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.
- 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.


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.

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

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

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

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

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

- Conservative flux approximations:

\[
f(u)|_{x=x_i} \approx \frac{1}{\Delta x} (\hat{f}_{i+1/2} - \hat{f}_{i-1/2})
\]

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

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

where

\[
\begin{align*}
\hat{f}_{i+1/2}^{(0)} & = \frac{1}{3} f(u_{i-2}) - \frac{7}{6} f(u_{i-1}) + \frac{11}{6} f(u_i), \\
\hat{f}_{i+1/2}^{(1)} & = -\frac{1}{6} f(u_{i-1}) + \frac{5}{6} f(u_i) + \frac{1}{3} f(u_{i+1}), \\
\hat{f}_{i+1/2}^{(2)} & = \frac{1}{3} f(u_i) + \frac{5}{6} f(u_{i+1}) - \frac{1}{6} f(u_{i+2}).
\end{align*}
\]
Nonlinear weights

\[ w_r = \frac{\alpha_r}{\alpha_1 + \alpha_2 + \alpha_3}, \quad \alpha_r = \frac{d_r}{(\epsilon + \beta_r)^2}, \quad r = 0, 1, 2. \]  

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

\[
\begin{align*}
\beta_0 & = \frac{13}{12} (f_{i-2} - 2f_{i-1} + f_i)^2 + \frac{1}{4} (f_{i-2} - 4f_{i-1} + 3f_i)^2, \\
\beta_1 & = \frac{13}{12} (f_{i-1} - 2f_i + f_{i+1})^2 + \frac{1}{4} (f_{i-1} - f_{i+1})^2, \\
\beta_2 & = \frac{13}{12} (f_i - 2f_{i+1} + f_{i+2})^2 + \frac{1}{4} (3f_i - 4f_{i+1} + f_{i+2})^2,
\end{align*}
\]  

\( f_j \) denotes \( f(u_j) \). \( \epsilon \) 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:

\[
\begin{align*}
f^+(u) & = \frac{1}{2} (f(u) + \alpha u), \\
f^-(u) & = \frac{1}{2} (f(u) - \alpha u),
\end{align*}
\]  

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.


- 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

\[ \beta_0 = (f_{i-2} - 4f_{i-1} + 3f_i)^2, \]
\[ \beta_1 = (f_{i-1} - f_{i+1})^2, \]
\[ \beta_2 = (3f_i - 4f_{i+1} + f_{i+2})^2. \]

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:

\[ U_{i+1/2}^{(1)} = U_i \quad \text{when} \quad u_{i+1/2} \geq 0 \]

\[ U_{i+1/2}^{(2)} = U_{i+1} \quad \text{when} \quad u_{i+1/2} < 0 \]

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

u: the x-direction fluid velocity
Fixed-point iterative schemes

- After WENO discretization, we obtain a nonlinear system

\[
0 = -\left(\hat{f}_{i+1/2,j} - \hat{f}_{i-1/2,j}\right)/\Delta x - \left(\hat{g}_{i,j+1/2} - \hat{g}_{i,j-1/2}\right)/\Delta y + h(u_{ij}, x, y), \quad i = 1, \ldots, N; j = 1, \ldots, M.
\]

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

\[
\begin{align*}
  u_{ij}^{n+1} &= u_{ij}^n + \frac{\gamma}{\alpha_x / \Delta x + \alpha_y / \Delta y} L(u^n_{i-r,j}, \cdots, u^n_{i+s,j}; u^n_{i,j}; u^n_{i,j-r}, \cdots, u^n_{i,j+s}) \\
  i &= 1, \cdots, N; j = 1, \cdots, M,
\end{align*}
\]

\[\Delta t_n = \frac{\gamma}{\alpha_x / \Delta x + \alpha_y / \Delta y}\]

\(\gamma\) actually represents the CFL number.

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

\[
\begin{align*}
  u_{ij}^{(1)} &= u_{ij}^n + \Delta t L(u^n_{i-r,j}, \cdots, u^n_{i+s,j}; u^n_{i,j}; u^n_{i,j-r}, \cdots, u^n_{i,j+s}), \quad i = 1, \cdots, N; j = 1, \cdots, M. \\
  u_{ij}^{(2)} &= \frac{3}{4} u_{ij}^{(1)} + \frac{1}{4} u_{ij}^n + \frac{1}{4} \Delta t L(u^{(1)}_{i-r,j}, \cdots, u^{(1)}_{i+s,j}; u^{(1)}_{i,j}; u^{(1)}_{i,j-r}, \cdots, u^{(1)}_{i,j+s}), \quad i = 1, \cdots, N; j = 1, \cdots, M. \\
  u_{ij}^{n+1} &= \frac{1}{3} u_{ij}^n + \frac{2}{3} u_{ij}^{(2)} + \frac{2}{3} \Delta t L(u^{(2)}_{i-r,j}, \cdots, u^{(2)}_{i+s,j}; u^{(2)}_{i,j}; u^{(2)}_{i,j-r}, \cdots, u^{(2)}_{i,j+s}), \quad i = 1, \cdots, N; j = 1, \cdots, M.
\end{align*}
\]
Fixed-point fast sweeping WENO schemes for hyperbolic conservation laws


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

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

• 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) \(i = 1 : N, \ j = M : 1.\)
Runge Kutta type fixed-point sweeping scheme

\[
\begin{align*}
    u_{ij}^{(1)} &= u_{ij}^n + \frac{\gamma}{\alpha_x/\Delta x + \alpha_y/\Delta y} L(u_{i-r,j}^*, \ldots, u_{i+s,j}^*, u_{ij}^n; u_{i,j-r}^*, \ldots, u_{i,j+s}^*), \\
    i &= i_1, \ldots, i_N; j = j_1, \ldots, j_M. \\

    u_{ij}^{(2)} &= u_{ij}^{(1)} + \frac{\gamma}{4(\alpha_x/\Delta x + \alpha_y/\Delta y)} L(u_{i-r,j}^{**}, \ldots, u_{i+s,j}^{**}; u_{ij}^{(1)}; u_{i,j-r}^{**}, \ldots, u_{i,j+s}^{**}), \\
    i &= i_1, \ldots, i_N; j = j_1, \ldots, 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}^{***}, \ldots, u_{i+s,j}^{***}; u_{ij}^{(2)}; u_{i,j-r}^{***}, \ldots, u_{i,j+s}^{***}), \\
    i &= i_1, \ldots, i_N; j = j_1, \ldots, j_M.
\end{align*}
\]
Numerical Example I: **Burgers equations**

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

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

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

<table>
<thead>
<tr>
<th>points</th>
<th>$L_1$ error</th>
<th>$L_1$ order</th>
<th>$L_\infty$ error</th>
<th>$L_\infty$ index $(i,j)$</th>
<th>$L_\infty$ order</th>
<th>iter #</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2.46e-6</td>
<td>1.23e-5</td>
<td>(9,9)</td>
<td></td>
<td></td>
<td>270</td>
<td>4.30e-2</td>
</tr>
<tr>
<td>20</td>
<td>6.31e-8</td>
<td>5.29</td>
<td>3.50e-7</td>
<td>(19,19)</td>
<td>5.14</td>
<td>342</td>
<td>0.23</td>
</tr>
<tr>
<td>40</td>
<td>1.37e-9</td>
<td>5.52</td>
<td>8.9e-9</td>
<td>(39,39)</td>
<td>5.28</td>
<td>513</td>
<td>1.42</td>
</tr>
<tr>
<td>80</td>
<td>2.95e-11</td>
<td>5.54</td>
<td>1.93e-10</td>
<td>(79,79)</td>
<td>5.54</td>
<td>855</td>
<td>9.54</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>points</th>
<th>$L_1$ error</th>
<th>$L_1$ order</th>
<th>$L_\infty$ error</th>
<th>$L_\infty$ index $(i,j)$</th>
<th>$L_\infty$ order</th>
<th>iter #</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2.46e-6</td>
<td>1.23e-5</td>
<td>(9,9)</td>
<td></td>
<td></td>
<td>159</td>
<td>2.41e-2</td>
</tr>
<tr>
<td>20</td>
<td>6.31e-8</td>
<td>5.29</td>
<td>3.50e-7</td>
<td>(19,19)</td>
<td>5.14</td>
<td>186</td>
<td>0.12</td>
</tr>
<tr>
<td>40</td>
<td>1.37e-9</td>
<td>5.52</td>
<td>8.9e-9</td>
<td>(39,39)</td>
<td>5.28</td>
<td>273</td>
<td>0.70</td>
</tr>
<tr>
<td>80</td>
<td>2.95e-11</td>
<td>5.54</td>
<td>2.07e-10</td>
<td>(79,79)</td>
<td>5.44</td>
<td>450</td>
<td>4.72</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>points</th>
<th>$L_1$ error</th>
<th>$L_1$ order</th>
<th>$L_\infty$ error</th>
<th>$L_\infty$ index $(i,j)$</th>
<th>$L_\infty$ order</th>
<th>iter #</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2.46e-6</td>
<td>1.23e-5</td>
<td>(9,9)</td>
<td></td>
<td></td>
<td>1195</td>
<td>0.18</td>
</tr>
<tr>
<td>20</td>
<td>6.31e-8</td>
<td>5.29</td>
<td>3.50e-7</td>
<td>(19,19)</td>
<td>5.14</td>
<td>1439</td>
<td>0.92</td>
</tr>
<tr>
<td>40</td>
<td>1.37e-9</td>
<td>5.52</td>
<td>8.9e-9</td>
<td>(39,39)</td>
<td>5.28</td>
<td>1822</td>
<td>3.16</td>
</tr>
<tr>
<td>80</td>
<td>2.95e-11</td>
<td>5.54</td>
<td>2.08e-10</td>
<td>(79,79)</td>
<td>5.43</td>
<td>3178</td>
<td>33.67</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>points</th>
<th>$L_1$ error</th>
<th>$L_1$ order</th>
<th>$L_\infty$ error</th>
<th>$L_\infty$ index $(i,j)$</th>
<th>$L_\infty$ order</th>
<th>iter #</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2.46e-6</td>
<td>1.23e-5</td>
<td>(9,9)</td>
<td></td>
<td></td>
<td>113</td>
<td>1.77e-2</td>
</tr>
<tr>
<td>20</td>
<td>6.31e-8</td>
<td>5.29</td>
<td>3.50e-7</td>
<td>(19,19)</td>
<td>5.14</td>
<td>133</td>
<td>8.39e-2</td>
</tr>
<tr>
<td>40</td>
<td>1.37e-9</td>
<td>5.52</td>
<td>8.9e-9</td>
<td>(39,39)</td>
<td>5.28</td>
<td>181</td>
<td>0.47</td>
</tr>
<tr>
<td>80</td>
<td>2.95e-11</td>
<td>5.54</td>
<td>2.07e-10</td>
<td>(79,79)</td>
<td>5.44</td>
<td>286</td>
<td>3.04</td>
</tr>
</tbody>
</table>
### Table 21: New Smoothness Indicator WENO5 with Runge-Kutta in time. CFL: 1.0

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>points</th>
<th>$L_1$ error</th>
<th>$L_1$ order</th>
<th>$L_\infty$ error</th>
<th>$L_\infty$ index $(i,j)$</th>
<th>$L_\infty$ order</th>
<th>iter #</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4.33e-6</td>
<td></td>
<td>1.41e-5</td>
<td>(9,9)</td>
<td></td>
<td>110</td>
<td>1.59e-2</td>
</tr>
<tr>
<td>20</td>
<td>9.33e-8</td>
<td>5.54</td>
<td>3.85e-7</td>
<td>(19,19)</td>
<td>5.19</td>
<td>136</td>
<td>8.00e-2</td>
</tr>
<tr>
<td>40</td>
<td>2.03e-9</td>
<td>5.53</td>
<td>1.08e-8</td>
<td>(39,39)</td>
<td>5.16</td>
<td>193</td>
<td>0.47</td>
</tr>
<tr>
<td>80</td>
<td>4.30e-11</td>
<td>5.56</td>
<td>2.82e-10</td>
<td>(79,79)</td>
<td>5.26</td>
<td>289</td>
<td>2.84</td>
</tr>
</tbody>
</table>
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 $200 \times 100$ 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.
• 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 $135^\circ$ oblique shock of $M_\infty = 2$ 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.


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

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

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 $\gamma_{1,2} = \frac{1}{11}$, $\gamma_{2,2} = \frac{10}{11}$, $\gamma_{1,3} = \frac{1}{11}$, $\gamma_{2,3} = \frac{10}{11}$, $\gamma_{3,3} = \frac{100}{111}$.

Step 3. Compute the smoothness indicators $\beta_k$, 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]:

$$\beta_k = \frac{2k - 2}{\alpha = 1} \int_{x_{i-1/2}}^{x_{i+1/2}} \Delta x^{2\alpha - 1} \left( \frac{d^\alpha p_k(x)}{dx^\alpha} \right)^2 dx, \quad k = 2, 3.$$

The only exception is $\beta_1$, 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 $\tau$
which depends on the absolute differences between the smoothness indicators is calculated: \( \tau = \left( \frac{\sum_{l_1=1}^{2} |\beta_3 - \beta_{l_1}|}{2} \right)^2 \). The nonlinear weights are then computed as

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

\( \varepsilon \) is a small value to avoid that the denominator becomes zero. In this paper, \( \varepsilon \) 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

<table>
<thead>
<tr>
<th>FE Jacobi, $\gamma=0.1$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$N \times N$</td>
<td>$L_1$ error</td>
</tr>
<tr>
<td>10 x 10</td>
<td>6.74E-04</td>
</tr>
<tr>
<td>20 x 20</td>
<td>1.30E-05</td>
</tr>
<tr>
<td>30 x 30</td>
<td>1.84E-06</td>
</tr>
<tr>
<td>40 x 40</td>
<td>4.49E-07</td>
</tr>
<tr>
<td>50 x 50</td>
<td>1.50E-07</td>
</tr>
<tr>
<td>60 x 60</td>
<td>6.08E-08</td>
</tr>
<tr>
<td>70 x 70</td>
<td>2.83E-08</td>
</tr>
<tr>
<td>80 x 80</td>
<td>1.46E-08</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RK Jacobi, $\gamma=1.0$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$N \times N$</td>
<td>$L_1$ error</td>
</tr>
<tr>
<td>10 x 10</td>
<td>7.41E-04</td>
</tr>
<tr>
<td>20 x 20</td>
<td>1.31E-05</td>
</tr>
<tr>
<td>30 x 30</td>
<td>1.85E-06</td>
</tr>
<tr>
<td>40 x 40</td>
<td>4.51E-07</td>
</tr>
<tr>
<td>50 x 50</td>
<td>1.50E-07</td>
</tr>
<tr>
<td>60 x 60</td>
<td>6.10E-08</td>
</tr>
<tr>
<td>70 x 70</td>
<td>2.84E-08</td>
</tr>
<tr>
<td>80 x 80</td>
<td>1.46E-08</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FE fast sweeping, $\gamma=1.0$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$N \times N$</td>
<td>$L_1$ error</td>
</tr>
<tr>
<td>10 x 10</td>
<td>6.62E-04</td>
</tr>
<tr>
<td>20 x 20</td>
<td>1.30E-05</td>
</tr>
<tr>
<td>30 x 30</td>
<td>1.84E-06</td>
</tr>
<tr>
<td>40 x 40</td>
<td>4.49E-07</td>
</tr>
<tr>
<td>50 x 50</td>
<td>1.50E-07</td>
</tr>
<tr>
<td>60 x 60</td>
<td>6.08E-07</td>
</tr>
<tr>
<td>70 x 70</td>
<td>2.83E-08</td>
</tr>
<tr>
<td>80 x 80</td>
<td>1.46E-08</td>
</tr>
</tbody>
</table>

Table 4: Example 4, A 2D Euler system of equations with source terms. Accuracy, iteration 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.

---

<table>
<thead>
<tr>
<th>γ : CFL number</th>
<th>iteration number</th>
<th>final time</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>12046</td>
<td>5.09</td>
<td>618.36</td>
</tr>
<tr>
<td>0.2</td>
<td>Not convergent</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**RK Jacobi scheme**

<table>
<thead>
<tr>
<th>γ : CFL number</th>
<th>iteration number</th>
<th>final time</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>11268</td>
<td>4.76</td>
<td>579.22</td>
</tr>
<tr>
<td>0.4</td>
<td>8454</td>
<td>4.76</td>
<td>436.33</td>
</tr>
<tr>
<td>0.5</td>
<td>6762</td>
<td>4.76</td>
<td>348.09</td>
</tr>
<tr>
<td>0.6</td>
<td>5634</td>
<td>4.76</td>
<td>289.89</td>
</tr>
<tr>
<td>0.7</td>
<td>Not convergent</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**FE fast sweeping scheme**

<table>
<thead>
<tr>
<th>γ : CFL number</th>
<th>iteration number</th>
<th>final time</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>3651</td>
<td>4.62</td>
<td>188.14</td>
</tr>
<tr>
<td>0.4</td>
<td>2722</td>
<td>4.59</td>
<td>140.03</td>
</tr>
<tr>
<td>0.5</td>
<td>2170</td>
<td>4.57</td>
<td>111.91</td>
</tr>
<tr>
<td>0.6</td>
<td>1934</td>
<td>4.89</td>
<td>98.92</td>
</tr>
<tr>
<td>0.7</td>
<td>Not convergent</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
A supersonic flow past an airfoil problem

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

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

\[ u_t + \nabla \cdot \vec{f}(u) = 0, \]  \hspace{1cm} (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 5\textsuperscript{th} 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.


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

- 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 \( \Omega^{l_1,l_2} \), 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 \( \Omega^{l_1,l_2} \), 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 \( \Omega^{N_L,N_L} \). 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

\[ d: \text{ Spatial dimension of the PDE} \]

\[ l: \text{ The most refined level} \]

\[ I_d = (l_1, \ldots, l_d): \text{ A sparse grid with levels } l_1, \ldots, l_d \text{ on each direction.} \]

\[ u_{l_d}^f: \text{ Numerical solution at the sparse grid } I_d \text{ after prolongation to the most refined grid} \]

\[ u_l^c: \text{ The solution after combination.} \]

The general formula for combination is,

\[ u_l^c = \sum_{m=l}^{l+d-1} (-1)^{d+l-(m+1)} \binom{d-1}{m-l} \sum_{|I_d|=m-(d-1)} u_{l_d}^f, \]
5th order WENO prolongation / interpolation

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

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

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

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

\[ u_{\text{WENO}}(x) = \sum_{k=0}^{2} w_k(x) P_k(x). \]
Example 1 (A 3D Linear equation):

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

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

### Computations on single grids:

<table>
<thead>
<tr>
<th>$N_h \times N_h \times N_h$</th>
<th>$L^1$ Error</th>
<th>Order</th>
<th>$L^\infty$</th>
<th>Order</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$80 \times 80 \times 80$</td>
<td>2.0810e-06</td>
<td>-</td>
<td>7.6743e-06</td>
<td>-</td>
<td>12.132</td>
</tr>
<tr>
<td>$160 \times 160 \times 160$</td>
<td>6.6581e-08</td>
<td>4.966</td>
<td>2.4607e-07</td>
<td>4.963</td>
<td>339.748</td>
</tr>
<tr>
<td>$320 \times 320 \times 320$</td>
<td>2.0825e-09</td>
<td>4.999</td>
<td>7.7131e-09</td>
<td>4.996</td>
<td>8461.060</td>
</tr>
<tr>
<td>$640 \times 640 \times 640$</td>
<td>6.5097e-11</td>
<td>5</td>
<td>2.4123e-10</td>
<td>4.999</td>
<td>231386.000</td>
</tr>
</tbody>
</table>

### Computations on sparse grids:

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

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

![Graphs showing numerical errors and CPU times comparison between regular and sparse Lagrangian methods.](image-url)
Nonlinear Burgers’ equation, around 80% CPU times are saved by using sparse grids on refined meshes

Example 5 (A 3D Burgers’ equation):
\[
\begin{align*}
\frac{u_t}{\frac{t^2}{2}} + \left( \frac{u^2}{2} \right)_x + \left( \frac{u^2}{2} \right)_y + \left( \frac{u^2}{2} \right)_z &= 0, \\
(x, y, z) &\in [-3,3] \times [-3,3] \times [-3,3]; \\
u(x, y, z, 0) &= 0.3 + 0.7 \sin \left( \frac{x}{2} (x + y + z) \right),
\end{align*}
\]  
with periodic boundary conditions. As that for the last example, we first apply both

### Computations on single grids:

<table>
<thead>
<tr>
<th>(N_h \times N_h \times N_h)</th>
<th>(L^1) Error</th>
<th>Order</th>
<th>(L^\infty)</th>
<th>Order</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>80 \times 80 \times 80</td>
<td>2.0866e-06</td>
<td>-</td>
<td>7.6725e-06</td>
<td>-</td>
<td>26.706</td>
</tr>
<tr>
<td>160 \times 160 \times 160</td>
<td>6.6687e-08</td>
<td>4.968</td>
<td>2.4606e-07</td>
<td>4.963</td>
<td>691.410</td>
</tr>
<tr>
<td>320 \times 320 \times 320</td>
<td>2.0836e-09</td>
<td>5</td>
<td>7.7132e-09</td>
<td>4.996</td>
<td>16868.220</td>
</tr>
<tr>
<td>640 \times 640 \times 640</td>
<td>6.5107e-11</td>
<td>5</td>
<td>2.4124e-10</td>
<td>4.999</td>
<td>444972.000</td>
</tr>
</tbody>
</table>

### Computations on sparse grids – Lagrange prolongation:

<table>
<thead>
<tr>
<th>(N_r)</th>
<th>Level</th>
<th>(N_h \times N_h \times N_h)</th>
<th>(L^1) Error</th>
<th>Order</th>
<th>(L^\infty)</th>
<th>Order</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3</td>
<td>80 \times 80 \times 80</td>
<td>1.4078e-04</td>
<td>-</td>
<td>1.1642e-03</td>
<td>-</td>
<td>8.632</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>160 \times 160 \times 160</td>
<td>7.4122e-07</td>
<td>7.569</td>
<td>9.2799e-06</td>
<td>6.971</td>
<td>171.818</td>
</tr>
<tr>
<td>40</td>
<td>3</td>
<td>320 \times 320 \times 320</td>
<td>2.4238e-09</td>
<td>8.256</td>
<td>2.2712e-08</td>
<td>8.675</td>
<td>3733.668</td>
</tr>
<tr>
<td>80</td>
<td>3</td>
<td>640 \times 640 \times 640</td>
<td>6.4885e-11</td>
<td>5.223</td>
<td>2.4064e-10</td>
<td>6.56</td>
<td>92752.380</td>
</tr>
</tbody>
</table>

### Computations on sparse grids – WENO prolongation:

<table>
<thead>
<tr>
<th>(N_r)</th>
<th>Level</th>
<th>(N_h \times N_h \times N_h)</th>
<th>(L^1) Error</th>
<th>Order</th>
<th>(L^\infty)</th>
<th>Order</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3</td>
<td>80 \times 80 \times 80</td>
<td>1.3225e-04</td>
<td>-</td>
<td>1.1997e-03</td>
<td>-</td>
<td>15.106</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>160 \times 160 \times 160</td>
<td>7.6655e-07</td>
<td>7.431</td>
<td>7.9147e-06</td>
<td>7.244</td>
<td>223.450</td>
</tr>
<tr>
<td>40</td>
<td>3</td>
<td>320 \times 320 \times 320</td>
<td>2.4830e-09</td>
<td>8.27</td>
<td>2.5650e-08</td>
<td>8.269</td>
<td>4144.632</td>
</tr>
<tr>
<td>80</td>
<td>3</td>
<td>640 \times 640 \times 640</td>
<td>6.4883e-11</td>
<td>5.258</td>
<td>2.4056e-10</td>
<td>6.736</td>
<td>96962.960</td>
</tr>
</tbody>
</table>
Numerical errors vs. CPU times

**Linear scheme**

**WENO5 scheme**
Shock wave case

Figure 3.9: y-z cut with x = 0

<table>
<thead>
<tr>
<th>End Time</th>
<th>Regular Grid Time</th>
<th>Sparse Grid Time</th>
<th>Sparse / Regular Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.52</td>
<td>86549.8</td>
<td>17893.4</td>
<td>0.2067</td>
</tr>
</tbody>
</table>

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:

\[ f_t + \mathbf{v} \cdot \nabla_x f + \mathbf{E}(t,x) \cdot \nabla_v f = L(f), \]  

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

\[ L(f) = \frac{\mu_\infty(\mathbf{v}) \rho(t,x) - f(t,x,v)}{\tau}, \]  

and \( \mu_\infty \) is an absolute Maxwellian distribution defined as

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

and

\[ \rho(t,x) = \int f(t,x,v) dv \]  

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

\[ \mathbf{E}(x) = -\nabla_x \Phi(x) \quad \text{with} \quad \Phi(x) = \frac{|x|^2}{2}. \]
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 \( x_2 = v_2 = 0 \)

Initial condition: 2D cut at \( v_1 = v_2 = 0 \)

5/20/2021
Sparse grid WENO5 simulations of 4D problem

2D cuts at $x_2 = v_2 = 0$:

![Figure 8.3: $t=1.0$, $x_2 = v_2 = 0$](image1)

![Figure 8.5: $t=3.0$, $x_2 = v_2 = 0$](image2)
CPU time comparison

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

- Sparse grid WENO5 simulations save 93% CPU time of that for single grid simulations.
- Comparable resolutions are obtained.
A simplified 3D Vlasov-Maxwell system

\[ f_t + \xi_2 f_{x_2} + (E_1 + \xi_2 B_3) f_{\xi_1} + (E_2 - \xi_1 B_3) f_{\xi_2} = 0, \]
\[ \frac{\partial B_3}{\partial t} = \frac{\partial E_1}{\partial x_2}, \]
\[ \frac{\partial E_1}{\partial t} = \frac{\partial B_3}{\partial x_2} - j_1, \]
\[ \frac{\partial E_2}{\partial t} = -j_2, \]

where \( x_2 \) is the spatial variable and \( \xi_1, \xi_2 \) are the velocity variables. The system is defined on the domain \( \Omega_x \times \Omega_\xi \). \( \Omega_x \) denotes the physical space and \( x_2 \in \Omega_x \). \( \Omega_\xi \) 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 \( \vec{E} = (E_1(x_2, t), E_2(x_2, t), 0) \) and the 1D magnetic field \( \vec{B} = (0, 0, B_3(x_2, t)) \). The current densities \( j_1(x_2, t) \) and \( j_2(x_2, t) \) are

\[ j_1 = \iiint_{\Omega_\xi} f(x_2, \xi_1, \xi_2, t) \xi_1 d\xi_1 d\xi_2, \quad j_2 = \iiint_{\Omega_\xi} f(x_2, \xi_1, \xi_2, t) \xi_2 d\xi_1 d\xi_2. \] (26)

The initial condition of the system is

\[ f(x_2, \xi_1, \xi_2, 0) = \frac{1}{\pi \beta} e^{-\xi_3^2/\beta} [\delta e^{-(\xi_1 - v_{0,1})^2/\beta} + (1 - \delta) e^{-(\xi_1 + v_{0,2})^2/\beta}], \]
\[ E_1(x_2, 0) = E_2(x_2, 0) = 0, \quad B_3(x_2, 0) = b \sin(k_0 x_2). \]

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].
• 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.