Journal of Computational Mathematics, Vol.24, No.2, 2006, 121–140.

HIGH RESOLUTION SCHEMES FOR CONSERVATION LAWS AND CONVECTION-DIFFUSION EQUATIONS WITH VARYING TIME AND SPACE GRIDS *1)

Hua-zhong Tang
(*LMAM, School of Mathematical Sciences, Peking University, Beijing 100871, China*)

Gerald Warnecke
(*Institut für Analysis und Numerik, Otto-von-Guericke Universität Magdeburg, 39106 Magdeburg, Germany*)

Abstract

This paper presents a class of high resolution local time step schemes for nonlinear hyperbolic conservation laws and the closely related convection–diffusion equations, by projecting the solution increments of the underlying partial differential equations (PDE) at each local time step. The main advantages are that they are of good consistency, and it is convenient to implement them. The schemes are L^∞ stable, satisfy a cell entropy inequality, and may be extended to the initial boundary value problem of general unsteady PDEs with higher–order spatial derivatives. The high resolution schemes are given by combining the reconstruction technique with a second order TVD Runge-Kutta scheme or a Lax-Wendroff type method, respectively.

The schemes are used to solve a linear convection–diffusion equation, the nonlinear inviscid Burgers' equation, the one– and two–dimensional compressible Euler equations, and the two–dimensional incompressible Navier–Stokes equations. The numerical results show that the schemes are of higher–order accuracy, and efficient in saving computational cost, especially, for the case of combining the present schemes with the adaptive mesh method [15]. The correct locations of the slow moving or stronger discontinuities are also obtained, although the schemes are slightly nonconservative.

Mathematics subject classification: 35L65, 65M06, 65M99, 76M12.
Key words: Hyperbolic conservation laws, Degenerate diffusion, High resolution scheme, Finite volume method, Local time discretization.

1. Introduction

This paper is aimed at the construction of numerical approximations for nonlinear hyperbolic conservation laws

$$\frac{\partial}{\partial t}u + \frac{\partial}{\partial x}f(u) = 0, \tag{1.1}$$

and the closely related convection–diffusion equations

$$\frac{\partial}{\partial t}u + \frac{\partial}{\partial x}f(u) = \frac{\partial}{\partial x}Q(u, u_x), \tag{1.2}$$

with given initial data $u(x,0) = u_0(x)$ and corresponding suitable boundary conditions. Here $u = u(x,t)$ is an m–vector of conserved quantities, $x \in \mathbb{R}^d$, $m, d \geq 1$, $f(u) = \big(f^1(u), \cdots, f^d(u)\big)$

* Received February 24, 2004; Final revised October 10, 2005.
1) This research was partially sponsored by the National Basic Research Program under the Grant 2005CB321703, National Natural Science Foundation of China (No. 10431050, 10576001), SRF for ROCS, SEM, the Alexander von Humboldt foundation, and the Deutsche Forschungsgemeinschaft (DFG Wa 633/10-3).

is a nonlinear convective flux vector, and $Q(u, u_x) = \big(Q^1(u, u_x), \cdots, Q^d(u, u_x)\big)$ is a dissipation flux vector satisfying the weak parabolicity condition $\nabla_v Q(u, v) \geq 0$, for all u and v in $\mathbb{R}^m$.

These equations are of great practical importance since they arise in fluid flows, for example, reactive flows, groundwater flows, non–Newtonian flows, traffic flows, two-phase flows in oil reservoirs etc. During the past few decades there has been an enormous amount of activity related to the construction of high resolution schemes for Eqs.(1.1) and (1.2), see for instance [4, 12] and references therein. Explicit high resolution methods have been proven to be very efficient in capturing moving discontinuities or fronts, such as shock waves etc. However, they need a small time step size satisfying a Courant-Friedrichs-Lewy (CFL) condition for Eq.(1.1) and a similar more restriction condition for Eq.(1.2), to guarantee stability. For an implicit scheme, the time step size is also often constrained by nonlinear convergence. The spatial step sizes and the "signal" speeds are the two main elements to limit a choice of the time step size. Hence, when solving numerically unsteady PDEs, it may occur that in some spatial regions there is the need for a smaller time step than in other regions. Typical examples are numerical simulations of viscous fluid flows on nonuniform meshes and other computations of solutions to PDEs on an adaptive mesh [3, 9, 15]. Due to the above reason, the large time step schemes [7, 18] and the local time step schemes [3, 5, 9] become attractive. The large time step schemes satisfy the CFL condition by automatically increasing the stencil with the size of the time step. They can give correctly the location of shocks with virtually no smearing, but they seem to be inconvenient in practical applications, especially in treating boundary conditions.

The local time step schemes are only restricted by a local stability condition rather than the traditional global stability condition dominated by the smallest cells. The schemes studied in [3, 5, 9] are conservative, but they suffer a loss of consistency near a time grid interface in terms of truncation errors, see Section 2. Recently, the local time step schemes are widely studied and extended to adaptive grid methods, see e.g. [8, 14, 16].

The discrete conservation of a numerical algorithm for (1.1) or (1.2) is important in order to keep the correct location of the discontinuities. Hou and LeFloch in [6] have shown that if a nonconservative scheme for (1.1) converges, it converges to a solution of $\partial_t u + \partial_x f(u) = \mu$, where μ is a Borel measure source term that is expected to be zero in the region where the solution u is smooth and concentrated where u is not smooth. Even so, nonconservative schemes are also valuable in some practical applications and have been implemented successfully, for example, in computations of compressible multi-fluids [2] and fluid flows on an overlapping grid [10]. Another kind of the nonconservative schemes are residual distribution methods for hyperbolic conservation laws [1].

The aim of this paper is to study high resolution local time discretization schemes for (1.1) and (1.2), by projecting the solution increments of the underlying PDEs at each local time step. Because of good consistency, they may be applied to solving the initial boundary value problem of general unsteady PDEs with higher–order spatial derivatives. Moreover, the schemes are L^∞ stable, and it is convenient to implement them. Although the schemes will loose locally the discrete conservation, correct shocks have been obtained numerically when computing the 1D scalar Burgers' equation and the cylindrical explosion problem of the Euler equations.

This paper is organized as follows. In Section 2, we first review and analyze the scheme of Osher and Sanders [9]. A simple projection of the solutions is used in their scheme, but it suffers a loss of local consistency with the governing equations. Motivated by their scheme, we present a class of high resolution local time step schemes for Eq. (1.1) by projecting directly the solution increments of the underlying PDEs at each local time step. They include consistent first order Euler type schemes, second–order Runge–Kutta type schemes, and second–order Lax–Wendroff type schemes with multi–time increments. The schemes suffer from a slight loss of conservativity at isolated time grid interfaces. It turns out though, that shock speeds are hardly perturbed, see our numerical computations in Section 4. Because of good consistency, we may extend them to solve unsteady PDEs with higher–order derivatives, for example, Eq. (1.2). In Section 3, our schemes will be extended to the initial boundary value problems of Eq. (1.2). In Section 4, the local time step schemes are used to solve a linear convection–diffusion equation, the Burgers' equation, the compressible Euler equations, and the incompressible Navier–Stokes equations. The purpose is to demonstrate the ability of obtaining a higher–order rate of convergence and saving computational costs of the present schemes. We conclude the paper with a few remarks in Section 5.

2. Numerical Schemes for Conservation Laws

For simplicity in this section we will mainly restrict our attention to one–dimensional scalar equations, i.e. $d = 1$ and $m = 1$. There does not exist a difficulty in extension of the schemes to the multidimensional case. In Section 4, we will use our schemes presented in this section to solve numerically the two dimensional compressible Euler equations and the incompressible Navier-Stokes equation.

Consider a set $\{x_j\}$ of the physical domain $\mathbb{R}$, where x_j denotes the coordinate of a discrete cell boundary, $j \in \mathbb{Z}$, and define the cell size $\Delta x_{j+\frac{1}{2}} = x_{j+1} - x_j$ and the cell $I_{j+\frac{1}{2}} = \{x | x_j < x < x_{j+1}\}$. The initial value function $u(x, 0) = u_0(x)$ will be approximated by the cell average over each cell $I_{j+\frac{1}{2}}$, i.e.

$$u_h(x,0) = \frac{1}{\Delta x_{j+\frac{1}{2}}} \int_{I_{j+\frac{1}{2}}} u_0(x)\ dx =: u^0_{j+\frac{1}{2}} \quad \text{for } x \in I_{j+\frac{1}{2}}. \tag{2.1}$$

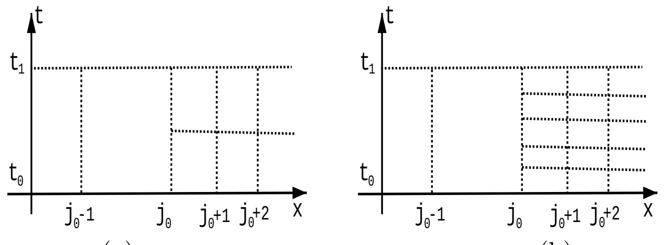

Figure 1: Non–uniform meshes in time and space.

2.1 Osher and Sanders' Scheme

In this subsection, we review the local time step scheme of Osher and Sanders presented in [9].

For convenience, we only consider two time increments here, $\Delta t_n = t_{n+1} - t_n$ and $\Delta t_n/2$, and assume that Δt_n is used for the cells with left boundary index in the set $\mathcal{D}_1 = \{j | j \le j_0 - 1\}$ and $\Delta t_n/2$ is for $\mathcal{D}_2 = \{j | j \ge j_0\}$, see Fig.1(a), $\Delta t_n = t_1 - t_0$ there. The Osher and Sanders scheme can be defined as

$$u^{n+\frac{1}{2}}_{j+\frac{1}{2}} = \begin{cases} u^n_{j+\frac{1}{2}}, & j \in \mathcal{D}_1, \\ u^n_{j+\frac{1}{2}} - \frac{1}{2}\lambda^n_{j+\frac{1}{2}} \Delta_+ h(u^n_{j-\frac{1}{2}}, u^n_{j+\frac{1}{2}}), & j \in \mathcal{D}_2, \end{cases} \tag{2.2}$$

$$u^{n+1}_{j+\frac{1}{2}} = u^n_{j+\frac{1}{2}} - \frac{1}{2}\lambda_{j+\frac{1}{2}} \Delta_+ \frac{1}{2} \left(h(u^n_{j-\frac{1}{2}}, u^n_{j+\frac{1}{2}}) + h(u^{n+1/2}_{j-\frac{1}{2}}, u^{n+\frac{1}{2}}_{j+\frac{1}{2}}) \right), \quad j \in \mathbb{Z}. \tag{2.3}$$

where $\lambda_{j+\frac{1}{2}} = \frac{\Delta t_n}{\Delta x_{j+\frac{1}{2}}}$, $u^n_{j+\frac{1}{2}}$ is a numerical cell average approximation of the solution $u(x, t)$ over the cell $I_{j+\frac{1}{2}}$ at $t = t_n$, Δ_+ denotes the forward difference operator, and $h(u^n_{j-\frac{1}{2}}, u^n_{j+\frac{1}{2}})$

is any numerical convective flux function of some three–point conservative scheme satisfying the consistancy condition: $h(u,u)=f(u)$. Examples of the numerical convective fluxes include the Godunov flux, the Lax–Friedrichs flux, and the Engquist–Osher flux etc. In our numerical computations, we will use the following numerical convective flux for a scalar equation on nonuniform meshes:

$$\begin{aligned} h(u_{j-\frac12},u_{j+\frac12}) =&\frac{\Delta_+x_j f(u_{j-\frac12})+\Delta_+x_{j-1}f(u_{j+\frac12})}{\Delta_+x_j+\Delta_+x_{j-1}} \\ &-\max_u\{|f'(u)|\}\max\{\Delta_+x_j,\Delta_+x_{j-1}\}\frac{u_{j+\frac12}-u_{j-\frac12}}{\Delta_+x_j+\Delta_+x_{j-1}}, \end{aligned} \tag{2.4}$$

which is monotone, that is to say, $h(u,v)$ is nondecreasing in the first variable and nonincreasing in the second variable.

For comparison in the following subsection, we rewrite the scheme (2.2)–(2.3) as:

$$(\delta u)^n_{j+\frac12} := -\lambda^n_{j+\frac12}\Delta_+ h(u^n_{j-\frac12},u^n_{j+\frac12}),\quad j\in\mathbb{Z}, \tag{2.5}$$

$$u^{n+\frac12}_{j+\frac12} = \begin{cases} u^n_{j+\frac12}, & j\in\mathcal{D}_1,\\ u^n_{j+\frac12}+\frac12(\delta u)^n_{j+\frac12}, & j\in\mathcal{D}_2,\end{cases} \tag{2.6}$$

$$(\delta u)^{n+\frac12}_{j+\frac12} := -\lambda^n_{j+\frac12}\Delta_+ h(u^{n+\frac12}_{j-\frac12},u^{n+\frac12}_{j+\frac12}),\quad j\in\mathbb{Z}, \tag{2.7}$$

$$u^{n+1}_{j+\frac12} = u^n_{j+\frac12}+\frac12\left((\delta u)^n_{j+\frac12}+(\delta u)^{n+\frac12}_{j+\frac12}\right),\quad j\in\mathbb{Z}. \tag{2.8}$$

The scheme (2.2)–(2.3) or (2.5)–(2.8) is constructed by using a simple projection of the solution at each local time step when the global time step size is available over any cell, see the first equation in (2.6). It is conservative in the global time step, because

$$\begin{aligned} u^{n+1}_{j_0-\frac12} =& u^n_{j_0-\frac12}-\frac12\lambda^n_{j_0-\frac12}\Big(h(u^n_{j_0-\frac12},u^n_{j_0+\frac12})-h(u^n_{j_0-\frac32},u^n_{j_0-\frac12}) \\ &+h(u^n_{j_0-\frac12},u^{n+\frac12}_{j_0+\frac12})-h(u^n_{j_0-\frac32},u^n_{j_0-\frac12})\Big),\\ u^{n+1}_{j_0+\frac12} =& u^n_{j_0+\frac12}-\frac12\lambda^n_{j_0+\frac12}\Big(h(u^n_{j_0+\frac12},u^n_{j_0+\frac32})-h(u^n_{j_0-\frac12},u^n_{j_0+\frac12}) \\ &+h(u^{n+\frac12}_{j_0+\frac12},u^{n+\frac12}_{j_0+\frac32})-h(u^n_{j_0-\frac12},u^{n+\frac12}_{j_0+\frac12})\Big). \end{aligned}$$

Osher and Sanders have proven convergence of corresponding approximate solutions to entropy solution of conservation laws (1.1). We refer the reader to [9] for details. However, the scheme (2.2)–(2.3) or (2.5)–(2.8) is non–conservative at each local time step, and the solutions at each local time step are not TVD, total variation diminishing. Moreover, the above scheme also suffers a loss of local consistency near a time grid interface in the sense of the truncation errors. To show this, we consider $u_t+au_x=0$ with a positive constant $a>0$ and a spatial mesh with a constant step size, i.e. $\Delta x_{j+\frac12}=\Delta x$. Then the scheme (2.2)–(2.3) at $x_{j_0+\frac12}$ becomes

$$u^{n+1}_{j_0+\frac12} = u^n_{j_0+\frac12}-\frac a2\lambda^n_{j+\frac12}(u^n_{j_0+\frac12}-u^n_{j_0-\frac12}+u^{n+\frac12}_{j_0+\frac12}-u^n_{j_0-\frac12}),$$

or

$$\frac{u^{n+1}_{j_0+\frac12}-u^n_{j_0+\frac12}}{\Delta t}+a\frac{\frac12(u^n_{j_0+\frac12}+u^{n+\frac12}_{j_0+\frac12})-u^n_{j_0-\frac12}}{\Delta x}=0.$$

If assuming that the exact solution $u(x,t)$ is smooth enough, then the following "modified" equation can be derived at $x_{j_0+\frac{1}{2}}$:

$$u_t + au_x = -\frac{a\Delta t}{4\Delta x}u_t + \frac{a\Delta x}{2}u_{xx} - \frac{\Delta t}{2}u_{tt} - \frac{a\Delta t^2}{8\Delta x}u_{tt} + \mathcal{O}(\Delta x^2, \Delta t^2, \Delta t^3/\Delta x).$$

Obviously, if $\frac{\Delta t}{\Delta x} \to c$, a finite constant, then the scheme is not locally consistent with the underlying PDEs, $u_t + au_x = 0$, even though $\Delta x \to 0$, and $\Delta t \to 0$.

Due to the local inconsistency, we may conclude that it is difficult to extend the scheme (2.2)–(2.3) to (1.2), by using a traditional way to discretize higher–order derivatives near a time grid interface. Also assuming that a uniformly spatial mesh with a constant space step size Δx is used, we discretize the term $(u_{xx})^{n+\frac{1}{2}}_{j_0+\frac{1}{2}}$ as

$$I := \frac{1}{\Delta x^2}(u^{n+\frac{1}{2}}_{j_0+\frac{3}{2}} - 2u^{n+\frac{1}{2}}_{j_0+\frac{1}{2}} + u^{n+\frac{1}{2}}_{j_0-\frac{1}{2}}).$$

Because $u^{n+\frac{1}{2}}_{j_0-\frac{1}{2}} = u^n_{j_0-\frac{1}{2}}$ as in (2.2), we have

$$I = \frac{1}{\Delta x^2}(u^{n+\frac{1}{2}}_{j_0+\frac{3}{2}} - 2u^{n+\frac{1}{2}}_{j_0+\frac{1}{2}} + u^n_{j_0-\frac{1}{2}}) = (u_{xx})^{n+\frac{1}{2}}_{j_0+\frac{1}{2}} - \frac{\Delta t}{2\Delta x^2}(u_t)^{n+\frac{1}{2}}_{j_0+\frac{1}{2}} + \cdots .$$

The truncation error is $\mathcal{O}(\Delta t/\Delta x^2)$, which is not ideal.

Combining the above predictor–corrector type local time step scheme with a reconstruction technique [17], Dawson and Kirby presented a high resolution local time scheme for conservation laws [5]. To our knowledge, it seems to be inconvenient to implement numerically their scheme, because the variables at different steps in the Runge–Kutta methods within $\mathcal{D}_1$ and $\mathcal{D}_2$ are interacting with each other, see Section 3.4 of their paper [5]. Moreover, their scheme will degenerate to first order accuracy in the sense of the truncation error, see also numerical results given in [5]. Besides the above analysis, another possibility to decrease accuracy of their scheme is the numerical approximation of the slope, which is similar to the previous analysis. For example, we use

$$S^{n+\frac{1}{2}}_{j_0+\frac{1}{2}} := \text{minmod}\left(\frac{\Delta_+ u^{n+\frac{1}{2}}_{j_0-\frac{1}{2}}}{\Delta_+ x_{j_0-\frac{1}{2}}}, \frac{\Delta_+ u^{n+\frac{1}{2}}_{j_0+\frac{1}{2}}}{\Delta_+ x_{j_0+\frac{1}{2}}}\right),$$

to approximate $(u_x)^{n+\frac{1}{2}}_{j_0+\frac{1}{2}}$, where the function minmod(a,b) is defined by

$$\text{minmod}(a,b) = \begin{cases} \text{sign}(a)\min\{|a|,|b|\}, & ab > 0, \\ 0, & ab \le 0. \end{cases} \tag{2.9}$$

Using the predictor step (2.2) and Taylor expansion, we have

$$\begin{aligned} S^{n+\frac{1}{2}}_{j_0+\frac{1}{2}} =& \text{minmod}\Big(\frac{u^{n+\frac{1}{2}}_{j_0+\frac{1}{2}} - u^n_{j_0-\frac{1}{2}}}{\Delta_+ x_{j_0-\frac{1}{2}}}, \frac{\Delta_+ u^{n+\frac{1}{2}}_{j_0+\frac{1}{2}}}{\Delta_+ x_{j_0+\frac{1}{2}}}\Big) = \text{minmod}\Big((u_x)^{n+\frac{1}{2}}_{j_0+\frac{1}{2}} + \mathcal{O}(\Delta_+ x_{j_0+\frac{1}{2}}), \\ & (u_x)^{n+\frac{1}{2}}_{j_0+\frac{1}{2}} + \frac{\Delta t_n}{2\Delta_+ x_{j_0-\frac{1}{2}}}(u_t)^{n+\frac{1}{2}}_{j_0+\frac{1}{2}} + \mathcal{O}(\Delta t_n^2/\Delta_+ x_{j_0-\frac{1}{2}}, \Delta_+ x_{j_0-\frac{1}{2}})\Big). \end{aligned}$$

It shows that $S^{n+\frac{1}{2}}_{j_0+\frac{1}{2}}$ is not a good approximation of $(u_x)^{n+\frac{1}{2}}_{j_0+\frac{1}{2}}$.

2.2 A New Local Time Step Scheme

2.1.1 First–order accurate time discretization

First, we consider the case of two time increments as in the last subsection.

A new local time step scheme can be given as follows:

$$(\delta u)^n_{j+\frac12} := -\lambda^n_{j+\frac12}\Delta_+ h(u^n_{j-\frac12}, u^n_{j+\frac12}), \quad j \in \mathbb{Z}, \tag{2.10}$$

$$u^{n+\frac12}_{j+\frac12} = u^n_{j+\frac12} + \frac12 (\delta u)^n_{j+\frac12}, \qquad j \in \mathbb{Z}, \tag{2.11}$$

$$(\delta u)^{n+\frac12}_{j+\frac12} := \begin{cases} (\delta u)^n_{j+\frac12}, & j \in \mathcal{D}_1, \\ -\lambda^n_{j+\frac12}\Delta_+ h(u^{n+\frac12}_{j-\frac12}, u^{n+\frac12}_{j+\frac12}), & j \in \mathcal{D}_2, \end{cases} \tag{2.12}$$

$$u^{n+1}_{j+\frac12} = u^{n+\frac12}_{j+\frac12} + \frac12 (\delta u)^{n+\frac12}_{j+\frac12}, \quad j \in \mathbb{Z}. \tag{2.13}$$

Equation (2.13) may also be rewritten as

$$u^{n+1}_{j+\frac12} = u^n_{j+\frac12} + \frac12 \left((\delta u)^n_{j+\frac12} + (\delta u)^{n+\frac12}_{j+\frac12} \right). \tag{2.14}$$

Although it has a same form as one in Eq.(2.8), two schemes are different. Actually, we have

$$u^{n+1}_{j_0-\frac12} = u^n_{j_0-\frac12} - \lambda^n_{j_0-\frac12}\left(h(u^n_{j_0-\frac12}, u^n_{j_0+\frac12}) - h(u^n_{j_0-\frac32}, u^n_{j_0-\frac12}) \right), \tag{2.15}$$

$$\begin{aligned} u^{n+1}_{j_0+\frac12} =& u^n_{j_0+\frac12} - \frac12 \lambda^n_{j_0+\frac12}\Big(h(u^n_{j_0+\frac12}, u^n_{j_0+\frac32}) - h(u^n_{j_0-\frac12}, u^n_{j_0+\frac12}) \\ &+ h(u^{n+\frac12}_{j_0+\frac12}, u^{n+\frac12}_{j_0+\frac32}) - h(u^{n+\frac12}_{j_0-\frac12}, u^{n+\frac12}_{j_0+\frac12}) \Big), \end{aligned} \tag{2.16}$$

for the scheme (2.10)–(2.13). It also shows the present scheme will be slightly nonconservative at points where the time step changes. But it has a good consistency, and may be implemented conveniently because a simple projection of the solution increments (δu) is used. Due to the good consistency, the present scheme can be extended to the PDEs with higher-order derivatives.

Moreover, we have.

Lemma 2.1. *Let $u_h(x,t)$ be defined by (2.1) and (2.10)–(2.13), the initial data be in the space $L^\infty \cap L^1 \cap BV$, and the flux $f(u)$ be locally Lipschitz continuous. If $h(u,v)$ is nondecreasing in the first variable and nonincreasing in the second variable, and a* **local CFL** *type of restriction is satisfied, that is*

$$\Lambda^n_{j+\frac12}\left(\frac{h(v_1,u) - h(v_2,u)}{v_1 - v_2} - \frac{h(w,v_1) - h(w,v_2)}{v_1 - v_2} \right) \le 1, \tag{2.17}$$

for all u, w, v_1 and v_2 between the values of u^n_{j+p} and $u^{n+\frac12}_{j+p}$ $(p = 0, \pm 1)$ and where $\Lambda_{j+\frac12}$ is defined by

$$\Lambda_{j+\frac12} = \begin{cases} \lambda^n_{j+\frac12}, & j \in \mathcal{D}_1, \\ \frac12 \lambda^n_{j+\frac12}, & j \in \mathcal{D}_2. \end{cases} \tag{2.18}$$

Then, we have

$$\min_j \{u^n_{j+\frac12}\} \le u^{n+\frac12}_{j+\frac12} \le \max_j \{u^n_{j+\frac12}\}, \tag{2.19}$$

$$\min_j \{u^n_{j+\frac12}\} \le u^{n+1}_{j+\frac12} \le \max_j \{u^n_{j+\frac12}\}, \tag{2.20}$$

and cell entropy inequalities hold

$$U(u^{n+\frac{1}{2}}_{j+\frac{1}{2}}) \le U(u^{n}_{j+\frac{1}{2}}) - \frac{1}{2}\lambda^n_{j+\frac{1}{2}}\big(H(u^n_{j+\frac{1}{2}}, u^n_{j+\frac{3}{2}}) - H(u^n_{j-\frac{1}{2}}, u^n_{j+\frac{1}{2}})\big), \tag{2.21}$$

$$U(u^{n+1}_{j+\frac{1}{2}}) \le U(u^{n+\frac{1}{2}}_{j+\frac{1}{2}}) - \frac{1}{2}\lambda^n_{j+\frac{1}{2}}\big(H(u^{n+\frac{1}{2}}_{j+\frac{1}{2}}, u^{n+\frac{1}{2}}_{j+\frac{3}{2}}) - H(u^{n+\frac{1}{2}}_{j-\frac{1}{2}}, u^{n+\frac{1}{2}}_{j+\frac{1}{2}})\big), \tag{2.22}$$

where $H(u^{n+\frac{1}{2}}_{j+\frac{1}{2}}, u^{n+\frac{1}{2}}_{j+\frac{3}{2}})$ is numerical entropy flux satisfying $H(u,u) = F(u)$, and (U,F) is any convex entropy pair of (1.1), $F'(u) = U'(u)f'(u)$.

The proof of the above lemma is not difficult, and will be omitted here. But we cannot derive a global numerical entropy condition in the sense of distributions and convergence analysis of the present scheme now.

Next, we extend the scheme (2.10)–(2.13) to a more general case with multi–time increments: Δt_n, $\alpha_1\Delta t_n, \cdots, \alpha_k\Delta t_n$, where $\sum_{l=1}^k \alpha_l = 1$, see Fig.1(b). Let us define $\beta_1 = \alpha_1$, $\beta_l = \sum_{i=1}^l \alpha_i$, $l = 2, \cdots, k$.

The algorithm can be described as follows (We name it *Algorithm I*):

Step 1: Compute increment $(\delta u)^n_{j+\frac{1}{2}}$ for all $j \in \mathbb{Z}$:

$$(\delta u)^n_{j+\frac{1}{2}} := -\lambda^n_{j+\frac{1}{2}}\Delta_+ h(u^n_{j-\frac{1}{2}}, u^n_{j+\frac{1}{2}}), \tag{2.23}$$

Step 2: Update solution at $t = t_n + \beta_1\Delta t_n$:

$$u^{n+\beta_1}_{j+\frac{1}{2}} = u^n_{j+\frac{1}{2}} + \alpha_1(\delta u)^n_{j+\frac{1}{2}}, \qquad j \in \mathbb{Z}, \tag{2.24}$$

Step 3: For $l = 2, \cdots, k$, do the following:

(a). Project increment $(\delta u)^{n+\beta_{l-1}}_{j+\frac{1}{2}}$ at local time step:

$$(\delta u)^{n+\beta_{l-1}}_{j+\frac{1}{2}} = \begin{cases} (\delta u)^n_{j+\frac{1}{2}}, & j \in \mathcal{D}_1, \\ -\lambda^n_{j+\frac{1}{2}}\Delta_+ h(u^{n+\beta_{l-1}}_{j-\frac{1}{2}}, u^{n+\beta_{l-1}}_{j+\frac{1}{2}}), & j \in \mathcal{D}_2, \end{cases} \tag{2.25}$$

(b). Update solution at $t = t_n + \beta_l\Delta t_n$ for all j:

$$u^{n+\beta_l}_{j+\frac{1}{2}} = u^{n+\beta_{l-1}}_{j+\frac{1}{2}} + \alpha_l(\delta u)^{n+\beta_{l-1}}_{j+\frac{1}{2}}. \tag{2.26}$$

The scheme (2.23)–(2.26) has the same properties as the scheme (2.10)–(2.13).

2.2.2 Higher–order accurate spatial discretization

In this subsection, we simply discuss the reconstruction technique [17] in order to improve the spatial accuracy of the previous schemes. For each local time step, we use the following piecewise linear function:

$$w^{n+\beta_l}_h(x,t) = u^{n+\beta_l}_{j+\frac{1}{2}} + S^{n+\beta_l}_{j+\frac{1}{2}}(x - x_{j+\frac{1}{2}}) =: w^{n+\beta_l}_{j+\frac{1}{2}}(x),\ (x,t) \in I_{j+\frac{1}{2}} \times [t_{n+\beta_l}, t_{n+\beta_{l+1}}[, \tag{2.27}$$

to replace the original piecewise constant function. Here $S^{n+\beta_l}_{j+\frac{1}{2}}$ is a numerical slope approximating $(u_x)^{n+\beta_l}_{j+\frac{1}{2}}$, $l = 0, 1, \cdots, k-1$, and $\beta_0 = 0$. There are many manners to define $S^{n+\beta_l}_{j+\frac{1}{2}}$.

The commonly used formulae are

$$S_{j+\frac{1}{2}}^{n+\beta_l} = \text{minmod}\left(S_{j+\frac{1}{2}}^{n+\beta_l,L}, S_{j+\frac{1}{2}}^{n+\beta_l,R}\right), \tag{2.28}$$

$$S_{j+\frac{1}{2}}^{n+\beta_l} = \left(\text{sign}(S_{j+\frac{1}{2}}^{n+\beta_l,L}) + \text{sign}(S_{j+\frac{1}{2}}^{n+\beta_l,R})\right) \frac{|S_{j+\frac{1}{2}}^{n+\beta_l,L}| \cdot |S_{j+\frac{1}{2}}^{n+\beta_l,R}|}{|S_{j+\frac{1}{2}}^{n+\beta_l,L}| + |S_{j+\frac{1}{2}}^{n+\beta_l,R}| + \varepsilon}, \tag{2.29}$$

$$S_{j+\frac{1}{2}}^{n+\beta_l} = \frac{(|S_{j+\frac{1}{2}}^{n+\beta_l,R}|^2 + \varepsilon^R) S_{j+\frac{1}{2}}^{n+\beta_l,L} + (|S_{j+\frac{1}{2}}^{n+\beta_l,L}|^2 + \varepsilon^L) S_{j+\frac{1}{2}}^{n+\beta_l,R}}{|S_{j+\frac{1}{2}}^{n+\beta_l,L}|^2 + |S_{j+\frac{1}{2}}^{n+\beta_l,R}|^2 + \varepsilon^L + \varepsilon^R}, \tag{2.30}$$

which are the Minmod limiter, the van Leer limiter, and the van Albada limiter, respectively. Here

$$S_{j+\frac{1}{2}}^{n+\beta_l,R} = \frac{\Delta_+ u_{j+\frac{1}{2}}^{n+\beta_l}}{\Delta_+ x_{j+\frac{1}{2}}}, \quad S_{j+\frac{1}{2}}^{n+\beta_l,L} = \frac{\Delta_+ u_{j-\frac{1}{2}}^{n+\beta_l}}{\Delta_+ x_{j-\frac{1}{2}}},$$

and ε^L and ε^R are taken as $3\Delta x_{j-\frac{1}{2}}$ and $3\Delta x_{j+\frac{1}{2}}$ in our numerical computations, respectively.

Then, the above *Algorithm I* can be modified as follows (We call it *Algorithm II*):

Step 1: Compute increment $(\delta u)_{j+\frac{1}{2}}^n$ for all $j \in \mathbb{Z}$:

$$(\delta u)_{j+\frac{1}{2}}^n := -\lambda_{j+\frac{1}{2}}^n \Delta_+ h(u_{j,L}^n, u_{j,R}^n), \tag{2.31}$$

Step 2: Update solution at $t = t_n + \beta_1 \Delta t_n$:

$$u_{j+\frac{1}{2}}^{n+\beta_1} = u_{j+\frac{1}{2}}^n + \alpha_1 (\delta u)_{j+\frac{1}{2}}^n, \qquad j \in \mathbb{Z}, \tag{2.32}$$

Step 3: For $l = 2, \cdots, k$, do the following:

(a). Project increment $(\delta u)_{j+\frac{1}{2}}^{n+\beta_{l-1}}$ at $t = t_n + \beta_{l-1}\Delta t_n$:

$$(\delta u)_{j+\frac{1}{2}}^{n+\beta_{l-1}} = \begin{cases} (\delta u)_{j+\frac{1}{2}}^n, & j \in \mathcal{D}_1, \\ -\lambda_{j+\frac{1}{2}}^n \Delta_+ h(u_{j,L}^{n+\beta_{l-1}}, u_{j,R}^{n+\beta_{l-1}}), & j \in \mathcal{D}_2, \end{cases} \tag{2.33}$$

(b). Update solution at $t = t_n + \beta_l \Delta t_n$ for all j:

$$u_{j+\frac{1}{2}}^{n+\beta_l} = u_{j+\frac{1}{2}}^{n+\beta_{l-1}} + \alpha_l (\delta u)_{j+\frac{1}{2}}^{n+\beta_{l-1}}, \tag{2.34}$$

where $u_{j,L}^{n+\beta_l} = w_{j-\frac{1}{2}}^{n+\beta_l}(x_j)$, $u_{j,R}^{n+\beta_l} = w_{j+\frac{1}{2}}^{n+\beta_l}(x_j)$.

2.2.3 Higher–order accurate time discretization

To increase the accuracy of the time discretization, we can use Runge–Kutta methods or Lax–Wendroff type methods to replace the forward Euler time discretization used in the first order method.

A second–order explicit TVD Runge–Kutta method [13], Heun's method, may be implemented as follows (We name it *Algorithm III*):

Step 1: Compute increment $(\delta u)_{j+\frac{1}{2}}^n$ for all $j \in \mathbb{Z}$:

$$(\delta u)_{j+\frac{1}{2}}^n := -\lambda_{j+\frac{1}{2}}^n \Delta_+ h(u_{j,L}^n, u_{j,R}^n), \tag{2.35}$$

Step 2: Update solution at $t = t_n + \beta_1 \Delta t$ for all $j \in \mathbb{Z}$:

$$\begin{aligned} u_{j+\frac{1}{2}}^{n+\beta_1,*} =& u_{j+\frac{1}{2}}^n + \alpha_1 (\delta u)_{j+\frac{1}{2}}^n, & (2.36)\\ (\delta u)_{j+\frac{1}{2}}^{n+\beta_1,*} =& -\lambda_{j+\frac{1}{2}}^n \Delta_+ h(u_{j,L}^{n+\beta_1,*}, u_{j,R}^{n+\beta_1,*}), & (2.37)\\ u_{j+\frac{1}{2}}^{n+\beta_1} =& \frac{1}{2}\left(u_{j+\frac{1}{2}}^n + u_{j+\frac{1}{2}}^{n+\beta_1,*} + \alpha_1 (\delta u)_{j+\frac{1}{2}}^{n+\beta_1,*}\right). & (2.38) \end{aligned}$$

Step 3: For $l = 2, \cdots, k$, project increment $(\delta u)_{j+\frac{1}{2}}^{n+\beta_{l-1}}$ and update solution $u_{j+\frac{1}{2}}^{n+\beta_l}$ at $t = t_n + \beta_{l-1}\Delta t_n$:

$$(\delta u)_{j+\frac{1}{2}}^{n+\beta_{l-1}} = \begin{cases} (\delta u)_{j+\frac{1}{2}}^n, & j \in \mathcal{D}_1, \\ -\lambda_{j+\frac{1}{2}}^n \Delta_+ h(u_{j,L}^{n+\beta_{l-1}}, u_{j,R}^{n+\beta_{l-1}}), & j \in \mathcal{D}_2, \end{cases} \tag{2.39}$$

$$u_{j+\frac{1}{2}}^{n+\beta_l,*} = \begin{cases} u_{j+\frac{1}{2}}^n + \beta_l (\delta u)_{j+\frac{1}{2}}^{n+\beta_{l-1}}, & j \in \mathcal{D}_1, \\ u_{j+\frac{1}{2}}^{n+\beta_{l-1}} + \alpha_l (\delta u)_{j+\frac{1}{2}}^{n+\beta_{l-1}}, & j \in \mathcal{D}_2, \end{cases} \tag{2.40}$$

$$(\delta u)_{j+\frac{1}{2}}^{n+\beta_l,*} = -\lambda_{j+\frac{1}{2}}^n \Delta_+ h(u_{j,L}^{n+\beta_l,*}, u_{j,R}^{n+\beta_l,*}), \tag{2.41}$$

$$u_{j+\frac{1}{2}}^{n+\beta_l} = \begin{cases} \frac{1}{2} u_{j+\frac{1}{2}}^n + \frac{1}{2}\left(u_{j+\frac{1}{2}}^{n+\beta_l,*} + \beta_l (\delta u)_{j+\frac{1}{2}}^{n+\beta_l,*}\right), & j \in \mathcal{D}_1, \\ \frac{1}{2} u_{j+\frac{1}{2}}^{n+\beta_{l-1}} + \frac{1}{2}\left(u_{j+\frac{1}{2}}^{n+\beta_l,*} + \alpha_l (\delta u)_{j+\frac{1}{2}}^{n+\beta_l,*}\right), & j \in \mathcal{D}_2. \end{cases} \tag{2.42}$$

The accuracy of *Algorithm III* in time can be verified by using the Taylor series expansion at x_{j_0}. To save space, we omit it.

We can also consider a higher–order Lax–Wendroff type time discretization for hyperbolic conservation laws (1.1). Using Taylor expansion, we have

$$u_{j+\frac{1}{2}}^{n+\beta_{l+1}} = u_{j+\frac{1}{2}}^{n+\beta_l} + \alpha_l \Delta t_n (u_t)_{j+\frac{1}{2}}^{n+\beta_l} + \frac{1}{2}(\alpha_l \Delta t_n)^2 (u_{tt})_{j+\frac{1}{2}}^{n+\beta_l} + \mathcal{O}(\Delta t_n^3), \tag{2.43}$$

where

$$(u_t)_{j+\frac{1}{2}}^{n+\beta_l} = -\big(f(u)_x\big)_{j+\frac{1}{2}}^{n+\beta_l},$$

and

$$(u_{tt})_{j+\frac{1}{2}}^{n+\beta_l} = \Big(\frac{\partial}{\partial x}\big(a^2(u) u_x\big)\Big)_{j+\frac{1}{2}}^{n+\beta_l}, \quad a(u) = f'(u).$$

The term $a^2(u)u_x$ in the above equation is a special form of a diffusion flux $Q(u, u_x)$. Thus it may be discretized in a same way described in Section 3. In the following, we use g_j to denote its approximation at the spatial grid interface x_j.

Now we have the following algorithm (We name it *Algorithm IV*):

Step 1: Compute increments $(\delta_t u)_{j+\frac{1}{2}}^n$ and $(\delta_{tt} u)_{j+\frac{1}{2}}^n$ for all $j \in \mathbb{Z}$:

$$\begin{cases} (\delta_t u)_{j+\frac{1}{2}}^n := -\lambda_{j+\frac{1}{2}}^n \Delta_+ h(u_{j,L}^n, u_{j,R}^n), \\ (\delta_{tt} u)_{j+\frac{1}{2}}^n := \frac{\Delta t_n}{2} \lambda_{j+\frac{1}{2}}^n \Delta_+ g_j^n, \end{cases} \tag{2.44}$$

Step 2: Update solution at $t = t_n + \beta_1 \Delta t_n$:

$$u_{j+\frac{1}{2}}^{n+\beta_1} = u_{j+\frac{1}{2}}^n + \alpha_1 (\delta_t u)_{j+\frac{1}{2}}^n + (\alpha_1)^2 (\delta_{tt} u)_{j+\frac{1}{2}}^n, \qquad j \in \mathbb{Z}, \tag{2.45}$$

Step 3: For $l = 2, \cdots, k$, do the following:

(a). Project increments $(\delta_t u)_{j+\frac{1}{2}}^{n+\beta_{l-1}}$ and $(\delta_{tt} u)_{j+\frac{1}{2}}^{n+\beta_{l-1}}$ at $t = t_n + \beta_{l-1}\Delta t_n$:

$$(\delta_t u)_{j+\frac{1}{2}}^{n+\beta_{l-1}} = \begin{cases} (\delta_t u)_{j+\frac{1}{2}}^{n}, & j \in \mathcal{D}_1, \\ -\lambda_{j+\frac{1}{2}}^{n} \Delta_+ h(u_{j,L}^{n+\beta_{l-1}}, u_{j,R}^{n+\beta_{l-1}}), & j \in \mathcal{D}_2, \end{cases} \tag{2.46}$$

and

$$(\delta_{tt} u)_{j+\frac{1}{2}}^{n+\beta_{l-1}} = \begin{cases} (\delta_{tt} u)_{j+\frac{1}{2}}^{n}, & j \in \mathcal{D}_1, \\ \frac{\Delta t_n}{2} \lambda_{j+\frac{1}{2}}^{n} \Delta_+ g_j^{n+\beta_{l-1}}, & j \in \mathcal{D}_2, \end{cases} \tag{2.47}$$

(b). Update solution at $t = t_n + \beta_l \Delta t_n$ for all j:

$$u_{j+\frac{1}{2}}^{n+\beta_l} = \begin{cases} u_{j+\frac{1}{2}}^{n} + \beta_l (\delta_t u)_{j+\frac{1}{2}}^{n+\beta_{l-1}} + (\beta_l)^2 (\delta_{tt} u)_{j+\frac{1}{2}}^{n+\beta_{l-1}}, & j \in \mathcal{D}_1, \\ u_{j+\frac{1}{2}}^{n+\beta_{l-1}} + \alpha_l (\delta_t u)_{j+\frac{1}{2}}^{n+\beta_{l-1}} + (\alpha_l)^2 (\delta_{tt} u)_{j+\frac{1}{2}}^{n+\beta_{l-1}}, & j \in \mathcal{D}_2, \end{cases} \tag{2.48}$$

where $u_j^{n+\beta_l} = \frac{1}{2}(u_{j,L}^{n+\beta_l} + u_{j,R}^{n+\beta_l})$.

For the above Lax–Wendroff type algorithm, we should project two "increments" in the local time step.

3. Numerical Schemes for Convection–Diffusion Equations

In this section we discuss the application of the local time step schemes to convection–diffusion equations (1.2).

The local time step schemes *Algorithm I–III* presented in the last section can be applied to the initial boundary value problems of unsteady parabolic and convection–diffusion equations. We need only use corresponding spatial discretizations to replace the solution increment in the previous algorithms. The spatial discretization may be finite difference, or finite element, or spectral methods.

For a finite volume approximation of the convection–diffusion equation (1.2), the increment can be computed simply as

$$(\delta u)_{j+\frac{1}{2}}^{n+\beta_l} = -\lambda_{j+\frac{1}{2}}^{n} \Delta_+ h(u_{j,L}^{n+\beta_l}, u_{j,R}^{n+\beta_l}) + \lambda_{j+\frac{1}{2}}^{n} \Delta_+ g_j^{n+\beta_l}, \tag{3.1}$$

where $g_j^{n+\beta_l}$ is the numerical diffusive flux approximating $Q(u, u_x)$:

$$g_j^{n+\beta_l} = Q\left(\frac{1}{2}(u_{j,L}^{n+\beta_l} + u_{j,R}^{n+\beta_l}), \frac{\Delta_+ u_{j-\frac{1}{2}}^{n+\beta_l}}{\Delta_+ x_{j-\frac{1}{2}}}\right). \tag{3.2}$$

Similarly, we can use

$$g_j^{n+\beta_l} = a^2(u_j^{n+\beta_l}) \frac{\Delta_+ u_{j-\frac{1}{2}}^{n+\beta_l}}{\Delta_+ x_{j-\frac{1}{2}}}, \tag{3.3}$$

for *Algorithm IV*.

4. Numerical Experiments

Several examples will be considered in this section. All of them have been used by several authors to test various numerical schemes. In our computations, the CFL number is taken as

0.48 and 0.24 for the one- and two- dimensional cases, respectively, unless stated otherwise. The codes are run on a Pentium-III PC with 800 MHz under a Linux environment.

Example 4.1. Linear Convection–Diffusion Equation. We will provide the results for the following one–dimensional linear convection–diffusion equation

$$\begin{cases} u_t + cu_x = au_{xx}, & (x,t) \in (0,2\pi) \times (0,T), \\ u(x,0) = \sin(x), & x \in (0,2\pi), \\ u(x+2\pi,t) = u(x,t), & t \in (0,T), \end{cases} \tag{4.1}$$

where c and $a \geq 0$ are both constant. The exact solution is $u(x,t) = e^{-at}\sin(x-ct)$. We compute the solution up to $T = 2$ and consider three cases: (1) $a = 0, c = 1$, and (2) $a = 1, c = 1$.

In this example, we will use the numerical convective flux given in Eq.(2.4) and the van Albada limiter (2.30) approximating the slopes.

First, we assume that the spatial step size Δx is used within the global domain, $\Delta t/2$ is within $[\pi, 1.5\pi]$ and Δt is in other regions. Tables 1–2 show the L^∞–, L^1–, and L^2–errors and corresponding convergence orders obtained by the *Algorithm III* with (3.1), and Table 3 shows the corresponding results obtained by the *Algorithm IV*.

Table 1: Example 4.1: The errors and convergence order for the case (1) by the *Algorithm III*.

N	L^∞–error	order	L^1–error	order	L^2–error	order
20	2.17e-02	–	9.81e-02	–	4.15e-02	–
40	5.32e-03	2.03	2.41e-02	2.03	1.00e-02	2.05
80	1.32e-03	2.01	5.78e-03	2.06	2.41e-03	2.05
160	3.30e-04	2.00	1.39e-03	2.06	5.82e-04	2.05

Table 2: Example 4.1: The errors and convergence order for the case (2) by the *Algorithm III*.

N	L^∞–error	order	L^1–error	order	L^2–error	order
20	2.14e-03	–	8.47e-03	–	3.77e-03	–
40	6.06e-04	1.82	2.42e-03	1.81	1.07e-03	1.82
80	1.62e-04	1.90	6.45e-04	1.91	2.86e-04	1.90
160	4.17e-05	1.96	1.67e-04	1.95	7.40e-04	1.95

Table 3: Same as Table 1, except for *Algorithm IV*.

N	L^∞–error	order	L^1–error	order	L^2–error	order
20	2.11e-02	–	9.51e-02	–	4.02e-02	–
40	5.02e-03	2.07	2.35e-02	2.02	9.80e-03	2.04
80	1.24e-03	2.02	5.68e-03	2.05	2.36e-03	2.05
160	3.13e-04	1.99	1.37e-03	2.05	5.73e-04	2.04

Next, we assume that the half step sizes $\Delta x/2$ and $\Delta t/2$ are used within $[\pi, 1.5\pi]$, while the step sizes Δx and Δt are in other region. Tables 4–5 show the results obtained by the *Algorithm III* with (3.1), while the data in Table 6 are obtained by the *Algorithm IV*.

Table 4: Example 4.1: The errors and convergence order for the case (1) by the *Algorithm III*.

N	L^∞–error	order	L^1–error	order	L^2–error	order
25	2.44e-02	–	8.02e-02	–	3.57e-02	–
50	6.25e-03	1.96	1.94e-02	2.05	8.46e-03	2.08
100	1.59e-03	1.97	4.66e-03	2.06	2.01e-03	2.07
200	3.98e-04	2.00	1.12e-03	2.06	4.85e-04	2.05

Table 5: Example 4.1: The errors and convergence order for the case (2) by *Algorithm III*.

N	L^∞–error	order	L^1–error	order	L^2–error	order
25	1.91e-03	–	7.21e-03	–	3.18e-03	–
50	5.79e-04	1.72	2.20e-03	1.71	9.65e-04	1.72
100	1.58e-04	1.87	6.02e-04	1.87	2.63e-04	1.88
200	4.10e-05	1.95	1.57e-04	1.94	6.85e-05	1.94

From these results, we can observe that second–order rates of convergence can be obtained for the linear convection–diffusion equation. If using the van Leer's limiter (2.29) to replace the van Albada limiter (2.30), similar results may be obtained. However, the Minmod limiter (2.28) is more diffusive.

Example 4.2. The Burgers' equation. This example is to solve the in-viscid Burgers equation

$$u_t + \left(\frac{u^2}{2}\right)_x = 0, \qquad 0 \le x \le 2\pi, \tag{4.2}$$

subject to the 2π-periodic initial data $u(x,0) = 0.5 + \sin(x), \quad x \in [0, 2\pi)$.

The solution propagates to the right, steepening until the critical time $t_c = 1$, at which a shock forms. In this computation, we also use the numerical convection flux given in (2.4) and the van Albada limiter (2.30) approximating slopes.

First, we assume that the half step sizes $\Delta x/2$ and $\Delta t/2$ are used within $[\pi, 1.5\pi]$, while the global step sizes Δx and Δt in the other domain. Table 7 shows the L^∞–, L^1–, and L^2–errors and corresponding convergence orders obtained by the *Algorithm III*.

Second, we assume that the half step sizes $\Delta x/2$ and three local time steps $0.5\Delta t$, $0.2\Delta t$, and $0.3\Delta t$ are used within $[\pi, 1.5\pi]$, while the global step sizes Δx and Δt are used in the other regions.

Tables 8 and 9 show the results obtained by the *Algorithm III* and *Algorithm IV* with three time levels, respectively.

It is observed that a second-order rate of convergence can be obtained by our local time step schemes. Fig. 2 also shows the solutions at $t = 2$, when the shock is well developed. The moving shock has been captured and followed correctly.

To further validate the ability of the local time step schemes in capturing correctly the moving shock, we solve a Riemann problem for the Burgers equation (4.2) with the initial data

$$u_0(x) = \begin{cases} 1.2, & x < 0, \\ 0.4, & x > 0. \end{cases}$$

The computational domain is divided such that a small spatial step size $\Delta x = 0.1$ and k local time increments with small time step $\Delta t/k$ are used within 32 sub-intervals Ω_i, $i = 1, \cdots, 32$, while the large step sizes $\Delta x = 1$ and $\Delta t = \frac{CFL\Delta x}{1.2}$ are in other sub-domains. The Ω_i are irregularly chosen as follows: $\Omega_1 = [6.6, 7[$, $\Omega_2 = [12, 12.5[$, $\Omega_3 = [13.5, 13.8[$, $\Omega_4 = [15.8, 16[$,

Table 6: Same as Table 4, except for *Algorithm IV*.

N	L^∞–error	order	L^1–error	order	L^2–error	order
25	2.14e-02	–	8.17e-02	–	3.59e-02	–
50	5.70e-03	1.91	2.07e-02	1.98	8.92e-03	2.01
100	1.47e-03	1.96	5.09e-03	2.02	2.18e-03	2.03
200	3.77e-04	1.96	1.24e-03	2.04	5.35e-04	2.03

Table 7: Example 4.2: The errors and convergence order for solution at $t = 0.8$

N	L^∞–error	order	L^1–error	order	L^2–error	order
25	6.35e-02	–	3.33e-02	–	3.14e-02	–
50	1.84e-02	1.79	8.35e-03	2.00	7.87e-03	2.00
100	4.39e-03	2.07	2.19e-03	1.93	1.96e-03	2.01
200	1.13e-03	1.96	5.63e-04	1.96	4.96e-04	1.98

$\Omega_5 = [22, 22.4[$, $\Omega_6 = [27.4, 27.8[$, $\Omega_7 = [32.8, 33.[$, $\Omega_8 = [38, 38.5[$, $\Omega_9 = [41.5, 41.6[$, $\Omega_{10} = [43.6, 44.1[$, $\Omega_{11} = [46.1, 46.7[$, $\Omega_{12} = [48.7, 49.1[$, $\Omega_{13} = [54.1, 54.5[$, $\Omega_{14} = [60.5, 61[$, $\Omega_{15} = [67, 67.4[$, $\Omega_{16} = [70.4, 70.6[$, $\Omega_{17} = [72.6, 72.9[$, $\Omega_{18} = [76.9, 77.1[$, $\Omega_{19} = [81.1, 81.3[$, $\Omega_{20} = [85.3, 85.6[$, $\Omega_{21} = [86.6, 86.8[$, $\Omega_{22} = [88.8, 89.1[$, $\Omega_{23} = [92.1, 92.3[$, $\Omega_{24} = [96.3, 96.5[$, $\Omega_{25} = [99.5, 99.7[$, $\Omega_{26} = [102.7, 102.9[$, $\Omega_{27} = [103.9, 104.3[$, $\Omega_{28} = [105.3, 105.5[$, $\Omega_{29} = [108.5, 108.6[$, $\Omega_{30} = [110.6, 110.9[$, $\Omega_{31} = [122.9, 123.3[$, and $\Omega_{32} = [125.3, 125.6[$.

The numerical shock speeds are recorded in Fig. 3 from $t = 0$ to 150 for CFL=0.4 and k time increments: $k = 1$, 2, 5, 10, 15, 20, 25, 30, 40, 50, 70, 100, and 200, respectively, while the averaged shock speeds are listed in Table 10. The numerical shock speed is calculated here as

$$S^n = \frac{1}{(u_L - u_R)\Delta t} \sum_{j=-\infty}^{\infty} (u^n_{j+\frac{1}{2}} - u^{n-1}_{j+\frac{1}{2}})(x_{j+1} - x_j). \tag{4.3}$$

From these data we can observe that the numerical shock speeds are hardly perturbed when increasing the number of the time increments. A comparison of the computed solutions at $t = 150$ is given in Fig. 4 for this case. The profile of the solutions are almost same as one obtained by the global time step scheme.

In Fig. 5, we record the numerical shock speeds from $t = 0$ to 150 for CFL=0.9 and k time increments: $k = 1$, 5, 10, 15, 20, 25, 30, 40, 50, 70, 100, and 200, respectively. The averaged shock speeds are reported in Table 11. A comparison of the computed solutions at $t = 150$ is given Fig. 6 for this case. In the above the minmod limiter has been used. Although the perturbation appeared in the numerical shock speed is larger that the former, the averaged shock speed is still correct and the computed solutions are acceptable. These computed results show that the moving shock can be captured correctly by our local time step schemes, even though they are slightly nonconservative at points where the time step changes.

It is also interesting to solve a much stronger and faster moving shock problem of the Burgers' equation with the initial data by using the present local time stepsize schemes

$$u(x, 0) = 17.4 + 13.3 \sin(\pi x), \quad -1 \le x \le 1.$$

This problem is introduced in [11] to test ability of the multi-domain WENO finite difference schemes with interpolation. The resulting numerical shock location may lag behind the exact shock location due to conservative errors [11]. Fig.7 shows the numerical solutions at $t = 4$ and 24 obtained by our local time step schemes. The solid line is obtained by the MUSCL scheme on a finer uniform mesh with 2000 cells. Although there is numerical oscillation, the calculated

Table 8: Same as Table 7, except for second mesh-partition.

N	L^∞–error	order	L^1–error	order	L^2–error	order
25	9.60e-02	–	4.65e-02	–	4.64e-02	–
50	2.50e-02	1.94	1.27e-02	1.87	1.27e-02	1.87
100	5.58e-03	2.16	3.10e-03	2.03	2.94e-03	2.11
200	1.22e-03	2.19	7.27e-04	2.09	6.57e-04	2.16

Table 9: Same as Table 8, except for *Algorithm IV*.

N	L^∞–error	order	L^1–error	order	L^2–error	order
25	1.00e-01	–	4.79e-02	–	4.65e-02	–
50	2.60e-02	1.94	1.31e-02	1.87	1.27e-02	1.87
100	5.71e-03	2.19	3.24e-03	2.02	3.06e-03	2.05
200	1.28e-03	2.16	7.69e-04	2.07	7.04e-04	2.12

shock location is correct. In our computations, the half step sizes $\Delta x/2$ and $\Delta t/2$ are used within $[-0.5, 0.5]$, while the global step sizes Δx and Δt in the other domain.

Example 4.3. One–dimensional Euler equations. We solve the following system of the Euler equations

$$U_t + F(U)_x = 0, \tag{4.4}$$

where $U = (\rho, \rho u, E)^T$, and $F(U) = \big(\rho u, \rho u^2 + p, u(E+p)\big)^T$.

Here ρ is the density, u is the velocity, E is the total energy, p is the pressure. We assume that the pressure is related to the total energy by $E = \frac{p}{\gamma-1} + \frac{1}{2}\rho u^2$ with $\gamma = 1.4$. The initial data are set such that the exact solution is

$$\rho(x,t) = 1 + 0.2\sin\big(\pi(x - ut)\big),\ u(x,t) = 0.7,\ p(x,t) = 1,\ x \in [0,2].$$

The periodic boundary condition is specified. We compute the solution up to $t = 2$. The errors and numerical orders of accuracy of the density ρ for our high resolution local time step algorithm *Algorithm III* with kinetic flux–vector splitting schemes are shown in Table 12. We can see that a second-order rate of convergence can also be obtained. Here the half step sizes $\Delta x/2$ and $\Delta t/2$ have been used within $[0.7, 1.3]$, while the global step sizes Δx and Δt have been used in other domain.

Example 4.4. Two–dimensional Euler equations. We solve the following system of two–dimensional Euler equations

$$U_t + F(U)_x + G(U)_y = 0, \tag{4.5}$$

where $U = (\rho, \rho u, \rho v, E)^T$, $F(U) = \big(\rho u, \rho u^2 + p, \rho uv, u(E+p)\big)^T$, and $G(U) = \big(\rho v, \rho uv, \rho v^2 + p, v(E+p)\big)^T$.

Table 10: Example 4.2: The averaged shock speed for CFL=0.4.

k	Shock speed	k	Shock Speed	k	Shock speed
1	0.800000172	20	0.799664386	50	0.799648687
2	0.799993110	25	0.799659362	70	0.799646677
10	0.799688536	30	0.799655796	100	0.799643219
15	0.799672775	40	0.799651374	200	0.799640447

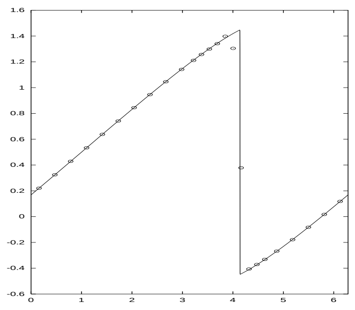

Figure 2: Numerical solution ("∘") and exact solution (solid line) of the periodic problem in Example 4.1 given at $t = 2$.

Table 11: Same as Table 10, except for CFL=0.9.

k	Shock speed	k	Shock Speed	k	Shock speed
1	0.800000174	50	0.799866552	200	0.799822289
2	0.799961043	100	0.799837461	500	0.799813063

Here ρ is the density, (u, v) is the velocity vector, E is the total energy, p is the pressure. We also assume that the pressure is related to the total energy by $E = \frac{p}{\gamma-1} + \frac{1}{2}\rho u^2 + \frac{1}{2}\rho v^2$ with $\gamma = 1.4$. The initial data are set such that the exact solution is

$$\rho(x, y, t) = 1 + 0.2 \sin\big(\pi(x + y - ut - vt)\big),$$
$$u(x, y, t) = 0.7,\ v(x, y, t) = 0.3,\ p(x, y, t) = 1.$$

The computational domain is taken as $[0, 2] \times [0, 2]$, and the periodic boundary conditions are specified in $x-$ and $y-$ directions. The algorithm is a multidimensional extension of the high resolution local time step scheme (2.35)–(2.42) with kinetic flux–vector splitting schemes. The computational domain is divided such that that the half step sizes $\Delta x/2$ and $\Delta y/2$ are used within $[0.7, 1.3]$, respectively, and $\Delta t/2$ is in $[0.7, 1.3] \times [0.7, 1.3]$, while the global step sizes Δx, Δy and Δt are in other domain. We compute the solution up to $t = 2$. The errors and numerical orders of accuracy of the density ρ are shown in Table 13. The results show that a second-order rate of convergence can also be obtained by our local time step scheme.

We also solve two-dimensional Euler equations with a slow moving shock in the positive

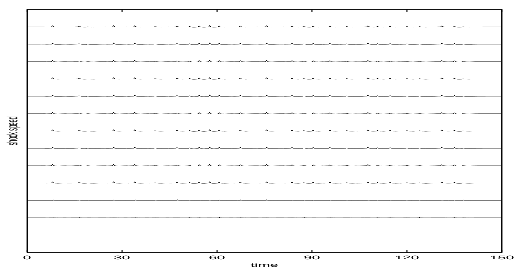

Figure 3: Numerical shock speeds are recorded for CFL=0.4 and k time increments: 1, 2, 5, 10, 15, 20, 25, 30, 40, 50, 70, 100, 200 (from bottom to top), respectively. CFL=0.4.

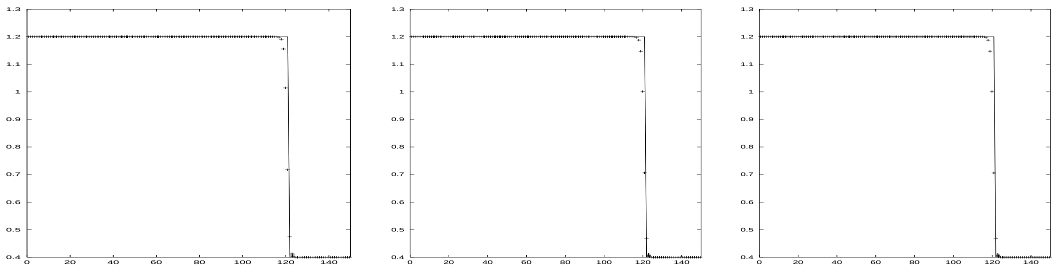

Figure 4: Comparisons of numerical solutions for CFL=0.4 and k time increments: k= 1, 70, and 200.

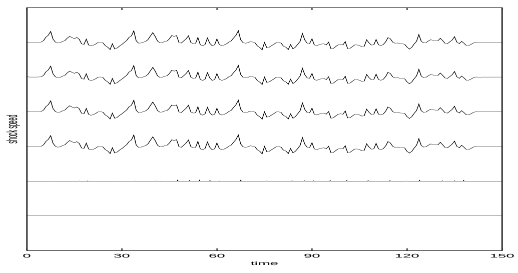

Figure 5: Numerical shock speeds are recorded for for CFL=0.9 and k time increments: k =1,2,50,100, 200, and 500 (from bottom to top), respectively.

y-direction. The initial data for the conservative variables

$$U(x,y,0) = \begin{cases} (1.4, 0, 4.2, 8.8)^T, & y < 0.65, \\ (5.38064, 0, 4.25971, 27.2576)^T, & y > 0.65, \end{cases} \tag{4.6}$$

are chosen such that the exact solution is a shock moving with speed $s = 0.015$ in the the positive y-direction, see [11]. The computational domain $[0, 2] \times [0, 2]$ is divided into 200×200 cells such that that the half step sizes $\Delta x/2$ and $\Delta y/2$ are used within $[0.7, 1.3]$, respectively, and $\Delta t/2$ is in $[0.7, 1.3] \times [0.7, 1.3]$, while the global step sizes Δx, Δy and Δt are in other domain. We compute the solution up to $t = 20$. The computed density is shown in Fig.8, compared with the exact solution (solid).

Example 4.5. Double Mach reflection for 2D Euler equations. In this example, we solve two-dimensional double Mach reflection problem for two-dimensional Euler equations by combining our moving mesh method developed in [15], in order to investigate efficiency of the local time step scheme when we combine it with the adaptive mesh methods. We refer the

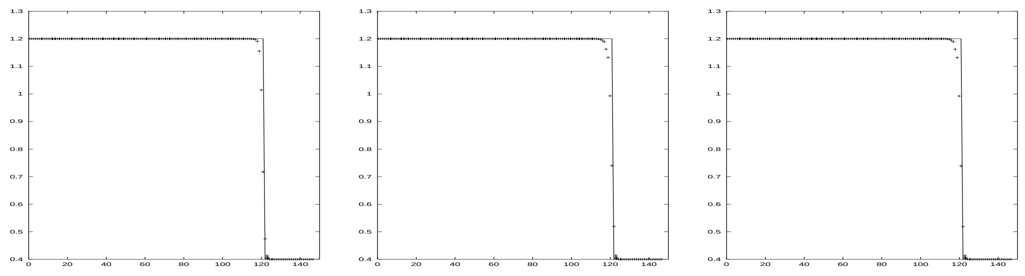

Figure 6: Comparison of numerical solutions for CFL=0.9 and k time increments: k = 1, 100, and 500, respectively.

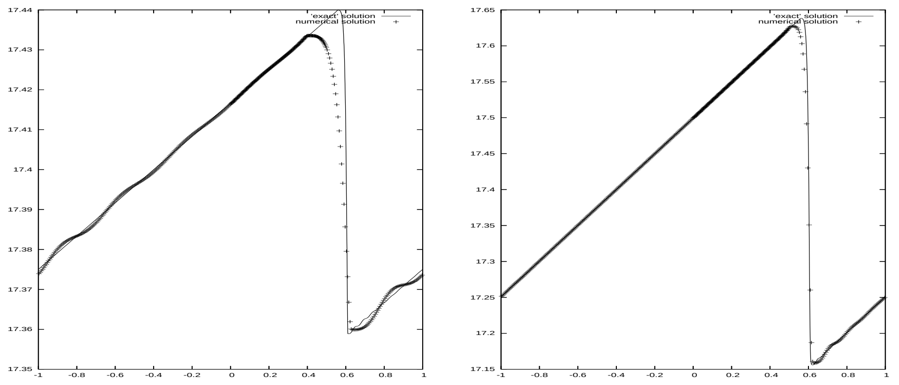

Figure 7: A much stronger and faster shock problem, see Case 3 of Example 3.1 in [11]. The computed solutions are obtained by solving inviscid Burgers' equation in 1D with 400 nonuniform cells. The CFL number is 0.48. Left: $t = 4$, right: $t = 24$.

Table 12: Example 4.3: The errors and convergence order for solutions at $t = 2$.

N	L^∞–error	order	L^1–error	order	L^2–error	order
26	8.26e-03	–	9.56e-03	–	7.28e-03	–
52	2.33e-03	1.83	2.43e-03	1.98	1.89e-03	1.95
104	5.93e-04	1.97	5.91e-04	2.04	4.65e-04	2.02
208	1.61e-04	1.88	1.43e-04	2.05	1.12e-04	2.05

readers to [15] for a detailed description of this problem as well as the adaptive mesh method. Here, the computational domain is divided into non-uniform 160×80 cells and the output time is taken as $t = 0.2$. Fig. 9 shows the adaptive mesh and the density contours with 24 equally spaced contour lines, calculated by using the local time step scheme and global time step scheme on the moving meshes, respectively. We see that there is not obvious difference between them. The CPU times are 4555.04 seconds for the local time step scheme and 9377.90 seconds for the global time step scheme estimated on the Pentium-IV with 3GHz under a Linux environment, respectively. It means that about 51.43% cost is saved in the present computation.

Example 4.6. Navier–Stokes equations. This example is to solve the Navier–Stokes equations:

$$\begin{cases} \omega_t + \nabla \cdot (\mathbf{u}\omega) = \frac{1}{Re}\Delta\omega, \quad \Delta\psi = \omega, \quad \mathbf{u} = \nabla^\perp\psi, \\ \mathbf{u} = \text{given on } \partial\Omega, \end{cases} \tag{4.7}$$

and check the accuracy of our local time step schemes, where $\nabla^\perp = (-\partial_y, \partial_x)$. The Reynolds number is taken as $Re = 100$ and the computaional domain is $[0, 2\pi] \times [0, 2\pi]$ with periodic boundary conditions, respectively. In this case, the algorithm used is an extension of the

Table 13: Example 4.4: The errors and convergence order for solutions at $t = 2$.

N×N	L^∞–error	order	L^1–error	order	L^2–error	order
26×26	1.76e-02	–	3.07e-02	–	1.72e-02	–
52×52	4.67e-03	1.91	7.41e-03	2.05	4.17e-03	2.04
104×104	1.26e-03	1.89	1.76e-03	2.07	9.95e-04	2.07
208×208	3.32e-04	1.92	4.14e-04	2.09	2.34e-04	2.09

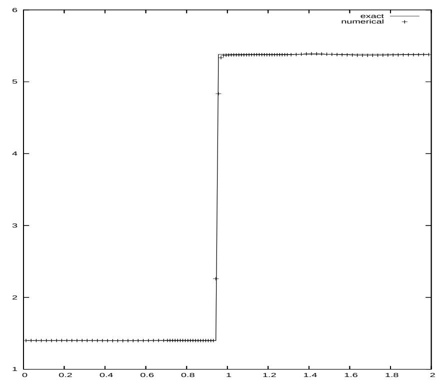

Figure 8: Slow moving shock problem, see Example 3.8 in [11]. The density $t = 20$ is obtained by solving Euler system of equations in 2D. The CFL number is 0.24. The solid line denotes the exact solution.

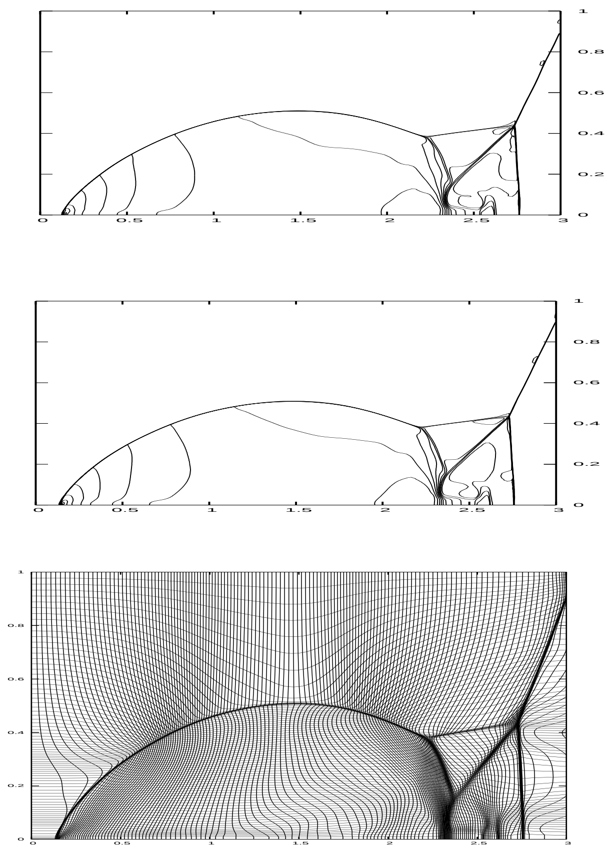

Figure 9: Example 4.5. The density contours and the adaptive mesh at $t = 0.2$. Top: the local time step scheme; middle: the global time step scheme; bottom: the adaptive mesh.

Algorithm III and (3.1). The discrete Possion equation for the stream function ψ is solved iteratively by a Jacobi type iteration.

The initial conditions are taken such that the exact solution of the problem is known

$$\omega(x,y,t) = -2\sin(x)\sin(y)e^{-\frac{2t}{Re}}, \quad \psi(x,y,t) = \sin(x)\sin(y)e^{-\frac{2t}{Re}},$$
$$u(x,y,t) = -\sin(x)\cos(y)e^{-\frac{2t}{Re}}, \quad v(x,y) = \cos(x)\sin(y)e^{-\frac{2t}{Re}}.$$

The computational domain is discretized such that the half step sizes $\Delta x/2$ and $\Delta y/2$ are within $[0.8, 1.2]$, respectively, and $\Delta t/2$ is in $[0.8, 1.2] \times [0.8, 1.2]$, while the global step sizes Δx, Δy and Δt are in other domain. Table 14 shows the errors and convergence orders for the vorticity function at $t = 2$.

Table 14: Example 4.6: The errors and convergence order for solutions at $t = 2$.

N×N	L^∞–error	order	L^1–error	order	L^2–error	order
24×24	2.60e-02	–	3.49e-01	–	7.13e-02	–
48×48	7.22e-03	1.85	6.89e-02	2.34	1.44e-02	2.31
96×96	1.75e-03	2.04	1.37e-02	2.33	2.94e-03	2.29
192×192	4.54e-04	1.95	3.17e-03	2.11	7.04e-04	2.06

5. Concluding Remarks

A class of the local time step schemes have been studied for nonlinear hyperbolic conservation laws (1.1) in this paper. The schemes are constructed by using a simple projection of the solution increments at each local time step. They are of good consistency and L^∞ stability, and satisfy the cell entropy inequalities, even though these properties still do not imply convergence of the schemes. High resolution schemes are derived by combining the initial reconstruction technique with second–order TVD Runge–Kutta methods or second order Lax–Wendroff type time discretization.

Because of the good consistency, our schemes have also been extended to the initial boundary value problems of unsteady parabolic and convection–diffusive equations, e.g. (1.2). We need only modify the corresponding spatial discretization operator in the right hand side term of the algorithms.

The methods have been used to solve numerically a linear convection–diffusion equation, the nonlinear inviscid Burgers' equation, the one– and two–dimensional compressible Euler equations, and the two–dimensional incompressible Navier–Stokes equations. From numerical results, we can observe that second–order rates of convergence have been obtained by the presented schemes; the local time step schemes can also save computational costs. Although the schemes are slightly nonconservative, the moving shocks can be captured correctly in our computations of two detailed examples.

It is possible to construct a conservative local time step scheme with a good consistency. As an example for the special case as shown in Fig.1a, we may apply a two-steps scheme for $j \in \mathcal{D}_1$, while a one-step scheme for $j \in \mathcal{D}_2$. However, it becomes expensive when the scheme is extended to the case with multi-time increments and higher-order accuracy.

It is an interesting topic to construct third– and higher–order accurate schemes with locally varying time and space grids.

Acknowledgments. The authors thank Professor Tao Tang for numerous discussions during the preparation of this work, and also thank the referees for many helpful suggestions.

References

[1] R. Abgrall and T. Barth, Residual distribution schemes for conservation laws via adaptive quadrature, *SIAM J. Sci. Comput.*, **24** (2002), 732-768.

[2] R. Abgrall and S. Karni, Computations of compressible multifluids, *J. Comput. Phys.*, **169** (2001), 594-623.

[3] M.J. Berger, On conservation at grid interfaces, *SIAM J. Numer. Anal.*, **24** (1987), 967-984.

[4] B. Cockburn, Discontinuous Galerkin methods for convection domainated problems. Available at <http://www.math.umn.edu/~cockburn>

[5] C. Dawson and R. Kirby, High resolution schemes for conservation laws with locally varying time steps, *SIAM J. Sci. Comput.*, **22** (2001), 2256–2281.

[6] T.Y. Hou and P.G. LeFloch, Why nonconservative schemes converge to wrong solutions: error analysis, *Math. Comput.*, **62** (1994), 497-530.

[7] R.J. LeVeque, Large time step shock-capturing techniques for scalar conservation laws, *SIAM J. Numer. Anal.*, **19** (1982), 1091–1109.

[8] S. Müller and Y. Stiriba, Fully adaptive multiscale schemes for conservation laws employing locally varying time stepping, Report No. 238, April 2004, IGPM, RWTH Aachen. Available at <http://www.igpm.rwth-aachen.de/~mueller/>

[9] S. Osher and R. Sanders, Numerical approximations to nonlinear conservation laws with locally varying time and space grids, *Math. Comput.*, **41** (1983), 321–336.

[10] E. Pärt-Enander and B. Sjögreen, Conservative and non-conservative interpolation between overlapping grids for finite volume solutions of hyperbolic problems, *Computers* & *Fluids*, **23** (1994), 551-574.

[11] K. Sebastian and C.-W. Shu, Multi domain WENO finite difference method with interpolation at sub-domain interfaces, *Journal of Scientific Computing*, **19** (2003), 405-438.

[12] C.-W. Shu, Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws, in *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*, B. Cockburn, C. Johnson, C.-W. Shu and E. Tadmor (Editor: A. Quarteroni), Lecture Notes in Mathematics, volume 1697, Springer, 1998, 325-432.

[13] C.-W. Shu and S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, *J. Comput. Phys.*, **77** (1988), 439–471.

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

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

[16] H.Z. Tang and G. Warnecke, A class of high resolution difference schemes for nonlinear Hamilton-Jacobi equations with varying time and space grids, *SIAM J. Sci. Comput.*, **26** (2005), 1415-1431

[17] B. van Leer, Towards the ultimate conservative difference scheme, V. A second order sequel to Godunov's method, *J. Comput. Phys.*, **32** (1979), 101–136.

[18] J.H. Wang and G. Warnecke, On entropy consistency of large time step scheme I. the Godunov and Glimm schemes, *SIAM J. Numer. Anal.*, **30** (1993), 1229-1251.