Communications on Applied Mathematics and Computation
https://doi.org/10.1007/s42967-022-00209-x

TECHNICAL NOTE

Sparse-Grid Implementation of Fixed-Point Fast Sweeping WENO Schemes for Eikonal Equations

Zachary M. Miksis[1] · **Yong-Tao Zhang**[1]

Received: 20 January 2022 / Revised: 29 June 2022 / Accepted: 10 August 2022

Abstract
Fixed-point fast sweeping methods are a class of explicit iterative methods developed in the literature to efficiently solve steady-state solutions of hyperbolic partial differential equations (PDEs). As other types of fast sweeping schemes, fixed-point fast sweeping methods use the Gauss-Seidel iterations and alternating sweeping strategy to cover characteristics of hyperbolic PDEs in a certain direction simultaneously in each sweeping order. The resulting iterative schemes have a fast convergence rate to steady-state solutions. Moreover, an advantage of fixed-point fast sweeping methods over other types of fast sweeping methods is that they are explicit and do not involve the inverse operation of any nonlinear local system. Hence, they are robust and flexible, and have been combined with high-order accurate weighted essentially non-oscillatory (WENO) schemes to solve various hyperbolic PDEs in the literature. For multidimensional nonlinear problems, high-order fixed-point fast sweeping WENO methods still require quite a large amount of computational costs. In this technical note, we apply sparse-grid techniques, an effective approximation tool for multidimensional problems, to fixed-point fast sweeping WENO methods for reducing their computational costs. Here, we focus on fixed-point fast sweeping WENO schemes with third-order accuracy (Zhang et al. 2006 [41]), for solving Eikonal equations, an important class of static Hamilton-Jacobi (H-J) equations. Numerical experiments on solving multidimensional Eikonal equations and a more general static H-J equation are performed to show that the sparse-grid computations of the fixed-point fast sweeping WENO schemes achieve large savings of CPU times on refined meshes, and at the same time maintain comparable accuracy and resolution with those on corresponding regular single grids.

Keywords Fixed-point fast sweeping methods · Weighted essentially non-oscillatory (WENO) schemes · Sparse grids · Static Hamilton-Jacobi (H - J) equations · Eikonal equations

Research was partially supported by the NSF Grant DMS-1620108.

✉ Yong-Tao Zhang
yzhang10@nd.edu

Zachary M. Miksis
zmiksis@nd.edu

[1] Department of Applied and Computational Mathematics and Statistics, University of Notre Dame, Notre Dame, IN 46556, USA

Published online: 10 October 2022

Mathematics Subject Classification 65N06 · 65N22 · 35L99

1 Introduction

In this technical note, we study an efficient approach to reduce the computational costs for solving the multidimensional Eikonal equations

$$\begin{cases} |\nabla \phi(\mathbf{x})| = f(\mathbf{x}), & \mathbf{x} \in \Omega \backslash \Gamma \subset \mathbb{R}^d, \\ \phi(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Gamma \subset \Omega, \end{cases} \tag{1}$$

where Ω is a d-dimension computational domain in $\mathbb{R}^d$, and Γ is a subset of Ω. The given functions $f(\mathbf{x})$ and $g(\mathbf{x})$ are Lipschitz continuous, and $f(\mathbf{x})$ is positive. The Eikonal equations are a very important class of static Hamilton-Jacobi (H-J) equations [7]

$$\begin{cases} H(\mathbf{x}, \nabla \phi(\mathbf{x})) = f(\mathbf{x}), & \mathbf{x} \in \Omega \backslash \Gamma \subset \mathbb{R}^d, \\ \phi(\mathbf{x}) = g(\mathbf{x}), & \mathbf{x} \in \Gamma \subset \Omega, \end{cases} \tag{2}$$

where H is the Hamiltonian. The numerical computations of Eikonal equations appear in many applications, such as the optimal control, image processing and computer vision, geometric optics, seismic waves, level set methods, etc.

Due to the nonlinearity of the equations and possible singularities in their solutions, it is challenging to design efficient and high-order accurate numerical methods for solving static H-J equations such as the Eikonal equations (1). In the literature, a popular approach is to discretize (1) into a nonlinear system and then design a fast numerical method to solve the nonlinear system. Among such methods are the fast marching method and the fast sweeping method. The fast marching method uses the Dijkstra's algorithm [8] and updates the solution by following the Eikonal equations' causality sequentially, e.g., see [32–34]. In the fast sweeping method [9, 16, 29, 30, 43, 44], Gauss-Seidel iterations with alternating orderings are combined with upwind schemes. Different from the fast marching method, the fast sweeping method is an iterative method and follows the Eikonal equations' causality along characteristics in a parallel way, i.e., each Gauss-Seidel iteration with a specific sweeping ordering covers a family of characteristics in a certain direction simultaneously.

The iterative framework of the fast sweeping method provides certain flexibility to incorporate high-order accuracy schemes for hyperbolic PDEs, such as weighted essentially non-oscillatory (WENO) methods [38, 42] or discontinuous Galerkin (DG) [19, 36, 40] methods, into it for developing high-order fast sweeping methods. In [41], fixed-point fast sweeping WENO methods were designed to solve static H-J equations. Different from other fast sweeping methods, fixed-point fast sweeping methods adopt the Gauss-Seidel idea and alternate sweeping strategy to the time-marching type of fixed-point iterations. They are explicit schemes and do not involve the inverse operation of nonlinear local systems which have to be done in other types of fast sweeping methods, and hence are much easier to be applied in solving various hyperbolic equations using any monotone numerical fluxes and high-order nonlinear WENO approximations. For example, how efficiently solving steady-state problems of hyperbolic conservation laws is important and challenging [4, 6]. In [20, 37], fixed-point fast sweeping WENO methods were applied in solving nonlinear hyperbolic conservation laws. Numerical experiments performed in [20, 37, 41] show that more than 50% of computational costs are saved by using fixed-point fast sweeping

methods rather than direct time-marching methods to converge to steady states of high-order WENO schemes.

Since high-order WENO methods require more operations than many other schemes due to their sophisticated nonlinearity and high-order accuracy, the associated computational costs increase significantly when the number of grid points is large for multidimensional problems. Sparse-grid techniques, an efficient approach for solving high-dimensional problems, have been developed in the literature to reduce the number of grid points needed in the simulations. See [3, 10] for a review. In 1991, sparse-grid techniques were introduced in [39] to reduce the number of degrees of freedom in the finite-element method. As an approach for the practical implementation of sparse-grid techniques, the sparse-grid combination technique was developed in [13]. The main idea of the sparse-grid combination technique is to compute the final solution as a linear combination of solutions on semi-coarsened grids, and the coefficients of the linear combination are taken, such that there is a canceling in leading-order error terms and the resulting accuracy order is kept to be the same as that on a single full grid. The sparse-grid combination technique was applied to linear schemes in [17, 18] in early time. Recently, it has been applied to nonlinear WENO schemes in [21, 22, 45] for solving hyperbolic conservation laws or convection-diffusion equations, where numerical results show that significant computational times are saved, while both accuracy and stability of the nonlinear WENO schemes are maintained for simulations on sparse grids. In this technical note, we follow the way in our previous work and apply the sparse-grid combination technique to fixed-point fast sweeping WENO methods for solving multidimensional Eikonal equations. Both a forward Euler (FE) fixed-point fast sweeping WENO scheme and a Runge-Kutta (RK) type fixed-point fast sweeping WENO scheme with third-order accuracy developed in [41] are used in this paper. The rest of the paper is organized as follows. In Sect. 2, we describe the algorithm how to apply the sparse-grid combination technique to the fixed-point fast sweeping WENO schemes. In Sect. 3, various numerical experiments including solving multidimensional Eikonal equations and a more general static H-J equation with smooth or non-smooth solutions are carried out to show that the sparse-grid computations of the fixed-point fast sweeping WENO schemes save a large amount of CPU time, especially on refined meshes, and at the same time maintain comparable simulation results with those on corresponding regular single grids. Conclusions are given in Sect. 4.

2 Description of the Numerical Algorithm

In this section, we first review the fixed-point fast sweeping WENO schemes in [41], and then describe the algorithm to implement them on sparse grids.

2.1 The Fixed-Point Fast Sweeping WENO Schemes

The fixed-point fast sweeping WENO schemes in [41] were developed by applying the Gauss-Seidel idea and alternating sweeping strategy to the time-marching schemes to solve the static H-J equations (2). In this paper, we use both the forward Euler fixed-point fast sweeping WENO scheme (FE FPFS-WENO) based on the forward Euler time-marching scheme and third-order WENO approximations to spatial derivatives, and the RK fixed-point fast sweeping WENO scheme (RK FPFS-WENO) based on the second-order total variation diminishing (TVD) RK time-marching scheme [35] and third-order WENO approximations to spatial

derivatives. Here, we take the two-dimensional (2D) case as an example to describe the methods, which is similar to higher dimensional cases. The computational domain Ω is partitioned by a Cartesian grid $\{(x_i, y_j), 1 \leqslant i \leqslant I, 1 \leqslant j \leqslant J\}$ with uniform grid sizes h_x and h_y in the x- and y-directions, respectively. Denote the viscosity numerical solution of (2) at a grid point (x_i, y_j) by $\phi_{i,j}$. The **FE fixed-point fast sweeping scheme** [41] has the form

$$\phi_{i,j}^{(n+1)} = \phi_{i,j}^{n} + \gamma \left(\frac{1}{\frac{\alpha_x}{h_x} + \frac{\alpha_y}{h_y}} \right) \left[f_{i,j} - \hat{H}((\phi_x^*)_{i,j}^-, (\phi_x^*)_{i,j}^+; (\phi_y^*)_{i,j}^-, (\phi_y^*)_{i,j}^+) \right], \tag{3}$$

and the RK fixed-point fast sweeping scheme [41] has the following form:

$$\phi_{i,j}^{(1)} = \phi_{i,j}^{n} + \gamma \left(\frac{1}{\frac{\alpha_x}{h_x} + \frac{\alpha_y}{h_y}} \right) \left[f_{i,j} - \hat{H}((\phi_x^*)_{i,j}^-, (\phi_x^*)_{i,j}^+; (\phi_y^*)_{i,j}^-, (\phi_y^*)_{i,j}^+) \right]; \tag{4}$$

$$\phi_{i,j}^{n+1} = \phi_{i,j}^{(1)} + \frac{1}{2}\gamma \left(\frac{1}{\frac{\alpha_x}{h_x} + \frac{\alpha_y}{h_y}} \right) \left[f_{i,j} - \hat{H}((\phi_x^{**})_{i,j}^-, (\phi_x^{**})_{i,j}^+; (\phi_y^{**})_{i,j}^-, (\phi_y^{**})_{i,j}^+) \right]. \tag{5}$$

Here, $\phi_{i,j}^n$ and $\phi_{i,j}^{n+1}$ are the numerical solution values at iteration steps n and $n+1$, respectively. $\phi_{i,j}^{(1)}$ is the numerical solution value at iteration step of the first RK stage. $f_{i,j}$ denotes the value of f at a grid point (x_i, y_j). $\hat{H}$ is a monotone numerical Hamiltonian [28]. $(\phi_x^*)_{i,j}^-$ and $(\phi_x^{**})_{i,j}^-$ are approximations of ϕ_x at the grid point (x_i, y_j) when the wind "blows" from the left to the right, while $(\phi_x^*)_{i,j}^+$ and $(\phi_x^{**})_{i,j}^+$ are approximations of ϕ_x at the grid point (x_i, y_j) when the wind "blows" from the right to the left. It is similar for y-direction approximations $(\phi_y^*)_{i,j}^-$, $(\phi_y^{**})_{i,j}^-$, $(\phi_y^*)_{i,j}^+$, and $(\phi_y^{**})_{i,j}^+$. The superscripts $*$ and $**$ indicate that unlike the usual FE or RK schemes, here, we always use the newest available values of ϕ in the schemes' stencils to compute these approximations of derivatives, according to the philosophy of Gauss-Seidel iterations. Namely, a numerical value (e.g., $\phi_{k,l}$) used on the computational stencil could be the value of the previous iteration step, or the new value which has been updated and available in the current iteration step, depending on the current sweeping direction of the iteration. γ is a parameter. To guarantee that the fixed-point iteration is a contractive mapping and converges, suitable values of γ need to be taken. In the context of time-marching schemes, γ is actually the Courant-Friedrichs-Lewy (CFL) number,

$$\alpha_x = \max_{\substack{A \leqslant u \leqslant B \\ C \leqslant v \leqslant D}} |H_1(u, v)|, \qquad \alpha_y = \max_{\substack{A \leqslant u \leqslant B \\ C \leqslant v \leqslant D}} |H_2(u, v)|. \tag{6}$$

$H_i(u, v)$ is the partial derivative of H with respect to the ith argument, or the Lipschitz constant of H with respect to the ith argument. $[A, B]$ is the value range for $\phi_x^{\pm}$, and $[C, D]$ is the value range for $\phi_y^{\pm}$. For the Eikonal equation (1), we have $\alpha_x = \alpha_y = 1$.

For first-order scheme, simple first-order upwind finite difference approximations for ϕ_x and ϕ_y are used. To obtain a high-order scheme, in [41] $(\phi_x^*)_{i,j}^-$, $(\phi_x^{**})_{i,j}^-$, $(\phi_x^*)_{i,j}^+$, $(\phi_x^{**})_{i,j}^+$, $(\phi_y^*)_{i,j}^-$,

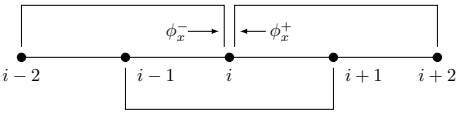


Fig. 1 Stencils of the third-order WENO approximations for derivatives

$(\phi_y^{**})_{i,j}^-$, $(\phi_y^{*})_{i,j}^+$, and $(\phi_y^{**})_{i,j}^+$ are computed by a third-order WENO scheme, which is also used in [42]. See Fig. 1 for an illustration of the computational stencils used. To simplify the notations, in the following, we omit the superscripts $*$ and $**$, with the understanding that the newest numerical values on the computational stencil of the WENO scheme are used whenever they are available. Again, in the following formulas, a numerical value (e.g., $\phi_{i-2,j}, \cdots, \phi_{i+2,j}$) used on the computational stencil could be the value of the previous iteration step, or the new value which has been updated and available in the current iteration step, depending on the current sweeping direction of the iteration. The WENO approximation of ϕ_x at the grid point (x_i, y_j) when the wind "blows" left-to-right is

$$(\phi_x)_{i,j}^- = (1 - w_-)\left(\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2h_x}\right) + w_-\left(\frac{3\phi_{i,j} - 4\phi_{i-1,j} + \phi_{i-2,j}}{2h_x}\right), \tag{7}$$

where

$$w_- = \frac{1}{1 + 2r_-^2}, \quad r_- = \frac{\epsilon + (\phi_{i,j} - 2\phi_{i-1,j} + \phi_{i-2,j})^2}{\epsilon + (\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j})^2}; \tag{8}$$

when the wind "blows" right-to-left, the WENO approximation is

$$(\phi_x)_{i,j}^+ = (1 - w_+)\left(\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2h_x}\right) + w_+\left(\frac{-\phi_{i+2,j} + 4\phi_{i+1,j} - 3\phi_{i,j}}{2h_x}\right), \tag{9}$$

where

$$w_+ = \frac{1}{1 + 2r_+^2}, \quad r_+ = \frac{\epsilon + (\phi_{i+2,j} - 2\phi_{i+1,j} + \phi_{i,j})^2}{\epsilon + (\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j})^2}. \tag{10}$$

Here, ϵ is a small value to avoid that the denominator becomes zero. The WENO approximations of ϕ_y are computed similarly. If we take $w_- = w_+ = 1/3$ in (7) and (9), then third-order linear upwind approximations are obtained. In this paper, we use the Lax-Friedrichs numerical Hamiltonian [28], which has the following form for a Hamiltonian $H(u, v)$:

$$\hat{H}^{LF}(u^-, u^+; v^-, v^+) = H\left(\frac{u^- + u^+}{2}, \frac{v^- + v^+}{2}\right) - \frac{1}{2}\alpha_x(u^+ - u^-) - \frac{1}{2}\alpha_y(v^+ - v^-), \tag{11}$$

where α_x and α_y have the same definition as (6).

We summarize the fixed-point fast sweeping WENO (FPFS-WENO) algorithm as follows.

(i) Initialization: according to the boundary condition $\phi(x, y) = g(x, y), (x, y) \in \Gamma$, assign exact values or interpolated values at grid points whose distances to Γ are less than or equal to $(m - 1)$ grid sizes, where m is the number of grid points in small stencils of WENO approximations. For example, $m = 3$ for the third-order WENO approximations used here. These values are fixed during iterations. For robust simulations, the solution from the non-fully converged (i.e., using a much larger convergence threshold value δ than that of the WENO sweeping used in Step (iii) below; specific values given in the numerical example section) first-order sweeping computation (i.e., using the first-order upwind approximations for these derivatives in $\hat{H}$ of the scheme (3) for the FE FPFS-WENO method or the scheme (4)–(5) for the RK FPFS-WENO

method) is used as the initial guess at all other grid points, while a big value (e.g., 10 in this paper) is used as the initial guess for the first-order sweeping computation.

(ii) Iterations: perform the Gauss-Seidel iterations (3) for the FE FPFS-WENO method or (4)–(5) for the RK FPFS-WENO method, with four alternating direction sweepings

(a) $i = 1 : I, j = 1 : J$;
(b) $i = I : 1, j = 1 : J$;
(c) $i = I : 1, j = J : 1$;
(d) $i = 1 : I, j = J : 1$.

For the RK FPFS-WENO method, each sweeping direction is completed in full for the first RK stage before moving to the second RK stage, and the sweeping direction of both stages should be same during one sweeping. High-order extrapolations are used for the ghost points when calculating the high-order WENO approximations of the derivatives for grid points on the boundary of the computational domain, as in [42].

(iii) Convergence: if

$$\|\phi^{n+1} - \phi^n\|_{L^\infty} \leqslant \delta,$$

where δ is a given convergence threshold value and $\|\cdot\|_{L^\infty}$ denotes the L^∞ norm, the iterations converge, and we stop the iterations.

2.2 The FPFS-WENO Schemes on Sparse Grids

In this section, we describe how to implement the FPFS-WENO methods on sparse grids by using the sparse-grid combination technique, for improving the method's efficiency in solving multidimensional Eikonal equations. Here, 2D cases are used to illustrate the idea. Algorithm procedures for higher dimensional cases are similar. We consider a square computational domain $[a, b]^2$ for simplicity of the description, and construct semi-coarsened sparse grids as the following. Note that the procedure here can be applied to any rectangular domain straightforwardly. The domain is first partitioned into the coarsest grid $\Omega^{0,0}$ with N_r cells in each direction and mesh size $H = \frac{b-a}{N_r}$. $\Omega^{0,0}$ is called a root grid. Then, a multi-level refinement on the root grid is done to construct a family of semi-coarsened grids $\{\Omega^{l_1,l_2}\}$ with mesh sizes $h_{l_1} = 2^{-l_1}H$ in the x-direction and $h_{l_2} = 2^{-l_2}H$ in the y-direction, where $l_1 = 0, 1, \cdots, N_L$ and $l_2 = 0, 1, \cdots, N_L$. The superscripts l_1, l_2 are the refinement levels relative to the root grid $\Omega^{0,0}$ in the x- and the y-directions, respectively, and N_L is the finest level. Here, the finest grid is Ω^{N_L,N_L} with the mesh size $h = 2^{-N_L}H$ in both x- and y-directions. Actually, Ω^{N_L,N_L} is corresponding to a single full grid in regular single-grid computations. Figure 2 is an illustration of 2D sparse grids $\{\Omega^{l_1,l_2}\}$ for one cell of a root grid, with $N_L = 3$. We apply the spare-grid combination techniques. The Eikonal equation (1) is not directly solved by the FPFS-WENO methods on a single full grid Ω^{N_L,N_L}, but on the set $\{\Omega^{l_1,l_2}\}_I$ of the following $(2N_L + 1)$ sparse grids:

$$\left\{\Omega^{0,N_L}, \Omega^{1,N_L-1}, \cdots, \Omega^{N_L-1,1}, \Omega^{N_L,0}\right\} \quad \text{and} \quad \left\{\Omega^{0,N_L-1}, \Omega^{1,N_L-2}, \cdots, \Omega^{N_L-2,1}, \Omega^{N_L-1,0}\right\}$$

with I being the index set

$$I = \{(l_1, l_2) | l_1 + l_2 = N_L \quad \text{or} \quad l_1 + l_2 = N_L - 1\}.$$

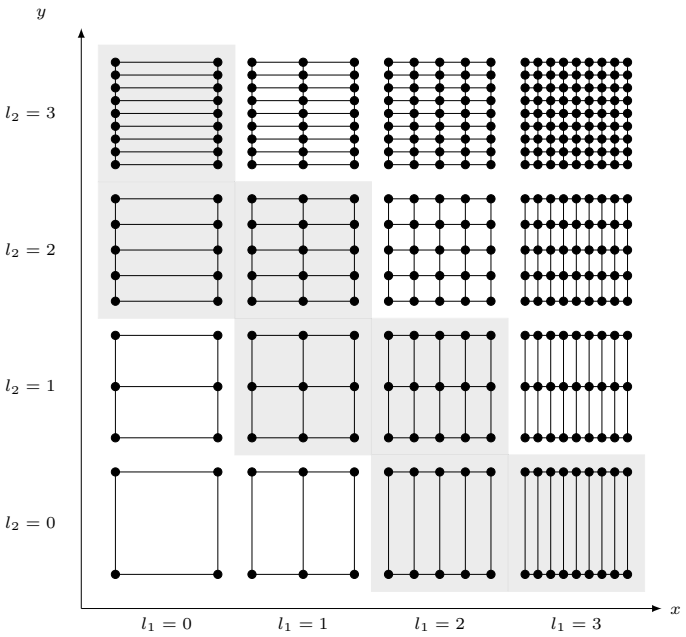


Fig. 2 Illustration of 2D sparse grids $\{\Omega^{l_1,l_2}\}$ for one cell of a root grid. Here, the cell indicated by the levels $l_1 = 0, l_2 = 0$ is one cell of the root grid $\Omega^{0,0}$, and the side length of the cell is H. The finest level $N_L = 3$. Highlighted grids are those on which PDEs are solved

Then, we have $(2N_L + 1)$ sets of numerical solutions $\{\boldsymbol{\Phi}^{l_1,l_2}\}_I$, where each set of numerical solutions is corresponding to each sparse grid in $\{\Omega^{l_1,l_2}\}_I$. The final step is to combine these sparse-grid solutions $\{\boldsymbol{\Phi}^{l_1,l_2}\}_I$ to obtain the final solution on the finest grid Ω^{N_L,N_L}. This is implemented by first using a prolongation operator P^{N_L,N_L} to map each sparse-grid solution $\hat{\boldsymbol{\Phi}}^{l_1,l_2}$ onto the finest grid Ω^{N_L,N_L}, and then combining these solutions to form the final solution $\hat{\boldsymbol{\Phi}}^{N_L,N_L}$ on Ω^{N_L,N_L}. Next, we describe the prolongation technique in details and then summarize the whole algorithm.

2.2.1 Prolongation Operator and WENO Interpolation

Given the numerical solution $\hat{\boldsymbol{\Phi}}^{l_1,l_2}$ on Ω^{l_1,l_2}, a prolongation operator P^{N_L,N_L} generates numerical values $P^{N_L,N_L}\hat{\boldsymbol{\Phi}}^{l_1,l_2}$ for all grid points on Ω^{N_L,N_L}. Prolongation is usually implemented by interpolation procedure. Studies in [13, 17, 18] for linear schemes and in [21, 22, 45] for nonlinear schemes show that the final solution resulting from the sparse-grid combination techniques can achieve similar accuracy orders as the numerical schemes, as long as the accuracy order of interpolations in the prolongations is not less than the accuracy order of the numerical schemes used to solve PDEs on sparse grids. Hence, we use third-order interpolations here for prolongations. If solutions are smooth, simple Lagrange interpolation can be used directly. The interpolations are carried out in the dimension by dimension way. In a 2D domain, first $(N_r 2^{l_1-1})$ quadratic polynomials $P_i^2(x)$, $i = 1, \cdots, N_r 2^{l_1-1}$, are constructed along the x-direction grid lines using the third-order Lagrange interpolation. Three adjacent grid points are used in each interpolation.

Each polynomial $P_i^2(x)$ is then evaluated on the grid points of Ω^{N_L,l_2} (the most refined grid in the x-direction). Then, the same interpolation procedure is performed in every grid line of the y-direction with a fixed x-coordinate on the grid Ω^{N_L,l_2}, and the obtained polynomials are evaluated on the grid points of Ω^{N_L,N_L} to get $P^{N_L,N_L}\hat{\Phi}^{l_1,l_2}$.

Because solutions of H-J equations may develop discontinuous derivatives and not be smooth, it is more robust to use WENO interpolations in the prolongation for a general case. Here, a third-order WENO interpolation is used, and detailed formulas are given as following. We describe the interpolation for a x-direction grid line, and it is similar for the y-direction. Given numerical values $\phi_{i-1,j}$, $\phi_{i,j}$, and $\phi_{i+1,j}$ at the grid points x_{i-1}, x_i, and x_{i+1} along the line $y = y_j$, we compute the third-order WENO interpolation $\phi_{\text{WENO}}(x)$ for any point $x \in [x_{i-1/2}, x_{i+1/2})$, where $x_{i-1/2} = (x_{i-1} + x_i)/2$ and $x_{i+1/2} = (x_i + x_{i+1})/2$. Let h be the grid size of the uniform mesh, we write the point x as $x = x_{i-1} + \tilde{\alpha}h$ with $\tilde{\alpha} \in [1/2, 3/2)$. The WENO interpolation is

$$\phi_{\text{WENO}}(x) = w_1 P^1_{(1)}(x) + w_2 P^1_{(2)}(x), \tag{12}$$

where $P^1_{(1)}(x)$ and $P^1_{(2)}(x)$ are second-order approximations computed as

$$P^1_{(1)}(x) = \tilde{\alpha}\phi_{i,j} - (\tilde{\alpha} - 1)\phi_{i-1,j}, \qquad P^1_{(2)}(x) = (\tilde{\alpha} - 1)\phi_{i+1,j} - (\tilde{\alpha} - 2)\phi_{i,j}. \tag{13}$$

The nonlinear weights w_1 and w_2 are computed as

$$w_1 = \frac{\tilde{w}_1}{\tilde{w}_1 + \tilde{w}_2}, \qquad w_2 = 1 - w_1 \tag{14}$$

with

$$\tilde{w}_1 = \frac{\gamma_1}{(\epsilon + \beta_1)^2}, \qquad \tilde{w}_2 = \frac{\gamma_2}{(\epsilon + \beta_2)^2}, \tag{15}$$

where $\gamma_1 = 1 - \tilde{\alpha}/2$, $\gamma_2 = \tilde{\alpha}/2$, $\beta_1 = (\phi_{i,j} - \phi_{i-1,j})^2$, and $\beta_2 = (\phi_{i+1,j} - \phi_{i,j})^2$. ϵ is a small positive number used to avoid the denominator becoming 0, and its value is specified in the next numerical experiment section.

2.2.2 Algorithm Summary

We summarize the algorithm of the FPFS-WENO schemes on sparse grids as following.

Algorithm: sparse-grid FPFS-WENO schemes

(i) Restriction step: perform the initialization step of the FPFS-WENO algorithm in Sect. 2.1 on the aforementioned $(2N_L + 1)$ sparse grids $\{\Omega^{l_1,l_2}\}_I$.

(ii) Sweeping step: on each sparse grid Ω^{l_1,l_2} in $\{\Omega^{l_1,l_2}\}_I$, perform the Gauss-Seidel iterations (3) for the FE FPFS-WENO method or (4)–(5) for the RK FPFS-WENO method, with four alternating direction sweepings as in the FPFS-WENO algorithm of Sect. 2.1, to solve the Eikonal equation (1). Then, we produce $(2N_L + 1)$ sets of converged solutions $\{\Phi^{l_1,l_2}\}_I$ for the Eikonal equation (1).

(iii) Prolongation step: on each sparse grid Ω^{l_1,l_2} in $\{\Omega^{l_1,l_2}\}_I$, use the prolongation operator P^{N_L,N_L} on $\boldsymbol{\Phi}^{l_1,l_2}$ to map it onto the most refined grid Ω^{N_L,N_L}, and obtain the solution $P^{N_L,N_L}\boldsymbol{\Phi}^{l_1,l_2}$.

(iv) Combination step: compute the final solution $\boldsymbol{\Phi}^{N_L,N_L}$ by taking the combination

$$\hat{\boldsymbol{\Phi}}^{N_L,N_L} = \sum_{l_1+l_2=N_L} P^{N_L,N_L}\boldsymbol{\Phi}^{l_1,l_2} - \sum_{l_1+l_2=N_L-1} P^{N_L,N_L}\boldsymbol{\Phi}^{l_1,l_2}. \tag{16}$$

In three-dimensional (3D) or higher dimensional cases, the algorithm follows similar procedure, while prolongation operations are carried out in additional spatial directions. The sparse-grid combination formula for higher dimensional problems is provided in the literature, e.g., [13]. In this technical notes, the following 3D formula is also used:

$$\begin{aligned}\hat{\boldsymbol{\Phi}}^{N_L,N_L,N_L} = &\sum_{l_1+l_2+l_3=N_L} P^{N_L,N_L,N_L}\boldsymbol{\Phi}^{l_1,l_2,l_3} - 2\sum_{l_1+l_2+l_3=N_L-1} P^{N_L,N_L,N_L}\boldsymbol{\Phi}^{l_1,l_2,l_3} \\ &+ \sum_{l_1+l_2+l_3=N_L-2} P^{N_L,N_L,N_L}\boldsymbol{\Phi}^{l_1,l_2,l_3}.\end{aligned} \tag{17}$$

Remark 1 Note that the adaptivity technique is a very effective approach to further improve the sparse-grid methods and achieve more efficient computations for solving multidimensional problems. Adaptive sparse-grid methods have been developed in the literature, e.g., [3, 12, 15, 39]. It will be a very interesting topic to combine the sparse-grid FPFS-WENO methods with the adaptivity technique. A key step here will be to develop efficient adaptive sparse-grid WENO methods. There are several ideas in the literature that may be used for this effort. For example, multiresolution analysis has been used to develop sparse-grid discontinuous Galerkin methods in [14]. This kind of multiresolution analysis methods may be used to develop sparse-grid WENO methods directly based on a hierarchical basis, without using the sparse-grid combination technique. Then, this kind of hierarchical basis WENO methods on sparse grids will be promising to be combined with adaptive approaches, since the hierarchical basis or the hierarchical structure in the schemes provides a convenient way to design error indicators in adaptive methods, as in [1, 14]. There are also approaches to develop adaptive sparse-grid combination technique which may be used. For example, a dimension-adaptive method was designed in [11] to obtain a generalized combination technique by employing adaptive index sets and iteratively adding new grids to the combination scheme. Spatially adaptive sparse-grid combination techniques were developed in [25, 26]. Detailed application of the adaptivity technique to the sparse-grid FPFS-WENO methods will be the next work.

3 Numerical Examples

In this section, we perform numerical experiments on solving multidimensional Eikonal equations to test the sparse-grid FPFS-WENO methods and show a large amount of CPU time savings by comparisons with corresponding single-grid simulations. Although theoretical error analysis on linear schemes for linear PDEs [13, 18] has been carried out to show that the sparse-grid combination leads to a canceling in leading-order errors of numerical solutions on semi-coarsened sparse grids, hence the accuracy order of the final solution of a sparse-grid computation is kept to be almost the same as that on the

corresponding single-grid simulation, such sparse-grid error analysis is very difficult to carry out for the WENO methods due to their high nonlinearity. Following our previous studies [21, 22, 45], numerical experiments are used to verify the third-order accuracy for the sparse-grid FPFS-WENO schemes in this note, rather than theoretical analysis. Specifically, mesh refinement studies are carried out to compute numerical convergence rates on successively refined grids, for problems with smooth solutions. In [45], two different approaches, "refine root grid" and "refine levels", are studied for mesh refinement in sparse-grid computations. For example, for 3D sparse grids with a $10 \times 10 \times 10$ root grid and $N_L = 3$, the finest grid is $80 \times 80 \times 80$. The "refine root grid" approach is to refine the root grid, while the total number of semi-coarsened sparse-grid levels $N_L + 1$ is kept unchanged. Therefore, if the root grid is refined once to be $20 \times 20 \times 20$, we obtain the finest grid $160 \times 160 \times 160$. The "refine levels" approach refines the sparse-grid levels, while keeping the root grid fixed. Therefore, if $N_L = 3$ is refined once to be $N_L = 4$ with the fixed $10 \times 10 \times 10$ root grid, the finest grid $160 \times 160 \times 160$ is also obtained. It is discovered in [45] that although with more levels, the "refine levels" approach saves more CPU time costs than the "refine root grid" approach, it has obvious accuracy order reductions for the nonlinear sparse-grid WENO schemes. The "refine root grid" approach can always achieve the desired accuracy order of the sparse-grid WENO scheme. In this technical note, we test both approaches in mesh refinement studies and draw the similar conclusion as that in [45]. $N_L = 3$ is used for the "refine root grid" approach in accuracy order test examples and all other sparse-grid computations for problems with non-smooth solutions.

We first test the sparse-grid FPFS-WENO methods on problems with smooth solutions to study its numerical accuracy orders. Then, the method is applied to problems with non-smooth solutions to show its nonlinear stability. For all numerical examples, we take $\epsilon = 10^{-6}$ in the WENO scheme for both the iterations and the WENO interpolation in the prolongation operator. The convergence threshold value is taken as $\delta = 10^{-11}$ for the third-order WENO sweeping, and we take $\delta = 10^{-4}$ in the non-fully-converged first-order sweeping to provide initial values for the WENO sweeping. As in [41], we select the largest γ value for each problem that provides the iteration convergence with the fastest speed on all semi-coarsened sparse grids in the sparse-grid combination, for the purpose of testing the computational efficiency of the algorithm. To identify the largest possible γ value for a problem, we gradually increase/decrease the value of γ from an initial value. In this section, we use N_h to denote the number of computational cells in one spatial direction of the most refined grid in sparse grids or the corresponding single grid.

Example 1 (A linear problem with a smooth solution) Consider the following 2D linear problem:

$$\phi_x + \phi_y = 0, \qquad (x, y) \in \Omega \backslash \Gamma, \tag{18}$$

where $\Omega = [0, 2\pi]^2$ and $\Gamma = \{(x, y) \in \Omega \,|\, x = 0 \text{ or } y = 0\}$. The inflow boundary conditions are applied on Γ,

$$\phi(x, 0) = \sin(x), \qquad \phi(0, y) = -\sin(y). \tag{19}$$

This problem has the exact solution

$$\phi(x, y) = \sin(x - y). \tag{20}$$

For this linear problem with a smooth solution, we first solve it by both the sparse-grid FE FPFS scheme and the sparse-grid RK FPFS scheme using the "refine root grid"

approach in the mesh refinement, with $\gamma = 1$ and the third-order linear upwind approximations to the derivatives, to verify the error analysis results for linear schemes applied to linear PDEs in the literature, e.g., [13, 17, 18]. The third-order Lagrange interpolation for prolongation is employed in sparse-grid computations. We perform simulations on both sparse grids and the corresponding single grids, and compare their results. The L^1 errors, L^∞ errors, and their numerical accuracy orders and CPU times are reported in Table 1 and Table 2. For both of schemes, the third-order accuracy is obtained for both sparse-grid computations using the "refine root grid" approach and the corresponding single-grid ones, along with the mesh refinement. This is consistent with the error analysis results for linear schemes in solving linear PDEs in [13, 17, 18]. Comparing the numerical errors of sparse-grid computations with the "refine root grid" approach and the corresponding single-grid ones, we observe that their L^1 errors are comparable. The L^∞ errors of sparse-grid computations are larger than the corresponding single-grid computations. In terms of computational efficiency, on refined meshes, we see around $60\% \sim 70\%$ and $50\% \sim 80\%$ CPU time saved for simulations on sparse grids vs single grids, for the FE FPFS scheme and the RK FPFS scheme, respectively, in this example. Also from Tables 1 and 2, we see that the numerical errors of the FE FPFS scheme and the RK FPFS scheme are similar. This is consistent with the expectation, since although the FE FPFS scheme and the RK FPFS scheme are different iterative schemes, here they converge to the same numerical steady-state solution (up to round-off errors) which is the solution of the algebraic system resulting from the spatial discretization of the equation. However, in terms of computational efficiency, in general, the FE FPFS scheme takes fewer CPU time costs and is more efficient than the RK FPFS scheme, since for most cases, the FE FPFS scheme needs fewer iteration steps to converge than the RK FPFS scheme.

Then, we use the "refine levels" approach to refine the meshes of the sparse-grid computations and observe the reduction of numerical accuracy orders. Since the numerical errors of the FE FPFS scheme and the RK FPFS scheme are similar, we report the results

Table 1 Example 1, a linear problem with a smooth solution. The FE FPFS scheme with the third-order linear upwind approximations, comparison of numerical errors and CPU times for computations on single grids and sparse grids. Third-order Lagrange interpolation for prolongation is employed in sparse-grid computations. N_r: number of cells in each spatial direction of a root grid. CPU: CPU time for a complete simulation. CPU time unit: seconds

Single grid						
	N_h	L^1 error	Order	L^∞ error	Order	CPU/s
	160	1.27×10^{-5}	–	4.91×10^{-5}	–	1.92
	320	1.59×10^{-6}	3.00	6.14×10^{-6}	3.00	10.60
	640	1.98×10^{-7}	3.00	7.68×10^{-7}	3.00	72.63
	1 280	2.47×10^{-8}	3.00	9.60×10^{-8}	3.00	535.69
Sparse grid						
N_r	N_h	L^1 error	Order	L^∞ error	Order	CPU/s
20	160	4.56×10^{-5}	–	5.21×10^{-4}	–	1.47
40	320	2.11×10^{-6}	4.43	1.63×10^{-4}	1.67	4.60
80	640	2.73×10^{-7}	2.95	1.35×10^{-5}	3.60	25.18
160	1 280	3.02×10^{-8}	3.18	1.56×10^{-6}	3.12	158.18

Table 2 Example 1, a linear problem with a smooth solution. The RK FPFS scheme with the third-order linear upwind approximations, comparison of numerical errors, and CPU times for computations on single grids and sparse grids. Third-order Lagrange interpolation for prolongation is employed in sparse-grid computations. Both the "refine root grid" approach (N_r refinement) and the "refine levels" approach (N_L refinement) are used. N_r: number of cells in each spatial direction of a root grid. N_L: the finest level in a sparse-grid computation. CPU: CPU time for a complete simulation. CPU time unit: seconds

Single grid						
	N_h	L^1 error	Order	L^∞ error	Order	CPU/s
	160	1.27×10^{-5}	–	4.91×10^{-5}	–	1.83
	320	1.59×10^{-6}	3.00	6.14×10^{-6}	3.00	11.74
	640	1.98×10^{-7}	3.00	7.68×10^{-7}	3.00	80.38
	1 280	2.47×10^{-8}	3.00	9.60×10^{-8}	3.00	748.34
Sparse grid — N_r refinement						
N_r	N_h	L^1 error	Order	L^∞ error	Order	CPU/s
20	160	4.56×10^{-5}	–	5.21×10^{-4}	–	1.51
40	320	2.11×10^{-6}	4.43	1.63×10^{-4}	1.67	6.08
80	640	2.73×10^{-7}	2.95	1.35×10^{-5}	3.60	22.63
160	1 280	3.02×10^{-8}	3.18	1.56×10^{-6}	3.12	159.19
Sparse grid — N_L refinement						
N_L	N_h	L^1 error	Order	L^∞ error	Order	CPU/s
2	160	9.30×10^{-6}	–	8.56×10^{-5}	–	1.53
3	320	2.11×10^{-6}	2.13	1.63×10^{-4}	–	6.08
4	640	7.96×10^{-7}	1.41	1.03×10^{-4}	–	15.41
5	1 280	3.67×10^{-7}	1.12	9.64×10^{-5}	–	53.50

of the sparse-grid RK FPFS scheme in Table 2 (N_L refinement case). Obvious accuracy order reduction is observed for the "refine levels" approach, which is consistent with our previous studies. Here, we see that the L^1 accuracy order is reduced to the first order and the L^∞ errors do not converge. Also, with more levels to obtain a specific finest grid, the "refine levels" approach saves more CPU time costs than the "refine root grid" approach. However, due to the accuracy order reduction, the "refine levels" approach has much larger numerical errors than the "refine root grid" approach when the meshes are refined. Hence, in general, to reach a level of small numerical error, the "refine root grid" approach has smaller computational costs than the "refine levels" approach.

Example 2 (A nonlinear problem with a smooth solution) We solve the 2D Eikonal equation (1) with the right-hand side function

$$f(x, y) = \frac{\pi}{2}\sqrt{\sin^2\left(\pi + \frac{\pi}{2}x\right) + \sin^2\left(\pi + \frac{\pi}{2}y\right)},$$

and the source point $\Gamma = (0, 0)$. The computational domain $\Omega = [-1, 1]^2$. The exact solution of the problem is

$$\phi(x, y) = \cos\left(\pi + \frac{\pi}{2}x\right) + \cos\left(\pi + \frac{\pi}{2}y\right).$$

At first, we use this example to verify that the proposed sparse-grid FPFS-WENO schemes with the "refine root grid" approach can achieve the desired accuracy order for a nonlinear problem with a smooth solution. The sparse-grid FE FPFS-WENO scheme and the sparse-grid RK FPFS-WENO scheme, with $\gamma = 1$, and the third-order WENO approximations to the derivatives are applied. Both the third-order Lagrange interpolation and the third-order WENO interpolation are used for prolongation in sparse-grid computations. We perform simulations on both sparse grids and the corresponding single grids, and compare their results. The L^1 errors, L^∞ errors, and their numerical accuracy orders and CPU times are reported in Table 3 and Table 4. It is observed that third-order accuracy is obtained for all cases, including sparse-grid computations with Lagrange or WENO prolongation and the corresponding single-grid ones, along with the mesh refinement. Comparing the numerical errors of sparse-grid computations and the corresponding single-grid ones, similar to Example 1, we observe that their L^1 errors are comparable, while sparse-grid computations with WENO prolongation have slightly larger errors. The L^∞ errors of sparse-grid computations are larger than the corresponding single-grid computations. In terms of computational efficiency, on refined meshes, we see around 50% $\sim$ 70% and 65% $\sim$ 70% CPU time saved for simulations on sparse grids vs single grids, for the FE FPFS-WENO scheme

Table 3 Example 2, a nonlinear problem with a smooth solution. The FE FPFS-WENO scheme, comparison of numerical errors and CPU times for computations on single grids and sparse grids. Both third-order Lagrange interpolation and WENO interpolation for prolongation are employed in sparse-grid computations. N_r: number of cells in each spatial direction of a root grid. CPU: CPU time for a complete simulation. CPU time unit: seconds

Single grid						
	N_h	L^1 error	Order	L^∞ error	Order	CPU/s
	160	1.05×10^{-6}	–	1.78×10^{-6}	–	4.25
	320	1.11×10^{-7}	3.24	1.71×10^{-7}	3.38	22.77
	640	1.37×10^{-8}	3.02	2.10×10^{-8}	3.30	149.41
	1 280	1.71×10^{-9}	3.00	2.61×10^{-9}	3.00	1 083.07
Sparse grid, Lagrange interpolation						
N_r	N_h	L^1 error	Order	L^∞ error	Order	CPU/s
20	160	3.28×10^{-6}	–	1.74×10^{-5}	–	12.80
40	320	2.70×10^{-7}	3.60	2.96×10^{-6}	2.55	28.30
80	640	2.34×10^{-8}	3.53	4.20×10^{-7}	2.82	72.88
160	1 280	2.27×10^{-9}	3.36	5.55×10^{-8}	2.92	343.70
Sparse grid, WENO interpolation						
N_r	N_h	L^1 error	Order	L^∞ error	Order	CPU/s
20	160	8.60×10^{-6}	–	4.72×10^{-3}	–	9.22
40	320	7.65×10^{-7}	3.49	1.21×10^{-3}	1.96	25.85
80	640	6.00×10^{-8}	3.67	2.72×10^{-4}	2.16	76.59
160	1 280	4.17×10^{-9}	3.84	2.00×10^{-5}	3.76	393.19

Table 4 Example 2, a nonlinear problem with a smooth solution. The RK FPFS-WENO scheme, comparison of numerical errors and CPU times for computations on single grids and sparse grids. Both third-order Lagrange interpolation and WENO interpolation for prolongation are employed in sparse-grid computations. N_r: number of cells in each spatial direction of a root grid. CPU: CPU time for a complete simulation. CPU time unit: seconds

Single grid						
	N_h	L^1 error	Order	L^∞ error	Order	CPU/s
	160	1.05×10^{-6}	–	1.78×10^{-6}	–	5.55
	320	1.11×10^{-7}	3.24	1.71×10^{-7}	3.38	30.59
	640	1.37×10^{-8}	3.02	2.10×10^{-8}	3.30	297.53
	1 280	1.71×10^{-9}	3.00	2.61×10^{-9}	3.00	1,401.50
Sparse grid, Lagrange interpolation						
N_r	N_h	L^1 error	Order	L^∞ error	Order	CPU/s
20	160	3.28×10^{-6}	–	1.74×10^{-5}	–	11.50
40	320	2.70×10^{-7}	3.60	2.96×10^{-6}	2.55	28.28
80	640	2.34×10^{-8}	3.53	4.20×10^{-7}	2.82	94.45
160	1 280	2.27×10^{-9}	3.36	5.55×10^{-8}	2.92	485.06
Sparse grid, WENO interpolation						
N_r	N_h	L^1 error	Order	L^∞ error	Order	CPU/s
20	160	8.60×10^{-6}	–	4.72×10^{-3}	–	11.74
40	320	7.65×10^{-7}	3.49	1.21×10^{-3}	1.96	29.57
80	640	6.00×10^{-8}	3.67	2.72×10^{-4}	2.16	87.74
160	1 280	4.17×10^{-9}	3.84	2.00×10^{-5}	3.76	494.72

and the RK FPFS-WENO scheme, respectively, in this nonlinear example. We also notice that on relatively coarse mesh (e.g., here, $N_h = 160, 320$ for the FE FPFS-WENO scheme and $N_h = 160$ for the RK FPFS-WENO scheme), it takes more CPU time for sparse-grid computations than the corresponding single-grid ones, due to the quite different iteration history on different semi-coarsened sparse grids. Also, similar to Example 1, from Table 3 and Table 4 we see that the numerical errors of the FE FPFS-WENO scheme and the RK FPFS-WENO scheme are similar, which is consistent with our expectation, since the FE FPFS-WENO scheme and the RK FPFS-WENO scheme converge to the same numerical steady-state solution (up to round-off errors) if the same spatial discretization for the equation is used. Again, in terms of computational efficiency, we observe that in general, the FE FPFS-WENO scheme is more efficient than the RK FPFS-WENO scheme, since for most cases, the FE FPFS-WENO scheme needs fewer iteration steps to converge than the RK FPFS-WENO scheme.

Second, we use the "refine levels" approach to refine the meshes of the sparse-grid computations and verify the reduction of numerical accuracy orders for this example. Also as in the previous example, since the numerical errors of the FE FPFS-WENO scheme and the RK FPFS-WENO scheme are similar, we report the results of the sparse-grid RK FPFS-WENO scheme in Table 5. Again, very obvious accuracy order reduction is observed for the "refine levels" approach, where the L^1 accuracy order is reduced to the first order and

Table 5 Example 2, a nonlinear problem with a smooth solution. The RK FPFS-WENO scheme. Numerical errors and CPU times for computations on sparse grids by using the "refine levels" approach to refine the meshes and observe the accuracy order reduction. Both third-order Lagrange interpolation and WENO interpolation for prolongation are employed in sparse-grid computations. N_L: the finest level in a sparse-grid computation. CPU: CPU time for a complete simulation. CPU time unit: seconds

N_L refinement, Lagrange interpolation						
N_L	N_h	L^1 error	Order	L^∞ error	Order	CPU/s
2	160	1.32×10^{-6}	–	2.68×10^{-6}	–	8.63
3	320	2.70×10^{-7}	2.29	2.96×10^{-6}	–	28.28
4	640	9.72×10^{-8}	1.47	2.98×10^{-6}	–	111.66
5	1 280	4.42×10^{-8}	1.14	2.98×10^{-6}	–	370.76
N_L refinement, WENO interpolation						
N_L	N_h	L^1 error	Order	L^∞ error	Order	CPU/s
2	160	2.35×10^{-6}	–	1.42×10^{-3}	–	8.87
3	320	7.65×10^{-7}	1.62	1.21×10^{-3}	–	29.57
4	640	3.37×10^{-7}	1.18	1.11×10^{-3}	–	112.29
5	1 280	1.61×10^{-7}	1.07	1.15×10^{-3}	–	378.36

the L^∞ errors do not converge. On the computational costs, with more levels to reach a refined mesh (e.g., $N_h = 1\,280$ here), the "refine levels" approach needs fewer CPU time costs than the "refine root grid" approach. But similar to Example 1, due to the accuracy order reduction, the "refine levels" approach has much larger numerical errors than the "refine root grid" approach when the meshes are refined. Therefore, this nonlinear example also shows that in general to reach a level of small numerical error, the "refine root grid" approach has smaller computational costs than the "refine levels" approach.

In the following, we apply the sparse-grid schemes to examples with non-smooth solutions to show their nonlinear stability and computational efficiency.

Example 3 (Two-sphere problem) We solve the 3D Eikonal equation (1) with $f(x, y, z) = 1$ on the computational domain $\Omega = [-3, 3]^3$. Γ are two spheres of equal radius 0.5 centered at $(-1, 0, 0)$ and $(\sqrt{1.5}, 0, 0)$. The exact solution of the problem is the distance function to Γ: $\phi(x, y, z) = \min(d_1, d_2)$, where

$$d_1 = \left| \sqrt{(x+1)^2 + y^2 + z^2} - 0.5 \right|,$$
$$d_2 = \left| \sqrt{(x-\sqrt{1.5})^2 + y^2 + z^2} - 0.5 \right|.$$

The solution of the problem is non-smooth. Singularities exist in the centers of each sphere and the plane that is equidistant from both spheres. The sparse-grid FE FPFS-WENO scheme and the sparse-grid RK FPFS-WENO scheme, with $\gamma = 0.8$ and the third-order WENO approximations to the derivatives are applied. The third-order WENO interpolation is used for prolongation in sparse-grid computations. We perform simulations on both sparse grids with $N_r = 80, N_L = 3$, and the corresponding $640 \times 640 \times 640$ single grid, and compare their results. Since the numerical solutions we obtain are similar to the

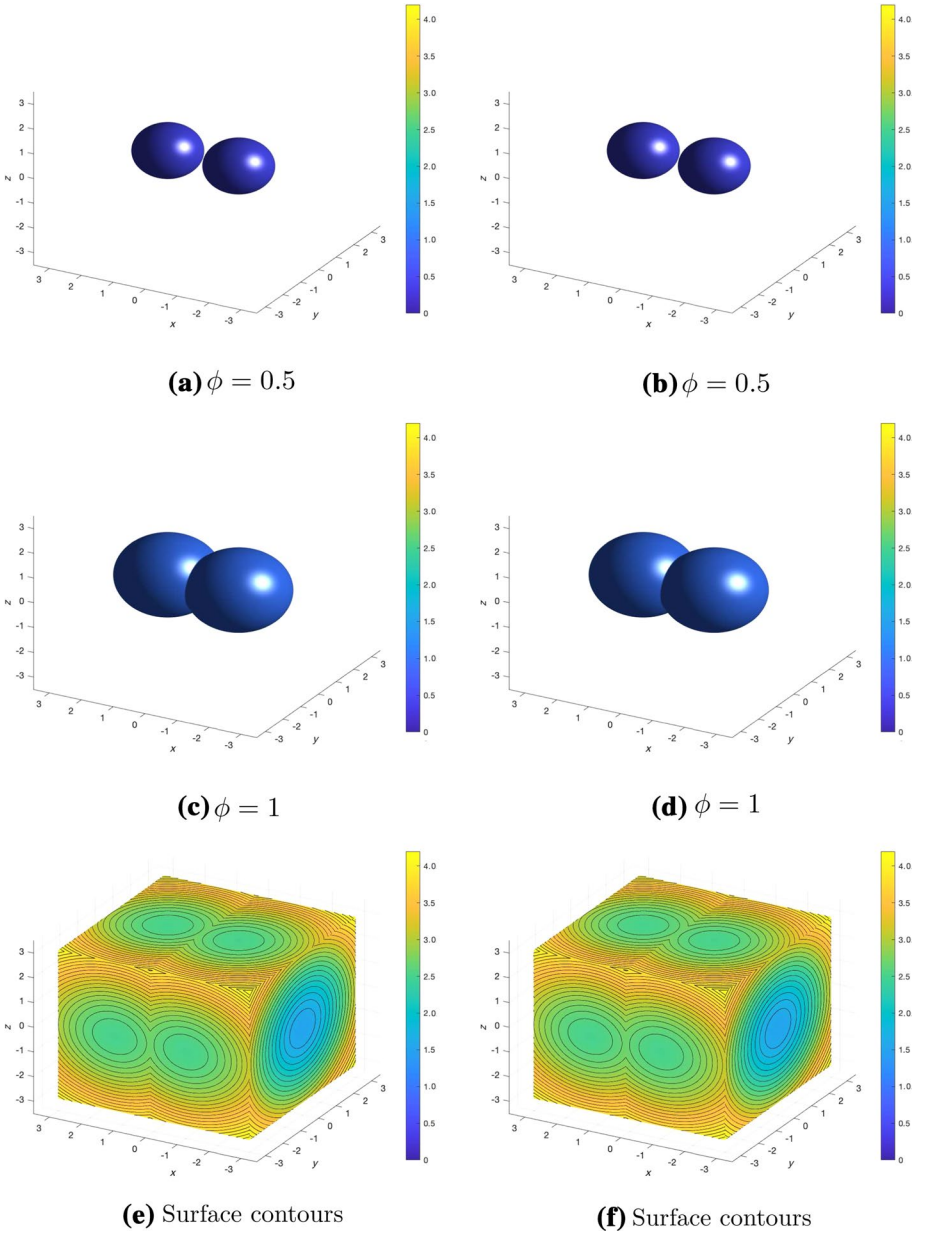


Fig. 3 Example 3, numerical solutions of the two-sphere problem by the RK FPFS-WENO scheme on sparse grids ($N_r = 80$ for root grid, finest level $N_L = 3$ in the sparse-grid computation) and the corresponding $640 \times 640 \times 640$ single grid, using the third-order WENO interpolation for prolongation in the sparse-grid combination. **a**, **c**, **e** single-grid results; **b**, **d**, **f** sparse-grid results; **a**, **b** the contour plots for $\phi = 0.5$; **c**, **d** the contour plots for $\phi = 1$; **e**, **f** the contour plots for the whole surface

FE FPFS-WENO scheme and the RK FPFS-WENO scheme, only the results of the RK FPFS-WENO scheme are shown in Fig. 3, to save space. We observe that the numerical solutions by the sparse-grid FPFS-WENO schemes and their corresponding single-grid simulations are comparable. The nonlinear stability and high-resolution properties of the FPFS-WENO schemes for resolving the non-smooth solution are preserved well in the sparse-grid simulations. We record the simulation CPU time costs to compare the computational efficiency. For the FE FPFS-WENO scheme, it takes 26 819.15 s of CPU time to complete the simulation in the sparse-grid computation, while 282 038.88 s of CPU time are needed for finishing the simulation in the corresponding single-grid computation. About 90% CPU time is saved by performing the FE FPFS-WENO simulation on the sparse grids here. For the RK FPFS-WENO scheme, it takes 42 674.31 s of CPU time to complete the simulation in the sparse-grid computation, while 458 311.48 s of CPU time are needed for finishing the simulation in the corresponding single-grid computation. So about 91% CPU time is saved by performing the RK FPFS-WENO simulation on the sparse grids. Also it is shown that the FE FPFS-WENO scheme is more efficient than the RK FPFS-WENO scheme for both sparse-grid computation and single-grid computation in this example.

Example 4 (Shape-from-shading) We solve the Eikonal equation (1) with the right-hand-side function

$$f(x,y) = 2\pi\sqrt{[\cos(2\pi x)\sin(2\pi y)]^2 + [\sin(2\pi x)\cos(2\pi y)]^2}.$$

The computational domain $\Omega = [0,1]\times[0,1]$. $\phi(x,y) = 0$ is prescribed at the boundary $\partial\Omega$ of the unit square. The boundary region $\Gamma = \{(\frac{1}{4},\frac{1}{4}),(\frac{3}{4},\frac{3}{4}),(\frac{1}{4},\frac{3}{4}),(\frac{3}{4},\frac{1}{4}),(\frac{1}{2},\frac{1}{2})\}\cup\partial\Omega$, consisting of five isolated points and $\partial\Omega$. The values at these five isolated points are specified as

$$g\Big(\frac{1}{4},\frac{1}{4}\Big) = g\Big(\frac{3}{4},\frac{3}{4}\Big) = g\Big(\frac{1}{4},\frac{3}{4}\Big) = g\Big(\frac{3}{4},\frac{1}{4}\Big) = 1, \qquad g\Big(\frac{1}{2},\frac{1}{2}\Big) = 2.$$

The exact solution of the problem is

$$\phi(x,y) = \begin{cases} \max(|\sin(2\pi x)\sin(2\pi y)|, 1+\cos(2\pi x)\cos(2\pi y)), \\ \quad \text{if } |x+y-1| < \frac{1}{2} \text{ and } |x-y| < \frac{1}{2}; \\ |\sin(2\pi x)\sin(2\pi y)|, \qquad \text{otherwise}, \end{cases}$$

which is *not* smooth. Actually the solution of this problem is the shape function, which has the brightness $I(x,y) = 1/\sqrt{1+f(x,y)^2}$ under vertical lighting. Details about this problem can be found in [31]. The sparse-grid FE FPFS-WENO scheme and the sparse-grid RK FPFS-WENO scheme, with $\gamma = 1$ and the third-order WENO approximations to the derivatives are applied. The third-order WENO interpolation is used for prolongation in sparse-grid computations. Simulations are carried out on both sparse grids with $N_r = 160, N_L = 3$ and the corresponding $1\,280\times 1\,280$ single grid, to compare their results. The numerical solutions we obtain are similar to the FE FPFS-WENO scheme and the RK FPFS-WENO scheme, so only the results of the RK FPFS-WENO scheme are reported in Fig. 4, to save space. It is observed that the numerical solutions by the sparse-grid FPFS-WENO schemes and their corresponding single-grid computations are comparable. As in the previous example, the nonlinear stability and high-resolution properties of the FPFS-WENO

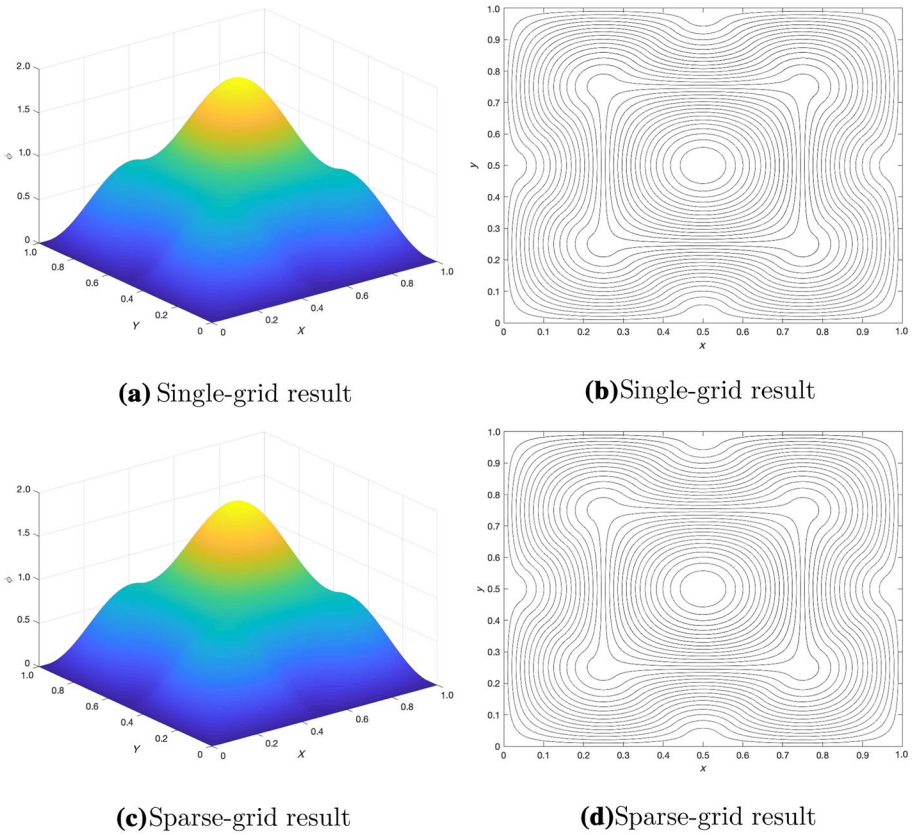


Fig. 4 Example 4, numerical solutions of the shape-from-shading problem by the RK FPFS-WENO scheme on sparse grids ($N_r = 160$ for root grid, finest level $N_L = 3$ in the sparse-grid computation) and the corresponding $1\,280 \times 1\,280$ single grid, using the third-order WENO interpolation for prolongation in the sparse-grid combination. **a**, **b** single-grid results; **c**, **d** sparse-grid results; **a**, **c** 3D view of the solutions; **b**, **d** the contour plots, 30 equally spaced contour lines from $\phi = 0$ to $\phi = 2$

schemes for resolving the non-smooth solution of this example are preserved well in the sparse-grid computations. Again, we record the simulation CPU time costs to compare their computational efficiency. It takes 468.13 s of CPU time to complete the simulation in the FE FPFS-WENO sparse-grid computation, while 1 228.65 s of CPU time are needed for finishing the simulation in the corresponding single-grid computation. About 62% CPU time is saved by carrying out the FE FPFS-WENO simulation on the sparse grids in this example. As well, it takes 670.86 s of CPU time to complete the simulation in the RK FPFS-WENO sparse-grid computation, while 1 510.27 s of CPU time are needed for finishing the simulation in the corresponding single-grid computation. About 56% CPU time is saved by carrying out the RK FPFS-WENO simulation on the sparse grids here. We also see that the FE FPFS-WENO scheme is more efficient than the RK FPFS-WENO scheme for both sparse-grid computation and single-grid computation in this example.

Example 5 (Voronoi diagram problem) We consider a Voronoi diagram problem as in [2, 27]. Given a set of points (called generators) in a domain, the Voronoi diagram divides the domain into regions in which all points inside the region are closest to the generator of that region than any other generators. This kind of problem has applications in many fields, including engineering, natural sciences, geometry, humanities, etc., for example, dividing a map into response regions for local fire stations. An essential part of solving a Voronoi diagram problem is to compute the minimum travel time to the closest generator by solving the Eikonal equation (1). Here, we solve both a 2D case and a 3D case.

Case 1 (2D). We solve the Eikonal equation (1) with $f(x, y) = 1$. The computational domain $\Omega = [0, 1]^2$. $\phi(x, y) = 0$ is prescribed at the points (the generators)

$$\Gamma = \left\{ \left(\frac{1}{4}, \frac{1}{5}\right), \left(\frac{1}{3}, \frac{1}{7}\right), \left(\frac{3}{5}, \frac{1}{5}\right), \left(\frac{3}{4}, \frac{1}{2}\right), \left(\frac{1}{2}, \frac{3}{4}\right), \left(\frac{1}{4}, \frac{1}{2}\right), \left(\frac{1}{7}, \frac{4}{5}\right), \left(\frac{1}{2}, \frac{1}{2}\right) \right\}.$$

The exact solution to the problem is the distance function to Γ, and it is *not* smooth. The sparse-grid FE FPFS-WENO scheme and the sparse-grid RK FPFS-WENO scheme, with $\gamma = 1$ and the third-order WENO approximations to the derivatives are applied. The third-order WENO interpolation is used for prolongation in sparse-grid computations. Simulations are performed on both sparse grids with $N_r = 160, N_L = 3$, and the corresponding $1\,280 \times 1\,280$ single grid, to compare their results. The results of the RK FPFS-WENO scheme are reported in Fig. 5 (the numerical solution of the FE FPFS-WENO scheme is similar, so we omit showing its pictures which are basically the same as Fig. 5). Again, we observe that the numerical solutions by the sparse-grid FPFS-WENO schemes and their corresponding single-grid computations are comparable, and the nonlinear stability and high-resolution properties of the FPFS-WENO schemes for resolving the non-smooth solution in this example are preserved well in the sparse-grid simulations. About the computational efficiency, by applying the FE FPFS-WENO scheme, it takes 680.57 s of CPU time to complete the simulation in the sparse-grid computation, while 1 364.39 s of CPU time are needed for finishing the simulation in the corresponding single-grid computation.

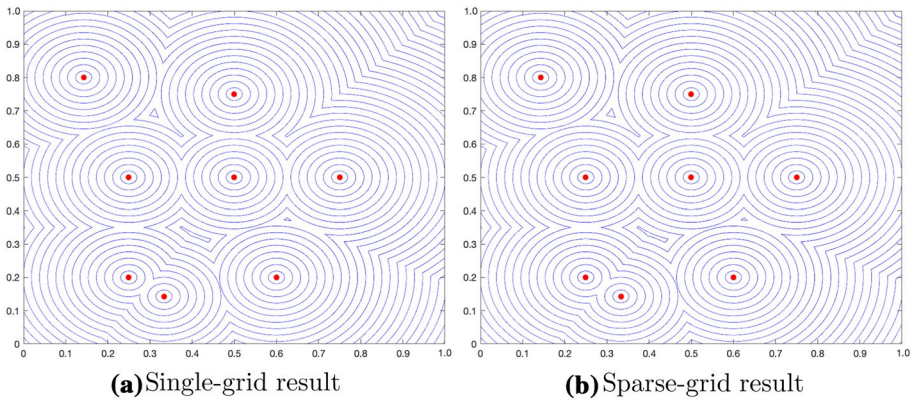


Fig. 5 Example 5, Case 1, numerical solutions of the 2D Voronoi diagram problem by the RK FPFS-WENO scheme on sparse grids ($N_r = 160$ for root grid, finest level $N_L = 3$ in the sparse-grid computation) and the corresponding $1\,280 \times 1\,280$ single grid, using the third-order WENO interpolation for prolongation in the sparse-grid combination. The contour plots, 30 equally spaced contour lines from $\phi = 0$ to $\phi = 0.558\,9$. Red points are the generators. **a** single-grid result; **b** sparse-grid result

About 50% CPU time is saved by performing the FE FPFS-WENO simulation on the sparse grids in this problem. For the RK FPFS-WENO scheme, it takes 634.42 s of CPU time to complete the simulation in the sparse-grid computation, while 1 591.41 s of CPU time are needed for finishing the simulation in the corresponding single-grid computation. About 60% CPU time is saved by performing the RK FPFS-WENO simulation on the sparse grids here. In this example, it is observed that the FE FPFS-WENO scheme is more efficient than the RK FPFS-WENO scheme for single-grid computation, but for sparse-grid computation, the RK FPFS-WENO scheme requires slightly less CPU time than the FE FPFS-WENO scheme, because the FE FPFS-WENO scheme needs more iterations than the RK FPFS-WENO scheme to converge on some semi-coarsened sparse grids.

Case 2 (3D). Now, we solve the 3D case, the Eikonal equation (1) with $f(x, y, z) = 1$. The computational domain $\Omega = [0, 1]^3$. $\phi(x, y, z) = 0$ is specified at the following generators:

$$\Gamma = \left\{ \left(\frac{1}{4}, \frac{1}{5}, \frac{1}{8}\right), \left(\frac{1}{3}, \frac{1}{7}, \frac{7}{9}\right), \left(\frac{3}{5}, \frac{1}{5}, \frac{4}{5}\right), \left(\frac{3}{4}, \frac{1}{2}, \frac{1}{4}\right), \left(\frac{1}{2}, \frac{3}{4}, \frac{4}{5}\right), \left(\frac{1}{4}, \frac{1}{2}, \frac{1}{2}\right), \left(\frac{1}{7}, \frac{4}{5}, \frac{3}{5}\right), \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{4}\right) \right\}.$$

The exact solution of the problem is the distance function to Γ in this 3D domain, and it is *not* a smooth function. The sparse-grid FE FPFS-WENO scheme and the sparse-grid RK FPFS-WENO scheme, with $\gamma = 0.8$, and the third-order WENO approximations to the derivatives are applied in solving this problem. The third order WENO interpolation is used for prolongation in sparse-grid computations. Simulations are performed on both sparse grids with $N_r = 80, N_L = 3$, and the corresponding $640 \times 640 \times 640$ single grid, to compare their results. The simulation results of the RK FPFS-WENO scheme are presented in Fig. 6 (the numerical solution of the FE FPFS-WENO scheme is similar, so we omit to show its pictures which are basically the same as Fig. 6). These simulation results show that the numerical solutions by the sparse-grid FPFS-WENO schemes and their corresponding single-grid computations are comparable, and the nonlinear stability and high-resolution properties of the FPFS-WENO schemes for resolving the non-smooth solution in this 3D example are preserved well in the sparse-grid simulations. In terms of computational efficiency, by applying the FE FPFS-WENO scheme, it takes 45 090.73 s of CPU time to complete the simulation in the sparse-grid computation, while 417 126.96 s of CPU time are needed for finishing the simulation in the corresponding single-grid computation. About 89% CPU time is saved by performing the FE FPFS-WENO simulation on the sparse grids in this 3D problem. For the RK FPFS-WENO scheme, it takes 65 425.44 s of CPU time to complete the simulation in the sparse-grid computation, while 672 078.25 s of CPU time are needed for finishing the simulation in the corresponding single-grid computation. About 90% CPU time is saved by performing the RK FPFS-WENO simulation on the sparse grids here. For this 3D example, the FE FPFS-WENO scheme is more efficient than the RK FPFS-WENO scheme for both sparse-grid computation and single-grid computation.

Example 6 (Boat-sail problem) In this example, we consider an extension of Voronoi diagram problems as in Example 5, boat-sail problems (see, e.g., [5, 23, 24]), which applies a flow field to a Voronoi diagram problem. An application of this kind of problems would be a boat trying to reach the nearest harbor or island on a moving river.

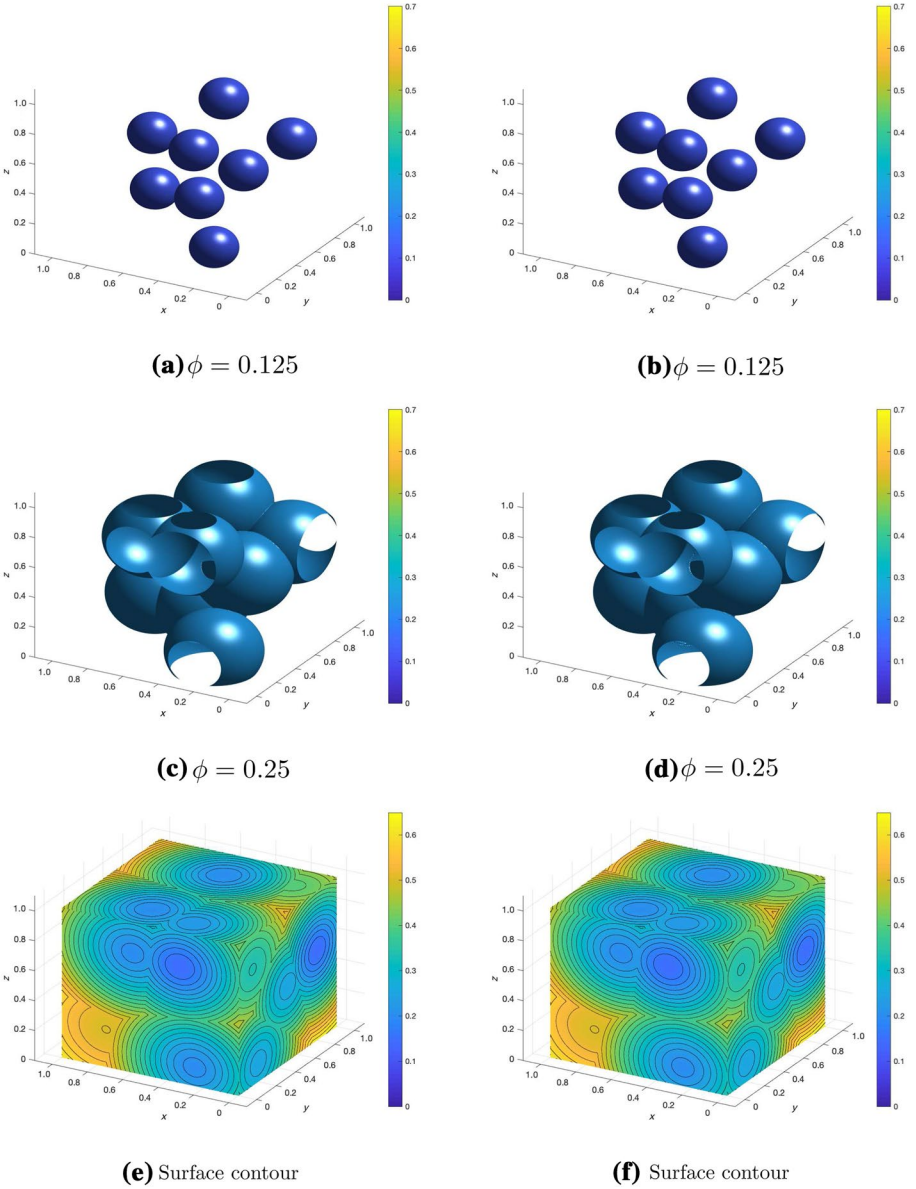


Fig. 6 Example 5, Case 2, numerical solutions of the 3D Voronoi diagram problem by the RK FPFS-WENO scheme on sparse grids ($N_r = 80$ for root grid, finest level $N_L = 3$ in the sparse-grid computation) and the corresponding $640 \times 640 \times 640$ single grid, using the third-order WENO interpolation for prolongation in the sparse-grid combination. **a**, **c**, **e** single-grid results; **b**, **d**, **f** sparse-grid results; **a**, **b** the contour plots for $\phi = 0.125$; **c**, **d** the contour plots for $\phi = 0.25$; **e**, **f** the contour plots for the whole surface

Case 1 (2D). Suppose that the river flows with some velocity $\mathbf{f} = (f_1, f_2)^{\mathrm{T}}$, and the boat travels at a maximum speed F, such that $F > |\mathbf{f}|$. The minimum travel time $\phi(x, y)$ from a point (x, y) to the nearest harbor or island can be found by solving the following static Hamilton-Jacobi equation:

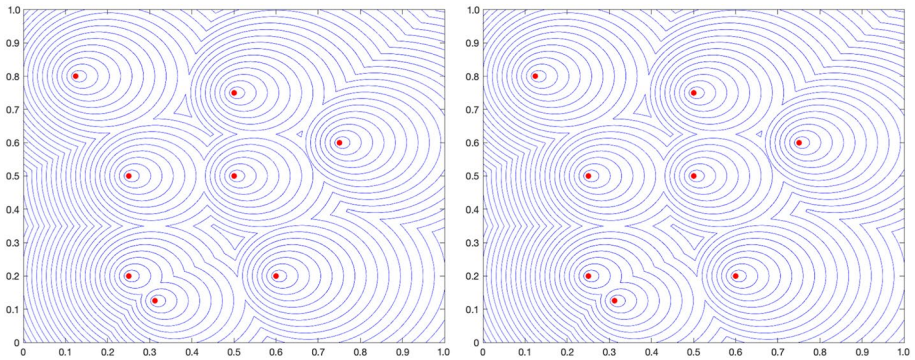
$$F|\nabla\phi| + \mathbf{f} \cdot \nabla\phi = 1, \quad \mathbf{x} \in \Omega \backslash \Gamma, \tag{21}$$

$$\phi(\mathbf{x}) = 0, \quad \mathbf{x} \in \Gamma, \tag{22}$$

where Γ is the locations of the harbors and islands. Here, we take $F = 1$ and $\mathbf{f} = (0.4, 0)^{\mathrm{T}}$. The computational domain is $\Omega = [0, 1]^2$ and harbor locations are

$$\Gamma = \left\{ \left(\frac{1}{4}, \frac{1}{5}\right), \left(\frac{5}{16}, \frac{1}{8}\right), \left(\frac{3}{5}, \frac{1}{5}\right), \left(\frac{3}{4}, \frac{3}{5}\right), \left(\frac{1}{2}, \frac{3}{4}\right), \left(\frac{1}{4}, \frac{1}{2}\right), \left(\frac{1}{8}, \frac{4}{5}\right), \left(\frac{1}{2}, \frac{1}{2}\right) \right\}.$$

The exact solution of the problem is also *not* smooth. The sparse-grid FE FPFS-WENO scheme and the sparse-grid RK FPFS-WENO scheme, with $\gamma = 1$ and the third-order WENO approximations to the derivatives are applied. The third-order WENO interpolation is used for prolongation in sparse-grid computations. Simulations are performed on both sparse grids with $N_r = 160, N_L = 3$, and the corresponding $1\,280 \times 1\,280$ single grid, to compare their numerical results. The obtained results of the RK FPFS-WENO scheme are reported in Fig. 7 (the numerical solution of the FE FPFS-WENO scheme is similar, so we omit to show its pictures which are basically the same as Fig. 7). As the previous examples, we observe that the numerical solutions by the sparse-grid FPFS-WENO schemes and their corresponding single-grid computations are comparable, and the nonlinear stability and high resolution properties of the FPFS-WENO schemes for resolving the non-smooth solution in this example are preserved well in the sparse-grid simulations. About computational efficiency, by applying the FE FPFS-WENO scheme, it takes 616.26 s of CPU time to complete the simulation in the sparse-grid computation, while 1 594.57 s of CPU time are needed for finishing the simulation in the corresponding single-grid computation. About



(a) Single-grid result **(b)** Sparse-grid result

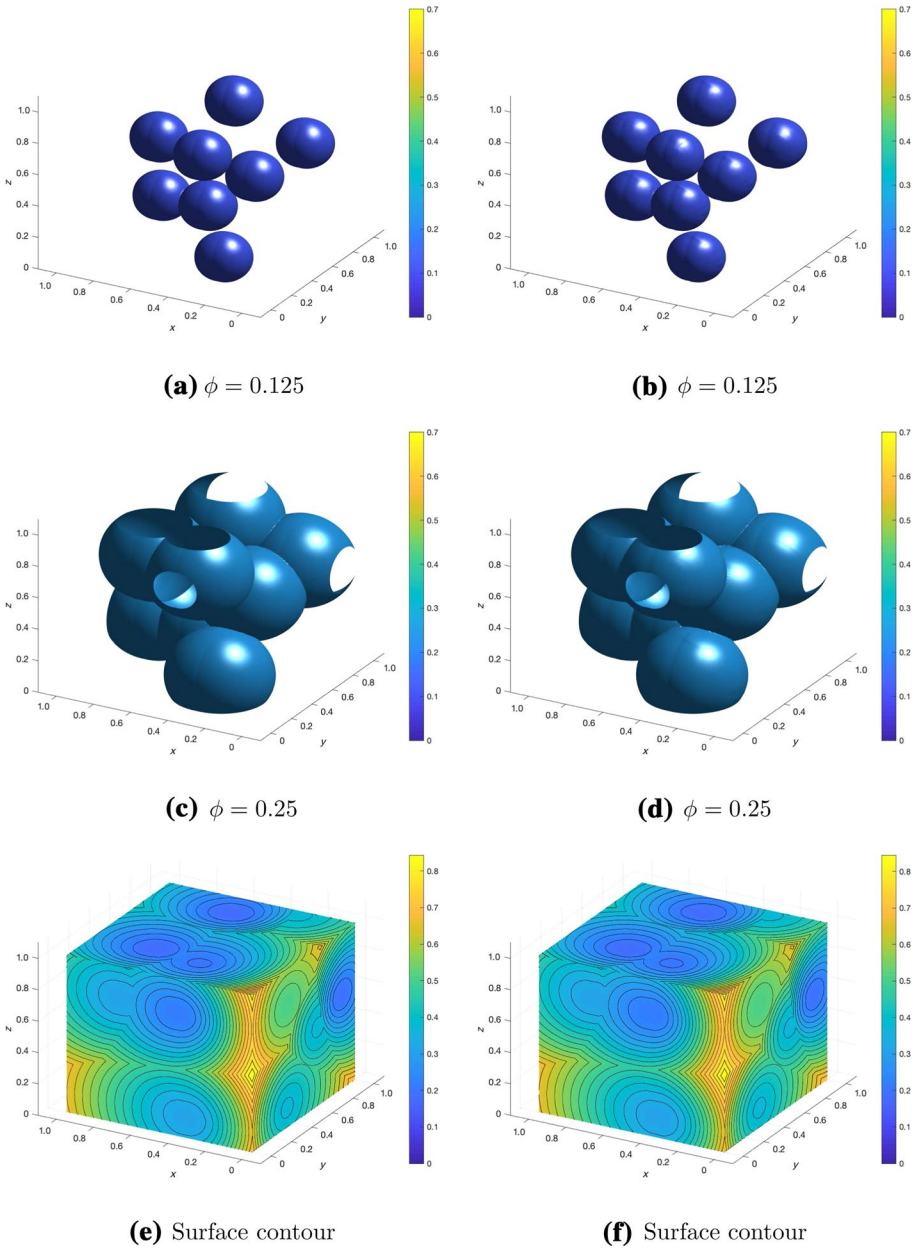
Fig. 7 Example 6, Case 1, numerical solutions of the 2D boat-sail problem by the RK FPFS-WENO scheme on sparse grids ($N_r = 160$ for root grid, finest level $N_L = 3$ in the sparse-grid computation) and the corresponding $1\,280 \times 1\,280$ single grid, using the third-order WENO interpolation for prolongation in the sparse-grid combination. The contour plots, 30 equally spaced contour lines from $\phi = 0$ to $\phi = 0.446\,1$. Red points are the harbor locations. **a** single-grid result; **b** sparse-grid result

61% CPU time is saved by performing the FE FPFS-WENO simulation on the sparse grids in this 2D example. For the RK FPFS-WENO scheme, it takes 892.71 s of CPU time to complete the simulation in the sparse-grid computation, while 2 155.63 s of CPU time are needed for finishing the simulation in the corresponding single-grid computation. About 59% CPU time is saved by performing the RK FPFS-WENO simulation on the sparse grids here. In this example, it is observed again that the FE FPFS-WENO scheme is more efficient than the RK FPFS-WENO scheme for both sparse-grid computation and single-grid computation.

Case 2 (3D). We consider the 3D case of the problem. The river flows with velocity $\mathbf{f} = (f_1, f_2, f_3)^{\mathrm{T}}$, and the boat travels at a maximum speed F, such that $F > |\mathbf{f}|$. The minimum travel time $\phi(x, y, z)$ from a point (x, y, z) to the nearest harbor or island can be found by solving the 3D version of the static Hamilton-Jacobi equation (21)–(22). We take $F = 1$ and $\mathbf{f} = (0.4, 0.4, 0)^{\mathrm{T}}$. The computational domain is $\Omega = [0, 1]^3$ and harbor locations are

$$\Gamma = \left\{ \left(\frac{1}{4}, \frac{1}{5}, \frac{1}{8}\right), \left(\frac{1}{3}, \frac{1}{7}, \frac{7}{9}\right), \left(\frac{3}{5}, \frac{1}{5}, \frac{4}{5}\right), \left(\frac{3}{4}, \frac{1}{2}, \frac{1}{4}\right), \left(\frac{1}{2}, \frac{3}{4}, \frac{4}{5}\right), \left(\frac{1}{4}, \frac{1}{2}, \frac{1}{2}\right), \left(\frac{1}{7}, \frac{4}{5}, \frac{3}{5}\right), \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{4}\right) \right\}.$$

Again, the exact solution of the 3D problem is *not* smooth. The sparse-grid FE FPFS-WENO scheme and the sparse-grid RK FPFS-WENO scheme, with $\gamma = 0.8$ and the third-order WENO approximations to the derivatives are applied in solving this 3D problem. The third-order WENO interpolation is used for prolongation in sparse-grid computations. Simulations are carried out on both sparse grids with $N_r = 80, N_L = 3$, and the corresponding $640 \times 640 \times 640$ single grid, for comparing the numerical results. The simulation results of the RK FPFS-WENO scheme are presented in Fig. 8 (the numerical solution of the FE FPFS-WENO scheme is similar, so we omit to show its pictures which are basically the same as Fig. 8). As the 2D case, the simulation results show that the numerical solutions by the sparse-grid FPFS-WENO schemes and their corresponding single-grid computations are comparable, and the nonlinear stability and high resolution properties of the FPFS-WENO schemes for resolving the non-smooth solution in this 3D boat-sail problem are preserved well in the sparse-grid simulations. In terms of computational efficiency, by applying the FE FPFS-WENO scheme, it takes 101 901.28 s of CPU time to complete the simulation in the sparse-grid computation, while 936 884.61 s of CPU time are needed for finishing the simulation in the corresponding single-grid computation. About 89% CPU time is saved by performing the FE FPFS-WENO simulation on the sparse grids in this 3D boat-sail problem. For the RK FPFS-WENO scheme, it takes 179 925.93 s of CPU time to complete the simulation in the sparse-grid computation, while 1 480 989.58 s of CPU time are needed for finishing the simulation in the corresponding single-grid computation. About 88% CPU time is saved by performing the RK FPFS-WENO simulation on the sparse grids here. In this 3D boat-sail problem, we also see that the FE FPFS-WENO scheme is more efficient than the RK FPFS-WENO scheme for both sparse-grid computation and single-grid computation, which is consistent with the observations for most of the other examples in this paper.



(a) $\phi = 0.125$ **(b)** $\phi = 0.125$

(c) $\phi = 0.25$ **(d)** $\phi = 0.25$

(e) Surface contour **(f)** Surface contour

Fig. 8 Example 6, Case 2, numerical solutions of the 3D boat-sail problem by the RK FPFS-WENO scheme on sparse grids ($N_r = 80$ for root grid, finest level $N_L = 3$ in the sparse-grid computation) and the corresponding $640 \times 640 \times 640$ single grid, using the third-order WENO interpolation for prolongation in the sparse-grid combination. **a**, **c**, **e** single-grid results; **b**, **d**, **f**: sparse-grid results; **a**, **b** the contour plots for $\phi = 0.125$; **c**, **d** the contour plots for $\phi = 0.25$; **e**, **f** the contour plots for the whole surface

4 Conclusions

In this technical note, we apply the sparse-grid combination technique to the third-order fixed-point fast sweeping WENO schemes for efficiently computing solutions of multidimensional Eikonal equations. Due to their sophisticated nonlinearity, more computational costs than many other schemes are needed in high-order WENO simulations, especially for multidimensional problems. Here, we follow our previous work and implement the FPFS-WENO schemes on sparse grids. A third-order WENO interpolation is applied in the prolongation step of the sparse-grid combination technique, for robust computations of non-smooth solutions of Eikonal equations in sparse-grid simulations. Numerical experiments on 2D and 3D problems are performed for the sparse-grid FPFS-WENO methods to show that a more efficient algorithm than the regular FPFS-WENO methods on single grids to solve the multidimensional Eikonal equations is achieved, with about $50\% \sim 90\%$ CPU time costs being saved on refined meshes, by comparing with the corresponding single-grid simulations in examples here.

In this technical note, we focus on the efficient implementation of the FPFS-WENO schemes on sparse grids and their numerical experiments. We would like to point out that there are still quite a few open problems to be investigated further for the sparse-grid methods here. For example, it is still an open problem on how to perform theoretical error analysis for such kind of nonlinear sparse-grid schemes, although that has been done for the linear schemes in solving linear time-dependent PDEs in the literature. In the numerical experiments, we find that the L^∞ errors of sparse-grid computations are larger than the corresponding single-grid computations. How to improve the accuracy of these sparse-grid schemes is an interesting and important problem. Furthermore, the sparse-grid FPFS-WENO methods studied here should be able to be extended to higher order accuracy and to solve more complicated static H-J equations. All of these important open problems will be our future work.

Compliance with Ethical Standards

Conflict of Interest The authors declare that there is no conflict of interest.

References

1. Alves, M.A., Cruz, P., Mendes, A., Magalhães, F.D., Pinho, F.T., Oliveira, P.J.: Adaptive multiresolution approach for solution of hyperbolic PDEs. Comput. Methods Appl. Mech. Eng. **191**, 3909–3928 (2002)
2. Aurenhammer, F.: Voronoi diagrams — a survey of a fundamental geometric data structure. ACM Comput. Surv. **23**, 345–405 (1991)
3. Bungartz, H.-J., Griebel, M.: Sparse grids. Acta Numer. **13**, 147–269 (2004)
4. Chen, W., Chou, C.-S., Kao, C.-Y.: Lax-Friedrichs fast sweeping methods for steady state problems for hyperbolic conservation laws. J. Comput. Phys. **234**, 452–471 (2012)
5. Chew, L.P., Drysdale, R.L.: Voronoi diagrams based on convex distance functions. In: SCG '85: Proceedings of the First Annual Symposium and Computational Geometry, New York, NY, Association for Computing Machinery, pp. 235–244 (1985)
6. Chou, C.-S., Shu, C.-W.: High order residual distribution conservative finite difference WENO schemes for steady state problems on non-smooth meshes. J. Comput. Phys. **214**, 698–724 (2006)
7. Crandall, M.G., Lions, P.L.: Viscosity solutions of Hamilton-Jacobi equations. Trans. Am. Math. Soc. **277**, 1–42 (1983)
8. Dijkstra, E.W.: A note on two problems in connection with graphs. Numer. Math. **1**, 269–271 (1959)

9. Fomel, S., Luo, S., Zhao, H.: Fast sweeping method for the factored Eikonal equation. J. Comput. Phys. **228**, 6440–6455 (2009)
10. Garcke, J.: Sparse grids in a nutshell. In: Garcke, J., Griebel, M. (eds) Sparse Grids and Applications, Lecture Notes in Computational Science and Engineering, vol. 88, pp. 57–80. Springer, New York (2013)
11. Gerstner, T., Griebel, M.: Dimension-adaptive tensor-product quadrature. Computing **71**, 65–87 (2003)
12. Griebel, M.: Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences. Computing **61**, 151–179 (1998)
13. Griebel, M., Schneider, M., Zenger, C.: A combination technique for the solution of sparse grid problems. In: Beauwens, R., de Groen, P. (eds) Iterative methods in linear algebra, pp. 263–281. North-Holland, Amsterdam (1992)
14. Guo, W., Cheng, Y.: An adaptive multiresolution discontinuous Galerkin method for time-dependent transport equations in multidimensions. SIAM J. Sci. Comput. **39**, A2962–A2992 (2017)
15. Hegland, M.: Adaptive sparse grids. ANZIAM J. **44**, C335–C353 (2002)
16. Kao, C.Y., Osher, S., Qian, J.: Lax-Friedrichs sweeping schemes for static Hamilton-Jacobi equations. J. Comput. Phys. **196**, 367–391 (2004)
17. Lastdrager, B., Koren, B., Verwer, J.: Solution of time-dependent advection-diffusion problems with the sparse-grid combination technique and a Rosenbrock solver. Comput. Methods Appl. Math. **1**, 86–99 (2001)
18. Lastdrager, B., Koren, B., Verwer, J.: The sparse-grid combination technique applied to time-dependent advection problems. Appl. Numer. Math. **38**, 377–401 (2001)
19. Li, F., Shu, C.-W., Zhang, Y.-T., Zhao, H.-K.: A second order discontinuous Galerkin fast sweeping method for Eikonal equations. J. Comput. Phys. **227**, 8191–8208 (2008)
20. Li, L., Zhu, J., Zhang, Y.-T.: Absolutely convergent fixed-point fast sweeping WENO methods for steady state of hyperbolic conservation laws. J. Comput. Phys. **443**, 1–24 (2021)
21. Lu, D., Chen, S., Zhang, Y.-T.: Third order WENO scheme on sparse grids for hyperbolic equations. Pure Appl. Math. Q. **14**, 57–86 (2018)
22. Lu, D., Zhang, Y.-T.: Krylov integration factor method on sparse grids for high spatial dimension convection-diffusion equations. J. Sci. Comput. **69**, 736–763 (2016)
23. Nishida, T., Sugihara, K.: Voronoi diagram in a flow field. In: Ibaraki, T., Katoh, N., Ono, H. (eds) Algorithms and Computation, ISAAC 2003 Lecture Notes in Computer Science, vol. 2906, pp. 26–35. Berlin, Springer (2003)
24. Nishida, T., Sugihara, K.: Boat-sail Voronoi diagram on a curved surface. Jpn. J. Ind. Appl. Math. **22**, 267–278 (2005)
25. Noordmans, J., Hemker, P.W.: Application of an adaptive sparse-grid technique to a model singular perturbation problem. Computing **65**, 357–378 (2000)
26. Obersteiner, M., Bungartz, H.-J.: A generalized spatially adaptive sparse grid combination technique with dimension-wise refinement. SIAM J. Sci. Comput. **43**, A2381–A2403 (2021)
27. Okabe, A., Boots, B., Sugihara, K., Chu, S.N.: Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, 2nd edn. Wiley Series in Probability and Statistics, Wiley, Hoboken, NJ (2000)
28. Osher, S., Shu, C.-W.: High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations. SIAM J. Numer. Anal. **28**, 907–922 (1991)
29. Qian, J., Zhang, Y.-T., Zhao, H.-K.: Fast sweeping methods for Eikonal equations on triangular meshes. SIAM J. Numer. Anal. **45**, 83–107 (2007)
30. Qian, J., Zhang, Y.-T., Zhao, H.-K.: A fast sweeping method for static convex Hamilton-Jacobi equations. J. Sci. Comput. **31**, 237–271 (2007)
31. Rouy, E., Tourin, A.: A viscosity solutions approach to shape-from-shading. SIAM J. Numer. Anal. **29**, 867–884 (1992)
32. Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. Proc. Natl. Acad. Sci. U.S.A **93**, 1591–1595 (1996)
33. Sethian, J.A., Vladimirsky, A.: Ordered upwind methods for static Hamilton-Jacobi equations. Proc. Natl. Acad. Sci. U.S.A. **98**, 11069–11074 (2001)
34. Sethian, J.A., Vladimirsky, A.: Ordered upwind methods for static Hamilton-Jacobi equations: theory and algorithms. SIAM J. Numer. Anal. **41**, 325–363 (2003)
35. Shu, C.-W.: Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. In: Quarteroni, A. (ed) Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, Lecture Notes in Mathematics, vol. 1697, pp. 325–432. Springer-Verlag, New York (1998)
36. Wu, L., Zhang, Y.-T.: A third order fast sweeping method with linear computational complexity for Eikonal equations. J. Sci. Comput. **62**, 198–229 (2015)

37. Wu, L., Zhang, Y.-T., Zhang, S., Shu, C.-W.: High order fixed-point sweeping WENO methods for steady state of hyperbolic conservation laws and its convergence study. Commun. Comput. Phys. **20**, 835–869 (2016)
38. Xiong, T., Zhang, M., Zhang, Y.-T., Shu, C.-W.: Fifth order fast sweeping WENO scheme for static Hamilton-Jacobi equations with accurate boundary treatment. J. Sci. Comput. **45**, 514–536 (2010)
39. Zenger, C.: Sparse grids. In: Hackbusch, W. (ed) Notes on Numerical Fluid Mechanics, vol. 31, pp. 241–251. Vieweg, Braunschweig (1991)
40. Zhang, Y.-T., Chen, S., Li, F., Zhao, H., Shu, C.-W.: Uniformly accurate discontinuous Galerkin fast sweeping methods for Eikonal equations. SIAM J. Sci. Comput. **33**, 1873–1896 (2011)
41. Zhang, Y.-T., Zhao, H.-K., Chen, S.: Fixed-point iterative sweeping methods for static Hamilton-Jacobi equations. Methods Appl. Anal. **13**, 299–320 (2006)
42. Zhang, Y.-T., Zhao, H.-K., Qian, J.: High order fast sweeping methods for static Hamilton-Jacobi equations. J. Sci. Comput. **29**, 25–56 (2006)
43. Zhao, H.-K.: A fast sweeping method for Eikonal equations. Math. Comput. **74**, 603–627 (2005)
44. Zhao, H., Osher, S., Merriman, B., Kang, M.: Implicit and non-parametric shape reconstruction from unorganized points using variational level set method. Comput. Vis. Image Underst. **80**, 295–319 (2000)
45. Zhu, X., Zhang, Y.-T.: Fast sparse grid simulations of fifth order WENO scheme for high dimensional hyperbolic PDEs. J. Sci. Comput. **87**, Article number 44, 1–38 (2021)