn, S.-Y. Lin, C.-W. Shu, TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws III: one dimensional systems, J. Comput. Phys. 84 (1989) 90–113.
[8] B. Cockburn, C.-W. Shu, TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws II: general framework, Math. Comput. 52 (1989) 411–435.
[9] B. Cockburn, C.-W. Shu, The Runge–Kutta local projection P1-discontinuous Galerkin finite element method for scalar conservation laws, Math. Model. Numer. Anal. (M^2AN) 25 (1991) 337–361.
[10] B. Cockburn, C.-W. Shu, The Runge–Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems, J. Comput. Phys. 141 (1998) 199–224.

[11] A. Dedner, C. Makridakis, M. Ohlberger, Error control for a class of Runge–Kutta discontinuous Galerkin methods for nonlinear conservation laws, SIAM J. Numer. Anal. 45 (2007) 514–538.
[12] K. Devine, J. Flaherty, Parallel adaptive hp-refinement techniques for conservation laws, Appl. Numer. Math. 20 (1996) 367–386.
[13] Y.N. Di, R. Li, T. Tang, P.W. Zhang, Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws, SIAM J. Sci. Comput. 26 (2005) 1036–1056.
[14] A. Harten, ENO schemes with subcell resolution, J. Comput. Phys. 83 (1989) 148–184.
[15] A. Harten, B. Engquist, S. Osher, S. Chakravathy, Uniformly high order accurate essentially non-oscillatory schemes III, J. Comput. Phys. 71 (1987) 231–303.
[16] R. Hartmann, P. Houston, Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations, J. Comput. Phys. 183 (2002) 508–532.
[17] P. Houston, B. Senior, E. Süli, Hp-discontinuous Galerkin finite element methods for hyperbolic problems: error analysis and adaptivity, Int. J. Numer. Methods Fluids 40 (2002) 153–169.
[18] G. Jiang, C.-W. Shu, Efficient implementation of weighted ENO schemes, J. Comput. Phys. 126 (1996) 202–228.
[19] C. Johnson, U. Navert, J. Pitkaranta, Finite element methods for linear hyperbolic problems, Comput. Methods Appl. Mech. Eng. 45 (1984) 285–312.
[20] C. Johnson, J. Pitkaranta, An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation, Math. Comput. 46 (1986) 1–26.
[21] L. Krivodonova, J. Xin, J.-F. Remacle, N. Chevaugeon, J. Flaherty, Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws, Appl. Numer. Math. 48 (2004) 323–338.
[22] P. Lesaint, P. Raviart, On a finite element method for solving the neutron transport equation, in: C. deBoor (Ed.), Mathematical Aspects of Finite Elements in Partial Differential Equations, Academic Press, New York, 1974, pp. 89–145.
[23] R. Li, T. Tang, P.W. Zhang, Moving mesh methods in multiple dimensions based on harmonic maps, J. Comput. Phys. 170 (2001) 562–588.
[24] J.T. Oden, Progress in adaptive methods in computational fluid dynamics, in: J.E. Flaherty et al. (Eds.), Adaptive Methods for Partial Differential Equations, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989, pp. 206–252.
[25] J. Qiu, C.-W. Shu, A comparison of troubled-cell indicators for Runge–Kutta discontinuous Galerkin mehtods using weighted essentially nonosillatory limiters, SIAM J. Sci. Comput. 27 (2005) 995–1013.
[26] J. Qiu, C.-W. Shu, Runge–Kutta discontinuous Galerkin method using WENO limiters, SIAM J. Sci. Comput. 26 (2005) 907–929.
[27] W.H. Reed, T.R. Hill, Triangular mesh methods for neutron transport equation, Technical Report la-ur-73-479, Los Alamos Scientific Laboratory, Los Alamos, NM, 1973.
[28] J.-F. Remacle, J. Flaherty, M. Shephard, An adaptive discontinuous Galerkin technique with an orthogonal basis applied to compressible flow problems, SIAM Rev. 45 (2003) 53–72.
[29] W.J. Rider, L.G. Margolin, Simple modifications of monotonicity-preserving limiters, J. Comput. Phys. 174 (2001) 473–488.
[30] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, J. Comput. Phys. 77 (1988) 439–471.
[31] J.M. Stockie, J.A. Mackenzie, R.D. Russell, A moving mesh method for one-dimensional hyperbolic conservation laws, SIAM J. Sci. Comput. 22 (2001) 1791–1813.
[32] A. Suresh, H.T. Huynh, Accurate monotonicity-preserving schemes with Runge–Kutta time stepping, J. Comput. Phys. 136 (1997) 83–99.
[33] H.Z. Tang, T. Tang, Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws, SIAM J. Numer. Anal. 41 (2003) 487–515.
[34] P. Woodward, P. Colella, The numerical simulation of two-dimensional fluid flow with strong shocks, J. Comput. Phys. 54 (1984) 115–117.