Fast computations of high order WENO methods for hyperbolic conservation laws

Yong-Tao Zhang

Department of Applied and Computational Mathematics and Statistics
University of Notre Dame

Introduction

- A high order WENO discretization of complicated multidimensional problems leads to large amount of operations and computational costs.
- Goal: achieve fast simulations by high order WENO methods (e.g., fifth order WENO scheme) for solving hyperbolic conservation laws.
- Fixed-point fast sweeping WENO schemes for solving steady state hyperbolic conservation laws.

  \- based on high order WENO fast sweeping methods for solving Hamilton-Jacobi equations. Utilize the hyperbolic properties of the PDE in the iterative scheme.

  \- Liang Wu, Yong-Tao Zhang, Shuhai Zhang, and Chi-Wang Shu, High order fixed-point sweeping WENO methods for steady state of hyperbolic conservation laws and its convergence study, Communications in Computational Physics, v20, (2016), pp. 835-869.

  \- Liang Li, Jun Zhu, Yong-Tao Zhang, Absolutely convergent fixed-point fast sweeping WENO methods for steady state of hyperbolic conservation laws, (2020), arXiv:2006.11885 [math.NA]. Submitted to Journal of Computational Physics.
- High order WENO scheme on sparse grids to solve high dimensional problems.

  \- Xiaozhi Zhu and Yong-Tao Zhang, Fast sparse grid simulations of fifth order WENO scheme for high dimensional hyperbolic PDEs, Journal of Scientific Computing, v87, (2021), article number: 44.

steady state problems of hyperbolic conservation laws

$$\nabla \cdot F(U) = h,$$

- With some appropriate boundary conditions.
- U is the vector of the unknown conservative variables.
- F(U) is the vector of flux functions.
- h is the source term.
- A spatial discretization leads to a large nonlinear system.

Motivated by high order WENO fast sweeping methods for solving static Hamilton-Jacobi equations.

Y.-T. Zhang, H. Zhao, J. Qian, Journal of Scientific Computing, 2006, v29: 25-56.
T. Xiong, M. Zhang, Y.-T. Zhang and C.-W. Shu, Journal of Scientific Computing, 2010, v45: 514-536.

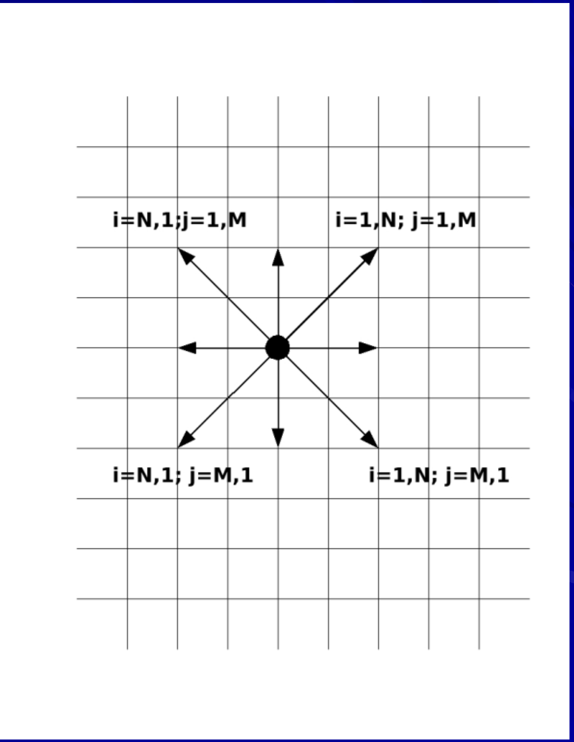
- A local solver based on a monotone numerical Hamiltonian, which is consistent with the causality of the PDE.
- Systematic orderings of all grid points, which can cover all directions of the characteristics.

  for example:

  on 2D rectangular mesh, the natural orderings give the sweeping directions

  (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.
- Solving the nonlinear system by Gauss-Seidel iterations along alternating directions.



Explicit high order fast sweeping WENO methods: Fixed-point sweeping schemes. 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.

For example: time marching with the 2nd order TVD-Runge Kutta:

$$
\begin{aligned}
\phi_{i,j}^{(1)} &= \phi_{i,j}^{n} + \Delta t \left[f_{i,j} - \hat{H}\left((\phi_x)_{i,j}^{n,-}, (\phi_x)_{i,j}^{n,+}; (\phi_y)_{i,j}^{n,-}, (\phi_y)_{i,j}^{n,+}\right) \right], \\
\phi_{i,j}^{n+1} &= \frac{1}{2}\phi_{i,j}^{n} + \frac{1}{2}\phi_{i,j}^{(1)} + \frac{1}{2}\Delta t \left[f_{i,j} - \hat{H}\left((\phi_x)_{i,j}^{(1),-}, (\phi_x)_{i,j}^{(1),+}; (\phi_y)_{i,j}^{(1),-}, (\phi_y)_{i,j}^{(1),+}\right) \right]
\end{aligned}
$$

Coupled with fast sweeping techniques (use new values in the stencil, and alternating sweeping directions):

$$
\phi_{i,j}^{(1)} = \phi_{i,j}^{n} + \gamma \left(\frac{1}{\frac{\alpha_x}{h_x} + \frac{\alpha_y}{h_y}} \right) \left[f_{i,j} - \hat{H}\left((\phi_x)_{i,j}^{-}, (\phi_x)_{i,j}^{+}; (\phi_y)_{i,j}^{-}, (\phi_y)_{i,j}^{+}\right) \right]
$$

$$
\phi_{i,j}^{n+1} = \phi_{i,j}^{(1)} + \frac{1}{2}\gamma \left(\frac{1}{\frac{\alpha_x}{h_x} + \frac{\alpha_y}{h_y}} \right) \left[f_{i,j} - \hat{H}\left((\phi_x)_{i,j}^{-}, (\phi_x)_{i,j}^{+}; (\phi_y)_{i,j}^{-}, (\phi_y)_{i,j}^{+}\right) \right]
$$

WENO discretization

- Base scheme: the fifth order finite difference WENO scheme with Lax-Friedrichs flux splitting.

  G.-S. Jiang and C.-W. Shu, JCP, 1996, v126, p. 202-228.

- Conservative flux approximations: $$f(u)_x|_{x=x_i} \approx \frac{1}{\Delta x}(\hat{f}_{i+1/2} - \hat{f}_{i-1/2})$$

- WENO5 approximation with five-point stencil to numerical fluxes (the case of positive wind):

$$\hat{f}_{i+1/2} = w_0 \hat{f}^{(0)}_{i+1/2} + w_1 \hat{f}^{(1)}_{i+1/2} + w_2 \hat{f}^{(2)}_{i+1/2}, \tag{2.3}$$

where

$$\begin{aligned}
\hat{f}^{(0)}_{i+1/2} &= \frac{1}{3} f(u_{i-2}) - \frac{7}{6} f(u_{i-1}) + \frac{11}{6} f(u_i), \\
\hat{f}^{(1)}_{i+1/2} &= -\frac{1}{6} f(u_{i-1}) + \frac{5}{6} f(u_i) + \frac{1}{3} f(u_{i+1}), \\
\hat{f}^{(2)}_{i+1/2} &= \frac{1}{3} f(u_i) + \frac{5}{6} f(u_{i+1}) - \frac{1}{6} f(u_{i+2}).
\end{aligned} \tag{2.4}$$

Nonlinear weights

$$w_r = \frac{\alpha_r}{\alpha_1 + \alpha_2 + \alpha_3}, \qquad \alpha_r = \frac{d_r}{(\epsilon + \beta_r)^2}, \qquad r = 0, 1, 2. \tag{2.5}$$

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

$$\begin{aligned} \beta_0 &= \frac{13}{12}(f_{i-2} - 2f_{i-1} + f_i)^2 + \frac{1}{4}(f_{i-2} - 4f_{i-1} + 3f_i)^2, \\ \beta_1 &= \frac{13}{12}(f_{i-1} - 2f_i + f_{i+1})^2 + \frac{1}{4}(f_{i-1} - f_{i+1})^2, \\ \beta_2 &= \frac{13}{12}(f_i - 2f_{i+1} + f_{i+2})^2 + \frac{1}{4}(3f_i - 4f_{i+1} + f_{i+2})^2, \end{aligned} \tag{2.6}$$

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

- For the negative wind, right-biased stencil is used. The formulae for negative and positive wind cases are symmetric with respect to the point $x_{i+1/2}$
- For the general case, Lax-Friedrichs flux splitting is performed:

$$f^+(u) = \frac{1}{2}(f(u) + \alpha u), \qquad f^-(u) = \frac{1}{2}(f(u) - \alpha u), \tag{2.7}$$

where $\alpha = \max_u |f'(u)|$. $f^+(u)$ is the positive wind part, and $f^-(u)$ is the negative wind part.

Early work on improvement of convergence to steady state for WENO schemes

- High order WENO schemes suffer from difficulties in their convergence to steady state solutions, e.g., the residue of WENO schemes often stops decreasing during their iterations.
- For example: studies in S. Zhang and C.-W. Shu (JSC, 2007, 2011) reveals that slight post-shock oscillations actually cause this problem.
- Slight post-shock oscillations can be mixed with multi-scale structures in complex fluids and cause problems in resolving the real physical phenomena.
- Two methods developed to reduce the slight post-shock oscillations.
  - New smoothness indicators.
  - Upwind-biased interpolation is used to form the Jacobian at the cell interface for the local characteristic decomposition.

The explicit formulae for the new smoothness indicator are

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

Derived via analyzing effects of different parts of the original smoothness indicator on numerical solution around shock waves. (S. Zhang and C.-W. Shu, JSC 2007).

- For systems of hyperbolic conservation laws, upwind-biased interpolation rather than the standard Roe average is used to form the Jacobian matrix at the cell interface for the local characteristic decomposition. (S. Zhang and C.-W. Shu, JSC 2011).

- the upwind-biased interpolation for the x-direction local characteristic decomposition:

$$U_{i+1/2} = U^{(1)} \text{ when } u_{i+1/2} \geq 0$$

$$U_{i+1/2} = U^{(2)} \text{ when } u_{i+1/2} < 0$$

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

u: the x-direction fluid velocity

$$u_{i+1/2} = \frac{\sqrt{\rho_i}}{\sqrt{\rho_i} + \sqrt{\rho_{i+1}}} u_i + \frac{\sqrt{\rho_{i+1}}}{\sqrt{\rho_i} + \sqrt{\rho_{i+1}}} u_{i+1}$$

Fixed-point iterative schemes

- After WENO discretization, we obtain a nonlinear system

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

- Time marching methods are essentially a Jacobi type fixed-point iterations. For example:

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

- This is actually forward Euler method with time step size: $\Delta t_n = \frac{\gamma}{\alpha_x/\Delta x + \alpha_y/\Delta y}$

γ actually represents the CFL number

- Another example, Jacobi type fixed-point iterations from the 3rd order TVD Runge-Kutta :

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

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

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

Fixed-point fast sweeping WENO schemes for hyperbolic conservation laws

L. Wu, Y.-T. Zhang, S. Zhang, and C.-W. Shu, High order fixed-point sweeping WENO methods for steady state of hyperbolic conservation laws and its convergence study, Communications in Computational Physics, v20, (2016), pp. 835-869.

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

$$i = i_1, \cdots, i_N; j = j_1, \cdots, j_M.$$

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

$$(1)\ i = 1:N, j = 1:M;$$

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

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

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

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 why we use the notation u^* to represent the values in the scheme (2.13), and $u^*_{k,l}$ could be $u^n_{k,l}$ or $u^{n+1}_{k,l}$, depending on the current sweeping direction.

Runge Kutta type fixed-point sweeping scheme

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

$$i = i_1, \cdots, i_N; j = j_1, \cdots, j_M.$$

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

$$i = i_1, \cdots, i_N; j = j_1, \cdots, j_M.$$

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

$$i = i_1, \cdots, i_N; j = j_1, \cdots, j_M.$$

Numerical Example I: **Burgers equations**

Here we further examine our scheme on a two-dimensional Burgers' equation with a source term,

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

with initial conditions,

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

Setting $\beta = 1.5$ leads to a smooth steady state solution

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

We use exact steady state solution on square boundaries. Also we use the same convergence criterion as before to test the accuracy, i.e. $Res_A < 1e-12$. Results are shown in Table 17 to 24. Again,

Table 17: Original WENO5 with Runge-Kutta in time. CFL: 1.0

points	L_1 error	L_1 order	L_∞ error	L_∞ index (i,j)	L_∞ order	iter #	CPU time
10	2.46e-6		1.23e-5	(9,9)		270	4.30e-2
20	6.31e-8	5.29	3.50e-7	(19,19)	5.14	342	0.23
40	1.37e-9	5.52	8.98e-9	(39,39)	5.28	513	1.42
80	2.95e-11	5.54	1.93e-10	(79,79)	5.54	855	9.54

Table 18: Original WENO5 with Sweeping(G-S) + RK3 in time. CFL: 1.0

points	L_1 error	L_1 order	L_∞ error	L_∞ index (i,j)	L_∞ order	iter #	CPU time
10	2.46e-6		1.23e-5	(9,9)		159	2.41e-2
20	6.31e-8	5.29	3.50e-7	(19,19)	5.14	186	0.12
40	1.37e-9	5.52	8.99e-9	(39,39)	5.28	273	0.70
80	2.95e-11	5.54	2.07e-10	(79,79)	5.44	450	4.72

Table 19: Original WENO5 with Forward Euler in time. CFL: 0.1

points	L_1 error	L_1 order	L_∞ error	L_∞ index (i,j)	L_∞ order	iter #	CPU time
10	2.46e-6		1.23e-5	(9,9)		1195	0.18
20	6.31e-8	5.29	3.50e-7	(19,19)	5.14	1439	0.92
40	1.37e-9	5.52	8.99e-9	(39,39)	5.28	1822	3.16
80	2.95e-11	5.54	2.08e-10	(79,79)	5.43	3178	33.67

Table 20: Original WENO5 with Sweeping(G-S) + Forward Euler in time. CFL: 1.0

points	L_1 error	L_1 order	L_∞ error	L_∞ index (i,j)	L_∞ order	iter #	CPU time
10	2.46e-6		1.23e-5	(9,9)		113	1.77e-2
20	6.31e-8	5.29	3.50e-7	(19,19)	5.14	133	8.39e-2
40	1.37e-9	5.52	8.99e-9	(39,39)	5.28	181	0.47
80	2.95e-11	5.54	2.07e-10	(79,79)	5.44	286	3.04

Table 21: New Smoothness Indicator WENO5 with Runge-Kutta in time. CFL: 1.0

points	L_1 error	L_1 order	L_∞ error	L_∞ index (i,j)	L_∞ order	iter #	CPU time
10	4.33e-6		1.41e-5	(9,9)		270	4.04e-2
20	9.33e-8	5.54	3.85e-7	(19,19)	5.19	351	0.22
40	2.03e-9	5.53	1.08e-8	(39,39)	5.16	528	1.35
80	4.30e-11	5.56	2.63e-10	(79,79)	5.36	867	8.88

Table 22: New Smoothness Indicator WENO5 with Sweeping(G-S) + RK3 in time. CFL: 1.0

points	L_1 error	L_1 order	L_∞ error	L_∞ index (i,j)	L_∞ order	iter #	CPU time
10	4.33e-6		1.41e-5	(9,9)		159	2.28e-2
20	9.33e-8	5.54	3.85e-7	(19,19)	5.19	198	0.12
40	2.03e-9	5.53	1.08e-8	(39,39)	5.16	279	0.67
80	4.30e-11	5.56	2.82e-10	(79,79)	5.26	447	4.41

Table 23: New Smoothness Indicator WENO5 with Forward Euler in time. CFL: 0.1

points	L_1 error	L_1 order	L_∞ error	L_∞ index (i,j)	L_∞ order	iter #	CPU time
10	4.33e-6		1.41e-5	(9,9)		1166	0.17
20	9.33e-8	5.54	3.85e-7	(19,19)	5.19	1409	0.84
40	2.03e-9	5.53	1.08e-8	(39,39)	5.16	1937	4.74
80	4.30e-11	5.56	2.83e-10	(79,79)	5.25	3209	32.04

Table 24: New Smoothness Indicator WENO5 with Sweeping(G-S) + Forward Euler in time. CFL: 1.0

points	L_1 error	L_1 order	L_∞ error	L_∞ index (i,j)	L_∞ order	iter #	CPU time
10	4.33e-6		1.41e-5	(9,9)		110	1.59e-2
20	9.33e-8	5.54	3.85e-7	(19,19)	5.19	136	8.00e-2
40	2.03e-9	5.53	1.08e-8	(39,39)	5.16	193	0.47
80	4.30e-11	5.56	2.82e-10	(79,79)	5.26	289	2.84

Numerical Example: A two-dimensional oblique shock

In this subsection, we simulate an oblique shock which has an angle of 135° with the positive x-direction, which is also tested in [9] and [10]. The flow Mach number on 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 5th WENO scheme (U1WENO) can settle down to 10^{-12}. U1WENO is also shown as the most efficient scheme among those offer the best results for this example in [10]. So here we use 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, 10^{-12}.

- Upwind interpolation in the local characteristic decomposition is used to improve the convergence of WENO5 scheme to steady state.

Table 29: 135° Oblique steady shock wave: U1WENO scheme with Forward Euler time marching.

γ: CFL number	iteration number	time until conv	CPU time
0.1	23729	22.26	5510
0.2	not convergent, hang at -3.1		

Table 30: 135° Oblique steady shock wave: U1WENO scheme with Forward Euler time marching and Fast Sweeping.

γ: CFL number	iteration number	time until conv	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 (blowed up)		

Table 31: 135° Oblique steady shock wave: U1WENO scheme with Runge-Kutta time marching.

γ: CFL number	iteration number	time until conv	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, hang at -2.2		
1.4	not convergent, hang at -1.9		

Table 32: 135° Oblique steady shock wave: U1WENO scheme with Runge-Kutta time marching and Fast Sweeping.

γ: CFL number	iteration number	time until conv	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 (blowed up)		
1.4	not convergent (blowed up)		

- **The fixed-point sweeping WENO scheme with forward Euler is the most efficient scheme among all of four different schemes, among all possible CFL numbers.**

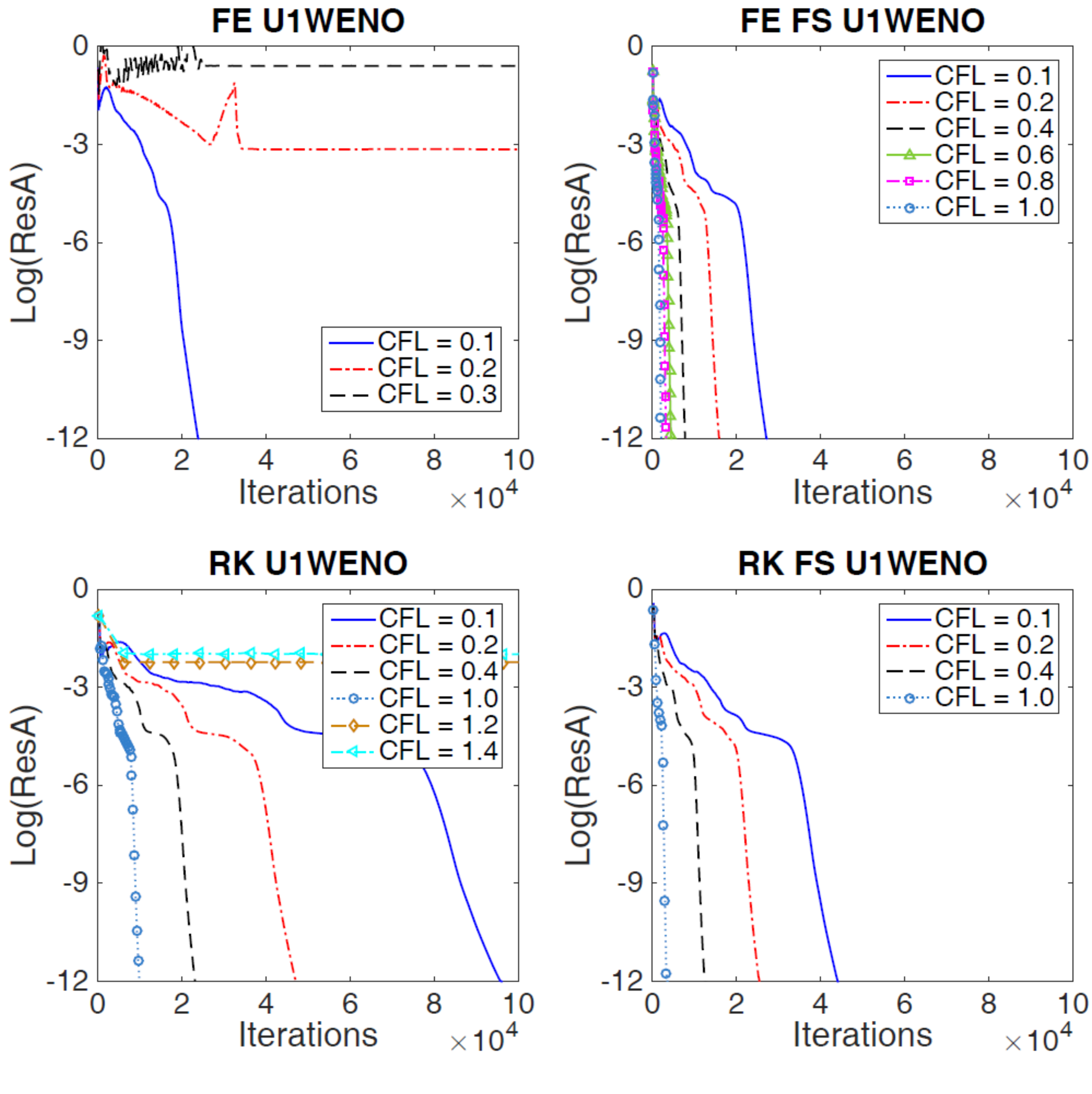


Figure 4: The evolution of average residue in terms of iterations of a 135° oblique shock of $M_\infty = 2$ by various U1WENO schemes and CFL numbers

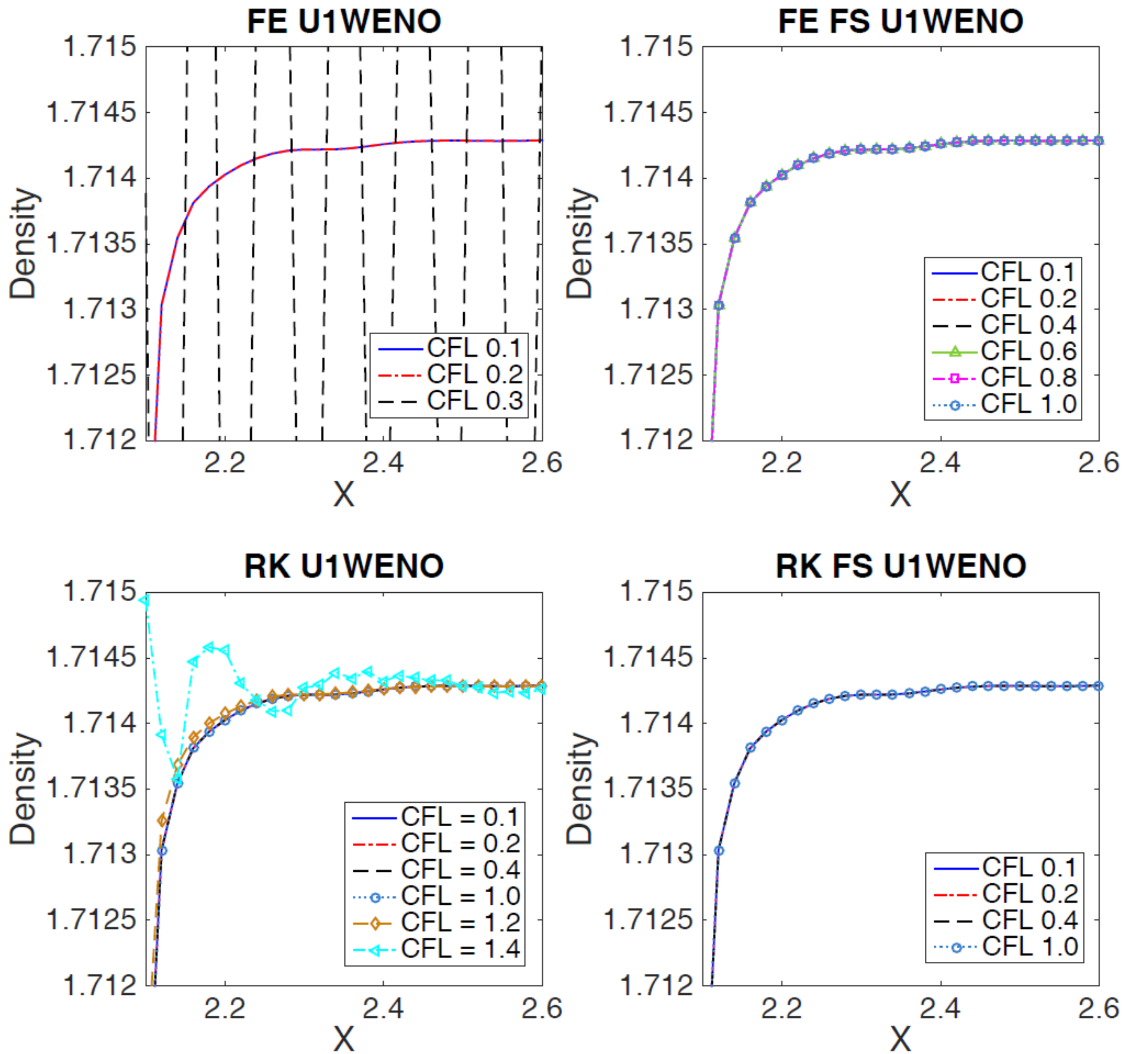
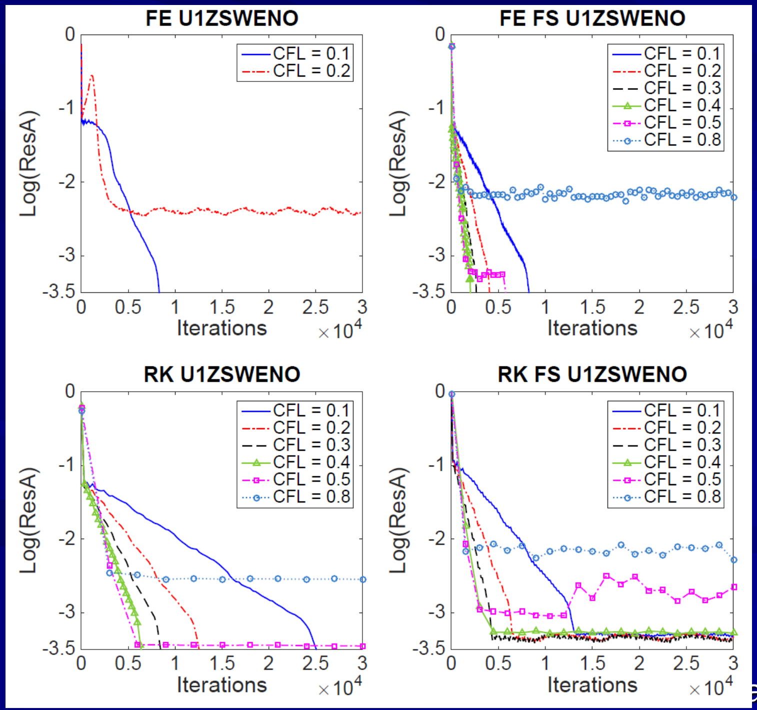


Figure 6: Zoomed density distribution (downstream) of a 135° oblique shock of $M_\infty = 2$ along the line $y = 1$ by various U1WENO schemes and CFL numbers

- Non-oscillatory shock transitions (In the "Zoomed in" small scales) are obtained for converged cases.

- Challenge: for some difficult examples, e.g., regular shock reflection problem, the iteration residue of the fixed-point fast sweeping WENO scheme still hangs at a truncation error level instead of converging to round-off levels.
- This motivates us to apply recent new WENO approximations to our fixed-point fast sweeping WENO methods for resolving this issue.
- J. Zhu and C.-W. Shu, A new type of multi-resolution WENO schemes with increasingly higher order of accuracy, Journal of Computational Physics, 375 (2018), 659-683.



Regular shock reflection problem. Iteration residues hang at $10^{-3.5}$ level.

Liang Li, Jun Zhu, Yong-Tao Zhang, Absolutely convergent fixed-point fast sweeping WENO methods for steady state of hyperbolic conservation laws, *submitted*. 2020, arXiv:2006.11885 [math.NA].

A fifth order multi-resolution WENO reconstruction (J. Zhu & C.-W. Shu, JCP, 2018) is applied in the fast sweeping iterations:

Reconstruction algorithm:

Step 1. We choose the central spatial stencils $T_k = \{I_{i+1-k}, \cdots, I_{i-1+k}\}$, $k = 1, 2, 3$, and reconstruct $2k-2$ degree polynomials $q_k(x)$ which satisfy

$$\frac{1}{\triangle x}\int_{x_{l-1/2}}^{x_{l+1/2}} q_k(x)dx = \overline{h}_l, l = i-k+1, \cdots, i-1+k; k = 1, 2, 3.$$

Step 2. Obtain equivalent expressions for these reconstruction polynomials of different degrees. To keep consistent notation, we denote $p_1(x) = q_1(x)$, with similar ideas for the central WENO schemes [13, 2, 14] as well, and compute

$$p_2(x) = \frac{1}{\gamma_{2,2}}q_2(x) - \frac{\gamma_{1,2}}{\gamma_{2,2}}p_1(x),$$

$$p_3(x) = \frac{1}{\gamma_{3,3}}q_3(x) - \frac{\gamma_{1,3}}{\gamma_{3,3}}p_1(x) - \frac{\gamma_{2,3}}{\gamma_{3,3}}p_2(x),$$

with $\gamma_{1,2} + \gamma_{2,2} = 1$, $\gamma_{1,3} + \gamma_{2,3} + \gamma_{3,3} = 1$, and $\gamma_{2,2} \neq 0, \gamma_{3,3} \neq 0$. In these expressions, $\gamma_{a,b}$ for $a = 1, \cdots, b$ and $b = 2, 3$ are the linear weights. Based on a balance between the sharp and essentially non-oscillatory shock transitions in nonsmooth regions and accuracy in smooth regions, following the practice in [6, 34, 17, 35, 37], we take the linear weights as $\gamma_{1,2} = \frac{1}{11}$, $\gamma_{2,2} = \frac{10}{11}$, $\gamma_{1,3} = \frac{1}{111}$, $\gamma_{2,3} = \frac{10}{111}$, $\gamma_{3,3} = \frac{100}{111}$.

Step 3. Compute the smoothness indicators β_k, which measure how smooth the functions $p_k(x)$ for $k = 2, 3$ are in the interval $[x_{i-1/2}, x_{i+1/2}]$. We use the same recipe for the smoothness indicators as that in [11, 21]:

$$\beta_k = \sum_{\alpha=1}^{2k-2}\int_{x_{i-1/2}}^{x_{i+1/2}} \triangle x^{2\alpha-1}(\frac{d^\alpha p_k(x)}{dx^\alpha})^2 dx, \qquad k = 2, 3.$$

The only exception is β_1, which is magnified from zero to a tiny value. See [37] for details.

Step 4. Compute the nonlinear weights based on the linear weights and the smoothness indicators. We adopt the WENO-Z type nonlinear weights as that in [1, 3]. First a quantity τ

which depends on the absolute differences between the smoothness indicators is calculated: $\tau = (\frac{\sum_{l_1=1}^{2}|\beta_3-\beta_{l_1}|}{2})^2$. The nonlinear weights are then computed as

$$\omega_{l_1} = \frac{\bar{\omega}_{l_1}}{\sum_{l_2=1}^{3}\bar{\omega}_{l_2}}, \quad \bar{\omega}_{l_1} = \gamma_{l_1,3}(1+\frac{\tau}{\varepsilon+\beta_{l_1}}), \qquad l_1 = 1,2,3.$$

ε is a small value to avoid that the denominator becomes zero. In this paper, ε is taken to be 10^{-6} for all numerical examples.

Step 5. The final reconstructed numerical flux $\hat{f}^{+}_{i+1/2,j}$ is given by

$$\hat{f}^{+}_{i+1/2,j} = \sum_{l_1=1}^{3}\omega_{l_1}p_{l_1}(x_{i+1/2}).$$

Accuracy and CPU time for a 2D Euler system

FE Jacobi, γ=0.1						
$N \times N$	L_1 error	L_1 order	L_∞ error	L_∞ order	iter♯	CPU time
10×10	6.74E-04	-	2.68E-03	-	5817	12.92
20×20	1.30E-05	5.69	3.58E-05	6.23	6804	30.63
30×30	1.84E-06	4.83	4.76E-06	4.98	8583	68.08
40×40	4.49E-07	4.90	1.13E-06	5.00	10613	138.61
50×50	1.50E-07	4.92	3.74E-07	4.96	12725	247.23
60×60	6.08E-08	4.94	1.51E-07	4.95	14931	404.63
70×70	2.83E-08	4.95	7.04E-08	4.97	17068	616.05
80×80	1.46E-08	4.96	3.62E-08	4.99	19093	886.75

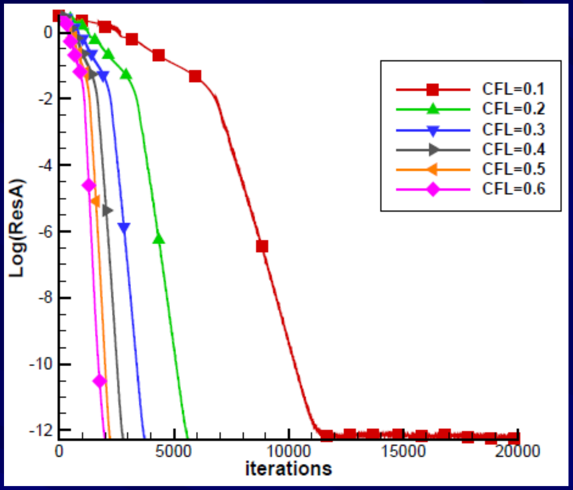
RK Jacobi, γ=1.0						
$N \times N$	L_1 error	L_1 order	L_∞ error	L_∞ order	iter♯	CPU time
10×10	7.41E-04	-	2.68E-03	-	1746	4.16
20×20	1.31E-05	5.82	3.58E-05	6.23	2037	11.06
30×30	1.85E-06	4.84	4.76E-06	4.98	2568	26.47
40×40	4.51E-07	4.91	1.13E-06	5.00	3174	52.63
50×50	1.50E-07	4.93	3.74E-07	4.96	3825	94.11
60×60	6.10E-08	4.95	1.51E-07	4.95	4488	155.11
70×70	2.84E-08	4.96	7.04E-08	4.97	5130	236.75
80×80	1.46E-08	4.96	3.62E-08	4.99	5739	340.70

FE fast sweeping, γ=1.0						
$N \times N$	L_1 error	L_1 order	L_∞ error	L_∞ order	iter♯	CPU time
10×10	6.62E-04	-	2.68E-03	-	560	1.26
20×20	1.30E-05	5.67	3.58E-05	6.23	653	4.02
30×30	1.84E-06	4.83	4.76E-06	4.98	821	10.69
40×40	4.49E-07	4.90	1.13E-06	5.00	1010	22.53
50×50	1.50E-07	4.92	3.74E-07	4.96	1213	42.39
60×60	6.08E-07	4.94	1.51E-07	4.95	1421	71.19
70×70	2.83E-08	4.95	7.04E-08	4.97	1622	110.55
80×80	1.46E-08	4.96	3.62E-08	4.99	1814	160.91

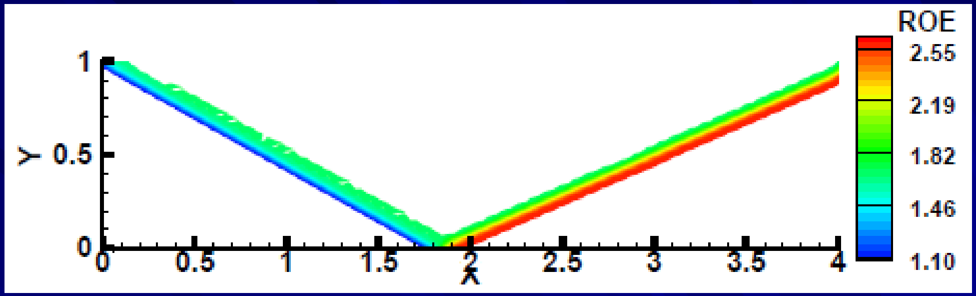
Table 4: Example 4, A 2D Euler system of equations with source terms. Accuracy, iteratio numbers and CPU times of three different iterative schemes. CPU time unit: second

The fast sweeping WENO scheme saves more than 50% CPU costs of that of the 3rd order TVD Runge-Kutta WENO scheme. Similar numerical errors and accuracy orders are obtained.

Regular shock reflection problem



Residue history of Fast sweeping WENO

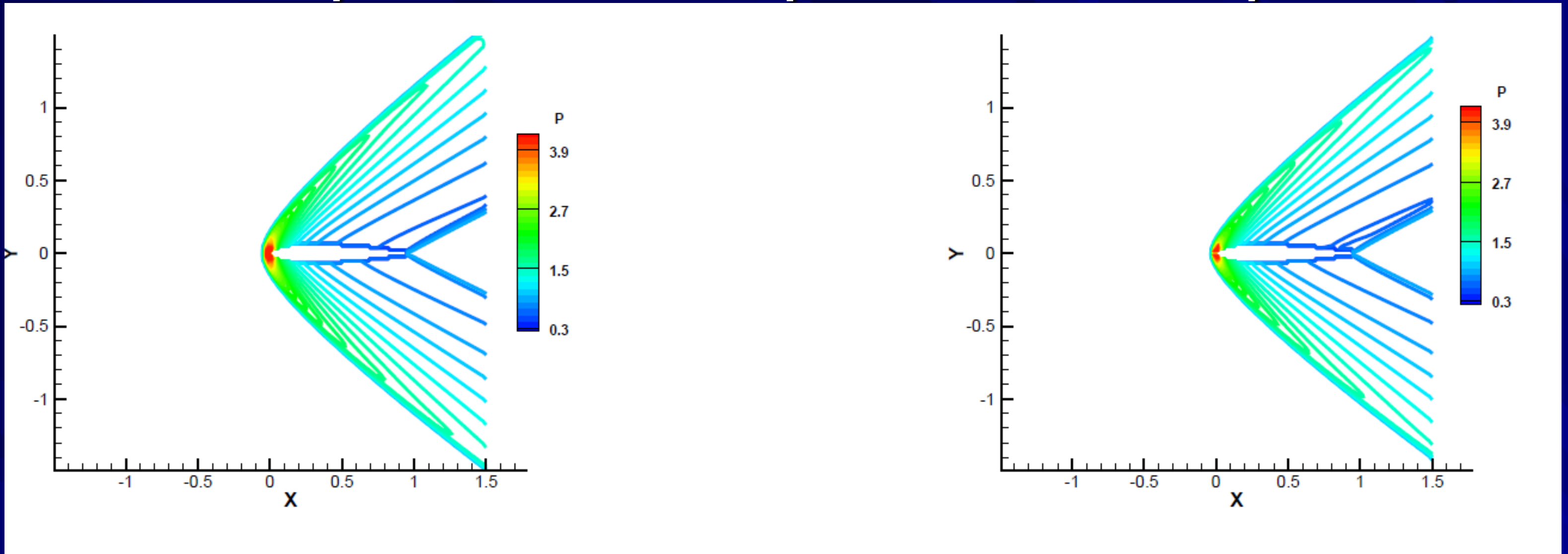


Contour plot of the FS WENO solution

FS WENO saves about 66% CPU costs of the TVD-RK WENO.

FE Jacobi scheme			
γ : CFL number	iteration number	final time	CPU time
0.1	12046	5.09	618.36
0.2	Not convergent	-	-
RK Jacobi scheme			
γ : CFL number	iteration number	final time	CPU time
0.3	11268	4.76	579.22
0.4	8454	4.76	436.33
0.5	6762	4.76	348.09
0.6	5634	4.76	289.89
0.7	Not convergent	-	-
FE fast sweeping scheme			
γ : CFL number	iteration number	final time	CPU time
0.3	3651	4.62	188.14
0.4	2722	4.59	140.03
0.5	2170	4.57	111.91
0.6	1934	4.89	98.92
0.7	Not convergent	-	-

A supersonic flow past an airfoil problem



(a) RK Jacobi scheme, Supersonic (b) FE fast sweeping scheme, Supersonic

Supersonic flow			
RK Jacobi scheme			
γ : CFL number	iteration number	final time	CPU time
0.9	18942	16.88	4243.36
1.0	17064	16.89	3822.81
1.1	Not convergent	-	-
FE fast sweeping scheme			
γ : CFL number	iteration number	final time	CPU time
0.9	8492	22.70	3288.59
1.0	4564	13.55	1728.37
1.1	4304	13.91	1618.44
1.2	Not convergent	-	-

Iteration residues converge to the level of 10^{-12}.

FS WENO saves about 58% CPU costs of the TVD-RK WENO.

X. Zhu and Y.-T. Zhang, Fast sparse grid simulations of fifth order WENO scheme for high dimensional hyperbolic PDEs, Journal of Scientific Computing, v87, (2021), article number: 44.

$$u_t + \nabla \cdot \vec{f}(u) = 0, \tag{1}$$

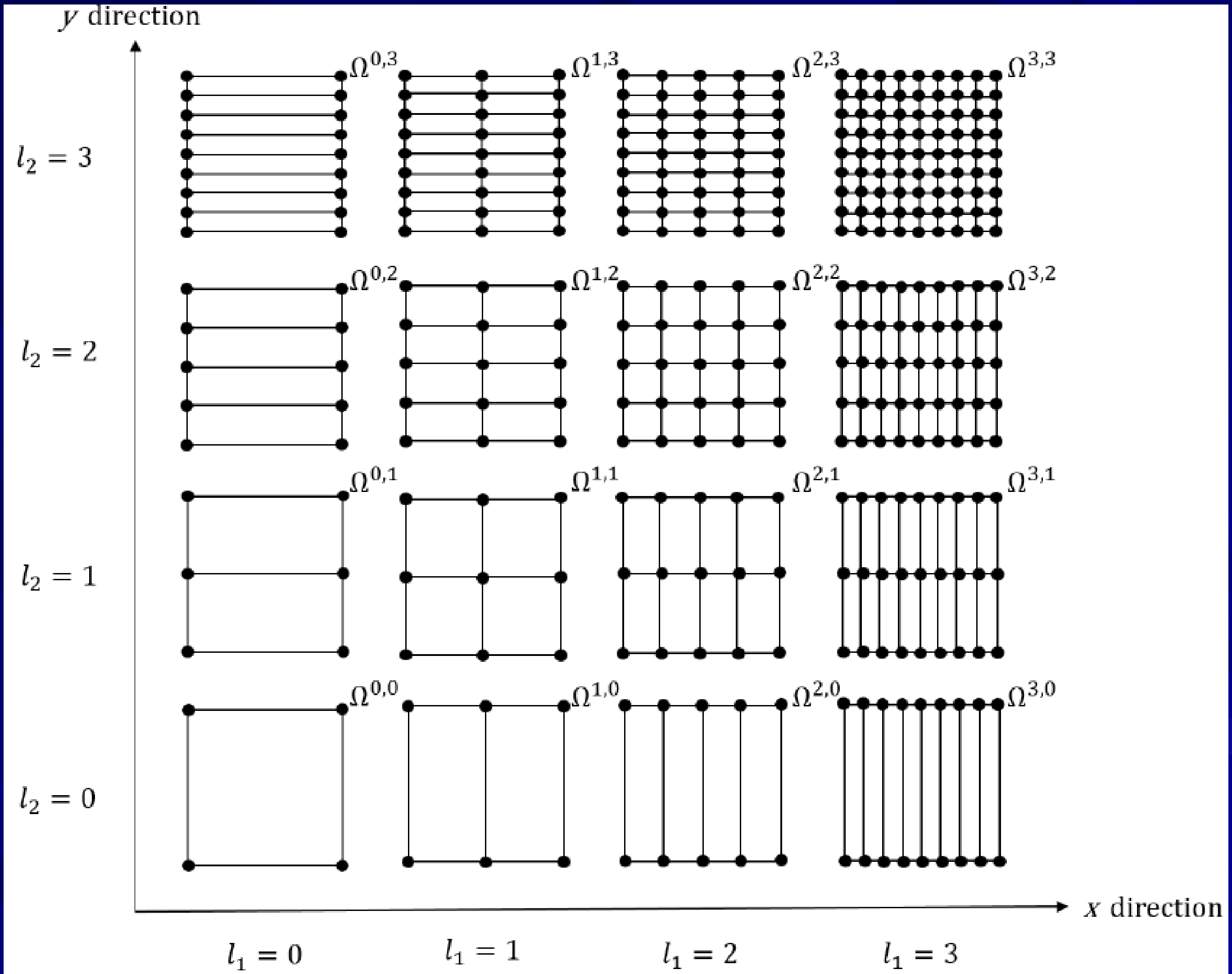
where $u(\vec{x}, t)$ is the unknown, and $\vec{f} = (f_1, \cdots, f_d)^T$ is the vector of flux functions

- Base scheme:

  --- Spatial discretization: high order (e.g the 5th order) finite difference WENO schemes with Lax-Friedrichs flux splitting.

  --- Time discretization: the third order TVD Runge-Kutta scheme.

Sparse grids



Semi-coarsened sparse grids with the finest level 3.

Sparse-grid combination technique

- The final solution is a linear combination of solutions on semi-coarsened grids, where the coefficients of the combination are chosen such that there is a canceling in leading-order error terms and the accuracy order can be kept to be the same as that on single full grids.
- Error analysis of linear schemes for linear PDEs has been performed in the following work which proved the above statement.

  -- Griebel, M., Schneider, M., Zenger, C., A combination technique for the solution of sparse grid problems, in: R. Beauwens, P. de Groen (Eds.), Iterative Methods in Linear Algebra, North-Holland, Amsterdam, 1992, pp. 263-281.

  -- Lastdrager, B., Koren, B., Verwer, J., The sparse-grid combination technique applied to time-dependent advection problems. Applied Numerical Mathematics, 2001. 38: p. 377-401.

  -- Lastdrager, B., Koren, B., Verwer, J., Solution of time-dependent advection-diffusion problems with the sparse-grid combination technique and a rosenbrock solver. Computational Methods in Applied Mathematics, 2001. 1: pp. 86-99.
- Error analysis of nonlinear schemes / nonlinear PDEs is still open.
- We use numerical experiments to show the accuracy order of WENO schemes for solving nonlinear problems can still be achieved with sparse-grid combination techniques.

Error analysis results for the 5th order linear schemes applied to a 2D linear advection PDE

- For the 2D linear PDE: $$c_t + a\partial_x c + b\partial_y c = 0$$

- Leading order errors in spatial direction for the 5th order linear scheme with the sparse grid combination technique:

$$-\frac{1}{60}h^5 \cdot T \cdot \left(a\frac{\partial^6 u}{\partial x^6} + b\frac{\partial^6 u}{\partial y^6}\right) + \frac{1}{3600}h^5 H^5 \cdot T^2 ab\left(1 - 31\log_2\frac{H}{h}\right)\frac{\partial^{12} u}{\partial x^6 \partial y^6}$$
$$+ O\left(h^6 \log_2 \frac{1}{h}\right)$$

  - H is the grid size of the root grid; h is the grid size of the most refine grid.

Sparse grid WENO scheme

Algorithm: WENO scheme with sparse-grid combination technique

- Step 1: Restrict the initial condition $u(x, y, 0)$ to $(2N_L + 1)$ sparse grids $\{\Omega^{l_1,l_2}\}_I$ defined above. Here "Restrict" means that functions are evaluated at grid points;
- Step 2: On each sparse grid Ω^{l_1,l_2}, solve the equation (1) by Runge-Kutta WENO scheme to reach the final time T. Then we get $(2N_L+1)$ sets of solutions $\{U^{l_1,l_2}\}_I$;
- Step 3: At the final time T,
  - on each grid Ω^{l_1,l_2}, apply prolongation operator P^{N_L,N_L} on U^{l_1,l_2}. Then we get $P^{N_L,N_L}U^{l_1,l_2}$, defined on the most refined mesh Ω^{N_L,N_L}. For smooth solutions, the regular Lagrange prolongation can be used directly. In general, WENO prolongation is used;
  - do the combination to get the final solution

$$\hat{U}^{N_L,N_L} = \sum_{l_1+l_2=N_L} P^{N_L,N_L}U^{l_1,l_2} - \sum_{l_1+l_2=N_L-1} P^{N_L,N_L}U^{l_1,l_2}. \tag{10}$$

For three dimensional (3D) or higher dimensional problems, the algorithm is similar although prolongation operations are performed in additional spatial directions. The sparse-grid combination formula for higher dimensional cases can be found in the literature (e.g. [5]). Specifically the 3D formula is

$$\hat{U}^{N_L,N_L,N_L} = \sum_{l_1+l_2+l_3=N_L} P^{N_L,N_L,N_L}U^{l_1,l_2,l_3} - 2\sum_{l_1+l_2+l_3=N_L-1} P^{N_L,N_L,N_L}U^{l_1,l_2,l_3}$$
$$+ \sum_{l_1+l_2+l_3=N_L-2} P^{N_L,N_L,N_L}U^{l_1,l_2,l_3}.$$

d-dimensional sparse-grid combination

d: Spatial dimension of the PDE

l: The most refined level

$I_d = (l_1, \ldots, l_d)$: A sparse grid with levels $l_1, \ldots, l_d$ on each direction.

$u^f_{I_d}$: Numerical solution at the sparse grid I_d after prolongation to the most refined grid

u^c_l: The solution after combination.

The general formula for combination is,

$$u^c_l = \sum_{m=l}^{l+d-1} (-1)^{d+l-(m+1)} \binom{d-1}{m-l} \sum_{|I_d|=m-(d-1)} u^f_{I_d},$$

5th order WENO prolongation / interpolation

$$u_{Lagr}(x) = \sum_{k=0}^{2} C_k(x) P_k(x)$$

$$C_0(x) = \frac{(x - x_{i+1})(x - x_{i+2})}{12h^2},$$
$$C_1(x) = \frac{(x - x_{i-2})(x - x_{i+2})}{-6h^2},$$
$$C_2(x) = \frac{(x - x_{i-2})(x - x_{i-1})}{12h^2}$$

$$w_k(x) = \frac{\tilde{C}_k(x)}{\tilde{C}_0(x) + \tilde{C}_1(x) + \tilde{C}_2(x)}, \qquad \tilde{C}_k(x) = \frac{C_k(x)}{(\epsilon + \beta_k)^2}, \qquad k = 1, 2, 3,$$

$$\beta_k = \sum_{l=1}^{2} h^{2l-1} \int_{x_{i-1/2}}^{x_{i+1/2}} \left(\frac{d^l}{dx^l} P_k(x) \right)^2 dx,$$

$$u_{WENO}(x) = \sum_{k=0}^{2} w_k(x) P_k(x).$$

Numerical example 1: Linear advection equation

Example 1 (A 3D Linear equation):

$$\begin{cases} u_t + u_x + u_y + u_z = 0, & -2 \le x \le 2, -2 \le y \le 2, -2 \le z \le 2; \\ u(x,y,z,0) = \sin(\frac{\pi}{2}(x+y+z)), & \end{cases} \tag{12}$$

with periodic boundary condition. We compute this 3D problem till final time $T = 1$

Computations on single grids:

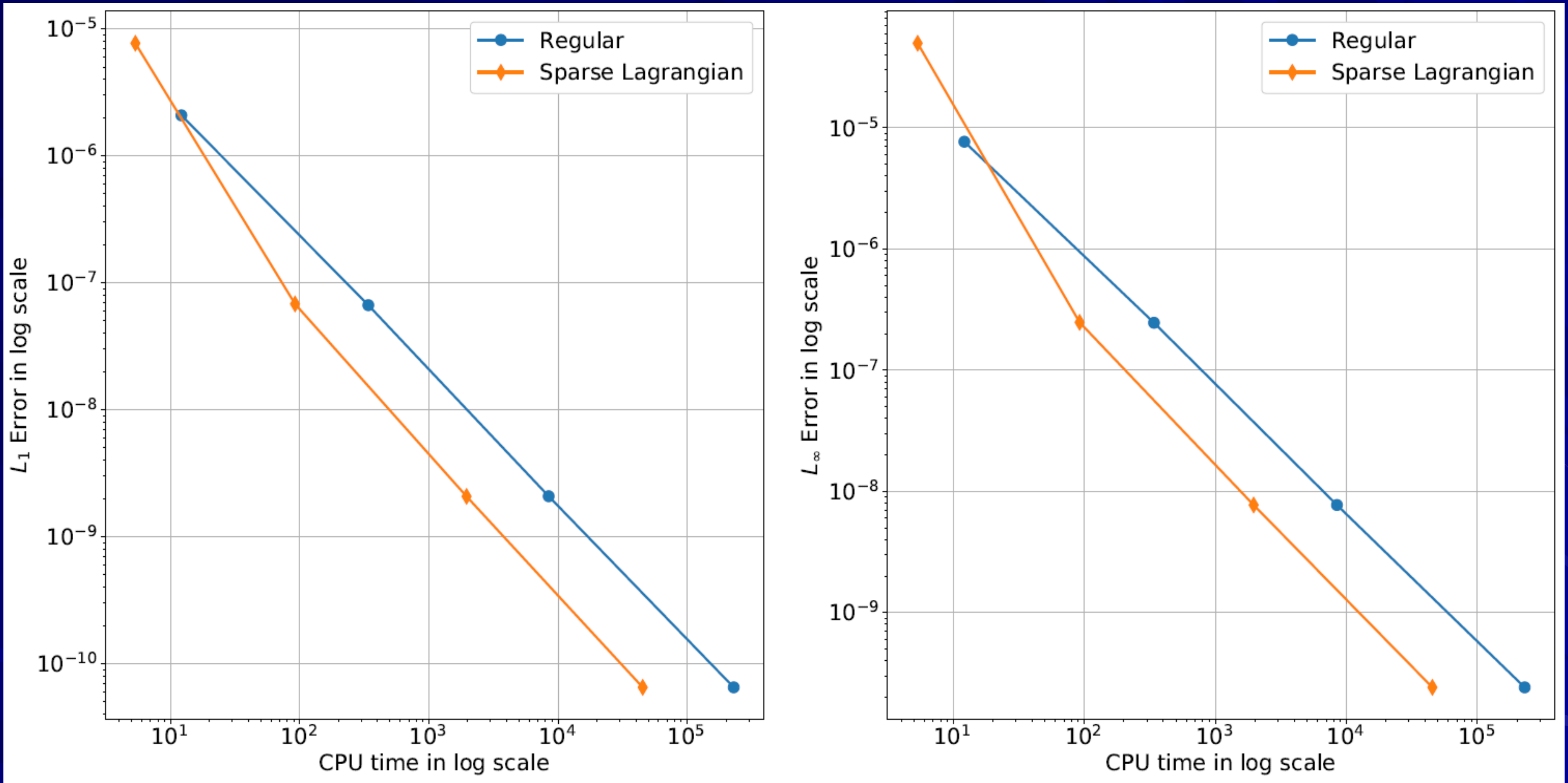
$N_h \times N_h \times N_h$	L^1 Error	Order	L^∞	Order	Time(s)
$80 \times 80 \times 80$	2.0810e-06	-	7.6743e-06	-	12.132
$160 \times 160 \times 160$	6.6581e-08	4.966	2.4607e-07	4.963	339.748
$320 \times 320 \times 320$	2.0825e-09	4.999	7.7131e-09	4.996	8461.060
$640 \times 640 \times 640$	6.5097e-11	5	2.4123e-10	4.999	231386.000

Computations on sparse grids:

N_r	Level	$N_h \times N_h \times N_h$	L^1 Error	Order	L^∞	Order	Time(s)
10	3	$80 \times 80 \times 80$	7.6908e-06	-	4.9805e-05	-	5.342
20	3	$160 \times 160 \times 160$	6.8082e-08	6.82	2.4792e-07	7.65	91.994
40	3	$320 \times 320 \times 320$	2.0761e-09	5.035	7.7136e-09	5.006	1957.354
80	3	$640 \times 640 \times 640$	6.5039e-11	4.996	2.4124e-10	4.999	45248.280

Sparse grid computations can save more than 80% CPU times on refined meshes to reach similar error levels as that on single grids.

Numerical errors vs. CPU times



Nonlinear Burgers' equation, around 80% CPU times are saved by using sparse grids on refined meshes

Example 5 (A 3D Burgers' equation):

$$\begin{cases} u_t + (\frac{u^2}{2})_x + (\frac{u^2}{2})_y + (\frac{u^2}{2})_z = 0, & (x,y,z) \in [-3,3]\times[-3,3]\times[-3,3]; \\ u(x,y,z,0) = 0.3 + 0.7\sin(\frac{\pi}{2}(x+y+z)), \end{cases} \quad (17)$$

with periodic boundary conditions. As that for the last example, we first apply both

Computations on single grids:

$N_h \times N_h \times N_h$	L^1 Error	Order	L^∞	Order	Time(s)
80 × 80 × 80	2.0866e-06	-	7.6725e-06	-	26.706
160 × 160 × 160	6.6687e-08	4.968	2.4606e-07	4.963	691.410
320 × 320 × 320	2.0836e-09	5	7.7132e-09	4.996	16868.220
640 × 640 × 640	6.5107e-11	5	2.4124e-10	4.999	444972.000

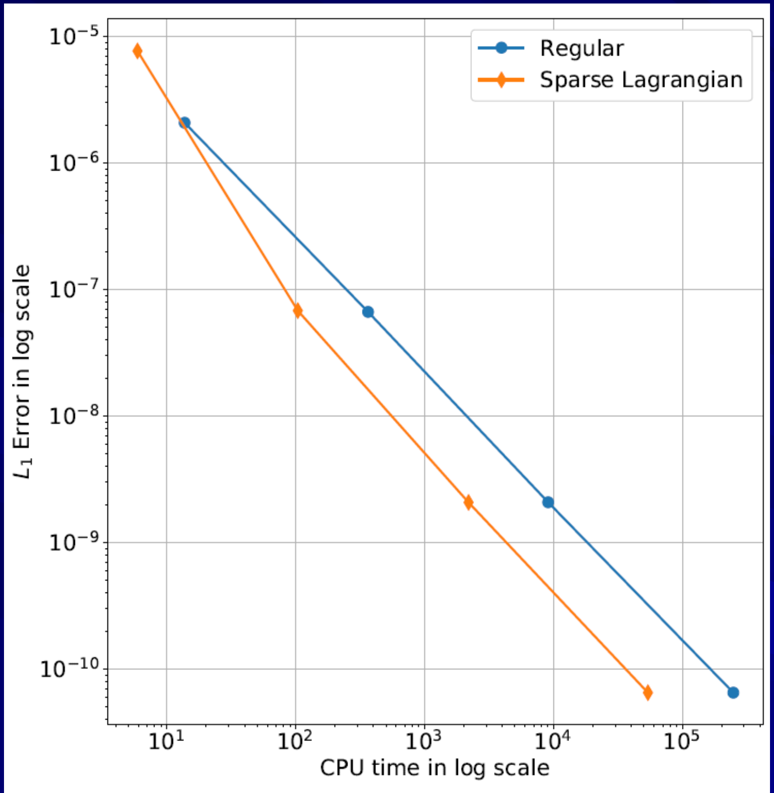
Computations on sparse grids – Lagrange prolongation:

N_r	Level	$N_h \times N_h \times N_h$	L^1 Error	Order	L^∞	Order	Time(s)
10	3	80 × 80 × 80	1.4078e-04	-	1.1642e-03	-	8.632
20	3	160 × 160 × 160	7.4122e-07	7.569	9.2799e-06	6.971	171.818
40	3	320 × 320 × 320	2.4238e-09	8.256	2.2712e-08	8.675	3733.668
80	3	640 × 640 × 640	6.4885e-11	5.223	2.4064e-10	6.56	92752.380

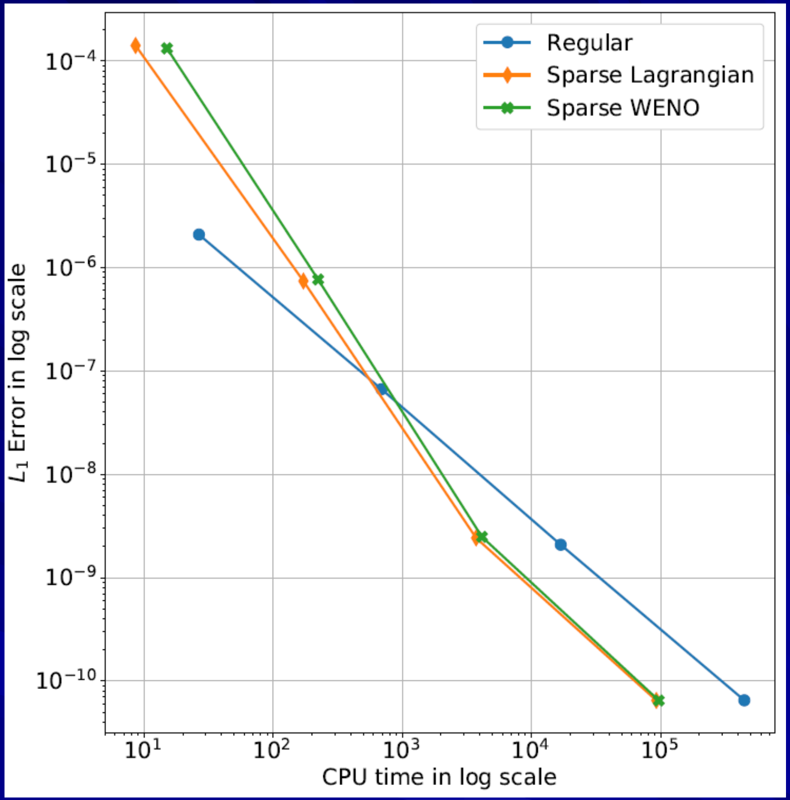
Computations on sparse grids – WENO prolongation:

N_r	Level	$N_h \times N_h \times N_h$	L^1 Error	Order	L^∞	Order	Time(s)
10	3	80 × 80 × 80	1.3225e-04	-	1.1997e-03	-	15.106
20	3	160 × 160 × 160	7.6655e-07	7.431	7.9147e-06	7.244	223.450
40	3	320 × 320 × 320	2.4830e-09	8.27	2.5650e-08	8.269	4144.632
80	3	640 × 640 × 640	6.4883e-11	5.258	2.4056e-10	6.736	96962.960

Numerical errors vs. CPU times

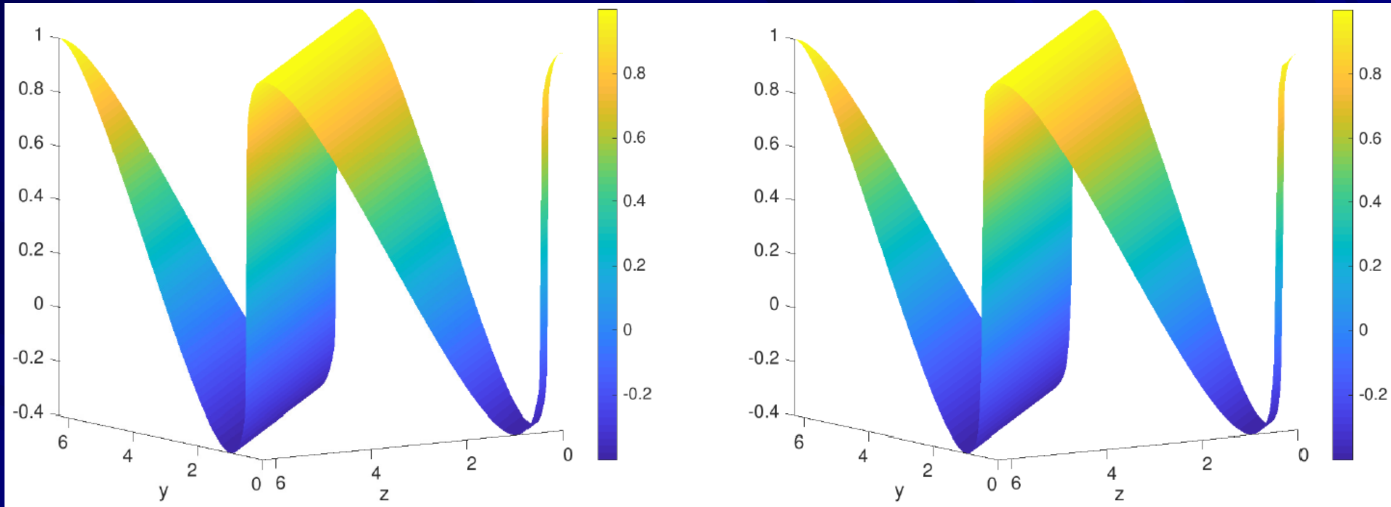


Linear scheme



WENO5 scheme

Shock wave case



(a) Regular

(b) Sparse

Figure 3.9: y-z cut with x = 0

End Time	Regular Grid Time	Sparse Grid Time	Sparse / Regular Ratio
0.52	86549.8	17893.4	0.2067

CPU time comparison

Application of the sparse grid WENO5 method to simulation of high dimensional Vlasov equation

An example of Vlasov-Boltzmann transport equation, the relaxation model:

From: A sparse grid DG method for high-dimensional transport equations and its application to kinetic simulations. W. Guo and Y. Cheng, SISC, v38, p. A3381, 2016.

$$f_t + \mathbf{v}\cdot\nabla_{\mathbf{x}} f + \mathbf{E}(t,\mathbf{x})\cdot\nabla_{\mathbf{v}} f = L(f), \tag{6.3}$$

where $L(f)$ denotes the linear relaxation operator

$$L(f) = \frac{\mu_\infty(\mathbf{v})\rho(t,\mathbf{x}) - f(t,\mathbf{x},\mathbf{v})}{\tau}, \tag{6.4}$$

and μ_∞ is an absolute Maxwellian distribution defined as

$$\mu_\infty(\mathbf{v}) = \frac{\exp(-\frac{|\mathbf{v}|^2}{2\theta})}{(2\pi\theta)^{d/2}}, \tag{6.5}$$

and

$$\rho(t,\mathbf{x}) = \int_{\mathbf{v}} f(t,\mathbf{x},\mathbf{v})\,d\mathbf{v} \tag{6.6}$$

denotes the macroscopic density. The external electric field $\mathbf{E}(t,x)$ is given by a known electrostatic potential

$$\mathbf{E}(\mathbf{x}) = -\nabla_{\mathbf{x}}\Phi(x) \;\; with \;\; \Phi(x) = \frac{|\mathbf{x}|^2}{2}. \tag{6.7}$$

Simulation of the 4D case

Initial condition:

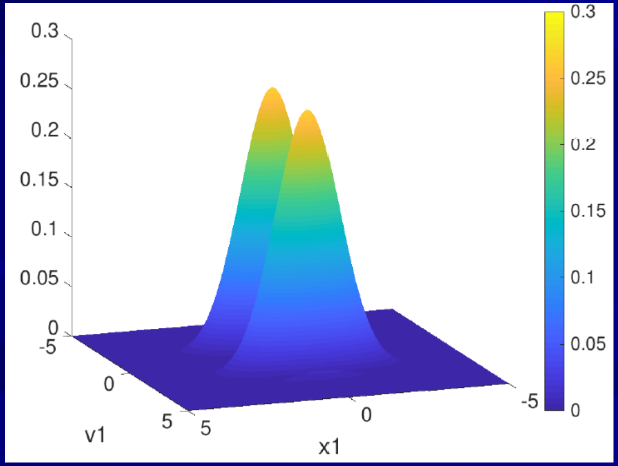
$$f(0,\mathbf{x},\mathbf{v}) = \frac{1}{s_2}\sin\left(\frac{x_1^2}{2}\right)^2 \cos\left(\frac{x_2^2}{2}\right)^2 \exp\left(-\frac{(x_1^2+x_2^2+v_1^2+v_2^2)}{2}\right)$$

Computational domain: [-5, 5] x [-5, 5] x [-5, 5] x [-5, 5]

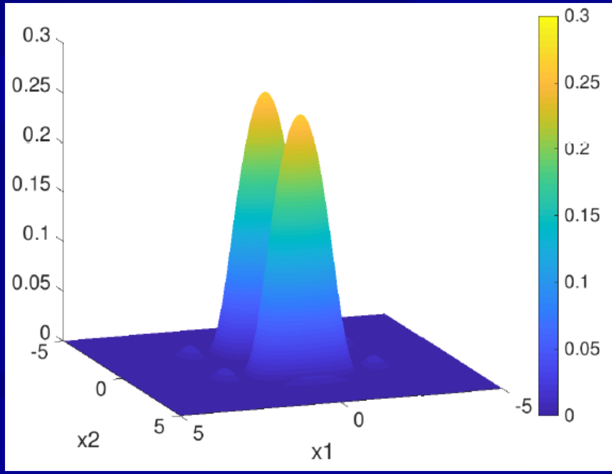
Boundary conditions: Zero Dirichlet boundary conditions.

Single grid: 80 x 80 x 80 x 80

Sparse grid: root grid 10 x 10 x 10 x 10; the most refined level 3.



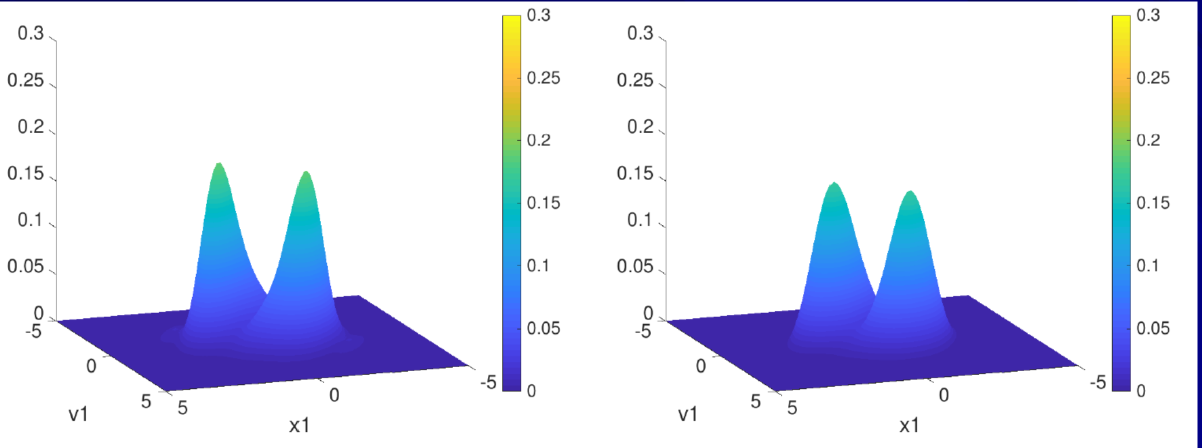
Initial condition:2D cut at x2=v2=0



Initial condition:2D cut at v1=v2=0

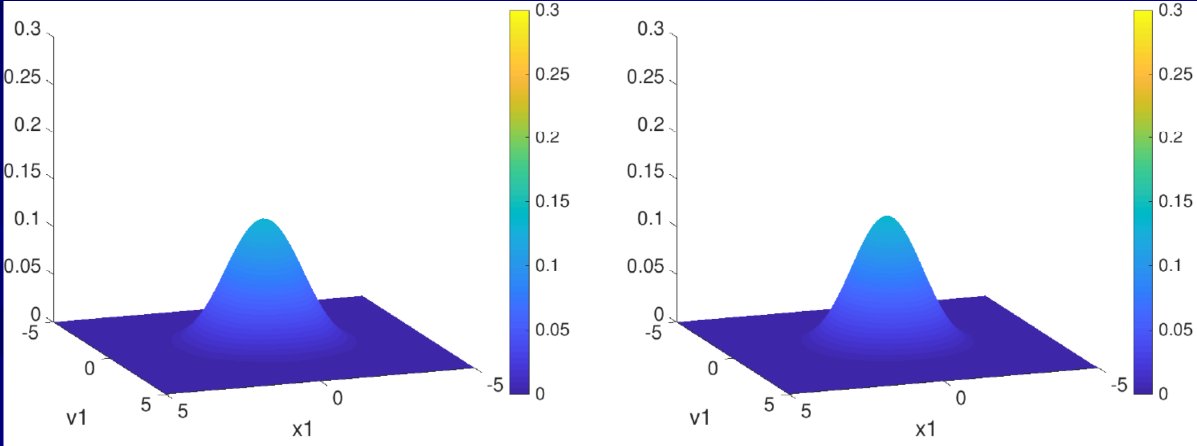
Sparse grid WENO5 simulations of 4D problem

2D cuts at x2=v2=0:



(a) Regular mesh (b) Sparse mesh

Figure 8.3: t=1.0, x2 = v2 = 0



(a) Regular mesh (b) Sparse mesh

Figure 8.5: t=3.0, x2 = v2 = 0

CPU time comparison

End Time	Regular CPU Time(s)	Regular std	Sparse CPU Time(s)	Sparse std	sparse/regular ratio
0.5	56231.125	49.6765	3918.4680	247.599210	0.06968
1.0	112539.000	2293.2772	7876.2325	72.197921	0.06998
2.0	225612.000	2278.5356	14468.5000	406.119687	0.06413
3.0	332706.250	2535.732554	22821.1250	409.147743	0.06827

- Sparse grid WENO5 simulations save 93% CPU time of that for single grid simulations.
- Comparable resolutions are obtained.

A simplified 3D Vlasov-Maxwell system

$$
\begin{aligned}
& f_t + \xi_2 f_{x_2} + (E_1 + \xi_2 B_3) f_{\xi_1} + (E_2 - \xi_1 B_3) f_{\xi_2} = 0, \\
& \frac{\partial B_3}{\partial t} = \frac{\partial E_1}{\partial x_2}, \\
& \frac{\partial E_1}{\partial t} = \frac{\partial B_3}{\partial x_2} - j_1, \\
& \frac{\partial E_2}{\partial t} = -j_2,
\end{aligned}
$$

where x_2 is the spatial variable and ξ_1, ξ_2 are the velocity variables. The system is defined on the domain $\Omega_x \times \Omega_\xi$. Ω_x denotes the physical space and $x_2 \in \Omega_x$. Ω_ξ is the velocity space and $(\xi_1, \xi_2) \in \Omega_\xi$. The probability distribution function of electrons $f = f(x_2, \xi_1, \xi_2, t)$. $E_1 = E_1(x_2, t)$ and $E_2 = E_2(x_2, t)$ are the electric field components. $B_3 = B_3(x_2, t)$ is the magnetic field component. The whole physical space has the 2D electric field $\vec{\mathbf{E}} = (E_1(x_2,t), E_2(x_2,t), 0)$ and the 1D magnetic field $\vec{\mathbf{B}} = (0, 0, B_3(x_2, t))$. The current densities $j_1(x_2, t)$ and $j_2(x_2, t)$ are

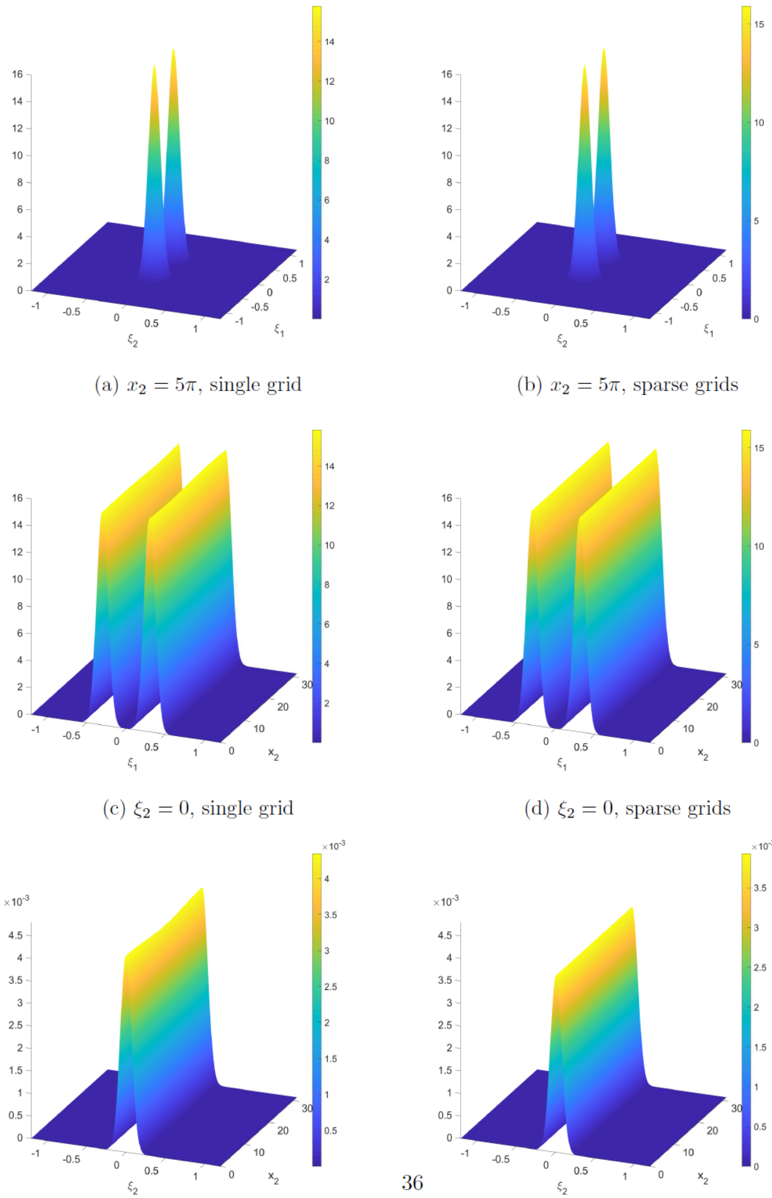
$$
j_1 = \iint_{\Omega_\xi} f(x_2, \xi_1, \xi_2, t)\xi_1 d\xi_1 d\xi_2, \quad j_2 = \iint_{\Omega_\xi} f(x_2, \xi_1, \xi_2, t)\xi_2 d\xi_1 d\xi_2. \tag{26}
$$

The initial condition of the system is

$$
\begin{aligned}
f(x_2, \xi_1, \xi_2, 0) &= \frac{1}{\pi\beta} e^{-\xi_2^2/\beta}[\delta e^{-(\xi_1 - v_{0,1})^2/\beta} + (1-\delta) e^{-(\xi_1 + v_{0,2})^2/\beta}], \\
E_1(x_2, 0) = E_2(x_2, 0) &= 0, \qquad B_3(x_2, 0) = b \sin(k_0 x_2).
\end{aligned}
$$

The computational domain is $\Omega_x = [0, 2\pi/k_0]$ and $\Omega_\xi = [-1.2, 1.2]^2$, with periodic boundary conditions applied to the system. The parameters are taken to be $\beta = 0.01, b = 0.001, \delta = 0.5, v_{0,1} = v_{0,2} = 0.3, k_0 = 0.2$ as in [35]. Here we use this interesting 3D problem to test the efficiency of the proposed fifth order sparse grid WENO scheme in this paper. For detailed physical explanations of the system and the parameters, we refer to [3, 35].

The example is from:
Z. Tao, W. Guo, Y. Cheng, Sparse grid discontinuous Galerkin methods for the Vlasov-Maxwell system. JCP: X, (2019), 3, 100022.



(a) $x_2 = 5\pi$, single grid (b) $x_2 = 5\pi$, sparse grids

(c) $\xi_2 = 0$, single grid (d) $\xi_2 = 0$, sparse grids

(e) $\xi_1 = 0$, single grid (f) $\xi_1 = 0$, sparse grids

36

Figure 11: Example 6, solution f of the 3D Vlasov-Maxwell system at time $T = 10$, by fifth order WENO scheme on sparse grids ($N_r = 20$ for root grid, finest level $N_L = 3$ in the sparse-grid computation) and the corresponding $160 \times 160 \times 160$ single grid. Plots of solutions in 2D planes with a fixed third direction coordinate. $CFL = 0.4$.

- CPU costs on sparse grids: 3585.81 seconds;
- CPU costs on single grid: 10331.04 seconds;
- 65% CPU time is saved.

Conclusions and some open problem

- We obtain efficient high order iterative schemes by combining fast sweeping methods with high order WENO techniques for solving steady state hyperbolic conservation laws.
- Very efficient computations to solve multi-dimensional hyperbolic PDEs can be achieved by using sparse grid techniques for high order WENO schemes.
- Some open problem for sparse grid WENO scheme: when shock profile is very sharp, some oscillations / noises can be observed. This is due to the last linear combination step. It will NOT affect the stability of the simulations because the linear combination step is only applied once at the final time step. How to resolve this issue is open and under further investigation.