High order fixed-point sweeping WENO methods for steady state of hyperbolic conservation laws and its convergence study

Liang Wu[1], Yong-Tao Zhang[2], Shuhai Zhang[3], Chi-Wang Shu[4]

ABSTRACT

Fixed-point iterative sweeping methods were developed in the literature to efficiently solve static Hamilton-Jacobi equations. This class of methods utilizes the Gauss-Seidel iterations and alternating sweeping strategy to achieve fast convergence rate. They take advantage of the properties of hyperbolic partial differential equations (PDEs) and try to cover a family of characteristics of the corresponding Hamilton-Jacobi equation in a certain direction simultaneously in each sweeping order. Different from other fast sweeping methods, fixed-point iterative sweeping methods have the advantages such as that they have explicit forms and do *not* involve inverse operation of nonlinear local systems. In principle, it can be applied in solving very general equations using any monotone numerical fluxes and high order approximations easily. In this paper, based on the recently developed fifth order WENO schemes which improve the convergence of the classical WENO schemes by removing slight post-shock oscillations, we design fifth order fixed-point sweeping WENO methods for efficient computation of steady state solution of hyperbolic conservation laws. Especially, we show that although the methods do *not* have linear computational complexity, they converge to steady state solutions much faster than regular time-marching approach by stability improvement for high order schemes with a forward Euler time-marching.

Key Words: Fixed-point sweeping methods, WENO methods, high order accuracy, steady state, hyperbolic conservation laws, convergence

[1] Department of Applied and Computational Mathematics and Statistics, University of Notre Dame, Notre Dame, IN 46556, USA. E-mail: lwu2@nd.edu

[2] Department of Applied and Computational Mathematics and Statistics, University of Notre Dame, Notre Dame, IN 46556, USA. E-mail: yzhang10@nd.edu

[3] State Key Laboratory of Aerodynamics, China Aerodynamics Research and Development Center, Mianyang, Sichuan 621000, China. E-mail: shuhai_zhang@163.com

[4] Division of Applied Mathematics, Brown University, Providence, RI 02912, USA. E-mail: shu@dam.brown.edu.
Research supported by ARO grant W911NF-15-1-0226 and NSF grant DMS-1418750.

1 Introduction

Steady state problems for hyperbolic partial differential equations (PDEs) are common mathematical models appearing in many applications, such as fluid mechanics, optimal control, differential games, image processing and computer vision, geometric optics, etc. Solution information of these boundary value problems propagates along characteristics starting from the boundary. Weighted essentially non-oscillatory (WENO) schemes are a popular class of high order numerical methods for spatial discretization of hyperbolic PDEs. They have the advantage of attaining uniform high order accuracy in smooth regions of the solution while maintaining sharp and essentially non-oscillatory transitions of discontinuities. WENO scheme was first constructed in [11] for a third-order accurate finite volume version. In [7], third- and fifth-order accurate finite difference WENO schemes in multi-space dimensions were constructed, with a general framework for the design of the smoothness indicators and nonlinear weights. To deal with complex domain geometries, WENO schemes on unstructured meshes were developed, e.g., see [6, 23, 9, 24, 28, 12].

A large nonlinear system is obtained after spatial discretization of a steady state hyperbolic PDE by a high order WENO scheme. It is still a challenging problem how to solve the large nonlinear system. There are at least two factors which may affect efficiency and robustness of computation. One is that a high order accurate shock capturing scheme such as a fifth order WENO scheme often suffers from difficulties in its convergence towards steady state solutions. In [21], A systematic study was carried out and discovered that slight post-shock oscillations actually cause this problem. A new smoothness indicator [21] and upwind-biased interpolation technique [20] have been developed to improve the convergence of fifth order WENO scheme for solving steady state of Euler systems. The other factor affecting the performance of computation is the iterative scheme designed for the nonlinear system. For a highly nonlinear system derived from high order WENO spatial discretization, one way is to solve it directly with Newton iterations or a more robust method such as the homotopy method [5]. Another way is to solve the large WENO system by fast sweeping technique [26]. Fast sweeping methods utilize alternating sweeping strategy to cover a family of characteristics in a certain direction simultaneously in each sweeping order. Coupled with the Gauss-Seidel iterations, these methods can achieve a fast convergence speed for computations of steady state solutions of hyperbolic PDEs. First order fast sweeping methods often achieve linear computational complexity for certain types of equations (e.g., see [27, 13, 14, 4]). There are additional difficulties to design high order fast sweeping methods with linear computational complexity, including lack of monotonicity of numerical solutions, much more complicated local nonlinear equations, wider stencils which make alternating sweeping less effective, etc. High order WENO fast sweeping method was developed in [26]. An explicit strategy was designed to avoid directly solving very complicated local nonlinear equations derived from WENO discretizations. The method was extended to a fifth order version in [19] with accurate boundary treatment techniques. This explicit strategy has been applied in Lax-Friedrichs fast sweeping method to solve steady state problems for hyperbolic conservation laws in [2]. High order WENO fast sweeping methods are much more efficient than classical time marching approach for solving steady state problems, although their computational complexity is not linear. Discontinuous Galerkin (DG) [3] fast sweeping methods achieve linear computational complexity due to very compact stencils which facilitate the propagation of upwind information. Second order DG fast sweeping methods were developed in [10, 22], and a third order DG fast sweeping method was recently developed in [18] for Eikonal equations. Although DG fast sweeping methods have linear computational complexity, a numerical comparison was performed in [18] and it shows that each method has its advantages. For the example studied in [18], it was found that the DG one is more efficient than the third order WENO fast sweeping method to obtain accurate results for the smooth region of the solution. On the other hand, the third order WENO fast sweeping method is more efficient for the regions with derivative singularities to get certain accuracy.

Another approach to explicitly incorporate high order WENO discretizations into fast sweeping techniques is by fixed-point iterative methods. Fixed-point iterative sweeping WENO methods were first developed in [25] for solving static Hamilton-Jacobi equations. Fixed-point iterative sweeping methods have the advantages such as that they have explicit forms and do *not* involve inverse operation of nonlinear local systems. In principle, they can be applied in solving very general equations using any monotone numerical fluxes and high order approximations (e.g. high order WENO approximations) easily. The approach has been applied to solve steady state solution of scalar hyperbolic conservation laws with the third order finite difference WENO method in [1]. In this paper, based on the recently developed fifth order WENO schemes which improve the convergence of the classical WENO schemes by removing slight post-shock oscillations, we design fifth order fixed-point sweeping WENO methods for efficiently solving steady state problems of hyperbolic conservation laws. Especially, we show that although the methods do *not* have linear computational complexity, they converge to steady state solutions much faster than regular time-marching approach. It is interesting to see that the acceleration of computation is essentially achieved via stability improvement for high order schemes with a forward Euler time-marching to steady state solutions.

The rest of the paper is organized as follows. The detailed algorithm is described in Section 2. In Section 3 we provide extensive numerical experiments to test and study the proposed methods. Comparisons of different methods are performed. Concluding remarks are given in Section 4.

2 Fifth order fixed-point sweeping WENO methods

Consider steady state problems of hyperbolic conservation laws with appropriate boundary conditions

$$\nabla \cdot F(U) = h, \tag{1}$$

where U is the vector of the unknown conservative variables, $F(U)$ is the vector of flux functions, and h is the source term. A spatial discretization of (1) usually leads to a large nonlinear system of N equations where N is the number of spatial grid points.

2.1 WENO discretization

In this paper, to discretize (1) we use the fifth order finite difference WENO (WENO5) scheme [7] with recently developed techniques to improve the convergence of WENO schemes for steady state computations [21, 20].

For the hyperbolic terms $f(u)_x + g(u)_y$, the conservative finite difference scheme we use approximates the point values at a uniform (or smoothly varying) grid (x_i, y_j) in a conservative fashion. Namely, the derivative $f(u)_x$ at (x_i, y_j) is approximated along the line $y = y_j$ by a conservative flux difference

$$f(u)_x|_{x=x_i, y=y_j} \approx \frac{1}{\Delta x}(\hat{f}_{i+1/2,j} - \hat{f}_{i-1/2,j}), \tag{2}$$

where for the fifth order WENO scheme the numerical flux $\hat{f}_{i+1/2,j}$ depends on the five-point values $f(u_{l,j})$, $l = i-2, i-1, i, i+1, i+2$, when the wind is positive (i.e., when $f'(u) \geq 0$ for the scalar case, or when the corresponding eigenvalue is positive for the system case with a local characteristic decomposition). This numerical flux $\hat{f}_{i+1/2,j}$ is written as a convex combination of three third order numerical fluxes based on three different substencils of three points each, and the combination coefficients depend on a "smoothness indicator" measuring the smoothness of the solution in each substencil. The detailed formula is

$$\hat{f}_{i+1/2,j} = w_0 \hat{f}^{(0)}_{i+1/2,j} + w_1 \hat{f}^{(1)}_{i+1/2,j} + w_2 \hat{f}^{(2)}_{i+1/2,j}, \tag{3}$$

where

$$\begin{aligned}
\hat{f}^{(0)}_{i+1/2,j} &= \frac{1}{3}f(u_{i-2,j}) - \frac{7}{6}f(u_{i-1,j}) + \frac{11}{6}f(u_{i,j}),\\
\hat{f}^{(1)}_{i+1/2,j} &= -\frac{1}{6}f(u_{i-1,j}) + \frac{5}{6}f(u_{i,j}) + \frac{1}{3}f(u_{i+1,j}),\\
\hat{f}^{(2)}_{i+1/2,j} &= \frac{1}{3}f(u_{i,j}) + \frac{5}{6}f(u_{i+1,j}) - \frac{1}{6}f(u_{i+2,j}).
\end{aligned} \tag{4}$$

Also

$$w_r = \frac{\alpha_r}{\alpha_1+\alpha_2+\alpha_3}, \qquad \alpha_r = \frac{d_r}{(\epsilon+\beta_r)^2}, \qquad r = 0,1,2. \tag{5}$$

$d_0 = 0.1, d_1 = 0.6, d_2 = 0.3$ are called the "linear weights", and $\beta_0, \beta_1, \beta_2$ are called the "smoothness indicators" with the explicit formulae

$$\begin{aligned}
\beta_0 &= \frac{13}{12}(f_{i-2,j} - 2f_{i-1,j} + f_{i,j})^2 + \frac{1}{4}(f_{i-2,j} - 4f_{i-1,j} + 3f_{i,j})^2,\\
\beta_1 &= \frac{13}{12}(f_{i-1,j} - 2f_{i,j} + f_{i+1,j})^2 + \frac{1}{4}(f_{i-1,j} - f_{i+1,j})^2,\\
\beta_2 &= \frac{13}{12}(f_{i,j} - 2f_{i+1,j} + f_{i+2,j})^2 + \frac{1}{4}(3f_{i,j} - 4f_{i+1,j} + f_{i+2,j})^2,
\end{aligned} \tag{6}$$

where $f_{k,l}$ denotes $f(u_{k,l})$. ϵ is a small positive number chosen to avoid the denominator becoming 0. We take $\epsilon = 10^{-6}$ in this paper.

When the wind is negative (i.e., when $f'(u) < 0$), right-biased stencil with numerical values $f(u_{i-1,j}), f(u_{i,j}), f(u_{i+1,j}), f(u_{i+2,j})$ and $f(u_{i+3,j})$ are used to construct a fifth order WENO approximation to the numerical flux $\hat{f}_{i+1/2,j}$. The formulae for negative and positive wind cases are symmetric with respect to the point $x_{i+1/2}$. For the general case of $f(u)$, we perform the "Lax-Friedrichs flux splitting"

$$f^+(u) = \frac{1}{2}(f(u) + \alpha u), \qquad f^-(u) = \frac{1}{2}(f(u) - \alpha u), \tag{7}$$

where $\alpha = \max_u |f'(u)|$. $f^+(u)$ is the positive wind part, and $f^-(u)$ is the negative wind part. Corresponding WENO approximations are applied to find numerical fluxes $\hat{f}^+_{i+1/2,j}$ and $\hat{f}^-_{i+1/2,j}$ respectively. Then $\hat{f}_{i+\frac{1}{2},j} = \hat{f}^+_{i+1/2,j} + \hat{f}^-_{i+1/2,j}$. Similar procedures are applied to the y direction for $g(u)_y$. Then we obtain a nonlinear system

$$0 = -(\hat{f}_{i+1/2,j} - \hat{f}_{i-1/2,j})/\Delta x - (\hat{g}_{i,j+1/2} - \hat{g}_{i,j-1/2})/\Delta y + h(u_{ij}, x_i, y_j), \quad i = 1, \cdots, N; j = 1, \cdots, M, \tag{8}$$

where $\hat{f}$, $\hat{g}$ are the numerical fluxes obtained by Lax-Friedrichs flux splitting and WENO approximation.

High order accuracy methods including the WENO methods suffer from difficulties in their convergence to steady state solutions. For example, as shown in [21], the residue of WENO schemes often stops decreasing during their iterations. The residue hangs at a level far above machine zero. A systematic study in [21] reveals that slight post-shock oscillations actually cause this problem, and a new smoothness indicator for the fifth order WENO scheme is designed to make the residue settle down to machine zero. In this paper, we use the new smoothness indicator instead of the original ones (6). The explicit formulae for the new smoothness indicators are

$$\begin{aligned} \beta_0 &= (f_{i-2,j} - 4f_{i-1,j} + 3f_{i,j})^2, \\ \beta_1 &= (f_{i-1,j} - f_{i+1,j})^2, \\ \beta_2 &= (3f_{i,j} - 4f_{i+1,j} + f_{i+2,j})^2. \end{aligned} \tag{9}$$

For systems of hyperbolic conservation laws, the local characteristic decomposition is often needed in high order accuracy schemes for solving strong shock problems. In [20], it is shown that the local characteristic decomposition has a close relationship with the slight post-shock oscillation. The slight post-shock oscillation often appears in a standard high order accuracy WENO simulation if the Roe average is used to form the Jacobian at the cell interface for the local characteristic decomposition. Again, the slight post-shock oscillation is responsible for the numerical residue to hang at a high level instead of settling down to machine zero when a fifth order WENO scheme is used to compute steady state solutions. To improve the convergence, upwind-biased interpolation is used to form the Jacobian in high order WENO schemes [20] and the slight post-shock oscillation can be removed or reduced significantly. The numerical residue can settle down to a much lower level than that by using the standard Roe average. In this paper, we incorporate the upwind-biased interpolation in the fifth order sweeping WENO scheme.

Upwind-biased interpolation uses only or main information from one side of the shock for grid points near the shock. In the upwind-biased interpolation for the $x-$direction local characteristic decomposition, we choose the physical variables on the cell interface $U_{i+1/2,j} = U^{(1)}$ when $u_{i+1/2,j} \geq 0$ (here u denotes the $x-$direction velocity in the Euler equations) and $U_{i+1/2,j} = U^{(2)}$ when $u_{i+1/2,j} < 0$, where $U^{(1)}$ and $U^{(2)}$ are the interpolated values on the cell interface, which are computed by the first order or the second order one-sided interpolation, or the higher order upwind-biased WENO interpolation. The first order upwind-biased interpolation turns out to be the most efficient and effective one to decrease the post-shock oscillations and drive the residue of high order

WENO schemes to machine zero or a much smaller value. Its formulae are

$$\begin{aligned} U^{(1)} &= U_{i,j}, \\ U^{(2)} &= U_{i+1,j}. \end{aligned} \tag{10}$$

To calculate $u_{i+1/2,j}$, the Roe average [15]

$$u_{i+1/2,j} = \frac{\sqrt{\rho_{i,j}}}{\sqrt{\rho_{i,j}} + \sqrt{\rho_{i+1,j}}} u_{i,j} + \frac{\sqrt{\rho_{i+1,j}}}{\sqrt{\rho_{i,j}} + \sqrt{\rho_{i+1,j}}} u_{i+1,j} \tag{11}$$

is used, where ρ is the density in the Euler equations. For the upwind-biased interpolation of the $y-$direction local characteristic decomposition in two dimensional case, similar procedure is followed by using the $y-$direction velocity v. We emphasize that the order of accuracy of the final WENO scheme does not depend on the order of interpolation in forming the Jacobian matrix for the local characteristic decomposition [7]. Hence the first order interpolation (10) here does not affect the high order accuracy of the final WENO scheme at all.

2.2 Fixed-point sweeping iterative schemes

Time marching approach for solving steady state problems is essentially a Jacobi type fixed-point iterative scheme for the nonlinear system (8). The right-hand-side (RHS) of (8) is a nonlinear function of numerical values at the grid points of computational stencils. We denote it by L and can write a Jacobi type fixed-point iterative scheme as the following

$$u_{ij}^{n+1} = u_{ij}^n + \frac{\gamma}{\alpha_x/\Delta x + \alpha_y/\Delta y} L(u_{i-r,j}^n, \cdots, u_{i+s,j}^n; u_{ij}^n; u_{i,j-r}^n, \cdots, u_{i,j+s}^n),$$
$$i = 1, \cdots, N; j = 1, \cdots, M, \tag{12}$$

where r, s are values which depend on the order of the WENO approximation. For the fifth order WENO scheme used in this paper, we have $r = s = 3$. n is the iteration step. $\alpha_x = \max_u |f'(u)|$ and $\alpha_y = \max_u |g'(u)|$ for the scalar equations, or they are the maximum absolute values of eigenvalues of the Jacobian matrices $f'(u)$ and $g'(u)$ for the system cases. They are the maximum characteristic speeds in each spatial direction. α_x and α_y are updated in every iteration. The parameter γ is chosen to be suitable values which can guarantee that the fixed-point iteration is a contractive mapping and it converges. In fact, the scheme (12) is the forward Euler (FE) time marching method with time step size $\Delta t_n = \frac{\gamma}{\alpha_x/\Delta x + \alpha_y/\Delta y}$. The parameter γ actually represents the CFL number. Since a higher order linear scheme with the forward Euler time discretization has linear stability issue, the Jacobi iterative scheme (12) needs many iteration steps to converge even with the help of a nonlinearly stable discretization such as WENO schemes. However, as shown in the numerical experiments (Section 3), by applying the Gauss-Seidel sweeping technique to the fixed-point scheme, we can obtain a much more efficient iterative scheme. The number of iteration steps to the steady state is reduced significantly and the CFL number γ is much larger than that in the Jacobi iteration (12).

The fast sweeping technique has two components, namely, the Gauss-Seidel philosophy and alternating direction iterations. By the Gauss-Seidel philosophy, the newest available numerical values of u are used in the interpolation stencils as long as they are available. The FE type fixed-point sweeping scheme can be written as

$$u_{ij}^{n+1} = u_{ij}^n + \frac{\gamma}{\alpha_x/\Delta x + \alpha_y/\Delta y} L(u_{i-r,j}^*, \cdots, u_{i+s,j}^*; u_{ij}^n; u_{i,j-r}^*, \cdots, u_{i,j+s}^*),$$
$$i = i_1, \cdots, i_N; j = j_1, \cdots, j_M. \tag{13}$$

Here the iterations do *not* just proceed in only one direction $i = 1 : N, j = 1 : M$ as the time-marching approach (12), but in the following four alternating directions repeatedly,

$$(1)\ i = 1 : N, j = 1 : M;$$
$$(2)\ i = N : 1, j = 1 : M;$$
$$(3)\ i = N : 1, j = M : 1;$$
$$(4)\ i = 1 : N, j = M : 1.$$

Since the strategy of alternating direction sweepings utilizes the characteristics property of hyperbolic PDEs, combining with the Gauss-Seidel philosophy, we are able to observe the acceleration of convergence speed, which will be shown in the following numerical experiments. By the Gauss-Seidel philosophy, we use the newest numerical values on the computational stencil of the WENO scheme whenever they are available. That is the reason we use the notation u^* to represent the values in the scheme (13), and $u_{k,l}^*$ could be $u_{k,l}^n$ or $u_{k,l}^{n+1}$, depending on the current sweeping direction.

Similarly for the third order TVD Runge-Kutta (RK) scheme (a RK type Jacobi iteration scheme) [17]

$$u_{ij}^{(1)} = u_{ij}^n + \Delta t_n L(u_{i-r,j}^n, \cdots, u_{i+s,j}^n; u_{ij}^n; u_{i,j-r}^n, \cdots, u_{i,j+s}^n), \quad i = 1, \cdots, N; j = 1, \cdots, M, \tag{14}$$

$$u_{ij}^{(2)} = \frac{3}{4}u_{ij}^n + \frac{1}{4}u_{ij}^{(1)} + \frac{1}{4}\Delta t_n L(u_{i-r,j}^{(1)}, \cdots, u_{i+s,j}^{(1)}; u_{ij}^{(1)}; u_{i,j-r}^{(1)}, \cdots, u_{i,j+s}^{(1)}), \quad i = 1, \cdots, N; j = 1, \cdots, M, \tag{15}$$

$$u_{ij}^{n+1}=\frac{1}{3}u_{ij}^{n}+\frac{2}{3}u_{ij}^{(2)}+\frac{2}{3}\Delta t_n L(u_{i-r,j}^{(2)},\cdots,u_{i+s,j}^{(2)};u_{ij}^{(2)};u_{i,j-r}^{(2)},\cdots,u_{i,j+s}^{(2)}),\quad i=1,\cdots,N;j=1,\cdots,M, \tag{16}$$

the RK type fixed-point sweeping scheme has the form

$$u_{ij}^{(1)}=u_{ij}^{n}+\frac{\gamma}{\alpha_x/\Delta x+\alpha_y/\Delta y}L(u_{i-r,j}^{*},\cdots,u_{i+s,j}^{*};u_{ij}^{n};u_{i,j-r}^{*},\cdots,u_{i,j+s}^{*}),$$

$$i=i_1,\cdots,i_N;j=j_1,\cdots,j_M, \tag{17}$$

$$u_{ij}^{(2)}=u_{ij}^{(1)}+\frac{\gamma}{4(\alpha_x/\Delta x+\alpha_y/\Delta y)}L(u_{i-r,j}^{**},\cdots,u_{i+s,j}^{**};u_{ij}^{(1)};u_{i,j-r}^{**},\cdots,u_{i,j+s}^{**}),$$

$$i=i_1,\cdots,i_N;j=j_1,\cdots,j_M, \tag{18}$$

$$u_{ij}^{n+1}=u_{ij}^{(2)}+\frac{2\gamma}{3(\alpha_x/\Delta x+\alpha_y/\Delta y)}L(u_{i-r,j}^{***},\cdots,u_{i+s,j}^{***};u_{ij}^{(2)};u_{i,j-r}^{***},\cdots,u_{i,j+s}^{***}),$$

$$i=i_1,\cdots,i_N;j=j_1,\cdots,j_M. \tag{19}$$

The above schemes (17)-(19) denote a complete iteration step n which includes three sub-iterations. Again, the complete iterations do *not* just proceed in only one direction $i=1:N,j=1:M$ as the time-marching approach (14)-(16), but in four alternating directions repeatedly. Note that the sweeping directions of the three sub-iterations (17)-(19) are the same inside a complete iteration step n. The newest numerical values are used on the computational stencil of the WENO scheme whenever they are available. That is the reason why we use notations such as u^*, u^{**}, u^{***} to represent the values in the scheme (17)-(19). For example $u_{k,l}^{*}$ could be $u_{k,l}^{n}$ or $u_{k,l}^{(1)}$, depending on the current sweeping direction; similarly $u_{k,l}^{**}$ could be $u_{k,l}^{(1)}$ or $u_{k,l}^{(2)}$, and $u_{k,l}^{***}$ could be $u_{k,l}^{(2)}$ or $u_{k,l}^{n+1}$. For RK type schemes, α_x and α_y are updated once in a complete iteration step, namely, three sub-iterations in a complete iteration have the same α_x and α_y values.

3 Numerical experiments

In this section, we use numerical experiments to test the efficiency of the fifth order sweeping WENO schemes. Computational efficiency of four different iterative schemes is compared. For the convenience of presentation, we name the scheme (12) FE Jacobi scheme, the scheme (13) FE sweeping scheme, the scheme (14)-(16) RK Jacobi scheme, and the scheme (17)-(19) RK sweeping scheme. With mesh refinement study, we compute L_1 and L_∞ numerical errors and accuracy orders. Grid point where the maximum error occurs is tracked, and it is called "L_∞ index i" in the following presented Tables. Iteration numbers and CPU times for each iterative method to converge are reported and compared. The convergence of the iterations is measured by the residue which is defined as

$$\mathrm{R}es_A = \sum_{i=1}^{N} \frac{|R_i|}{N}, \tag{20}$$

where the local residue

$$R_i = \frac{\partial u}{\partial t}\bigg|_i = \frac{u_i^{n+1} - u_i^n}{\Delta t_n}, \tag{21}$$

and N is total number of grid points and n is the iteration step. $\Delta t_n = \frac{\gamma}{\alpha_x/\Delta x + \alpha_y/\Delta y}$. For every test case of every example in this section, we count number of iterations for the methods to reach convergence. For most cases, the convergence criterion is set to be $\mathrm{R}es_A < 10^{-12}$ except that in some examples we study the levels that the residues can reach. Note that number of iterations reported in every Table here counts a complete update of numerical values in all grid points once as one iteration.

3.1 Example 1. Burgers' equation

We consider the following one-dimensional Burgers' equation with a source term

$$u_t + \left(\frac{u^2}{2}\right)_x = \sin(x)\cos(x), \qquad x \in [\frac{1}{4}\pi, \frac{3}{4}\pi], \tag{22}$$

and compute its steady state solution. The initial condition $u(x,0) = \beta\sin(x)$ is used as the initial guess in the iterations. An inflow boundary condition is imposed at the left boundary $x = \frac{1}{4}\pi$ with $u(\frac{1}{4}\pi, t) = \frac{\sqrt{2}}{2}$. And at the right boundary $x = \frac{3}{4}\pi$, the outflow boundary condition is applied. If $\beta > 1$, the unique steady state solution for this problem is $u(x,\infty) = \sin(x)$. We take $\beta = 2$ in this example. Four different iterative schemes based on the WENO5 discretization with either the original smoothness indicators or the new smoothness indicators are used to compute the steady state solution. For the outflow boundary point $x = \frac{3}{4}\pi$ itself and ghost points to the right of $x = \frac{3}{4}\pi$ in the stencil of WENO5 scheme, extrapolation by a degree 4 polynomial is used to compute numerical values at them. $\mathrm{R}es_A < 10^{-12}$ is used as the iteration convergence criterion. The results for four different iterative schemes with the original smoothness indicators (6) in the WENO5 are presented in Table 1. And the results for these iterative schemes with the new smoothness indicators (9) in the WENO5 are reported in Table 2. For this example, the WENO5 with original smoothness indicators has no difficulty to reach convergence. We observe that all schemes achieve similar numerical errors and fifth order accuracy when they converge, and maximum errors generally occur at grid points close to the right boundary. In terms of algorithm efficiency, the direct forward Euler scheme with WENO5 (i.e., the FE Jacobi scheme (12)) needs very small CFL number $\gamma = 0.1$ to achieve the convergence. This is because a forward Euler time discretization with a very high order linear spatial discretization (even a high order linear upwind one) suffers from linear stability problem. The nonlinear stable WENO discretization can help against linear instability. As a result, the FE Jacobi scheme can converge with a small CFL number which leads to large iteration numbers and the most CPU time among these four iterative schemes. With a high order TVD RK scheme (the third order here), the RK Jacobi scheme (14)-(16) under WENO5 discretization is both linearly and nonlinearly stable. Hence a much larger CFL number $\gamma = 1.0$ can be used. The iteration numbers and CPU costs are reduced a lot by using the RK Jacobi scheme rather than the FE Jacobi scheme. Fast sweeping techniques improve the convergence of Jacobi schemes significantly, as shown in Tables 1 and 2 for the performance of the RK sweeping scheme (17)-(19) and the FE sweeping scheme (13). On the most refined mesh for this example, we can see that the RK sweeping scheme just needs about 30% iteration number and CPU time of the RK Jacobi scheme to converge, while the FE sweeping scheme only needs about 5% iteration number and CPU time of the FE Jacobi scheme. Furthermore, it is interesting to see that with the fast sweeping technique, the FE sweeping scheme can also use a large CFL number $\gamma = 1.0$. So it suggests that the fast sweeping technique improves the linear stability of the forward Euler scheme when it is applied in a high order spatial scheme. For a steady state calculation, since the accuracy in the time direction is not a concern, the forward Euler time marching actually has an advantage that it is just a simple one stage method comparing to multi-stage Runge-Kutta schemes. However, due to its linear stability issue with a high order spatial scheme, it is not practically useful. Now this problem is solved by using the fast sweeping technique, i.e., using the FE sweeping scheme rather than the FE Jacobi scheme. Actually, as shown in Tables 1 and 2, the FE sweeping scheme is the most efficient one among all four iterative methods. For RK type schemes, the RK sweeping scheme and the RK Jacobi scheme converge at similar γ values.

Next we use this example to show that the fixed-point sweeping method can be applied in arbitrary monotone fluxes, not only just the Lax-Friedrichs flux splitting (7). An alternative formulation of WENO schemes, developed in [17, 8], needs to be used for constructing numerical fluxes based on the point values of the numerical solution. In this alternative formulation, the numerical fluxes $\hat{f}_{i+1/2}$ are obtained by Taylor expansion. The fifth order accuracy is achieved by using

$$\hat{f}_{i+\frac{1}{2}} = f_{i+\frac{1}{2}} - \frac{1}{24}\Delta x^2 f_{xx}|_{i+\frac{1}{2}} + \frac{7}{5760}\Delta x^4 f_{xxxx}|_{i+\frac{1}{2}}, \tag{23}$$

where the first term can be approximated by any monotone flux

$$f_{i+\frac{1}{2}} = h(u^-_{i+\frac{1}{2}}, u^+_{i+\frac{1}{2}}). \tag{24}$$

The values $u^+_{i+1/2}$ and $u^-_{i+1/2}$ are obtained by the WENO5 approximations based on the point values of the numerical solution. Here we test three different monotone fluxes including the Godunov flux

$$h(a,b) = \begin{cases} \min_{a\le x\le b} f(u) \text{ if } a \le b, \\ \max_{b\le x\le a} f(u) \text{ if } a > b; \end{cases} \tag{25}$$

the Engquist-Osher flux

$$h(a,b) = \int_0^a \max(f'(u),0)du + \int_0^b \min(f'(u),0)du + f(0); \tag{26}$$

and the Lax-Friedrichs flux

$$h(a,b) = \frac{1}{2}[f(a) + f(b) - \alpha(b-a)], \tag{27}$$

where $\alpha = \max_u |f'(u)|$ is a constant and the maximum is taken over the relevant range of u. For the other terms in (23), as that pointed out in [8], they only need lower order approximations and they contribute much less to spurious oscillations due to at least Δx^2 in their coefficients. Specifically, $f_{xx}|_{i+\frac{1}{2}}$ should be approximated at least by a scheme with third order accuracy due to the Δx^2 term and $f_{xxxx}|_{i+\frac{1}{2}}$ should have at least a first order accuracy approximation due to the Δx^4 term in (23). For this example, we have

$$\begin{aligned} f(u)_{xx} &= (u_x)^2 + u \cdot u_{xx}, & (28) \\ f(u)_{xxxx} &= 3(u_{xx})^2 + 4\ u_x \cdot u_{xxx} + u \cdot u_{xxxx}. & (29) \end{aligned}$$

Central differences are used to approximate u_x, u_{xx}, u_{xxx} and u_{xxxx} as following:

$$\begin{aligned} u'(x)|_{i+\frac{1}{2}} &= \frac{1}{24h}(u_{i-1} - 27u_i + 27u_{i+1} - u_{i+2}) + O(h^4), & (30) \\ u''(x)|_{i+\frac{1}{2}} &= \frac{1}{18h^2}(-u_{i-1} + 81u_i + 81u_{i+1} - u_{i+2}) - \frac{80}{18h^2}(u^+_{i+\frac{1}{2}} + u^-_{i+\frac{1}{2}}) + O(h^3), & (31) \\ u'''(x)|_{i+\frac{1}{2}} &= \frac{1}{h^3}(-u_{i-1} + 3u_i - 3u_{i+1} + u_{i+2}) + O(h^2), & (32) \\ u''''(x)|_{i+\frac{1}{2}} &= \frac{8}{3h^4}(u_{i-1} - 9u_i - 9u_{i+1} + u_{i+2}) + \frac{8\cdot 8}{3h^4}(u^+_{i+\frac{1}{2}} + u^-_{i+\frac{1}{2}}) + O(h). & (33) \end{aligned}$$

Note that to approximate $u_{i+\frac{1}{2}}$, we directly use the average of $u^+_{i+\frac{1}{2}}$ and $u^-_{i+\frac{1}{2}}$, which are both fifth order approximations of $u_{i+\frac{1}{2}}$. Since the FE sweeping scheme is the most efficient one among all four iterative methods, we applied the Lax-Friedrichs flux, the Godunov flux and the Engquist-Osher flux in the FE sweeping scheme to solve this problem. The numerical results are reported in Table 3 and Table 4 for the original smoothness indicators and the new smoothness indicators respectively. We observe that all tests achieve fifth order accuracy up to round-off errors. Slight reductions of the L_∞ order for the $N = 320$ mesh are due to accumulations of round-off errors. In terms of efficiency, the FE sweeping scheme has similar performance for different monotone fluxes.

Remark. This example can have different steady state solutions for different initial conditions. For a specific initial condition, there is an unique steady state solution. Our sweeping methods are based on time-marching schemes. While the methods have time-step size constraint by the CFL condition, they converge to the stable steady state for a specific well-posed initial-boundary-value problem.

3.2 Example 2. 1D shallow water equation

In this example, we apply these iterative schemes in solving a one-dimensional system, the shallow water equation

$$\begin{pmatrix} h \\ hu \end{pmatrix}_t + \begin{pmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \end{pmatrix}_x = \begin{pmatrix} 0 \\ -ghb_x \end{pmatrix},$$

where h denotes the water height, u is the velocity of the fluid, $b(x)$ represents the bottom topography, and g is the gravitational constant. We consider the smooth bottom topography given by

$$b(x) = 5e^{-\frac{2}{5}(x-5)^2}, \qquad x \in [0, 10].$$

This problem has the steady state solution

$$h + b = 10, \quad hu = 0.$$

The exact solution is used as the initial guess in the iterations. Since the exact steady state solution of the PDE does not satisfy the numerical schemes, we can observe convergence behavior of iterative schemes starting from it. We impose the exact solution for numerical values at boundary points. The fifth order WENO scheme with the first order upwind biased interpolation (10) (called "U1WENO5" in [20]) is used in the iterative schemes. The results for four different iterative schemes with the original smoothness indicators (6) in the U1WENO5 are presented in Table 5. And the results for these iterative schemes with the new smoothness indicators (9) in the fifth order WENO scheme with the first order upwind biased interpolation (called "U1ZSWENO5" in [20]) are reported in Table 6. We observe that all schemes achieve similar numerical errors and higher than fifth order accuracy in this example. Maximum errors generally occur at grid points

Table 1: Example 1. Accuracy, the grid point where maximum error occurs (L_∞ index i), iteration numbers and CPU times of four different iterative schemes. The original smoothness indicators (6) are used in WENO5. CPU time unit: second.

<table>
<tr><td colspan="8">RK Jacobi, $\gamma = 1.0$</td></tr>
<tr><td>N</td><td>L_1 error</td><td>L_1 order</td><td>L_∞ error</td><td>L_∞ index i</td><td>L_∞ order</td><td>iter #</td><td>CPU time</td></tr>
<tr><td>10</td><td>1.15e-4</td><td></td><td>1.88e-4</td><td>9</td><td></td><td>183</td><td>1.82e-3</td></tr>
<tr><td>20</td><td>2.85e-6</td><td>5.33</td><td>5.94e-6</td><td>17</td><td>4.98</td><td>258</td><td>4.36e-3</td></tr>
<tr><td>40</td><td>8.31e-8</td><td>5.10</td><td>1.86e-7</td><td>37</td><td>5.00</td><td>357</td><td>1.19e-2</td></tr>
<tr><td>80</td><td>2.17e-9</td><td>5.26</td><td>5.26e-9</td><td>79</td><td>5.14</td><td>615</td><td>4.12e-2</td></tr>
<tr><td>160</td><td>5.24e-11</td><td>5.38</td><td>1.30e-10</td><td>159</td><td>5.34</td><td>1185</td><td>0.16</td></tr>
<tr><td>320</td><td>1.11e-12</td><td>5.56</td><td>2.94e-12</td><td>316</td><td>5.46</td><td>1818</td><td>0.49</td></tr>
<tr><td colspan="8">RK Sweeping, $\gamma = 1.0$</td></tr>
<tr><td>N</td><td>L_1 error</td><td>L_1 order</td><td>L_∞ error</td><td>L_∞ index i</td><td>L_∞ order</td><td>iter #</td><td>CPU time</td></tr>
<tr><td>10</td><td>1.15e-4</td><td></td><td>1.88e-4</td><td>9</td><td></td><td>150</td><td>1.60e-3</td></tr>
<tr><td>20</td><td>2.85e-6</td><td>5.33</td><td>5.94e-6</td><td>17</td><td>4.98</td><td>168</td><td>2.80e-3</td></tr>
<tr><td>40</td><td>8.31e-8</td><td>5.10</td><td>1.86e-7</td><td>37</td><td>5.00</td><td>198</td><td>6.47e-3</td></tr>
<tr><td>80</td><td>2.17e-9</td><td>5.26</td><td>5.26e-9</td><td>79</td><td>5.14</td><td>273</td><td>1.80e-2</td></tr>
<tr><td>160</td><td>5.24e-11</td><td>5.37</td><td>1.31e-10</td><td>159</td><td>5.32</td><td>408</td><td>5.40e-2</td></tr>
<tr><td>320</td><td>1.17e-12</td><td>5.48</td><td>3.09e-12</td><td>317</td><td>5.41</td><td>612</td><td>0.16</td></tr>
<tr><td colspan="8">FE Jacobi, $\gamma = 0.1$</td></tr>
<tr><td>N</td><td>L_1 error</td><td>L_1 order</td><td>L_∞ error</td><td>L_∞ index i</td><td>L_∞ order</td><td>iter #</td><td>CPU time</td></tr>
<tr><td>10</td><td>1.15e-4</td><td></td><td>1.88e-4</td><td>9</td><td></td><td>1034</td><td>9.58e-3</td></tr>
<tr><td>20</td><td>2.85e-6</td><td>5.33</td><td>5.94e-6</td><td>17</td><td>4.98</td><td>1391</td><td>2.26e-2</td></tr>
<tr><td>40</td><td>8.31e-8</td><td>5.10</td><td>1.86e-7</td><td>37</td><td>5.00</td><td>1656</td><td>5.36e-2</td></tr>
<tr><td>80</td><td>2.17e-9</td><td>5.26</td><td>5.26e-9</td><td>79</td><td>5.14</td><td>2190</td><td>0.15</td></tr>
<tr><td>160</td><td>5.24e-11</td><td>5.37</td><td>1.31e-10</td><td>159</td><td>5.33</td><td>3996</td><td>0.54</td></tr>
<tr><td>320</td><td>1.13e-12</td><td>5.53</td><td>2.93e-12</td><td>317</td><td>5.48</td><td>6737</td><td>1.79</td></tr>
<tr><td colspan="8">FE Sweeping, $\gamma = 1.0$</td></tr>
<tr><td>N</td><td>L_1 error</td><td>L_1 order</td><td>L_∞ error</td><td>L_∞ index i</td><td>L_∞ order</td><td>iter #</td><td>CPU time</td></tr>
<tr><td>10</td><td>1.15e-4</td><td></td><td>1.88e-4</td><td>9</td><td></td><td>104</td><td>1.05e-3</td></tr>
<tr><td>20</td><td>2.85e-6</td><td>5.33</td><td>5.94e-6</td><td>17</td><td>4.98</td><td>128</td><td>2.26e-3</td></tr>
<tr><td>40</td><td>8.31e-8</td><td>5.10</td><td>1.86e-7</td><td>37</td><td>5.00</td><td>151</td><td>5.27e-3</td></tr>
<tr><td>80</td><td>2.17e-9</td><td>5.26</td><td>5.26e-9</td><td>79</td><td>5.14</td><td>178</td><td>1.14e-2</td></tr>
<tr><td>160</td><td>5.24e-11</td><td>5.37</td><td>1.31e-10</td><td>159</td><td>5.32</td><td>222</td><td>2.79e-2</td></tr>
<tr><td>320</td><td>1.17e-12</td><td>5.48</td><td>3.23e-12</td><td>317</td><td>5.35</td><td>328</td><td>8.33e-2</td></tr>
</table>

Table 2: Example 1. Accuracy, the grid point where maximum error occurs (L_∞ index i), iteration numbers and CPU times of four different iterative schemes. The new smoothness indicators (9) are used in WENO5. CPU time unit: second.

<table>
<tr><td colspan="8">RK Jacobi, $\gamma = 1.0$</td></tr>
<tr><td>N</td><td>L_1 error</td><td>L_1 order</td><td>L_∞ error</td><td>L_∞ index i</td><td>L_∞ order</td><td>iter #</td><td>CPU time</td></tr>
<tr><td>10</td><td>1.60e-4</td><td></td><td>3.44e-4</td><td>6</td><td></td><td>204</td><td>1.85e-3</td></tr>
<tr><td>20</td><td>3.99e-6</td><td>5.32</td><td>8.04e-6</td><td>14</td><td>5.42</td><td>264</td><td>4.23e-3</td></tr>
<tr><td>40</td><td>1.02e-7</td><td>5.29</td><td>2.16e-7</td><td>32</td><td>5.22</td><td>381</td><td>1.20e-2</td></tr>
<tr><td>80</td><td>2.64e-9</td><td>5.27</td><td>5.92e-9</td><td>71</td><td>5.19</td><td>627</td><td>3.95e-2</td></tr>
<tr><td>160</td><td>6.78e-11</td><td>5.28</td><td>1.65e-10</td><td>157</td><td>5.16</td><td>1152</td><td>0.15</td></tr>
<tr><td>320</td><td>1.48e-12</td><td>5.52</td><td>4.10e-12</td><td>319</td><td>5.33</td><td>1842</td><td>0.46</td></tr>
<tr><td colspan="8">RK Sweeping, $\gamma = 1.0$</td></tr>
<tr><td>N</td><td>L_1 error</td><td>L_1 order</td><td>L_∞ error</td><td>L_∞ index i</td><td>L_∞ order</td><td>iter #</td><td>CPU time</td></tr>
<tr><td>10</td><td>1.60e-4</td><td></td><td>3.44e-4</td><td>6</td><td></td><td>144</td><td>1.30e-3</td></tr>
<tr><td>20</td><td>3.99e-6</td><td>5.32</td><td>8.04e-6</td><td>14</td><td>5.42</td><td>177</td><td>2.71e-3</td></tr>
<tr><td>40</td><td>1.02e-7</td><td>5.29</td><td>2.16e-7</td><td>32</td><td>5.22</td><td>204</td><td>5.88e-3</td></tr>
<tr><td>80</td><td>2.64e-9</td><td>5.27</td><td>5.92e-9</td><td>71</td><td>5.19</td><td>288</td><td>1.68e-2</td></tr>
<tr><td>160</td><td>6.77e-11</td><td>5.28</td><td>1.65e-10</td><td>157</td><td>5.17</td><td>417</td><td>4.79e-2</td></tr>
<tr><td>320</td><td>1.64e-12</td><td>5.37</td><td>4.29e-12</td><td>319</td><td>5.26</td><td>624</td><td>0.15</td></tr>
<tr><td colspan="8">FE Jacobi, $\gamma = 0.1$</td></tr>
<tr><td>N</td><td>L_1 error</td><td>L_1 order</td><td>L_∞ error</td><td>L_∞ index i</td><td>L_∞ order</td><td>iter #</td><td>CPU time</td></tr>
<tr><td>10</td><td>1.60e-4</td><td></td><td>3.44e-4</td><td>6</td><td></td><td>922</td><td>7.48e-3</td></tr>
<tr><td>20</td><td>3.99e-6</td><td>5.32</td><td>8.04e-6</td><td>14</td><td>5.42</td><td>1407</td><td>2.14e-2</td></tr>
<tr><td>40</td><td>1.02e-7</td><td>5.29</td><td>2.16e-7</td><td>32</td><td>5.22</td><td>1781</td><td>5.28e-2</td></tr>
<tr><td>80</td><td>2.64e-9</td><td>5.27</td><td>5.92e-9</td><td>71</td><td>5.19</td><td>2529</td><td>0.15</td></tr>
<tr><td>160</td><td>6.77e-11</td><td>5.28</td><td>1.65e-10</td><td>159</td><td>5.17</td><td>4390</td><td>0.54</td></tr>
<tr><td>320</td><td>1.57e-12</td><td>5.43</td><td>4.11e-12</td><td>319</td><td>5.32</td><td>7066</td><td>1.74</td></tr>
<tr><td colspan="8">FE Sweeping, $\gamma = 1.0$</td></tr>
<tr><td>N</td><td>L_1 error</td><td>L_1 order</td><td>L_∞ error</td><td>L_∞ index i</td><td>L_∞ order</td><td>iter #</td><td>CPU time</td></tr>
<tr><td>10</td><td>1.60e-4</td><td></td><td>3.44e-4</td><td>6</td><td></td><td>226</td><td>1.75e-3</td></tr>
<tr><td>20</td><td>3.99e-6</td><td>5.32</td><td>8.04e-6</td><td>14</td><td>5.42</td><td>127</td><td>1.87e-3</td></tr>
<tr><td>40</td><td>1.02e-7</td><td>5.29</td><td>2.16e-7</td><td>32</td><td>5.22</td><td>147</td><td>4.67e-3</td></tr>
<tr><td>80</td><td>2.64e-9</td><td>5.27</td><td>5.92e-9</td><td>71</td><td>5.19</td><td>193</td><td>1.12e-2</td></tr>
<tr><td>160</td><td>6.77e-11</td><td>5.28</td><td>1.65e-10</td><td>159</td><td>5.17</td><td>234</td><td>2.70e-2</td></tr>
<tr><td>320</td><td>1.64e-12</td><td>5.37</td><td>4.47e-12</td><td>316</td><td>5.20</td><td>320</td><td>7.44e-2</td></tr>
</table>

Table 3: Example 1. Accuracy, the grid point where maximum error occurs (L_∞ index i), iteration numbers and CPU times of the FE sweeping scheme with different monotone fluxes. The original smoothness indicators (6) are used in WENO5. CPU time unit: second. $\gamma = 0.9$.

Lax-Friedrichs flux

N	L_1 error	L_1 order	L_∞ error	L_∞ index i	L_∞ order	iter #	CPU time
10	5.66e-6		1.21e-5	9		132	0.001
20	1.98e-7	4.84	4.80e-7	19	4.66	173	0.003
40	6.43e-9	4.94	1.63e-8	39	4.88	199	0.006
80	2.04e-10	4.98	5.25e-10	79	4.95	241	0.015
160	6.34e-12	5.01	1.65e-11	159	4.99	363	0.046
320	1.91e-13	5.06	8.79e-13	319	4.23	564	0.14

Godunov flux

N	L_1 error	L_1 order	L_∞ error	L_∞ index i	L_∞ order	iter #	CPU time
10	4.15e-6		9.46e-6	9		152	0.001
20	1.42e-7	4.87	3.65e-7	19	4.69	195	0.003
40	4.58e-9	4.95	1.22e-8	39	4.91	231	0.007
80	1.45e-10	4.98	3.90e-10	79	4.97	287	0.019
160	4.49e-12	5.01	1.22e-11	159	5.00	403	0.05
320	1.34e-13	5.07	5.76e-13	315	4.40	642	0.16

Engquist-Osher flux

N	L_1 error	L_1 order	L_∞ error	L_∞ index i	L_∞ order	iter #	CPU time
10	4.15e-6		9.46e-6	9		152	0.001
20	1.42e-7	4.87	3.65e-7	19	4.69	195	0.003
40	4.58e-9	4.95	1.22e-8	39	4.91	231	0.007
80	1.45e-10	4.98	3.90e-10	79	4.97	287	0.018
160	4.49e-12	5.01	1.22e-11	159	5.00	403	0.049
320	1.34e-13	5.07	5.76e-13	315	4.40	642	0.16

Table 4: Example 1. Accuracy, the grid point where maximum error occurs (L_∞ index i), iteration numbers and CPU times of the FE sweeping scheme with different monotone fluxes. The new smoothness indicators (9) are used in WENO5. CPU time unit: second. $\gamma = 0.9$.

<table>
<tr><td colspan="8">Lax-Friedrichs flux</td></tr>
<tr><td>N</td><td>L_1 error</td><td>L_1 order</td><td>L_∞ error</td><td>L_∞ index i</td><td>L_∞ order</td><td>iter #</td><td>CPU time</td></tr>
<tr><td>10</td><td>6.21e-6</td><td></td><td>1.38e-5</td><td>9</td><td></td><td>130</td><td>0.001</td></tr>
<tr><td>20</td><td>2.01e-7</td><td>4.95</td><td>4.89e-7</td><td>19</td><td>4.82</td><td>176</td><td>0.003</td></tr>
<tr><td>40</td><td>6.47e-9</td><td>4.96</td><td>1.63e-8</td><td>39</td><td>4.90</td><td>193</td><td>0.006</td></tr>
<tr><td>80</td><td>2.05e-10</td><td>4.98</td><td>5.27e-10</td><td>79</td><td>4.95</td><td>301</td><td>0.018</td></tr>
<tr><td>160</td><td>6.43e-12</td><td>4.99</td><td>1.66e-11</td><td>159</td><td>4.99</td><td>362</td><td>0.042</td></tr>
<tr><td>320</td><td>1.99e-13</td><td>5.01</td><td>6.79e-13</td><td>315</td><td>4.61</td><td>572</td><td>0.14</td></tr>
<tr><td colspan="8">Godunov flux</td></tr>
<tr><td>N</td><td>L_1 error</td><td>L_1 order</td><td>L_∞ error</td><td>L_∞ index i</td><td>L_∞ order</td><td>iter #</td><td>CPU time</td></tr>
<tr><td>10</td><td>4.62e-6</td><td></td><td>1.08e-5</td><td>9</td><td></td><td>136</td><td>0.001</td></tr>
<tr><td>20</td><td>1.44e-7</td><td>5.00</td><td>3.72e-7</td><td>19</td><td>4.86</td><td>205</td><td>0.003</td></tr>
<tr><td>40</td><td>4.61e-9</td><td>4.97</td><td>1.22e-8</td><td>39</td><td>4.93</td><td>241</td><td>0.007</td></tr>
<tr><td>80</td><td>1.45e-10</td><td>4.99</td><td>3.91e-10</td><td>79</td><td>4.97</td><td>307</td><td>0.018</td></tr>
<tr><td>160</td><td>4.55e-12</td><td>5.00</td><td>1.24e-11</td><td>157</td><td>4.98</td><td>585</td><td>0.067</td></tr>
<tr><td>320</td><td>1.40e-13</td><td>5.02</td><td>5.58e-13</td><td>315</td><td>4.47</td><td>634</td><td>0.15</td></tr>
<tr><td colspan="8">Engquist-Osher flux</td></tr>
<tr><td>N</td><td>L_1 error</td><td>L_1 order</td><td>L_∞ error</td><td>L_∞ index i</td><td>L_∞ order</td><td>iter #</td><td>CPU time</td></tr>
<tr><td>10</td><td>4.62e-6</td><td></td><td>1.08e-5</td><td>9</td><td></td><td>136</td><td>0.001</td></tr>
<tr><td>20</td><td>1.44e-7</td><td>5.00</td><td>3.72e-7</td><td>19</td><td>4.86</td><td>205</td><td>0.003</td></tr>
<tr><td>40</td><td>4.61e-9</td><td>4.97</td><td>1.22e-8</td><td>39</td><td>4.93</td><td>241</td><td>0.007</td></tr>
<tr><td>80</td><td>1.45e-10</td><td>4.99</td><td>3.91e-10</td><td>79</td><td>4.97</td><td>307</td><td>0.018</td></tr>
<tr><td>160</td><td>4.55e-12</td><td>5.00</td><td>1.24e-11</td><td>157</td><td>4.98</td><td>585</td><td>0.067</td></tr>
<tr><td>320</td><td>1.40e-13</td><td>5.02</td><td>5.58e-13</td><td>315</td><td>4.47</td><td>634</td><td>0.15</td></tr>
</table>

Table 5: Example 2. Accuracy, the grid point where maximum error occurs (L_∞ index i), iteration numbers and CPU times of four different iterative schemes. The original smoothness indicators (6) are used in U1WENO5. CPU time unit: second.

RK Jacobi, $\gamma = 1.0$							
N	L_1 error	L_1 order	L_∞ error	L_∞ index i	L_∞ order	iter #	CPU time
80	2.57e-6	-	1.39e-5	38	-	846	0.17
160	3.85e-8	6.06	2.43e-7	82	5.84	1440	0.57
320	6.05e-10	5.99	3.75e-9	158	6.02	2361	1.86
RK Sweeping, $\gamma = 1.0$							
N	L_1 error	L_1 order	L_∞ error	L_∞ index i	L_∞ order	iter #	CPU time
80	2.57e-6	-	1.39e-5	38	-	303	5.56e-2
160	3.85e-8	6.06	2.43e-7	78	5.84	516	0.19
320	6.05e-10	5.99	3.75e-9	162	6.02	1071	0.79
FE Jacobi, $\gamma = 0.1$							
N	L_1 error	L_1 order	L_∞ error	L_∞ index i	L_∞ order	iter #	CPU time
80	2.57e-6	-	1.39e-5	42	-	3054	0.57
160	3.85e-8	6.06	2.43e-7	82	5.84	4818	1.79
320	-	-	-	-	-	not conv	-
FE Sweeping, $\gamma = 1.0$							
N	L_1 error	L_1 order	L_∞ error	L_∞ index i	L_∞ order	iter #	CPU time
80	2.57e-6	-	1.39e-5	42	-	175	3.15e-2
160	3.85e-8	6.06	2.43e-7	82	5.84	272	0.10
320	6.05e-10	5.99	3.75e-9	162	6.02	460	0.34

near the middle of the domain where the extrema point is. About convergence of these iterative schemes, similar to Example 1, the direct forward Euler scheme with U1WENO5 / U1ZSWENO5 (i.e., the FE Jacobi scheme) needs very small CFL number $\gamma = 0.1$ to achieve the convergence for $N = 80$ and $N = 160$. For $N = 320$, residues of the direct forward Euler scheme with U1WENO5 / U1ZSWENO5 stop at the level of 10^{-9} and fail to reach the convergence criterion $\mathrm{R}es_A < 10^{-12}$. Large number of iterations and the most CPU time among four iterative schemes are needed for the FE Jacobi scheme. With the fast sweeping technique, the direct forward Euler scheme with U1WENO5 / U1ZSWENO5 (i.e., the FE sweeping scheme) converges with much smaller iteration numbers and CPU times, and it turns out to be the most efficient scheme among these four iterative schemes. A much larger CFL number $\gamma = 1.0$ can be used and the scheme converges for the most refined mesh $N = 320$ without any difficulty. Specifically, CPU times of the FE sweeping scheme is about 5% of that of the FE Jacobi scheme, 18% of that of the RK Jacobi scheme, and around 50% of that of the RK sweeping scheme.

Table 6: Example 2. Accuracy, the grid point where maximum error occurs (L_∞ index i), iteration numbers and CPU times of four different iterative schemes. The new smoothness indicators (9) are used in U1ZSWENO5. CPU time unit: second.

<table>
<tr><td colspan="8">RK Jacobi, $\gamma = 1.0$</td></tr>
<tr><td>N</td><td>L_1 error</td><td>L_1 order</td><td>L_∞ error</td><td>L_∞ index i</td><td>L_∞ order</td><td>iter #</td><td>CPU time</td></tr>
<tr><td>80</td><td>2.48e-6</td><td>-</td><td>2.33e-5</td><td>40</td><td>-</td><td>861</td><td>0.16</td></tr>
<tr><td>160</td><td>3.82e-8</td><td>6.02</td><td>2.60e-7</td><td>80</td><td>6.49</td><td>1449</td><td>0.54</td></tr>
<tr><td>320</td><td>6.00e-10</td><td>5.99</td><td>3.58e-9</td><td>160</td><td>6.18</td><td>2373</td><td>1.76</td></tr>
<tr><td colspan="8">RK Sweeping, $\gamma = 1.0$</td></tr>
<tr><td>N</td><td>L_1 error</td><td>L_1 order</td><td>L_∞ error</td><td>L_∞ index i</td><td>L_∞ order</td><td>iter #</td><td>CPU time</td></tr>
<tr><td>80</td><td>2.48e-6</td><td>-</td><td>2.33e-5</td><td>40</td><td>-</td><td>306</td><td>5.27e-2</td></tr>
<tr><td>160</td><td>3.82e-8</td><td>6.02</td><td>2.60e-7</td><td>80</td><td>6.49</td><td>516</td><td>0.18</td></tr>
<tr><td>320</td><td>6.00e-10</td><td>5.99</td><td>3.58e-9</td><td>160</td><td>6.18</td><td>924</td><td>0.64</td></tr>
<tr><td colspan="8">FE Jacobi, $\gamma = 0.1$</td></tr>
<tr><td>N</td><td>L_1 error</td><td>L_1 order</td><td>L_∞ error</td><td>L_∞ index i</td><td>L_∞ order</td><td>iter #</td><td>CPU time</td></tr>
<tr><td>80</td><td>2.48e-6</td><td>-</td><td>2.33e-5</td><td>40</td><td>-</td><td>2847</td><td>0.50</td></tr>
<tr><td>160</td><td>3.82e-8</td><td>6.02</td><td>2.60e-7</td><td>80</td><td>6.49</td><td>4996</td><td>1.73</td></tr>
<tr><td>320</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>not conv</td><td>-</td></tr>
<tr><td colspan="8">FE Sweeping, $\gamma = 1.0$</td></tr>
<tr><td>N</td><td>L_1 error</td><td>L_1 order</td><td>L_∞ error</td><td>L_∞ index i</td><td>L_∞ order</td><td>iter #</td><td>CPU time</td></tr>
<tr><td>80</td><td>2.48e-6</td><td>-</td><td>2.33e-5</td><td>40</td><td>-</td><td>322</td><td>5.45e-2</td></tr>
<tr><td>160</td><td>3.82e-8</td><td>6.02</td><td>2.60e-7</td><td>80</td><td>6.49</td><td>273</td><td>9.31e-2</td></tr>
<tr><td>320</td><td>6.00e-10</td><td>5.99</td><td>3.58e-9</td><td>160</td><td>6.18</td><td>462</td><td>0.31</td></tr>
</table>

3.3 Example 3. 2D Burgers' equation

In this example we test the iterative schemes for solving the steady state of a two-dimensional problem, the two-dimensional Burgers' equation with a source term

$$u_t + \Big(\frac{1}{\sqrt{2}}\frac{u^2}{2}\Big)_x + \Big(\frac{1}{\sqrt{2}}\frac{u^2}{2}\Big)_y = \sin(\frac{x+y}{\sqrt{2}})\cos(\frac{x+y}{\sqrt{2}}), \qquad (x,y) \in \Big[\frac{\pi}{4\sqrt{2}}, \frac{3\pi}{4\sqrt{2}}\Big] \times \Big[\frac{\pi}{4\sqrt{2}}, \frac{3\pi}{4\sqrt{2}}\Big].$$

The initial condition

$$u(x,y,0) = \beta \sin\Big(\frac{x+y}{\sqrt{2}}\Big)$$

is used as the initial guess in the iterations. Take $\beta = 1.5$ and the exact steady state solution is

$$u(x,y,\infty) = \sin\Big(\frac{x+y}{\sqrt{2}}\Big).$$

For the purpose of testing the schemes, we impose exact steady state solution on boundary points. The results for four different iterative schemes with the original smoothness indicators (6) in the WENO5 are presented in Table 7. The results for these iterative schemes with the new smoothness indicators (9) in the WENO5 are reported in Table 8. Similar as the 1D Burgers' equation, We observe that all schemes achieve similar numerical errors and fifth order accuracy when they converge, and maximum errors generally occur at grid points close to the boundary. Comparing the iterative schemes' efficiency, we obtain the same conclusion as that for the 1D problems. The forward Euler sweeping scheme (the FE sweeping scheme) is the most efficient one among all four iterative methods as shown in Table 7 and Table 8.

3.4 Example 4. One-dimensional steady shock

Now we test the iterative schemes for solving steady state of Euler systems. First we consider a one-dimensional steady shock problem

$$U_t + F(U)_x = 0, \tag{34}$$

where $U = (\rho, \rho u, e)^T$, $F(U) = (\rho u, \rho u^2 + p, u(e+p))^T$. Here ρ, u, e are the density, velocity, and total energy respectively. p is the pressure which is related to the total energy by $e = \frac{p}{\gamma'-1} + \frac{1}{2}\rho u^2$, and the ratio of specific heat $\gamma' = 1.4$.

The computational domain is $x \in [-1, 1]$. It is divided to 400 uniformly spaced mesh points. The initial condition of the flow Mach number at the left of the shock is $M_\infty = 2$. The shock is located at $x = 0$. Periodic boundary conditions are applied. The initial condition used to start the

Table 7: Example 3. Accuracy, the grid point where maximum error occurs (L_∞ index i), iteration numbers and CPU times of four different iterative schemes. The original smoothness indicators (6) are used in WENO5. CPU time unit: second.

RK Jacobi, $\gamma = 1.0$							
$N \times N$	L_1 error	L_1 order	L_∞ error	L_∞ index (i,j)	L_∞ order	iter #	CPU time
10×10	2.46e-6		1.23e-5	(9,9)		270	4.30e-2
20×20	6.31e-8	5.29	3.50e-7	(19,19)	5.14	342	0.23
40×40	1.37e-9	5.52	8.98e-9	(39,39)	5.28	513	1.42
80×80	2.95e-11	5.54	1.93e-10	(79,79)	5.54	855	9.54
RK Sweeping, $\gamma = 1.0$							
$N \times N$	L_1 error	L_1 order	L_∞ error	L_∞ index (i,j)	L_∞ order	iter #	CPU time
10×10	2.46e-6		1.23e-5	(9,9)		159	2.41e-2
20×20	6.31e-8	5.29	3.50e-7	(19,19)	5.14	186	0.12
40×40	1.37e-9	5.52	8.99e-9	(39,39)	5.28	273	0.70
80×80	2.95e-11	5.54	2.07e-10	(79,79)	5.44	450	4.72
FE Jacobi, $\gamma = 0.1$							
$N \times N$	L_1 error	L_1 order	L_∞ error	L_∞ index (i,j)	L_∞ order	iter #	CPU time
10×10	2.46e-6		1.23e-5	(9,9)		1195	0.18
20×20	6.31e-8	5.29	3.50e-7	(19,19)	5.14	1439	0.92
40×40	1.37e-9	5.52	8.99e-9	(39,39)	5.28	1822	3.16
80×80	2.95e-11	5.54	2.08e-10	(79,79)	5.43	3178	33.67
FE Sweeping, $\gamma = 1.0$							
$N \times N$	L_1 error	L_1 order	L_∞ error	L_∞ index (i,j)	L_∞ order	iter #	CPU time
10×10	2.46e-6		1.23e-5	(9,9)		113	1.77e-2
20×20	6.31e-8	5.29	3.50e-7	(19,19)	5.14	133	8.39e-2
40×40	1.37e-9	5.52	8.99e-9	(39,39)	5.28	181	0.47
80×80	2.95e-11	5.54	2.07e-10	(79,79)	5.44	286	3.04

Table 8: Example 3. Accuracy, the grid point where maximum error occurs (L_∞ index i), iteration numbers and CPU times of four different iterative schemes. The new smoothness indicators (9) are used in WENO5. CPU time unit: second.

RK Jacobi, $\gamma = 1.0$							
$N \times N$	L_1 error	L_1 order	L_∞ error	L_∞ index (i,j)	L_∞ order	iter #	CPU time
10×10	4.33e-6		1.41e-5	(9,9)		270	4.04e-2
20×20	9.33e-8	5.54	3.85e-7	(19,19)	5.19	351	0.22
40×40	2.03e-9	5.53	1.08e-8	(39,39)	5.16	528	1.35
80×80	4.30e-11	5.56	2.63e-10	(79,79)	5.36	867	8.88
RK Sweeping, $\gamma = 1.0$							
$N \times N$	L_1 error	L_1 order	L_∞ error	L_∞ index (i,j)	L_∞ order	iter #	CPU time
10×10	4.33e-6		1.41e-5	(9,9)		159	2.28e-2
20×20	9.33e-8	5.54	3.85e-7	(19,19)	5.19	198	0.12
40×40	2.03e-9	5.53	1.08e-8	(39,39)	5.16	279	0.67
80×80	4.30e-11	5.56	2.82e-10	(79,79)	5.26	447	4.41
FE Jacobi, $\gamma = 0.1$							
$N \times N$	L_1 error	L_1 order	L_∞ error	L_∞ index (i,j)	L_∞ order	iter #	CPU time
10×10	4.33e-6		1.41e-5	(9,9)		1166	0.17
20×20	9.33e-8	5.54	3.85e-7	(19,19)	5.19	1409	0.84
40×40	2.03e-9	5.53	1.08e-8	(39,39)	5.16	1937	4.74
80×80	4.30e-11	5.56	2.83e-10	(79,79)	5.25	3209	32.04
FE Sweeping, $\gamma = 1.0$							
$N \times N$	L_1 error	L_1 order	L_∞ error	L_∞ index (i,j)	L_∞ order	iter #	CPU time
10×10	4.33e-6		1.41e-5	(9,9)		110	1.59e-2
20×20	9.33e-8	5.54	3.85e-7	(19,19)	5.19	136	8.00e-2
40×40	2.03e-9	5.53	1.08e-8	(39,39)	5.16	193	0.47
80×80	4.30e-11	5.56	2.82e-10	(79,79)	5.26	289	2.84

iterations is given by the Rankine-Hugoniot jump condition [16] as follows:

$$U(x,0)=\begin{cases}U_l, & \text{if } x<0;\\ U_r, & \text{if } x>0,\end{cases} \tag{35}$$

where $\begin{pmatrix} p_l \\ \rho_l \\ u_l \end{pmatrix} = \begin{pmatrix} \frac{1}{\gamma' M_\infty^2} \\ 1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} p_r \\ \rho_r \\ u_r \end{pmatrix} = \begin{pmatrix} p_l \frac{2\gamma' M_\infty^2-(\gamma'-1)}{\gamma'+1} \\ \frac{\frac{\gamma'+1}{\gamma'-1}\frac{p_r}{p_l}+1}{\frac{\gamma'+1}{\gamma'-1}+\frac{p_r}{p_l}} \\ \sqrt{\gamma' \frac{(2+(\gamma'-1)M_\infty^2)p_r}{(2\gamma' M_\infty^2+(1-\gamma'))\rho_r}} \end{pmatrix}$. The initial condition is also the exact solution of the steady state for this problem. However as before, since the exact steady state solution of the PDE does not satisfy the numerical schemes, we can observe convergence behavior of iterative schemes starting from it. The initial condition will evolve to numerical steady states of the schemes.

This example is used to study convergence improvement of fifth order WENO scheme for solving steady state solutions of Euler equations in [21, 20]. Here our focus is to examine the effect of Gauss-Seidel sweeping technique on reduction of iteration number and computational time required for convergence to steady state. In [21], it is shown that the fifth order WENO schemes with the new smoothness indicator (called “ZSWENO” scheme) can reduce the average residue for this example down to machine zero. Also it is pointed out in [20] that both the new smoothness indicator used in ZSWENO and the technique proposed in [20] with the first order, the second order and WENO upwind-biased interpolation can remove post-shock oscillation completely in this 1D steady shock problem. So here we just use the fifth order WENO schemes with the new smoothness indicator (i.e., the ZSWENO scheme) in the four different iterative schemes to solve this problem.

First, the FE Jacobi iterative scheme (12) is used to solve this example. Different CFL numbers γ are tested. Similar as previous examples, although the fifth order linear scheme with forward Euler is linearly unstable, the nonlinear stable WENO procedure can help in stabilizing the scheme. As a result, the scheme (12) can converge with a small CFL number γ. In this example, numerical tests show that γ needs to be less than or equal to 0.2. For different CFL numbers γ, number of iterations required to achieve convergence (i.e. to satisfy $\mathrm{R}es_A < 10^{-12}$), the final time and total CPU time when convergence is obtained are reported in Table 9. If $\gamma = 0.3$, the residue hangs at $10^{-1.3}$ and does not decrease till the pre-set maximum iteration number $100,000$ is reached. In Figure 1, the residue history in terms of iterations for different CFL numbers using the FE Jacobi scheme with the fifth order ZSWENO is presented in the upper-left subgraph (the picture with title “FE ZSWENO”). The numerical solutions of density ρ when the scheme converges (for $\gamma = 0.1,\ 0.2$) or when the pre-set maximum iteration number is reached (for $\gamma = 0.3$) are presented in the upper-left subgraphs of Figures 2 and 3. We can see that for $\gamma = 0.3$, the numerical solution itself suffers from oscillation both in upstream and downstream of the shock and the residue can not settle down to a low value. There is no oscillation observed for $\gamma = 0.1,\ 0.2$, for which the FE Jacobi iterative scheme converges.

Then we further test the FE sweeping method with the fifth order ZSWENO and different CFL numbers to solve this problem. In the presented pictures, the results by the FE sweeping method with the fifth order ZSWENO have the title “FE FS ZSWENO”. When γ is less than or equal to 1.1, the scheme converges. The residue blows up when $\gamma = 1.2$. Number of iterations required for convergence, the final time and total CPU time when the scheme converges are reported in Table 10 for different CFL numbers. We observe that the iteration number required for convergence is significantly reduced comparing with the FE Jacobi scheme, due to that much larger CFL numbers can be used by using the fast sweeping technique. The total CPU time is largely saved (the FE sweeping needs 3.76 seconds with the largest possible CFL number while the FE Jacobi needs 14.16 seconds). It is also worth noting that for $\gamma = 0.1,\ 0.2$ where both the FE Jacobi scheme and the FE sweeping scheme converge, numbers of iterations required for convergence of the FE sweeping scheme is slightly smaller than that of the FE Jacobi scheme (see Tables 9 and 10). However, CPU time for the FE sweeping scheme is more than that of the FE Jacobi scheme. The reason is that since the newest numerical values are always used whenever available in the Gauss-Seidel procedure, it is needed to calculate $\hat{f}_{i+\frac{1}{2}}$, the numerical flux at $i + \frac{1}{2}$, twice. Namely, one is for updating numerical value at the point i and the other is for that at the point $i + 1$ since the newest numerical values in the stencil for computing $\hat{f}_{i+\frac{1}{2}}$ are different when updating values at i and $i + 1$. This results in extra computational time for the FE sweeping method. Despite of this extra computation, the large CFL number made possible by the FE sweeping method leads to a more efficient scheme, i.e. fewer iterations and less computational time required for convergence than that of the FE Jacobi method. Residue history of the FE sweeping scheme for various CFL numbers is shown in the upper-right subgraph of Figure 1. The numerical solutions of density ρ when the scheme converges are presented in the upper-right subgraphs of Figure 2 and Figure 3. No post-shock oscillation is observed as expected.

It is also interesting to see the computation efficiency effects by using Gauss-Seidel sweeping method on the RK type iterative schemes. The RK Jacobi scheme with the fifth order ZSWENO is given the title “RK ZSWENO” in the presented pictures. The RK sweeping scheme with the fifth order ZSWENO is given the title “RK FS ZSWENO”. We test different CFL numbers. The RK

Table 9: Example 4. 1D steady shock. The FE Jacobi scheme with the fifth order ZSWENO is used. Number of iterations, the final time and total CPU time when convergence is obtained. CPU time unit: second.

γ: CFL number	iteration number	final time	CPU time
0.1	27353	9.12	27.36
0.2	14146	9.43	14.16
0.3	not convergent		

Table 10: Example 4. 1D steady shock. The FE sweeping scheme with the fifth order ZSWENO is used. Number of iterations, the final time and total CPU time when convergence is obtained. CPU time unit: second.

γ: CFL number	iteration number	final time	CPU time
0.1	26344	8.78	44.85
0.2	12270	8.18	20.95
0.3	7958	7.96	13.61
0.5	4992	8.32	8.71
0.8	3054	8.14	5.32
1.0	2426	8.09	4.20
1.1	2162	7.93	3.76
1.2	not convergent		

Table 11: Example 4. 1D steady shock. The RK Jacobi scheme with the fifth order ZSWENO is used. Number of iterations, the final time and total CPU time when convergence is obtained. CPU time unit: second.

γ: CFL number	iteration number	final time	CPU time
0.1	78954	8.77	82.21
0.4	19812	8.81	21.08
1.0	7596	8.44	8.07
1.1	6867	8.39	7.30
1.4	5286	8.22	5.52
1.5	not convergent		

Table 12: Example 4. 1D steady shock. The RK sweeping scheme with the fifth order ZSWENO is used. Number of iterations, the final time and total CPU time when convergence is obtained. CPU time unit: second.

γ: CFL number	iteration number	final time	CPU time
0.1	44964	5.00	76.99
0.4	9480	4.21	16.47
1.0	3882	4.31	6.77
1.1	3504	4.28	6.10
1.2	not convergent		

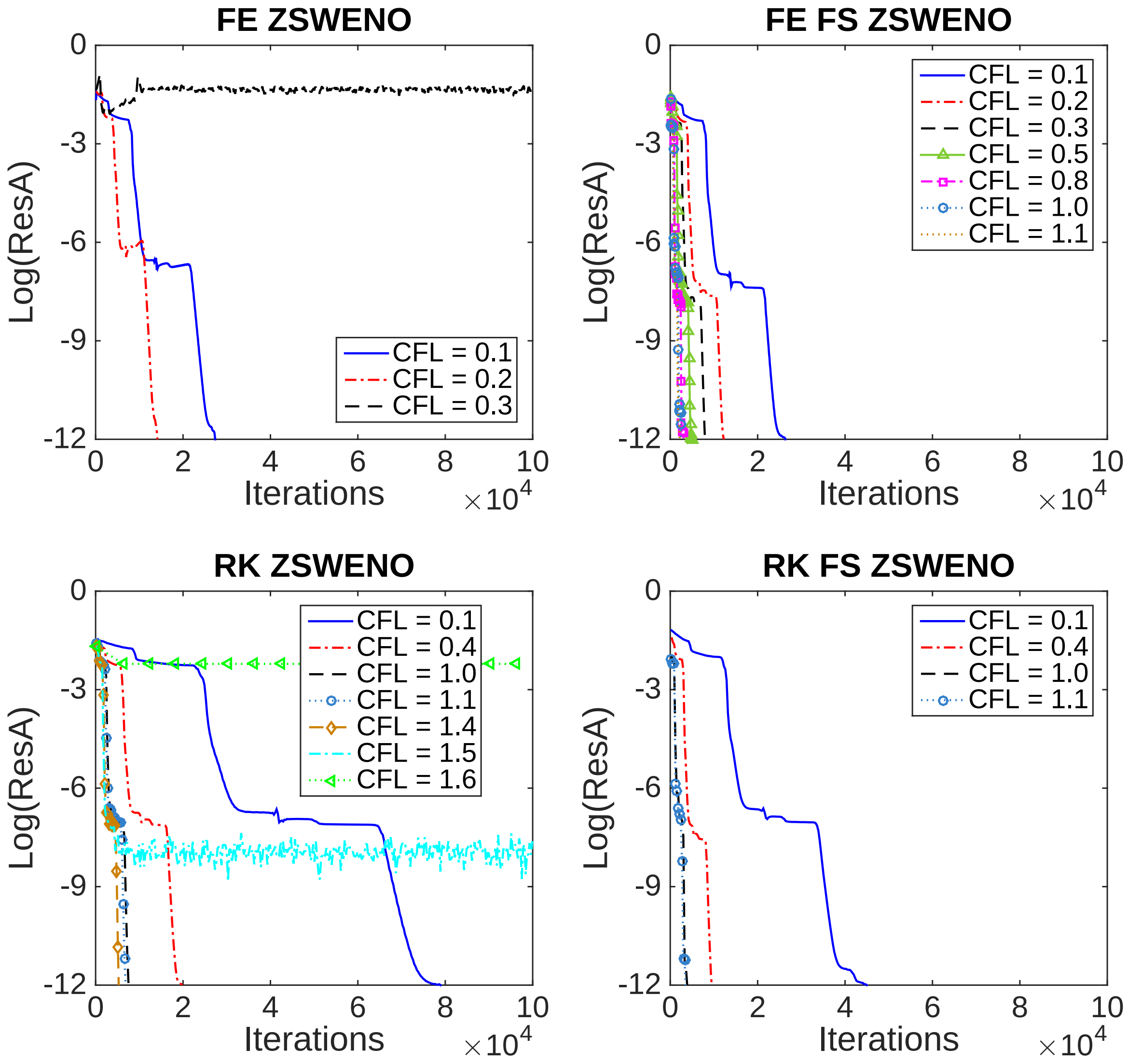

Figure 1: Example 4. 1D steady shock problem. The evolution history of average residue along with iterations of four different iterative schemes with different CFL numbers. Top left: the FE Jacobi scheme; top right: the FE sweeping scheme; bottom left: the RK Jacobi scheme; bottom right: the RK sweeping scheme.

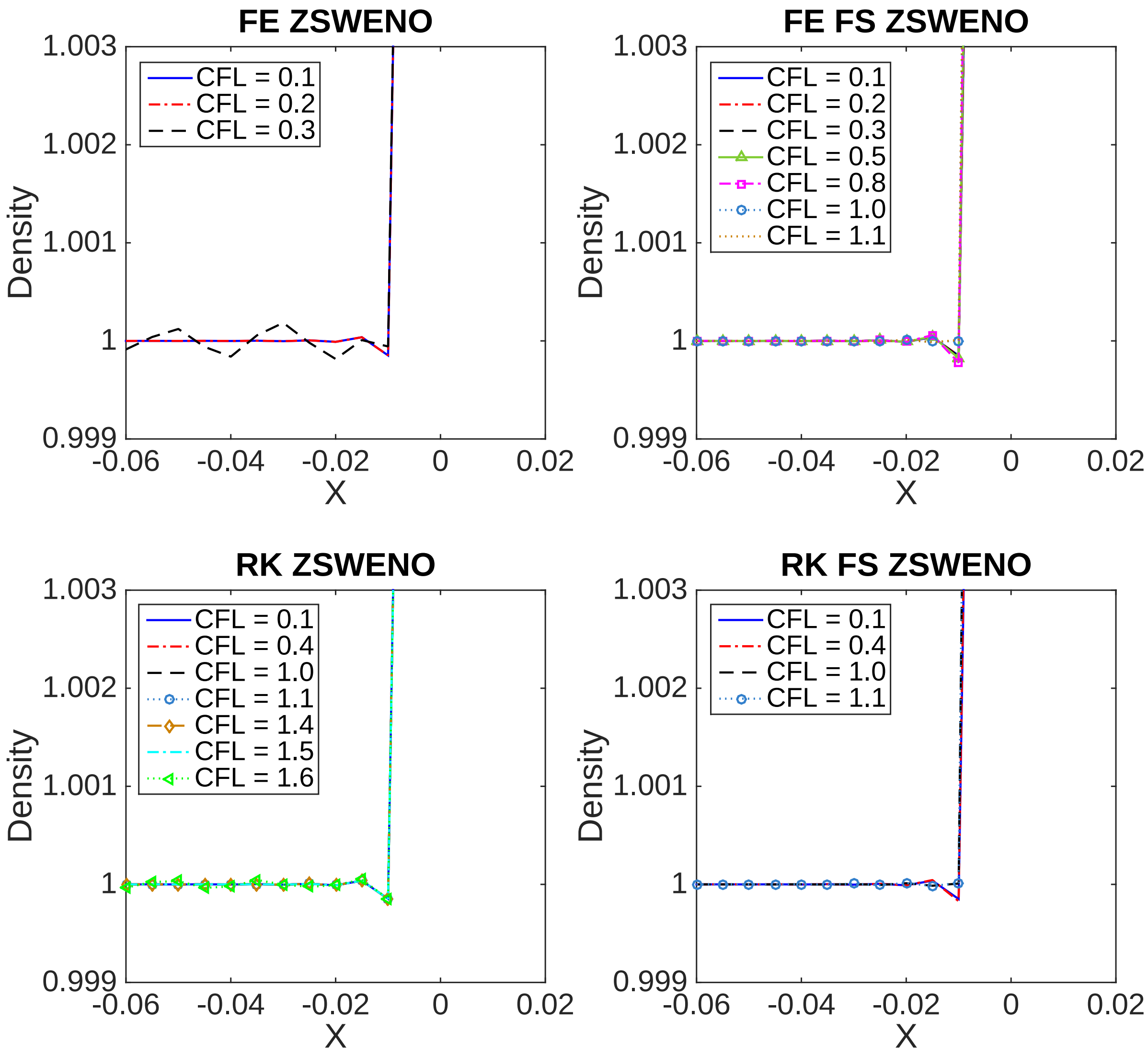

Figure 2: Example 4. 1D steady shock problem. Zoomed density distribution (upstream) of the solutions of four different iterative schemes with different CFL numbers. Top left: the FE Jacobi scheme; top right: the FE sweeping scheme; bottom left: the RK Jacobi scheme; bottom right: the RK sweeping scheme.

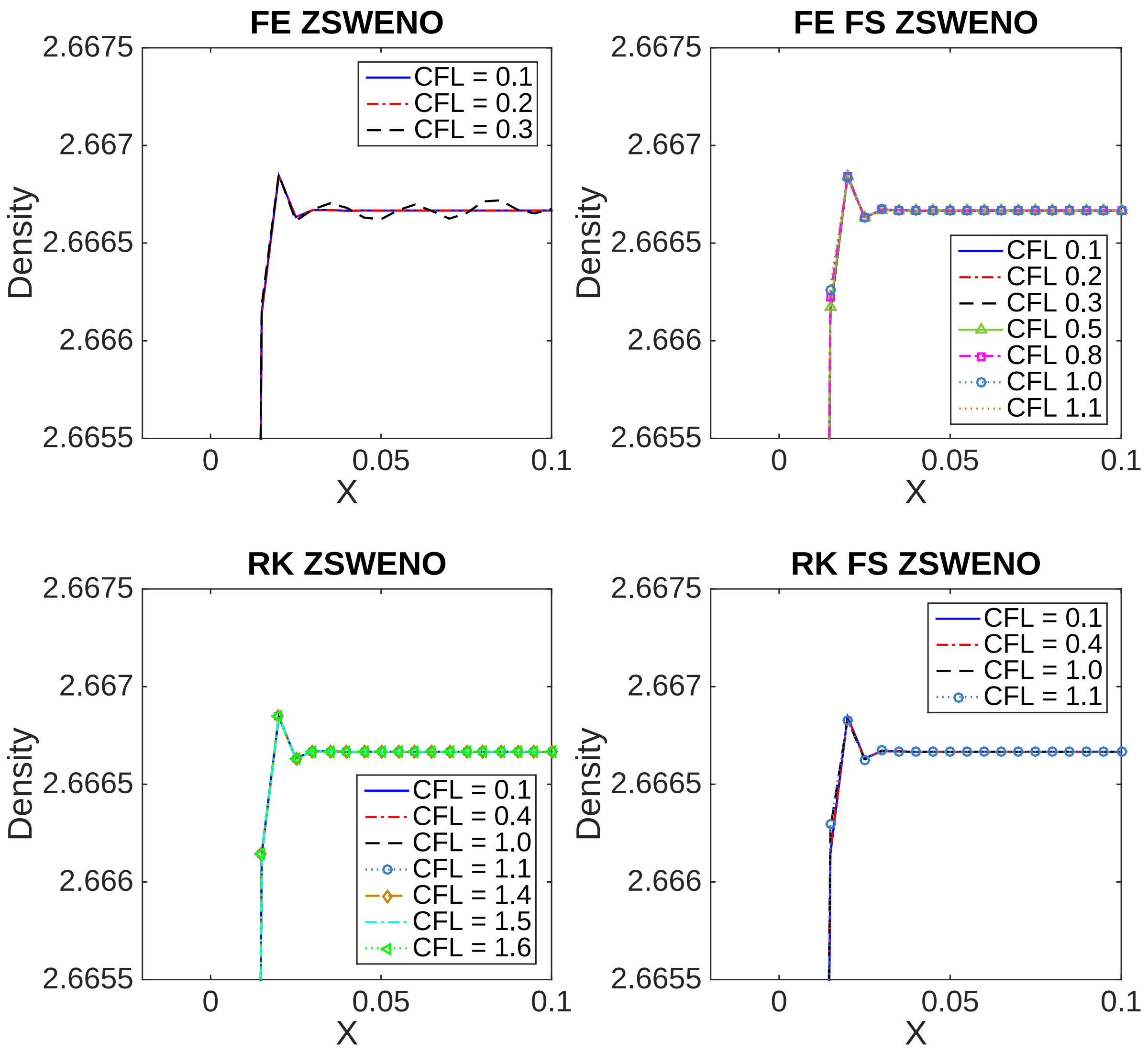

Figure 3: Example 4. 1D steady shock problem. Zoomed density distribution (downstream) of the solutions of four different iterative schemes with different CFL numbers. Top left: the FE Jacobi scheme; top right: the FE sweeping scheme; bottom left: the RK Jacobi scheme; bottom right: the RK sweeping scheme.

Jacobi scheme converges if γ is less than or equal to 1.4, i.e., the convergence criterion $\mathrm{R}es_A < 10^{-12}$ is satisfied. Number of iterations required for convergence, the final time and total CPU time when the scheme converges for different CFL numbers are shown in Table 11. If $\gamma = 1.5$, the residue oscillates between 10^{-7} and 10^{-8} till the pre-set maximum iteration number $100,000$ is reached. The residue hangs at $10^{-2.2}$ for $\gamma = 1.6$. Residue history of the RK Jacobi scheme for various CFL numbers is shown in the lower-left subgraph of Figure 1. The numerical solutions of density ρ when the scheme converges (for $\gamma = 0.1,\ 0.4,\ 1.0,\ 1.1,\ 1.4$) or when the pre-set maximum iteration number is reached (for $\gamma = 1.5,\ 1.6$) are presented in the lower-left subgraphs of Figure 2 and Figure 3. We can see that for $\gamma = 1.5,\ 1.6$ even the residues cannot settle down to a low level, the numerical solutions are acceptable for this example, although it is not the general case. Then we further test different CFL numbers for the RK sweeping scheme with the fifth order ZSWENO. If γ is less than or equal to 1.1, the RK sweeping scheme converges. The residue blows up when $\gamma = 1.2$. Number of iterations required for convergence, the final time and total CPU time when the scheme converges for different CFL numbers are reported in Table 12. Again, with the Gauss-Seidel sweeping technique, we observe that the iteration number required for convergence is significantly reduced, around by half for the same γ. However, due to the same reason discussed earlier that $\hat{f}_{i+\frac{1}{2}}$, the numerical flux at $i + \frac{1}{2}$ needs to be calculated twice in Gauss-Seidel procedure for the RK sweeping method, although the iteration number is reduced to about half of that of the RK Jacobi method, the computational cost is saved with an amount less than half for the same CFL number. Residue history of the RK sweeping scheme for various CFL numbers is shown in the lower-right subgraph of Figure 1. The numerical solutions of density ρ (for $\gamma = 0.1,\ 0.4,\ 1.0,\ 1.1$) are presented in the lower-right subgraphs of Figure 2 and Figure 3.

For this one-dimensional steady shock problem, we draw the same conclusion as previous examples. The FE sweeping method is the most efficient approach for fifth order WENO computation of the steady state problem among the four methods discussed here, in terms of both iteration number and CPU time. This is further verified by the following two dimensional simulations of Euler systems.

3.5 Example 5. A two-dimensional oblique steady shock

In this subsection, we use these four iterative methods to simulate a two-dimensional oblique steady shock problem, which is also tested in [21] and [20]. The shock has an angle of 135° with the positive x-direction. The flow Mach number at the left of the shock is $M_\infty = 2$. The computational domain is $0 \leq x \leq 4$ and $0 \leq y \leq 2$. The initial oblique shock passes the point $(3, 0)$. The domain is divided into 200×100 equally spaced points with $\Delta x = \Delta y$. With periodic boundary condition along the shock direction implemented, the residue of the first order upwind biased interpolation fifth order WENO scheme (U1WENO) can settle down to 10^{-12} as that shown in [20]. U1WENO is also shown as the most efficient scheme among those to offer the best results for this example in [20]. So here we use the fifth order U1WENO as our WENO scheme for this example to study the effect of introducing Gauss-Seidel sweeping method on the reduction of iteration number and computational time. Convergence criterion is set to the same value as before, i.e., 10^{-12}.

First, the FE Jacobi scheme with the fifth order U1WENO scheme (entitled "FE U1WENO" in the presented pictures) is used to solve this example. Different CFL numbers γ are tested. Similar to the one-dimensional steady shock example, the numerical tests show that the scheme only converges when γ is less than or equal to 0.1. Number of iterations required for convergence (i.e. $\mathrm{R}es_A < 10^{-12}$), the final time and total CPU time when the scheme converges are reported in Table 13 for different CFL numbers. When $\gamma = 0.2$, the residue hangs at $10^{-3.1}$ till the pre-set maximum iteration number $100,000$ is reached. For $\gamma = 0.3$, the residue hangs at an even higher level. In Figure 4, the residue history in terms of iterations for different CFL numbers using the FE Jacobi scheme with the fifth order U1WENO is presented in the upper-left subgraph. The numerical density distribution near the shock along the horizontal line $y = 1$ when the scheme converges (for $\gamma = 0.1$) or when the pre-set maximum iteration number is reached (for $\gamma = 0.2,\ 0.3$) are presented in the upper-left subgraphs of Figure 5 and Figure 6. We can see that for $\gamma = 0.2$, even the residue cannot settle down to 10^{-12}, the numerical density distribution is acceptable. However for $\gamma = 0.3$, the numerical solution has huge post-shock oscillations.

Then we test the FE sweeping scheme with the fifth order U1WENO (entitled "FE FS U1WENO" in the presented pictures). If γ is less than or equal to 1.0, the scheme converges. The residue blows up when $\gamma = 1.2$. Number of iterations required for convergence, the final time and total CPU time when the scheme converges are reported in Table 14 for different CFL numbers. Similar to the observation from the one-dimensional steady shock, the iteration number required for convergence is significantly reduced comparing with the FE Jacobi scheme, due to that much larger CFL numbers can be used by using the fast sweeping technique. The total CPU time is largely saved. The FE sweeping scheme only needs 616 seconds CPU time with the largest possible CFL number while the FE Jacobi scheme needs 5510 seconds CPU time to achieve convergence. Residue history in terms of iterations for the FE sweeping scheme with various CFL numbers is shown in the upper-right subgraph of Figure 4. The numerical density distribution near the shock along the horizontal line $y = 1$ when the scheme converges are presented in the upper-right subgraphs of Figure 5 and Figure 6.

We further examine the computational efficiency effects of the Gauss-Seidel sweeping method on the RK iterative schemes. The RK Jacobi scheme with the fifth order U1WENO is given the title "RK U1WENO" in the presented pictures. The RK sweeping scheme with the fifth order U1WENO is given the title "RK FS U1WENO". We test different CFL numbers. Number of iterations required for convergence, the final time and total CPU time when the scheme converges are reported in Table 15 for the RK Jacobi scheme and Table 16 for the RK sweeping scheme. Both schemes converge if γ is less than or equal to 1.0. The residue of the RK Jacobi scheme hangs around $10^{-2.2}$ if $\gamma = 1.2$, and around $10^{-1.9}$ if $\gamma = 1.4$. The residue of the RK sweeping scheme blows up for $\gamma = 1.2$ and $\gamma = 1.4$. From Table 15 and Table 16, for a fixed γ, we can see that the RK sweeping method needs much fewer iterations than the RK Jacobi method to achieve convergence and also less CPU costs. In the bottom pictures of Figure 4, we present the evolution of average residues in terms of iterations for both the RK Jacobi scheme and the RK sweeping scheme. We observe no oscillation in numerical density distribution for both schemes when they converge in the bottom pictures of Figure 5 and Figure 6. However, post-shock oscillation can be observed in Figure 6 for the RK Jacobi scheme with $\gamma = 1.4$, in which case the scheme does not converge.

Overall, comparing the computational costs of all four different iterative schemes for this two-dimensional problem, we conclude that the FE sweeping scheme with the fifth order U1WENO is still the most efficient one in terms of both iteration number and CPU time. This is consistent with the conclusion obtained in the previous examples.

3.6 Example 6. Regular shock reflection

In this subsection, the regular shock reflection problem is used to test the iterative schemes. This problem is a typical benchmark problem of two dimensional steady flow. The computational domain is chosen to be $[0, 4.128] \times [0, 1]$ such that the impinging shock wave and the reflected shock wave pass through two corners of the top boundary, see [20]. The computational grid is 200×50. This example is a special and difficult problem since even the techniques proposed in [21, 20] can not make average residue settle down to machine zero. The first order upwind biased interpolation fifth order WENO scheme with new smoothness indicator (U1ZSWENO) is the best converged

Table 13: Example 5. 135° Oblique steady shock wave. The FE Jacobi scheme with the fifth order U1WENO is used. Number of iterations, the final time and total CPU time when convergence is obtained. CPU time unit: second.

γ: CFL number	iteration number	final time	CPU time
0.1	23729	22.26	5510
0.2	not convergent		

Table 14: Example 5. 135° Oblique steady shock wave. The FE sweeping scheme with the fifth order U1WENO is used. Number of iterations, the final time and total CPU time when convergence is obtained. CPU time unit: second.

γ: CFL number	iteration number	final time	CPU time
0.1	27017	25.36	8569
0.2	15749	29.56	4960
0.4	7689	28.87	2428
0.6	4317	24.31	1357
0.8	3137	23.56	988
1.0	1953	18.32	616
1.2	not convergent		

Table 15: Example 5. 135° Oblique steady shock wave. The RK Jacobi scheme with the fifth order U1WENO is used. Number of iterations, the final time and total CPU time when convergence is obtained. CPU time unit: second.

γ: CFL number	iteration number	final time	CPU time
0.1	95463	29.87	22152
0.2	46734	29.24	10907
0.4	22788	28.52	5309
1.0	9507	29.74	2213
1.2	not convergent		
1.4	not convergent		

Table 16: Example 5. 135° Oblique steady shock wave. The RK sweeping scheme with the fifth order U1WENO is used. Number of iterations, the final time and total CPU time when convergence is obtained. CPU time unit: second.

γ: CFL number	iteration number	final time	CPU time
0.1	43755	13.69	13655
0.2	25059	15.68	7826
0.4	12123	15.17	3812
1.0	3027	9.47	951
1.2	not convergent		
1.4	not convergent		

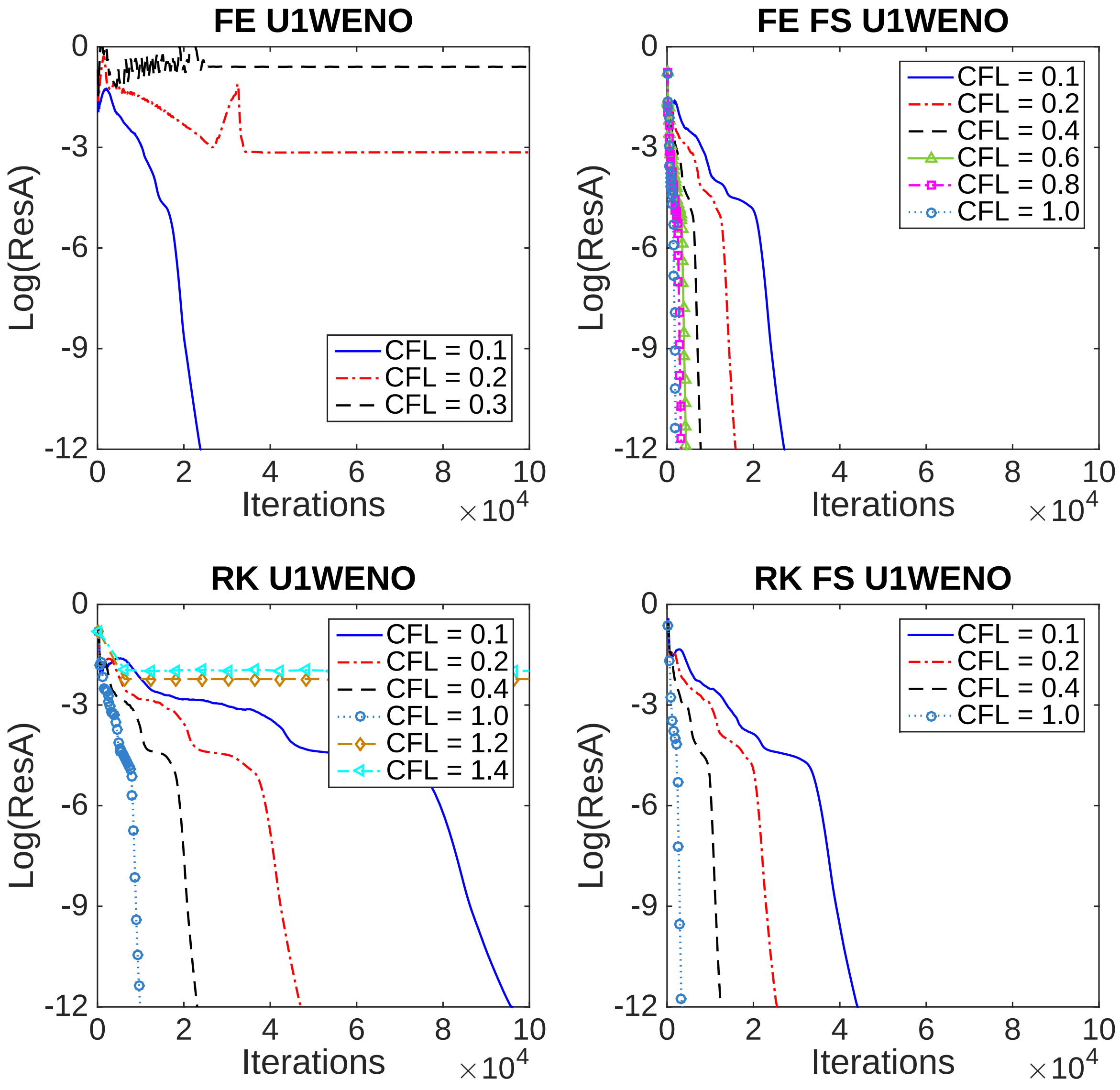

Figure 4: Example 5. 135° oblique shock of $M_\infty = 2$ problem. The evolution history of average residue along with iterations of four different iterative schemes with different CFL numbers. Top left: the FE Jacobi scheme; top right: the FE sweeping scheme; bottom left: the RK Jacobi scheme; bottom right: the RK sweeping scheme.

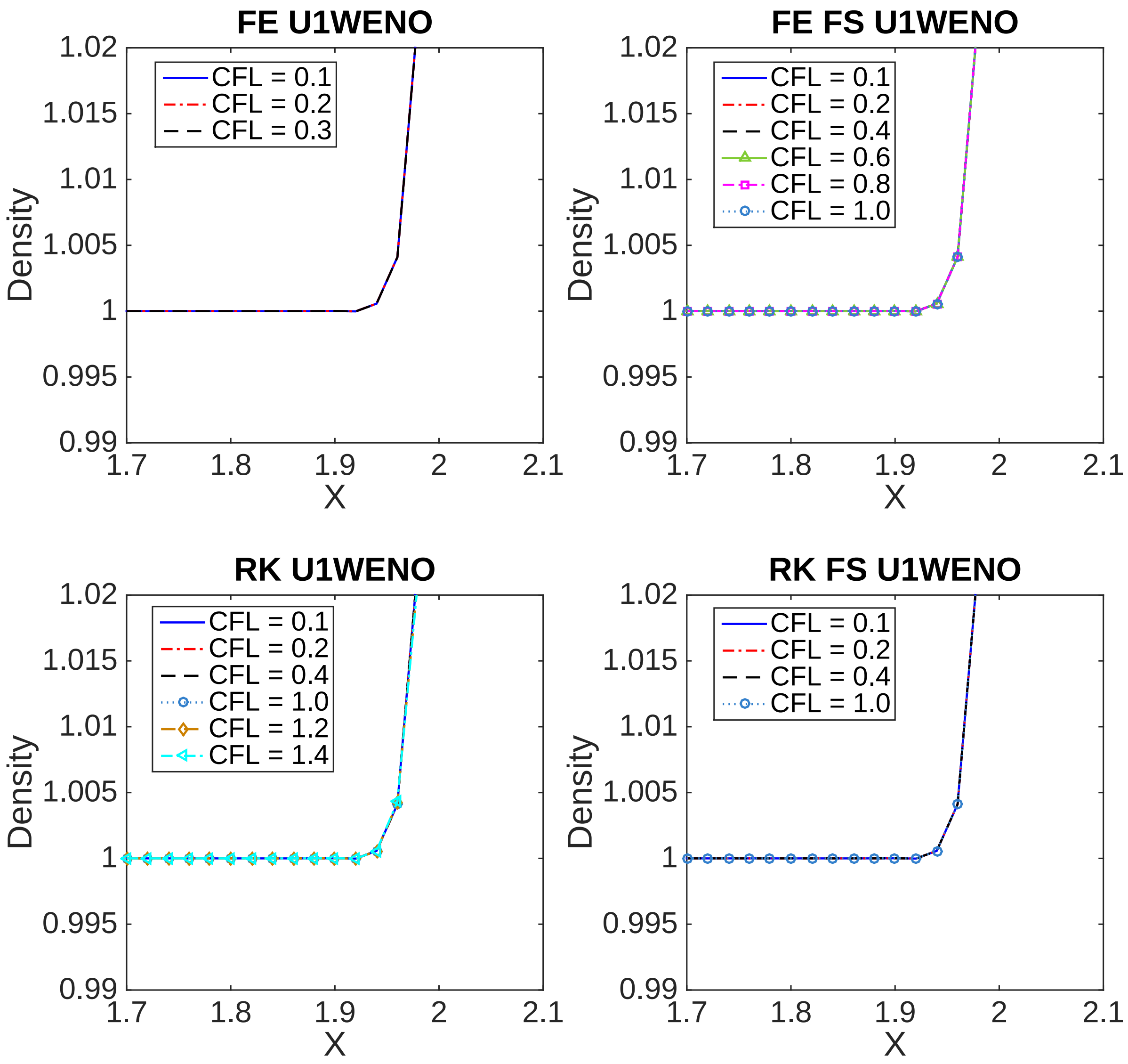

Figure 5: Example 5. 135° oblique shock of $M_\infty = 2$ problem. Zoomed density distribution (upstream) of the solutions along the line $y = 1$ by four different iterative schemes with different CFL numbers. Top left: the FE Jacobi scheme; top right: the FE sweeping scheme; bottom left: the RK Jacobi scheme; bottom right: the RK sweeping scheme.

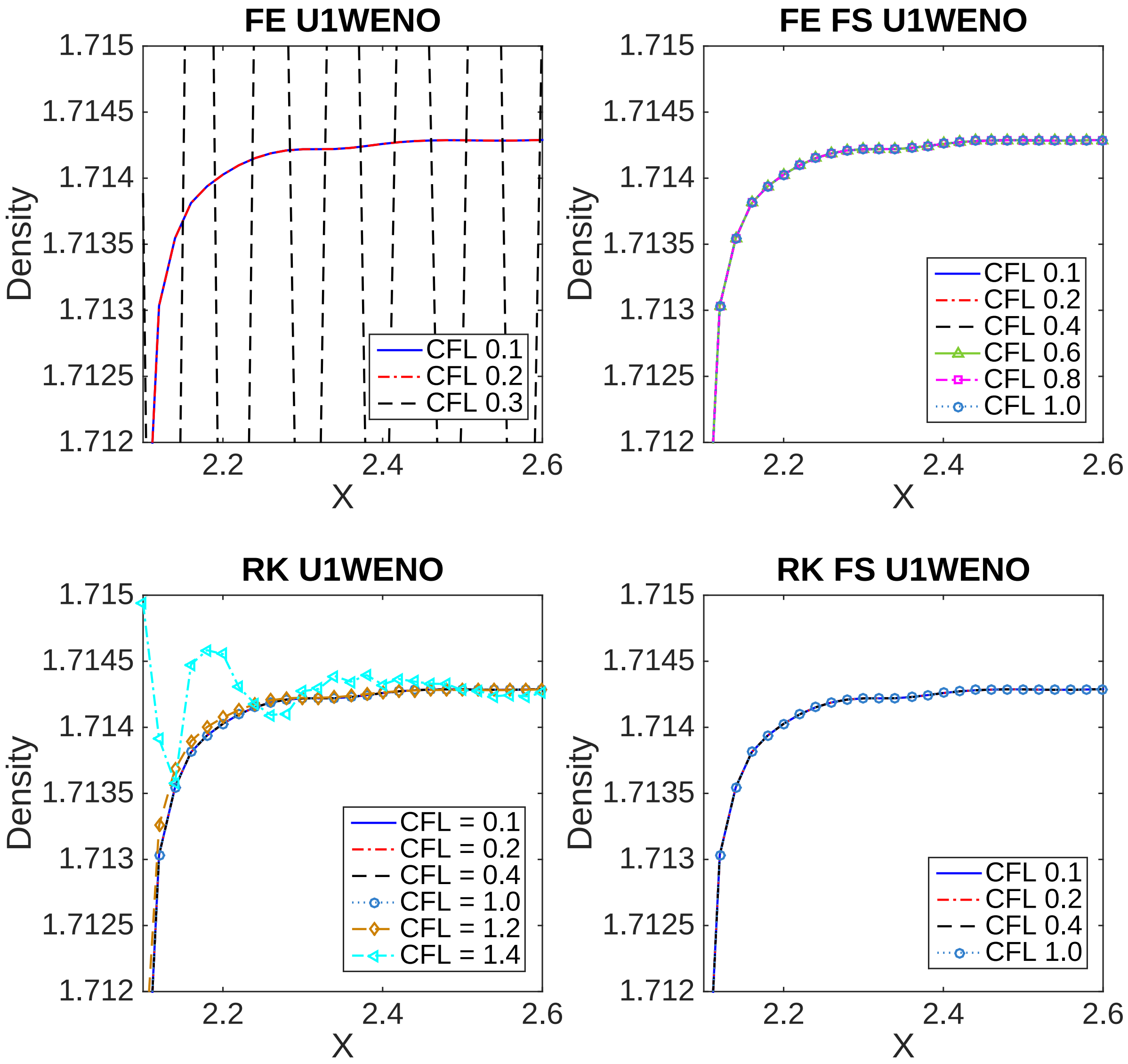

Figure 6: Example 5. 135° oblique shock of $M_\infty = 2$ problem. Zoomed density distribution (downstream) of the solutions along the line $y = 1$ by four different iterative schemes with different CFL numbers. Top left: the FE Jacobi scheme; top right: the FE sweeping scheme; bottom left: the RK Jacobi scheme; bottom right: the RK sweeping scheme.

scheme for this example as that shown in [20], giving an average residue of $10^{-4.4}$ when applying characteristic boundary treatment. So here we use the U1ZSWENO scheme as our base WENO scheme in the four iterative schemes to study the computational efficiency improvement by the Gauss-Seidel sweeping method.

We observe that when characteristic boundary condition is applied as in [20], the average residue of the FE sweeping scheme with the fifth order U1ZSWENO (entitled "FE FS U1ZSWENO" in the presented pictures) cannot be driven down to $10^{-4.4}$, the level achieved by the U1ZSWENO scheme with the original Runge-Kutta time marching in [20] (i.e., the RK Jacobi scheme, entitled "RK U1ZSWENO" in the presented pictures). Such inconsistency makes it difficult to compare the computation efficiency, i.e. iteration number and CPU time required for convergence, for the FE sweeping scheme and the RK Jacobi scheme because their residues settle down to different levels. So here rather than the characteristic boundary condition, we apply exact values obtained by Rankine-Hugoniot condition on the left, the right, and the top boundaries, see [21]. Reflective boundary condition is applied on the bottom boundary. Using this boundary treatment, residues of both schemes can reach $10^{-3.5}$. Even though it is greater than $10^{-4.4}$, the consistency makes it convenient for us to compare computation efficiency of the FE sweeping scheme (the most efficient one in all previous examples) and the classical time marching method used in [21, 20]. Hence for this example, we choose $10^{-3.5}$ as the convergence criterion.

In Table 17, Table 18 and Table 19, number of iterations required for convergence, the final time and total CPU time when the scheme converges are reported for the FE Jacobi scheme with the fifth order U1ZSWENO (entitled "FE U1ZSWENO" in the presented pictures), the FE sweeping scheme with the fifth order U1ZSWENO, and the RK Jacobi scheme with the fifth order U1ZSWENO respectively. The FE Jacobi scheme needs to use a small CFL number $\gamma = 0.1$ to achieve convergence. The residue hangs at $10^{-2.4}$ if $\gamma = 0.2$. With the help of fast sweeping technique, the FE sweeping scheme can achieve much larger CFL numbers. From numerical results, we see that the CFL numbers of the FE sweeping scheme can be as large as those for the classical RK scheme. With $\gamma = 0.1$ to 0.4 both schemes can reach the convergence criterion $10^{-3.5}$. It is interesting to observe that for $\gamma = 0.5$, the residue of the FE sweeping scheme finally reaches $10^{-3.5}$ after an unusually many iterations. It is further verified by the upper-right subgraph of Figure 7, which shows that the residue hangs at $10^{-3.4}$ for a while before it finally settles down to $10^{-3.5}$. On the other hand, the residue for the RK Jacobi scheme with $\gamma = 0.5$ always hangs at $10^{-3.4}$ till the pre-set maximum iteration number is reached. This indicates that $\gamma = 0.5$ is on the boundary of

CFL numbers to reach the convergence criterion $10^{-3.5}$. If $\gamma = 0.8$, the residue of the FE sweeping scheme hangs at $10^{-2.3}$. For the RK Jacobi scheme, it hangs at $10^{-2.5}$ if $\gamma = 0.8$. For this problem, the average residues of the RK sweeping scheme hang at $10^{-3.3}$ even for very small CFL number $\gamma = 0.1$ and cannot meet the convergence criterion $10^{-3.5}$ till the pre-set maximum iteration number (30,000 for this example) is reached. The details can be observed in Figure 7, in which we present the evolution of average residues along with iterations for these four iterative schemes. Numerical zoomed density distributions are shown in Figure 8. 2D contour plots of numerical solutions are presented in Figure 9. Again, for this example, the plots of numerical solutions are acceptable even for the cases that the average residues are not able to reach a low level. Overall no post-shock oscillation is observed. From the presented data, we draw the same conclusion as the previous examples, i.e., the FE sweeping scheme has the least CPU cost (316 seconds for the CFL number $\gamma = 0.4$) and is the most efficient scheme to compute steady state for this shock reflection problem.

4 Concluding remarks

To compute steady state of hyperbolic conservation laws, the forward Euler time marching is preferred since only one stage and one step is used, as time direction accuracy has no effects on the numerical accuracy of steady state solutions. However, a higher order spatial scheme (e.g., a fifth order scheme) with the forward Euler time marching is linearly unstable. Nonlinear stable schemes such as WENO schemes can help to stabilize the computation by the forward Euler but it requires a very small CFL number to converge to steady state, as shown in this paper. Hence the computation is very inefficient. In this paper, based on fifth order WENO schemes which improve the convergence of the classical WENO schemes by removing slight post-shock oscillations, we design fifth order fixed-point sweeping WENO methods for steady state of hyperbolic conservation laws. It is discovered that the fast sweeping technique can largely improve the stability of high order spatial scheme with the forward Euler time marching. Extensive numerical experiments are performed to compare four different iterative schemes including the regular forward Euler and Runge-Kutta time marching methods, and the ones coupled with fast sweeping technique. All numerical examples show that the forward Euler time discretization with fast sweeping technique is the most efficient approach for fifth order WENO computations of the steady state of hyperbolic conservation laws problems. Our fixed-point iterations are based on time-marching schemes. The advantage is that the computed steady state is stable and carries physical properties of the system and the initial condition. The methods have time-step size constraint by the CFL condition. It is

Table 17: Example 6. Regular shock reflection. The FE Jacobi scheme with the fifth order U1ZSWENO is used. Convergence criterion is $10^{-3.5}$. Number of iterations, the final time and total CPU time when convergence is obtained. CPU time unit: second.

γ: CFL number	iteration number	final time	CPU time
0.1	8303	9.15	970
0.2	not convergent		

Table 18: Example 6. Regular shock reflection. The FE sweeping scheme with the fifth order U1ZSWENO is used. Convergence criterion is $10^{-3.5}$. Number of iterations, the final time and total CPU time when convergence is obtained. CPU time unit: second.

γ: CFL number	iteration number	final time	CPU time
0.1	8249	9.10	1292
0.2	4053	8.94	634
0.3	2661	8.80	415
0.4	2017	8.90	316
0.5	5797	31.99	900
0.8	not convergent		

Table 19: Example 6. Regular shock reflection. The RK Jacobi scheme with the fifth order U1ZSWENO is used. Convergence criterion is $10^{-3.5}$. Number of iterations, the final time and total CPU time when convergence is obtained. CPU time unit: second.

γ: CFL number	iteration number	final time	CPU time
0.1	25068	9.21	2961
0.2	12558	9.23	1475
0.3	8454	9.32	994
0.4	6378	9.38	751
0.5	not convergent		
0.8	not convergent		

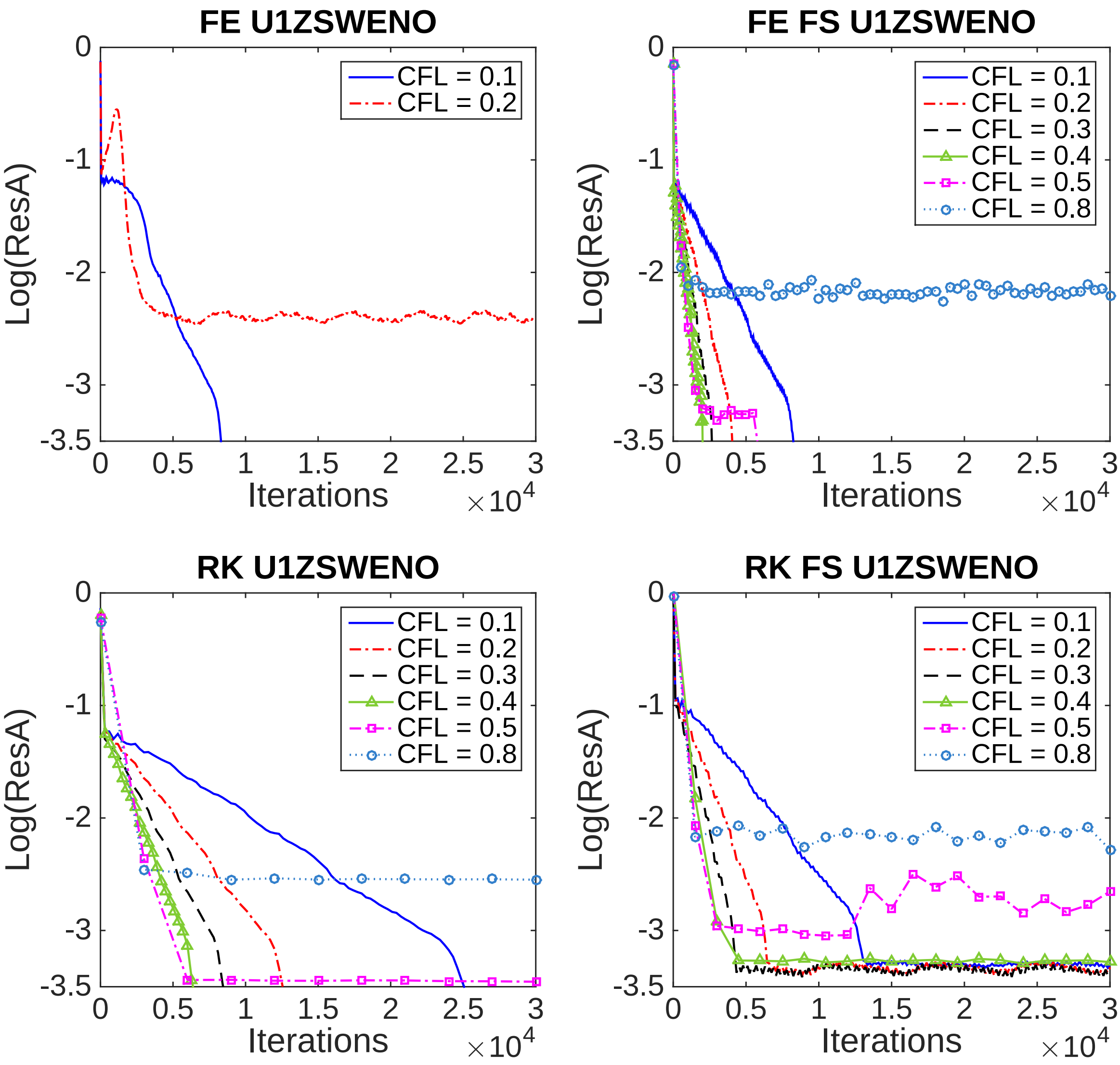

Figure 7: Example 6. Regular shock reflection. The evolution history of average residue along with iterations of four different iterative schemes with different CFL numbers. Top left: the FE Jacobi scheme; top right: the FE sweeping scheme; bottom left: the RK Jacobi scheme; bottom right: the RK sweeping scheme.

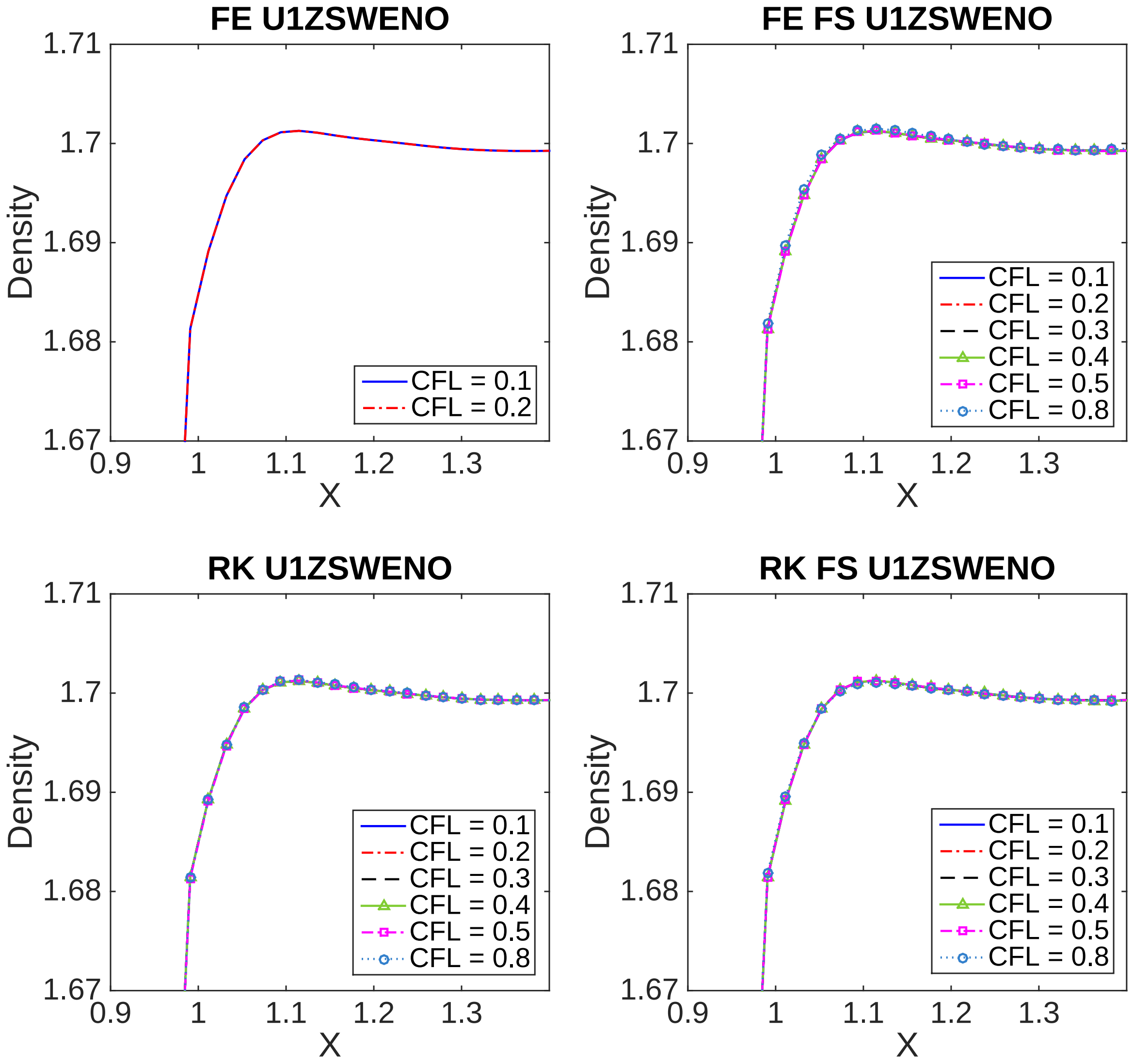

Figure 8: Example 6. Regular shock reflection. Zoomed density distribution near the impinging shock along the line $y = 0.5$ by four different iterative schemes with different CFL numbers. Top left: the FE Jacobi scheme; top right: the FE sweeping scheme; bottom left: the RK Jacobi scheme; bottom right: the RK sweeping scheme.

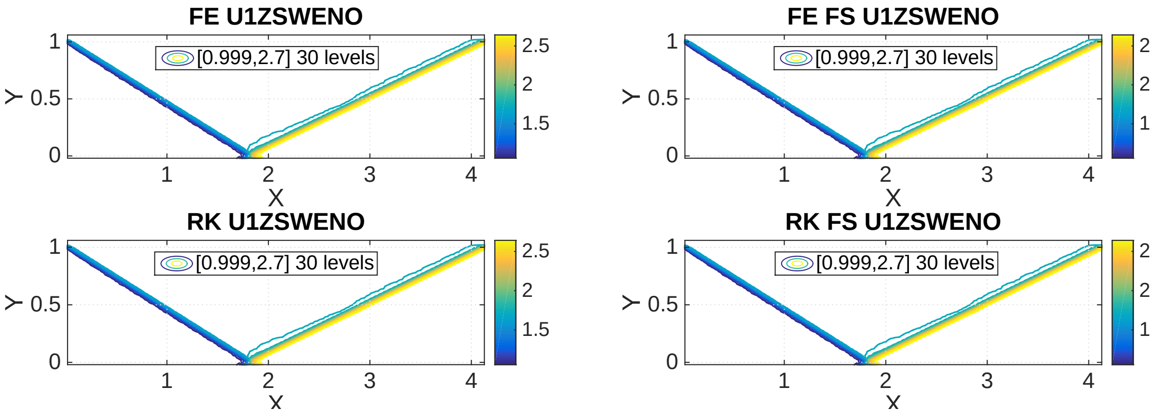

Figure 9: Example 6. Regular shock reflection. Density contours of numerical solutions by four different iterative schemes with different CFL numbers. Top left: the FE Jacobi scheme; top right: the FE sweeping scheme; bottom left: the RK Jacobi scheme; bottom right: the RK sweeping scheme. CFL number $\gamma = 0.1$ for the FE Jacobi scheme and $\gamma = 0.4$ for the rest of three schemes.

interesting to design other fixed-point sweeping methods which are independent of time-marching schemes, and they are free of the CFL-condition and have linear computational complexity as that in the homotopy method [5]. The keys are how to design a fixed-point method which is a contractive mapping for nonlinear problems and how to overcome the possible singularity difficulty since the iterations may not follow the physical time marching steps. This is one of our future work.

References

[1] S. Chen, *Fixed-point fast sweeping WENO methods for steady state solution of scalar hyperbolic conservation laws*, International Journal of Numerical Analysis and Modeling, 11 (2014), 117–130.

[2] W. Chen, C.-S. Chou and C.-Y. Kao, *Lax-Friedrichs fast sweeping methods for steady state problems for hyperbolic conservation laws*, Journal of Computational Physics, 234 (2012), 452-471.

[3] B. Cockburn and C.-W. Shu, *Runge-Kutta discontinuous Galerkin methods for convection-dominated problems*, Journal of Scientific Computing, 16 (2001), 173–261.

[4] S. Fomel, S. Luo and H. Zhao, *Fast sweeping method for the factored eikonal equation*, Journal of Computational Physics, 228 (2009), 6440–6455.

[5] W. Hao, J.D. Hauenstein, C.-W. Shu, A.J. Sommese, Z. Xu and Y.-T. Zhang, *A homotopy method based on WENO schemes for solving steady state problems of hyperbolic conservation laws*, Journal of Computational Physics, 250, (2013), 332–346.

[6] C. Hu and C.-W. Shu, *Weighted essentially non-oscillatory schemes on triangular meshes*, Journal of Computational Physics, 150, (1999), 97–127.

[7] G.-S. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, Journal of Computational Physics, 126 (1996), 202–228.

[8] Y. Jiang, C.-W. Shu and M. Zhang, *An alternative formulation of finite difference weighted ENO schemes with Lax-Wendroff time discretization for conservation laws*, SIAM Journal on Scientific Computing, 35 (2013), A1137-A1160.

[9] D. Levy, S. Nayak, C.-W. Shu and Y.-T. Zhang, *Central WENO schemes for Hamilton-Jacobi equations on triangular meshes*, SIAM Journal on Scientific Computing, 28 (2006), 2229–2247.

[10] F. Li, C.-W. Shu, Y.-T. Zhang and H.-K. Zhao, *A second order discontinuous Galerkin fast sweeping method for Eikonal equations*, Journal of Computational Physics, 227 (2008), 8191-8208.

[11] X.-D. Liu, S. Osher and T. Chan, *Weighted essentially non-oscillatory schemes*, Journal of Computational Physics, 115 (1994), 200-212.

[12] Y. Liu and Y.-T. Zhang, *A robust reconstruction for unstructured WENO schemes*, Journal of Scientific Computing, 54, (2013), pp. 603-621.

[13] J. Qian, Y.-T. Zhang and H.-K. Zhao, *Fast sweeping methods for Eikonal equations on triangular meshes*, SIAM Journal on Numerical Analysis, 45 (2007), 83–107.

[14] J. Qian, Y.-T. Zhang and H.-K. Zhao, *A fast sweeping method for static convex Hamilton-Jacobi equations*, Journal of Scientific Computing, 31 (2007), 237–271.

[15] P. L. Roe, *Approximate Riemann solvers, parameter vectors, and difference schemes*, Journal of Computational Physics, 43 (1981), 357–372.

[16] M.A. Saad, *Compressible Fluid Flow.* Prentice Hall, New York, (1993).

[17] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes*, Journal of Computational Physics, 77 (1988), 439-471.

[18] L. Wu and Y.-T. Zhang, *A third order fast sweeping method with linear computational complexity for Eikonal equations*, Journal of Scientific Computing, 62 (2015), 198-229.

[19] T. Xiong, M. Zhang, Y.-T. Zhang and C.-W. Shu, *Fast sweeping fifth order WENO scheme for static Hamilton-Jacobi equations with accurate boundary treatment*, Journal of Scientific Computing, 45 (2010), 514–536.

[20] S. Zhang, S. Jiang and C.-W. Shu, *Improvement of convergence to steady state solutions of Euler equations with the WENO schemes*, Journal of Scientific Computing, 47 (2011), 216–238.

[21] S. Zhang and C.-W. Shu, *A new smoothness indicator for the WENO schemes and its effect on the convergence to steady state solutions*, Journal of Scientific Computing, 31 (2007), 273–305.

[22] Y.-T. Zhang, S. Chen, F. Li, H. Zhao and C.-W. Shu, *Uniformly accurate discontinuous Galerkin fast sweeping methods for Eikonal equations*, SIAM Journal on Scientific Computing, 33 (2011), 1873-1896.

[23] Y.-T. Zhang and C.-W. Shu, *High order WENO schemes for Hamilton-Jacobi equations on triangular meshes*, SIAM Journal on Scientific Computing, 24 (2003), 1005-1030.

[24] Y.-T. Zhang and C.-W. Shu, *Third order WENO schemes on three dimensional tetrahedral meshes*, Communications in Computational Physics, 5, (2009), 836–848.

[25] Y.-T. Zhang, H.-K. Zhao and S. Chen, *Fixed-point iterative sweeping methods for static Hamilton-Jacobi equations*, Methods and Applications of Analysis, 13 (2006), 299–320.

[26] Y.-T. Zhang, H.-K. Zhao and J. Qian, *High order fast sweeping methods for static Hamilton-Jacobi equations*, Journal of Scientific Computing, 29 (2006), 25–56.

[27] H.-K. Zhao, *A fast sweeping method for Eikonal equations*, Math. Comp., 74 (2005), 603-627.

[28] J. Zhu, J. Qiu, C.-W. Shu and M. Dumbser, *Runge-Kutta discontinuous Galerkin method using WENO limiters II: unstructured meshes*, Journal of Computational Physics, 227, (2008), 4330-4353.