Positivity-preserving high-order schemes for conservation laws on arbitrarily distributed point clouds with a simple WENO limiter

Jie Du[1] and Chi-Wang Shu[2]

Abstract

This is an extension of our earlier work [9] in which a high order stable method was constructed for solving hyperbolic conservation laws on arbitrarily distributed point clouds. An algorithm of building a suitable polygonal mesh based on the random points was given and the traditional discontinuous Galerkin (DG) method was adopted on the constructed polygonal mesh. Numerical results in [9] show that the current scheme will generate spurious numerical oscillations when dealing with solutions containing strong shocks. In this paper, we adapt a simple weighted essentially non-oscillatory (WENO) limiter, originally designed for DG schemes on two-dimensional unstructured triangular meshes [27], to our high order method on polygonal meshes. The objective of this simple WENO limiter is to simultaneously maintain uniform high order accuracy of the original method in smooth regions and control spurious numerical oscillations near discontinuities. The WENO limiter we adopt is particularly simple to implement and will not harm the conservativeness and compactness of the original method. Moreover, we also extend the maximum-principle-satisfying limiter for the scalar case and the positivity-preserving limiter for the Euler system to our method. Numerical results for both scalar equations and Euler systems of compressible gas dynamics are provided to illustrate the good behavior of these limiters.

Key Words: WENO limiter, positivity-preserving, arbitrarily distributed point cloud, conservation laws, high order.

[1]Department of Mathematics, The Chinese University of Hong Kong, Hong Kong, P.R. China. E-mail: jdu@math.cuhk.edu.hk

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

1 Introduction

In this paper, we are interested in solving the following two-dimensional hyperbolic conservation law

$$\begin{cases} u_t + \nabla \cdot \mathbf{F}(u) = 0, \\ u(\mathbf{x}, 0) = u_0(\mathbf{x}), \end{cases} \tag{1}$$

in the computational domain $\Omega \in \mathbb{R}^2$, where $\mathbf{F} = (f(u), g(u))$ is the flux vector, $\mathbf{x} = (x, y) \in \Omega$ and $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$. Here, u, f and g can be either scalars or vectors. We assume that an arbitrarily distributed point cloud, namely a finite set of isolated, unstructured points $\{\mathbf{x}_i\}_{i=1}^N$, together with the values of the initial condition at these points $\{u_0(\mathbf{x}_i),\ i = 1, \cdots, N\}$, is given, and we seek an algorithm to obtain the point values of the numerical solution in this point cloud for later time. One possible scenario for such a set-up could be that the point clouds are locations of the observation posts or places where measurements are being made, and evolution data would need to be predicted and compared with future measurements. Unlike traditional problems where a grid or mesh is given and the initial condition is assumed given as a function, here we only have the knowledge of the initial values on the arbitrarily distributed point cloud. Hence, it is difficult to apply the classical well developed grid- or mesh-based computational methods to this problem directly, such as the finite difference (FD) methods, the finite volume (FV) methods, and the finite element (FE) methods. Meshless methods [1, 14] are alternatives to traditional mesh-based methods. They provide numerical solutions in terms of nodes without using any mesh to connect them or using a background mesh only minimally. However, to our best knowledge, there are few papers devoted to meshless methods for solving time-dependent hyperbolic conservation laws [18], and conservation and stability appear to be particularly difficult for meshless methods for such PDEs.

Recently, we designed a high order stable method for this problem in [9]. In order to utilize traditional mesh-based methods which have many important good properties, we provided a way to generate a suitable mesh based on the given point cloud. Each cell in the mesh is a polygon, and contains a minimum number of points in the original point cloud so that a polynomial of a pre-defined degree can be constructed to represent the initial condition to high order accuracy. Once the polygonal mesh is constructed, we march the piecewise polynomial numerical solution in time by choosing the classical discontinuous Galerkin (DG) methods. Due to the good properties of the DG method, the new constructed method is conservative, stable and high order accurate, both for linear and nonlinear equations.

The main difficulty in solving the conservation laws (1) is that solutions may contain discontinuities even if the initial conditions are smooth. As we can see in the numerical examples in [9], when dealing with solutions containing strong shocks, our current scheme on the polygonal mesh will generate spurious numerical oscillations, just as DG schemes without limiters on regular triangular or rectangular meshes will do. These spurious oscillations may lead to nonlinear instability and eventual blow-ups of the codes. Therefore, we need to apply nonlinear limiters to control these oscillations for our polygonal mesh.

To achieve the full potential of high order accuracy and efficiency of our method, we would like to find a robust high order limiting procedure to simultaneously maintain uniform high order accuracy in smooth regions and control spurious numerical oscillations near discontinuities. There are many successful works based on the WENO methodology [10, 11] for DG methods on two-dimensional unstructured triangular meshes, which would serve such a purpose. Zhu et al. [22] designed limiters using the usual WENO reconstruction. They use the cell averages in an adaptive stencil to reconstruct the values of the solutions at certain points in the target cell. Note that the DG method is compact, that is, it uses the information only from the target cell and its immediate neighboring cells. However, the reconstruction stencil in [22] contains not only the immediate neighboring cells of the target cell but also the neighbors' neighbors. To reduce the width of the reconstruction stencil, [13] adopted a Hermite type WENO procedure, which uses not only the cell averages but also the first derivative or first order moment information in the stencil. However the information of neighbors' neighbors is still needed for higher order methods. Also, it is complicated to perform the usual WENO procedure or the Hermite type WENO procedure on unstructured meshes, with the possibility of negative linear weights, as we would need to use extra special treatments to handle them [19]. Recently, a new and simple WENO limiter [27] was designed. This WENO limiter attempts to reconstruct the entire polynomial on the target cell, instead of reconstructing point values or moments in the classical WENO reconstructions. In fact, the entire reconstruction polynomial is just a convex combination of polynomials on the target cell and its immediate neighboring cells (with suitable adjustments for conservation). Hence, it will not harm the compactness of the DG method. Also, the linear weights are always positive.

All the above mentioned WENO limiters are designed for DG methods on triangular meshes. In our method [9] to solve conservation laws on random points, the generated mesh consists of polygonal cells. Each polygon within the mesh can have arbitrary number of edges and can be in any shape (even non-convex). Also, two neighboring polygonal cells usually have more than one common edges. As far as we know, there has been no prior work on WENO limiters on such a complex mesh. To control the numerical oscillations near discontinuities in our method on random points, we extend the work in [27] to the polygonal mesh. Numerical examples in this paper show that we do achieve the purpose to simultaneously maintain uniform high order accuracy in smooth regions and control spurious numerical oscillations near discontinuities on the complex polygonal mesh.

An important property of the unique entropy solution to the scalar conservation law (1) is that it satisfies a strict maximum principle, i.e., if

$$M = \max_{\mathbf{x}} u_0(\mathbf{x}), \qquad m = \min_{\mathbf{x}} u_0(\mathbf{x}), \tag{2}$$

then $u(\mathbf{x}, t) \in [m, M]$ for any $\mathbf{x} \in \Omega$ and $t > 0$. In particular, the solution will not be negative if $m \geqslant 0$. For hyperbolic conservation law systems, the entropy solutions generally do not satisfy the maximum principle. However, some important quantities should be non-negative physically. For instance, density and pressure in compressible Euler equations, and water height in shallow water equations. In practice, failure of preserving positivity of such quantities may cause blow-ups of the computation. In [23] and [24], genuinely high order accurate maximum-principle-satisfying and positivity-preserving schemes were designed for scalar conservation laws and compressible Euler equations on rectangular meshes.

An important component of the maximum-principle-satisfying and the positivity-preserving limiters is to find a quadrature rule in the target cell which is accurate enough, contains Gauss quadrature points on each edge of the cell, and has positive weights. [26] developed a special quadrature rule which satisfies the above conditions over a triangle and thus made a nontrivial extension of the aforementioned schemes to triangular meshes. [20] further discussed an extension to the reactive Euler equations and proposed a slightly different but very robust and simpler implementation to enforce the positivity of pressure. In the quadrature rules in all these methods, one need to evaluate the point values on quadrature points inside the target cell. The implementation is relevantly expensive if the approximation polynomials are not available, as in the finite volume WENO method. [25] proposed an alternative and simpler implementation to achieve the same maximum principle or positivity. Instead of computing all the point values inside the cell, one only need to compute the value at one single point, determined by the mean value theorem at an unknown location, which results in a reduction of computational cost and complexity of the procedure for WENO schemes. The same trick was also used in [21] for the shallow water equations.

In this paper, our high order method on random points with the WENO limiter does not in general satisfy a strict maximum-principle for the scalar case or a positivity-preserving property for the Euler system. Hence, we also extend the aforementioned limiters into our scheme. Note that we generate a polygonal mesh based on the given point cloud and apply DG method upon it. Hence, the first thing we need to do is to find a suitable quadrature rule on the complex polygonal mesh. The easiest way is to cut the polygon into small triangles and then use the quadrature rule in [26] for each triangle, which contains all Gauss quadrature points on each edges of the triangle and several points inside the triangle. By doing so, the number of quadrature points inside the polygon is huge. The cost will be large even though we know the approximation polynomials on each cell in our method. In fact, we do not need to consider the quadrature points on the edges of triangles which are inside the polygon. Hence, we use the trick in [21, 25] and extend it to our polygonal mesh, namely lumping all points inside the cell into just one. Thus, we just need values on this one (artificial) point plus all the points on the polygonal cell boundary to construct the limiter. When enforcing the positivity of the pressure for the Euler system, we adopt the idea in [20] to simplify the implementation.

This paper is organized as follows. We first review the high order method [9] in Section 2. In Section 3, we describe the details of how to introduce the WENO limiting procedure to this method. In Section 4, we describe the detailed procedure to construct a maximum-principle-satisfying limiter for the scalar case and a positivity-preserving limiter for the Euler system. In Section 5, numerical experiments are provided to verify the accuracy and stability of our scheme with these limiters. Finally, concluding remarks are provided in Section 6.

2 Formulation of the high order method on random points

For self consistency, in this section, we give a brief overview of the high order method in [9] to solve our problem. In Section 2.1, we first recall how to generate an appropriate mesh based on the given point cloud and approximate the initial discrete data with a function. Then in Section 2.2, we show the formulation of the classical DG method on our constructed polygonal mesh.

2.1 Mesh generation and the initial data approximation

At the initial time, we are given a set of random points $\{\mathbf{x}_i\}_{i=1}^N$, as shown by the blue points in Figure 1. We need to cover the computational domain Ω with an appropriate mesh $\{V_j\}_{j=1}^M$, based on the given point cloud. The purpose is to represent the initial condition within each cell V_j with a polynomial in $P^k(V_j)$, by interpolating or fitting the given initial values on the given points in this cell. Here, $P^k(V_j)$ is the space of polynomials of degree up to k on V_j. Hence, each cell should contain at least $K = \frac{(k+1)(k+2)}{2}$ points, where K is the degree of freedom of $P^k(V_j)$. The locations of the given points in each cell is crucial. For example, when $k = 1$, the three interpolation points can not be aligned along a straight line.

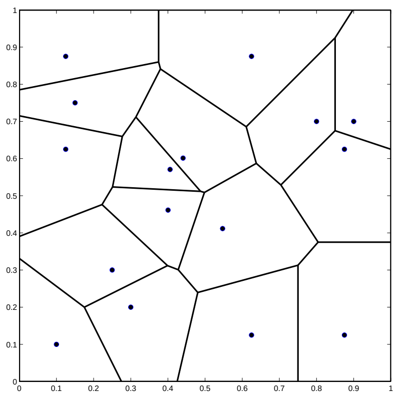


Figure 1: **Voronoi diagram for sixteen random points**

The mesh generation includes two steps. The first step is to divide the computational domain into small Voronoi regions $\{\widehat{V}_i\}_{i=1}^N$. Each Voronoi region $\widehat{V}_i$ contains only one point $\mathbf{x_i}$ in the give point cloud and consists of all locations in Ω closer to $\mathbf{x_i}$ than to any other given point in the cloud:

$$\widehat{V}_i = \{\mathbf{x} \in \Omega \quad | \quad |\mathbf{x} - \mathbf{x}_i| < |\mathbf{x} - \mathbf{x}_j| \quad for\ j = 1, \cdots, N, j \neq i\}. \tag{3}$$

We call $\mathbf{x}_i$ as the generator of $\widehat{V}_i$. Note that Voronoi regions are all polygons. The set $\{\widehat{V}_i\}_{i=1}^N$ forms a tessellation of Ω and is called a Voronoi diagram. Figure 1 shows the Voronoi diagram for the given random points. For a comprehensive treatment, see [15].

The second step is to carefully group these small Voronoi regions into cells. Each cell consists of at least K adjacent Voronoi regions. The locations of the corresponding generator points should enable us to interpolate or fit the initial values with a polynomial in P^k. Considering the cell V_j, we denote the generator points in it as $\{\mathbf{x}_l^j\}_{l=1}^L$. By choosing a set of basis functions $\{\phi_m(\mathbf{x})\}_{m=1}^K$ in V_j, we attempt to interpolate the initial solution in V_j by

$$u_j^0(\mathbf{x}) = \sum_{m=1}^{K} \alpha_m \phi_m(\mathbf{x}),\ \mathbf{x} \in V_j, \tag{4}$$

such that

$$u_j^0(\mathbf{x}_l^j) = u_0(\mathbf{x}_l^j),\ l = 1, \cdots, L. \tag{5}$$

By denoting $A = (a_{l,m})$ with $a_{l,m} = \phi_m(\mathbf{x}_l^j)$, $\vec{\alpha} = (\alpha_m)$ and $\vec{b} = (u_0(\mathbf{x}_l^j))$, we can rewrite Equations (4) and (5) into the following matrix version:

$$A\vec{\alpha} = \vec{b}. \tag{6}$$

When $L = K$, and if A is invertible, we can solve the above system of equations to obtain $\vec{\alpha}$. We denote the condition number of A as $\kappa(A)$, which gives a bound on how inaccurate the solution will be. We can easily see that each row of A relates to one generator point. During the grouping procedure, even when $L < K$, we can still compute $\kappa(A)$, which can be viewed as a measure of closeness to a rank loss [8]. Hence, a crucial rule in our algorithm is to bound $\kappa(A)$ each time we add points into V_j, by using a threshold value δ. We summarize the algorithm to group Voronoi regions into cells as follows. For a more detailed explanation, we refer to [9].

- We check all generator points one by one, starting from $\mathbf{x}_1$. If $\mathbf{x}_i$ has not been distributed into any cells, we now start to create a new cell, say, V_j, and denote $\mathbf{x}_1^j = \mathbf{x}_i$.

- Add all available immediate neighbors (immediate neighbors that have not been assigned to any cells) of $\mathbf{x}_1^j$ into V_j one by one in ascending order of their index numbers by checking the corresponding $\kappa(A)$.

- If the number of points in V_j is still less than K, we use the following algorithm to add only one point each time, until the total number of points reaches K: we find out all available immediate neighboring points of V_j and rank them by the number of their immediate neighbors in V_j in descending order. We check them one by one until we find one that the corresponding $\kappa(A)$ is less than δ.

- If we still can not find enough points that can pass the condition number test after checking all available immediate neighbors of V_j, we put aside $\mathbf{x}_i$ for a while, set $\mathbf{x}_1^j = \mathbf{x}_{i+1}$, and use the above algorithm to construct V_j again. At the end of the algorithm, if $\mathbf{x}_i$ is still available, we add it to the nearest existing cell. By doing so, the number of points in this cell will be larger than K and we need to solve the following least squares problem

$$\min_{\vec{\alpha}} ||\vec{b} - A\vec{\alpha}||_2 \tag{7}$$

  to determine the fitting polynomial.

2.2 DG method on the polygonal mesh

In the previous section, we have already divided Ω into polygonal cells $\{V_j\}_{j=1}^M$ and approximated the initial value with a function $u^0(\mathbf{x})$: $u^0(\mathbf{x})|_{V_j} = u_j^0(\mathbf{x}), 1 \leqslant j \leqslant M$. It belongs to the following finite element space consisting of piecewise polynomials

$$V_h^k = \{v : v|_{V_j} = v_j \in P^k(V_j), 1 \leqslant j \leqslant M\}, \tag{8}$$

which is just the solution as well as the test function space in the DG method. In this section, we adopt the Runge-Kutta discontinuous Galerkin (RKDG) method carried out by Cockburn et al. in a series of papers [3, 4, 5, 6, 7] to march the initial function in time. It uses DG discretization in space and the Runge-Kutta method in time.

The DG method in space is defined as: find the unique solution $u_h \in V_h^k$ such that, for all test functions $v_h \in V_h^k$ and all $1 \leqslant j \leqslant M$, we have

$$\int_{V_j} (u_h)_t v_h d\mathbf{x} - \int_{V_j} \mathbf{F}(u_h) \cdot \nabla v_h d\mathbf{x} + \int_{\partial V_j} \widehat{F}^n v_h ds = 0, \tag{9}$$

where $\mathbf{n}$ is the outward unit normal vector of the cell boundary ∂V_j. $\widehat{F}^n = \widehat{F}^n(u_h^-, u_h^+, \mathbf{n})$ is a monotone numerical flux define on ∂V_j for the scalar case or an approximate Riemann solver for the system case, consistent with $\mathbf{F} \cdot \mathbf{n}$. Here u_h^- and u_h^+ are the values of u_h on ∂V_j taken from the inside the cell V_j and the outside of V_j, respectively.

The semi-discrete scheme (9) can be written as

$$(u_h)_t = L(u_h, t)$$

where $L(u_h, t)$ is a spatial discretization operator. For the time discretization, we use the total variation diminishing (TVD) third order Runge-Kutta method [17]. Starting from $u^n \in V_h^k$ at time level n, we compute u^{n+1} by

$$\begin{aligned} u^{(1)} &= u^n + \Delta t L(u^n, t^n), \\ u^{(2)} &= \frac{3}{4}u^n + \frac{1}{4}u^{(1)} + \frac{1}{4}\Delta t L(u^{(1)}, t^n + \Delta t), \\ u^{n+1} &= \frac{1}{3}u^n + \frac{2}{3}u^{(2)} + \frac{2}{3}\Delta t L(u^{(2)}, t^n + \frac{1}{2}\Delta t). \end{aligned} \tag{10}$$

3 The WENO limiter

As explained in [9], the method is conservative, stable, and high order accurate. However, numerical results show that it will generate spurious numerical oscillations when dealing with solutions containing strong shocks. In this section, we extend the WENO limiter in [27] designed for the DG method on triangular meshes to our method on polygonal meshes. The goal is to control the oscillations for shocked flows as well as to maintain the original high order accuracy in smooth regions.

Note that the TVD third order Runge-Kutta method is just a convex combination of first order forward Euler steps. For simplicity, we only consider the forward Euler time discretization in this section. We denote the numerical solution on the n-th time level as $u^n \in V_h^k$ and denote

$$u^n|_{V_j}(\mathbf{x}) = u_j^n(\mathbf{x}) \in P^k(V_j). \tag{11}$$

Starting from $u^n \in V_h^k$, we first reconstruct it to obtain a new function $u^{n,new} \in V_h^k$ and then match the new function in time. That is, find $u^{n+1} \in V_h^k$, such that for all test functions $v_h \in V_h^k$ and all $1 \leqslant j \leqslant M$, we have

$$\int_{V_j} \frac{u_j^{n+1} - u_j^{n,new}}{\Delta t} v_h d\mathbf{x} - \int_{V_j} \mathbf{F}(u^{n,new}) \cdot \nabla v_h d\mathbf{x} + \int_{\partial V_j} \widehat{F}^{n,new} v_h ds = 0, \tag{12}$$

where $\widehat{F}^{n,new}$ is the flux computed by using the new function $u^{n,new}$. For the high order Runge-Kutta method, we only need to repeat the same procedure for each Runge-Kutta inner stage.

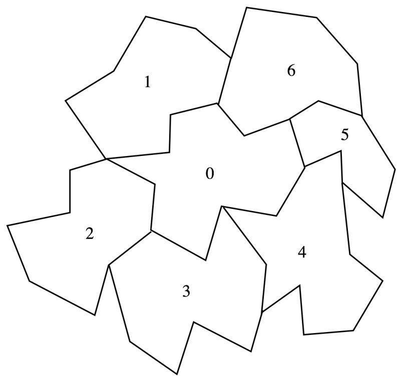


Figure 2: The stencil of the DG method

One can see that when using the DG method to solve u^{n+1} on the target V_j, we only need to use functions on V_j and its immediate neighboring cells, as shown in Figure 2. We denote this stencil as $S_j = \{V_j, V_{j(1)}, V_{j(2)}, \cdots, V_{j(N_j)}\}$, where N_j is the number of immediate neighboring cells of V_j. Note that unlike the triangular mesh case, N_j is usually not equal to the the number of edges of V_j since two neighboring cells usually share more than one common edges. In the WENO limiting procedure on V_j, we use the same stencil to obtain $u^{n,new}$. Hence, the WENO limiter can maintain the compactness of the DG method which makes the method very suitable for parallel computing. For convenience, we denote the solution polynomials on the stencil S_j as

$$p_0(\mathbf{x}) = u_j^n(\mathbf{x}), \qquad p_l(\mathbf{x}) = u_{j(l)}^n(\mathbf{x}),\ l = 1, \cdots, N_j. \tag{13}$$

3.1 The WENO limiting procedure for the scalar case

In this section, we consider conservation laws as shown in Eq. (1), where u, f and g are scalars. Before we perform the WENO limiter, we first need to find out all troubled cells which contain possible shocks and need the limiting procedure. As in [27], we adopt the KXRCF shock detection technique [12]. We divide the boundary ∂V_j into two parts: ∂V_j^- and ∂V_j^+, where the flow is into $(\mathbf{F}'(u)\cdot\mathbf{n} < 0)$ and out of $(\mathbf{F}'(u)\cdot\mathbf{n} > 0)$ V_j respectively. The target cell V_j is identified as a troubled cell when

$$\frac{\left|\int_{\partial V_j^-}\big(p_0(\mathbf{x}) - p_l(\mathbf{x})\big)ds\right|}{h^{\frac{k+1}{2}}|\partial V_i^-|\cdot\|p_0\|} > C_k, \tag{14}$$

where C_k is a constant. We choose h as the maximum distance between two points in V_j. $p_l(\mathbf{x})$ denote solutions on the neighboring cells sharing ∂V_j^-, and $\|\cdot\|$ is the standard L^2 norm in V_j. We remark that the KXRCF troubled cell indicator is just one of the many possibilities and may not be the best one. We use it here for its simplicity, as our main focus of this paper is not on troubled cell indicators. We refer the readers to [16] for a detailed discussion about different troubled-cell indicators.

Assuming that V_j is a troubled cell, we now reconstruct $u_j^n(\mathbf{x})$ to obtain $u_j^{n,new}(\mathbf{x})$. In order to maintain the original cell average of u_j^n in cell V_j, which is essential to keep the conservativeness, we modify solutions on the neighboring cells as:

$$\tilde{p}_l(\mathbf{x}) = p_l(\mathbf{x}) - \frac{1}{|V_j|}\int_{V_j} p_l(\mathbf{x})d\mathbf{x} + \frac{1}{|V_j|}\int_{V_j} p_0(\mathbf{x})d\mathbf{x}, \qquad l = 1, 2, \cdots, N_j, \tag{15}$$

where $|V_j|$ is the area of V_j. For notational consistency, we also denote $\tilde{p}_0(\mathbf{x}) = p_0(\mathbf{x})$. The nonlinear WENO reconstructed polynomial on cell V_j is defined by a convex combination of these modified polynomials:

$$u_j^{n,new}(\mathbf{x}) = \sum_{l=0}^{N_j} \omega_l \tilde{p}_l(\mathbf{x}). \tag{16}$$

From Eq. (16), we know that $u_j^{n,new}$ has the same cell average as $p_0(\mathbf{x})$ as long as $\sum_{l=0}^{N_j} \omega_l = 1$. Hence, we define the normalized nonlinear weights as

$$\omega_l = \frac{\tilde{\omega}_l}{\sum_{m=0}^{N_j} \tilde{\omega}_m}, \qquad l = 0, 1, \cdots, N_j, \tag{17}$$

where the non-normalized nonlinear weights $\tilde{\omega}_l$ are defined as

$$\tilde{\omega}_l = \frac{\gamma_l}{(\varepsilon + \beta_l)^2}. \tag{18}$$

Here $\varepsilon > 0$ is introduced to avoid the denominator to become 0. We take $\varepsilon = 10^{-6}$ in all our numerical tests. β_l is the smoothness indicator, which measures how smooth the function $\tilde{p}_l$ is on the target cell V_j. As in [2, 11], we define β_l as

$$\beta_l = \sum_{|s|=1}^{k} |V_j|^{|s|-1} \int_{V_j} \left(\frac{\partial^{|s|}}{\partial x^{s_1} \partial y^{s_2}} \tilde{p}_l\right)^2 d\mathbf{x}, \tag{19}$$

where $s = (s_1, s_2)$. When $\tilde{p}_l$ is not smooth, then β_l will be large and hence ω_l will be relevantly small.

The linear weights $\{\gamma_l\}$ are a set of positive numbers adding up to one. Note that since $\tilde{p}_l$ for $l = 0, 1, \cdots, N_j$ are all $(k+1)$-th order approximations to the exact solution in smooth regions, there are no extra requirements on the linear weights in order to maintain the original high order accuracy. As discussed in [27], since for smooth solutions the central cell is usually the best one, we put a larger linear weight on the central cell than on the neighboring cells. In this paper, we take

$$\gamma_0 = 1 - 0.0005 N_j, \qquad \gamma_l = 0.0005, \qquad l = 1, \cdots, N_j, \tag{20}$$

which can maintain the original high order accuracy in smooth regions and can keep essentially non-oscillatory shock transitions in our numerical examples.

For each cell V_j, $j = 1, \cdots, M$, if it is a troubled cell, we replace the entire solution polynomial u_j^n with the new reconstructed polynomial $u_j^{n,new}$, which is a convex combination of polynomials on this cell and its immediate neighboring cells. If the cell V_j is not a troubled cell, we just let $u_j^{n,new} = u_j^n$. After the WENO limiter procedure, we just replace u^n with $u^{n,new}$ to march to the next time level by solving Equation (12).

3.2 The WENO limiting procedure for the Euler system

Let us consider the two-dimensional Euler system which is given by

$$\mathbf{u}_t + \mathbf{f}(\mathbf{u})_x + \mathbf{g}(\mathbf{u})_y = \mathbf{0},$$
$$\mathbf{u} = \begin{pmatrix} \rho \\ m \\ n \\ E \end{pmatrix}, \mathbf{f}(\mathbf{u}) = \begin{pmatrix} m \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{pmatrix}, \mathbf{g}(\mathbf{u}) = \begin{pmatrix} n \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{pmatrix}. \tag{21}$$

Here, ρ is the density, $(u, v)^T$ is the velocity vector, $m = \rho u$ and $n = \rho v$ are the momenta. E is the total energy, and p is the pressure, with $p(\mathbf{u}) = (\gamma - 1)\big(E - \frac{1}{2}\rho(u^2 + v^2)\big)$. For convenience, we also denote the solution polynomials on cell V_j and its immediate neighbors as

$$\mathbf{p}_0(\mathbf{x}) = \mathbf{u}_j^n(\mathbf{x}), \qquad \mathbf{p}_l(\mathbf{x}) = \mathbf{u}_{j(l)}^n(\mathbf{x}), \quad l = 1, \cdots, N_j, \tag{22}$$

respectively. Each of them is a 4-component vector and each component is a k-th degree polynomial.

As in the scalar case, we first identify the troubled cells using the KXRCF technique. The boundary ∂V_j is also divided into two parts: ∂V_j^- and ∂V_j^+, where the flow is into ($(u, v)^T \cdot \mathbf{n} < 0$) and out of ($(u, v)^T \cdot \mathbf{n} > 0$) V_j respectively. In the system case, we take both the density ρ and the total energy E as the indicator variables. The target cell V_j is identified as a troubled cell when

$$\frac{\left| \int_{\partial V_j^-} \big(\rho_0(\mathbf{x}) - \rho_l(\mathbf{x})\big) ds \right|}{h^{\frac{k+1}{2}} |\partial V_j^-| \cdot \|\rho_0\|} > C_k, \tag{23}$$

or

$$\frac{|\int_{\partial V_j^-}\big(E_0(\mathbf{x})-E_l(\mathbf{x})\big)ds|}{h^{\frac{k+1}{2}}|\partial V_j^-|\cdot\|E_0\|}>C_k, \tag{24}$$

where $\rho_l(\mathbf{x})$ and $E_l(\mathbf{x})$ denote the density polynomials and the total energy polynomials on the neighboring cells sharing the edge(s) in ∂V_i^-.

Assuming V_j is a troubled cell, we now compute a new polynomial $\mathbf{u}_j^{n,new}$ on V_j to replace the original one. To maintain the original cell average of $\mathbf{p}_0(\mathbf{x})$ in cell V_j, we compute as before the modified polynomial vectors $\tilde{\mathbf{p}}_l(\mathbf{x})$, $l=0,\cdots,N_j$, corresponding to the cell V_j and its immediate neighboring cells. In order to achieve better non-oscillatory qualities, the WENO limiter is used with a local characteristic field decomposition. In the triangular mesh case in [27], the characteristic-wise WENO limiting procedure is performed along the normal directions of each edge of V_j. For the polygonal mesh in this paper, we make a small modification. We denote $\mathbf{n}_l=(n_{lx},n_{ly})^T, l=1,\cdots,N_j$ as the unit vector pointing from the center of V_j to the center of the neighboring cell $V_{j(l)}$. Here we simply compute the center of a cell as the arithmetic mean of the locations of all generator points in this cell. We perform the characteristic-wise WENO limiting procedure in each $\mathbf{n}_l$ direction to reconstruct a new polynomial vector $(\mathbf{p}_0)_l^{new}$ and then combine them to get $\mathbf{u}_j^{n,new}$.

In the $\mathbf{n}_l$ direction, we first denote the associate Jacobian matrix as $\big(\mathbf{f}'(\mathbf{u}),\mathbf{g}'(\mathbf{u})\big)^T\cdot\mathbf{n}_l$, Then the matrix with the left eigenvectors of such Jacobian matrix as rows is

$$L_l=\begin{pmatrix}\frac{B_2+(un_{lx}+vn_{ly})/c}{2} & -\frac{B_1u+n_{lx}/c}{2} & -\frac{B_1v+n_{ly}/c}{2} & \frac{B_1}{2}\\ n_{ly}u-n_{lx}v & -n_{ly} & n_{lx} & 0\\ 1-B_2 & B_1u & B_1v & -B_1\\ \frac{B_2-(un_{lx}+vn_{ly})/c}{2} & -\frac{B_1u-n_{lx}/c}{2} & -\frac{B_1v-n_{ly}/c}{2} & \frac{B_1}{2}\end{pmatrix}, \tag{25}$$

and the matrix with the right eigenvectors as columns is

$$R_l=\begin{pmatrix}1 & 0 & 1 & 1\\ u-cn_{lx} & -n_{ly} & u & u+cn_{lx}\\ v-cn_{ly} & n_{lx} & v & v+cn_{ly}\\ H-c(un_{lx}+vn_{ly}) & -n_{ly}u+n_{lx}v & \frac{u^2+v^2}{2} & H+c(un_{lx}+vn_{ly})\end{pmatrix}, \tag{26}$$

where $c=\sqrt{\gamma p/\rho}$, $B_1=(\gamma-1)/c^2$, $B_2=B_1(u^2+v^2)/2$ and $H=(E+p)/\rho$. Now we use the following steps to obtain $(\mathbf{p}_0)_l^{new}$:

- We first project the modified polynomial vectors $\tilde{\mathbf{p}}_m, m = 0, 1, \cdots, N_j$ into the characteristic fields:

$$\bar{\mathbf{p}}_m = L_l \cdot \tilde{\mathbf{p}}_m, \qquad m = 0, 1, \cdots, N_j. \tag{27}$$

- We perform the scalar WENO limiting procedure that has been specified in the last subsection on each component of these vectors, and obtain a 4-component vector $\bar{\mathbf{p}}_0^{new}$.

- The new polynomial vector $(\mathbf{p}_0)_l^{new}$ in the $\mathbf{n}_l$ direction is then computed by projecting $\bar{\mathbf{p}}_0^{new}$ back into the physical space:

$$(\mathbf{p}_0)_l^{new} = R_l \cdot \bar{\mathbf{p}}_0^{new}. \tag{28}$$

After we have performed the above procedure in all directions, the final new 4-component solution polynomial vector on the troubled cell V_j is defined as

$$\mathbf{u}_j^{n,new} = \frac{\sum_{l=1}^{N_j} (\mathbf{p}_0)_l^{new} |V_{j(l)}|}{\sum_{l=1}^{N_j} |V_{j(l)}|}. \tag{29}$$

For each cell V_j, $j = 1, \cdots, M$ in the computational domain Ω, if it is a troubled cell, we perform the characteristic-wise WENO limiting procedure discussed above and replace the entire solution polynomial $\mathbf{u}_j^n$ with the new 4-component solution polynomial vector $\mathbf{u}_j^{n,new}$ in (29). If the cell V_i is not a troubled cell, we just let $\mathbf{u}_j^{n,new} = \mathbf{u}_j^n$. Then we use the new polynomial to march to the next time level as in Equation (12).

4 Positivity-preserving limiter

Our method with WENO limiters described in the last section does not satisfy the maximum-principle for the scalar conservation laws and the positivity-preserving property for the Euler system directly. Hence, in this section, we extend the maximum-principle-satisfying limiters and the positivity-preserving limiters in [20, 21, 25, 26] to our method. In each time stage, we first use the WENO limiter to reconstruct solution polynomials in those troubled cells in order to control oscillations. Then we further modify solutions in each cell before we march to the next time stage, such that the cell averages on the next time level will satisfy the maximum-principle for the scalar case or the positivity-preserving property for the Euler system.

Based on the given random point cloud, we have shown how to divide the domain Ω into polygonal cells in Section 2.1. From the procedure of mesh generation, we know that the numbers of edges for different polygonal cells can be totally different. Also, the shape of each cell is arbitrary. In the following, we first show a quadrature rule on such a complex polygonal mesh to compute the cell averages of the solutions in Section 4.1. Based on this quadrature rule, limiters for the scalar conservation laws and the Euler system will be shown in Section 4.2 and Section 4.3, respectively.

4.1 Decomposition of the cell average

Consider an arbitrary polygonal cell V_j with m_j edges. In this section, we aim to find a suitable quadrature rule to compute the cell average of the solution on V_j. Assume $q(\mathbf{x}) \in P^k(V_j)$, then the cell average of $q(\mathbf{x})$ on V_j is defined as

$$\bar{q}_j = \frac{1}{|V_j|} \int_{V_j} q(\mathbf{x}) d\mathbf{x}. \tag{30}$$

Note that in the DG method, the edge integral in Equation (9) is approximated by the $(k+1)$-point Gauss quadrature. The quadrature rule we are going to find should be exact for functions in $P^k(V_j)$ and include all the Gauss quadrature points on ∂V_j. Also, all quadrature weights should be positive. In the following, We denote the β-th Gauss quadrature point on the i-th edge of V_j as $\mathbf{x}^j_{i,\beta}$, $i = 1, \cdots, m_j$, $\beta = 1, \cdots, k+1$. Also, we use ω_β to denote the β-th quadrature weight of the $(k+1)$-point Gauss rule on $[-\frac{1}{2}, \frac{1}{2}]$. Besides the Gauss quadrature rule, we also consider the l-point Gauss-Lobatto rule on $[-\frac{1}{2}, \frac{1}{2}]$ with $2l - 3 \geqslant k$. We denote the corresponding α-th quadrature weight as $\hat{\omega}_\alpha$.

As shown in Figure 3, we subdivide V_j into several non-overlapping sub-domains such that those sub-domains which share edges with ∂V_j are all triangles. Suppose there are n_j

$(n_j \leqslant m_j)$ such triangles on the boundary of V_j, which are denoted as $V_j^{(r)}$, $r = 1, \cdots, n_j$. For the case in Figure 3, we have $n_j = 8$. We further denote the rest region of V_j as $V_j^{(n_j+1)} = V_j \backslash (\bigcup_{r=1}^{n_j} V_j^{(r)})$, which is a polygon. Thus, we can divide the computation of $\bar{q}_j$ into several parts:

$$\bar{q}_j = \frac{1}{|V_j|}\int_{V_j} q(\mathbf{x})d\mathbf{x} = \frac{1}{|V_j|}\sum_{r=1}^{n_j}\int_{V_j^{(r)}} q(\mathbf{x})d\mathbf{x} + \frac{1}{|V_j|}\int_{V_j^{(n_j+1)}} q(\mathbf{x})d\mathbf{x}. \tag{31}$$

For the computation on each triangle $V_j^{(r)}$, $1 \leqslant r \leqslant n_j$, we adopt the special numerical quadrature derived in [26]:

$$\frac{1}{|V_j^{(r)}|}\int_{V_j^{(r)}} q(\mathbf{x})d\mathbf{x} = \sum_{\mathbf{x}\in Q_j^{(r)}} \omega_{\mathbf{x}} q(\mathbf{x}). \tag{32}$$

Here $Q_j^{(r)}$ is the set of quadrature points in $V_j^{(r)}$, which includes all $k+1$ Gauss quadrature points on each edge of $V_j^{(r)}$. Also, it is derived in [26] that $\omega_{\mathbf{x}} = \frac{2}{3}\omega_\beta \hat{\omega}_1$ when $\mathbf{x}$ is the β-th Gauss point on any edge of $V_j^{(r)}$. Hence, we can further compute $\bar{q}_j$ as

$$\begin{aligned}
\bar{q}_j &= \sum_{r=1}^{n_j} \frac{|V_j^{(r)}|}{|V_j|}\Big(\sum_{\mathbf{x}\in Q_j^{(r)}\bigcap \partial V_j} \omega_{\mathbf{x}} q(\mathbf{x}) + \sum_{\mathbf{x}\in Q_j^{(r)}\backslash \partial V_j} \omega_{\mathbf{x}} q(\mathbf{x})\Big) + \frac{1}{|V_j|}\int_{V_j^{(n_j+1)}} q(\mathbf{x})d\mathbf{x} \\
&= \sum_{i=1}^{m_j}\sum_{\beta=1}^{k+1} a_{i,\beta}^j q(\mathbf{x}_{i,\beta}^j) + \sum_{r=1}^{n_j}\sum_{\mathbf{x}\in Q_j^{(r)}\backslash \partial V_j} \frac{|V_j^{(r)}|}{|V_j|}\omega_{\mathbf{x}} q(\mathbf{x}) + \frac{1}{|V_j|}\int_{V_j^{(n_j+1)}} q(\mathbf{x})d\mathbf{x},
\end{aligned} \tag{33}$$

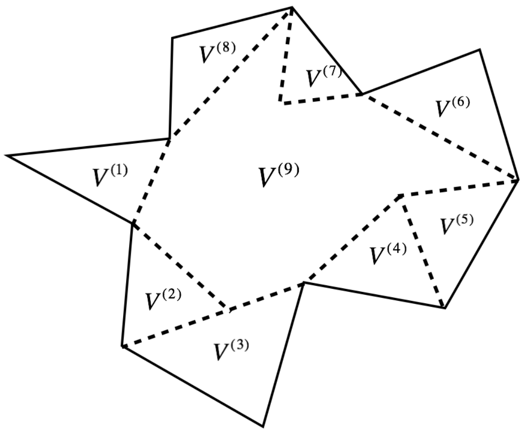


Figure 3: cell subdivision

where $a_{i,\beta}^j = \frac{|V_j^{(r)}|}{|V_j|}\frac{2}{3}\omega_\beta\hat{\omega}_1$ if $\mathbf{x}_{i,\beta}^j \in \partial V_j^{(r)}$.

If we denote

$$\begin{aligned} \xi_j &= \frac{\sum_{r=1}^{n_j}\sum_{\mathbf{x}\in Q_j^{(r)}\backslash\partial V_j}\frac{|V_j^{(r)}|}{|V_j|}\omega_{\mathbf{x}}q(\mathbf{x}) + \frac{1}{|V_j|}\int_{V_j^{(n_j+1)}} q(\mathbf{x})d\mathbf{x}}{1-\sum_{i=1}^{m_j}\sum_{\beta=1}^{k+1}a_{i,\beta}^j} \\ &= \frac{\bar{q}_j - \sum_{i=1}^{m_j}\sum_{\beta=1}^{k+1}a_{i,\beta}^j q(\mathbf{x}_{i,\beta}^j)}{1-\sum_{i=1}^{m_j}\sum_{\beta=1}^{k+1}a_{i,\beta}^j}, \end{aligned} \tag{34}$$

then we obtain a numerical quadrature rule

$$\bar{q}_j = \sum_{i=1}^{m_j}\sum_{\beta=1}^{k+1}a_{i,\beta}^j q(\mathbf{x}_{i,\beta}^j) + (1-\sum_{i=1}^{m_j}\sum_{\beta=1}^{k+1}a_{i,\beta}^j)\xi_j. \tag{35}$$

It is easy to see that

$$\sum_{i=1}^{m_j}\sum_{\beta=1}^{k+1}a_{i,\beta}^j < \sum_{r=1}^{n_j}\sum_{\beta=1}^{k+1}\frac{|V_j^{(r)}|}{|V_j|}2\omega_\beta\hat{\omega}_1 = \sum_{r=1}^{n_j}\frac{|V_j^{(r)}|}{|V_j|}2\hat{\omega}_1 \leqslant 1. \tag{36}$$

Hence, all quadrature weights in the new quadrature are positive. Note that it is easy to prove that there exists some $\mathbf{x}_j^* \in V_j$, such that $q(\mathbf{x}_j^*) = \xi_j$. In fact, we do not need to know the exact location of $\mathbf{x}_j^*$, we only need the know the value of q at this point.

4.2 Positivity-preserving limiter for scalar conservation laws

We consider the scalar conservation law (1) in this section. Based on the polygonal mesh we established in Section 2.1, let us first discuss the first order global Lax-Friedrich scheme on each polygon V_j:

$$u_j^{n+1} = u_j^n - \frac{\Delta t}{|V_j|}\sum_{i=1}^{m_j}\widehat{F}(u_j^n, u_{j[i]}^n, \mathbf{n}_j^i)l_j^i = H(u_j^n, u_{j[1]}^n, \cdots, u_{j[m_j]}^n), \tag{37}$$

where l_j^i is the length of the i-th edge of V_j, denoted by e_j^i, with outward unit normal vector $\mathbf{n}_j^i$. $j[i]$ denotes the index of the neighboring polygon along e_j^i. The global Lax-Friedrichs flux is defined by

$$\widehat{F}(u, v, \mathbf{n}) = \frac{1}{2}(\mathbf{F}(u)\cdot\mathbf{n} + \mathbf{F}(v)\cdot\mathbf{n} - a(v-u)), \tag{38}$$

where $a = \max_{u,v} |\mathbf{F}'(u) \cdot \mathbf{n}|$. This flux satisfies the conservativity

$$\widehat{F}(u, v, \mathbf{n}) = -\widehat{F}(v, u, -\mathbf{n}). \tag{39}$$

Following the standard proof on the triangular mesh, we can easily prove that $H(\cdot, \cdots, \cdot)$ is a monotone increasing function with respect to each argument under the CFL condition

$$a\frac{\Delta t}{|V_j|}\sum_{i=1}^{m_j} l_j^i \leqslant 1. \tag{40}$$

Now let us consider our high order scheme. For simplicity, we only discuss the Euler forward time discretization. Note that after the WENO limiting procedure described in the last section, we obtain a new solution polynomial $u^{n,new}$, which has the same cell average as u^n on V_j. By taking the test function v_h as 1, we can rewrite Equation (12) as

$$\bar{u}_j^{n+1} = \bar{u}_j^n - \frac{\Delta t}{|V_j|}\sum_{i=1}^{m_j}\int_{e_j^i} \widehat{F}(u_i^{int(V_j)}, u_i^{ext(V_j)}, \mathbf{n}_j^i)ds, \tag{41}$$

where $u_i^{int(V_j)}$ and $u_i^{ext(V_j)}$ are the values of $u^{n,new}$ on the edge e_j^i obtained from the interior and the exterior of V_j. By using the $(k+1)$-point Gauss quadrature to compute the edge integral, the scheme becomes

$$\bar{u}_j^{n+1} = \bar{u}_j^n - \frac{\Delta t}{|V_j|}\sum_{i=1}^{m_j}\sum_{\beta=1}^{k+1} \widehat{F}(u_{i,\beta}^{int(V_j)}, u_{i,\beta}^{ext(V_j)}, \mathbf{n}_j^i)\omega_\beta l_j^i, \tag{42}$$

where $u_{i,\beta}^{int(V_j)}$ and $u_{i,\beta}^{ext(V_j)}$ denote the values of $u^{n,new}$ evaluated at $\mathbf{x}_{i,\beta}^j$ from the interior and the exterior of the cell V_j respectively.

By using the quadrature rule we derived in the last section, we can prove the following theorem:

Theorem 1. *For the scheme (42) to satisfy the maximum principle*

$$m \leqslant \bar{u}_j^{n+1} \leqslant M, \tag{43}$$

a sufficient condition is that each $u_j^{n,new}(\mathbf{x}) \in [m, M]$, $\forall \mathbf{x} \in S_k^j$ *and* $\xi_j \in [m, M]$, *where* $S_k^j = \{\mathbf{x}_{i,\beta}^j,\ i = 1, \cdots, m_j,\ \beta = 1, \cdots, k+1\}$ *and* ξ_j *is computed by letting* $q(\mathbf{x}) =$

$u_j^{n,new}(\mathbf{x})$ in (34), under the CFL condition

$$\alpha\frac{\Delta t}{|V_j|}\sum_{i=1}^{m_j} l_j^i \leqslant min_r \frac{|V_j^{(r)}|}{|V_j|}\frac{2}{3}\hat{\omega}_1, \tag{44}$$

where $\hat{\omega}_1$ is the quadrature weight of the l-point Gauss-Lobatto rule on $[-\frac{1}{2},\frac{1}{2}]$ for the first quadrature point. For $k=2,3$, $\hat{\omega}_1=\frac{1}{6}$ and for $k=4,5$, $\hat{\omega}_1=\frac{1}{12}$.

Proof. Rewrite Equation (42) as

$$\begin{aligned}\bar{u}_j^{n+1} &= \bar{u}_j^n - \frac{\Delta t}{|V_j|}\sum_{i=1}^{m_j}\sum_{\beta=1}^{k+1}\widehat{F}(u_{i,\beta}^{int(V_j)}, u_{i,\beta}^{ext(V_j)}, \mathbf{n}_j^i)\omega_\beta l_j^i \\ &= \bar{u}_j^n - \frac{\Delta t}{|V_j|}\sum_{\beta=1}^{k+1}\omega_\beta\big(\sum_{i=1}^{m_j}\widehat{F}(u_{i,\beta}^{int(V_j)}, u_{i,\beta}^{ext(V_j)}, \mathbf{n}_j^i)l_j^i\big).\end{aligned} \tag{45}$$

Then decompose the flux term inside the bracket:

$$\begin{aligned}&\sum_{i=1}^{m_j}\widehat{F}(u_{i,\beta}^{int(V_j)}, u_{i,\beta}^{ext(V_j)}, \mathbf{n}_j^i)l_j^i \\ =\ &\sum_{i=1}^{m_j-1}\big(\widehat{F}(u_{i,\beta}^{int(V_j)}, u_{i,\beta}^{ext(V_j)}, \mathbf{n}_j^i)l_j^i + \widehat{F}(u_{i,\beta}^{int(V_j)}, u_{m_j,\beta}^{int(V_j)}, -\mathbf{n}_j^i)l_j^i\big) \\ +\ &\sum_{i=1}^{m_j-1}\widehat{F}(u_{m_j,\beta}^{int(V_j)}, u_{i,\beta}^{int(V_j)}, \mathbf{n}_j^i)l_j^i + \widehat{F}(u_{m_j,\beta}^{int(V_j)}, u_{m_j,\beta}^{ext(V_j)}, \mathbf{n}_j^{m_j})l_j^{m_j},\end{aligned} \tag{46}$$

where the conservativity of the flux in Equation(39) is used.

Using the quadrature rule (35), we obtain

$$\begin{aligned}\bar{u}_j^{n+1} &= \sum_{i=1}^{m_j}\sum_{\beta=1}^{k+1}a_{i,\beta}^j u_{i,\beta}^{int(V_j)} + (1-\sum_{i=1}^{m_j}\sum_{\beta=1}^{k+1}a_{i,\beta}^j)\xi_j \\ &- \frac{\Delta t}{|V_j|}\sum_{\beta=1}^{k+1}\omega_\beta\big(\sum_{i=1}^{m_j}\widehat{F}(u_{i,\beta}^{int(V_j)}, u_{i,\beta}^{ext(V_j)}, \mathbf{n}_j^i)l_j^i\big) \\ &= (1-\sum_{i=1}^{m_j}\sum_{\beta=1}^{k+1}a_{i,\beta}^j)\xi_j + \sum_{\beta=1}^{k+1}\sum_{i=1}^{m_j}a_{i,\beta}^j H_{i,\beta},\end{aligned} \tag{47}$$

where

$$H_{i,\beta} = u_{i,\beta}^{int(V_j)} - \frac{\omega_\beta\Delta t}{a_{i,\beta}^j|V_j|}\big[\widehat{F}(u_{i,\beta}^{int(V_j)}, u_{i,\beta}^{ext(V_j)}, \mathbf{n}_j^i)l_j^i + \widehat{F}(u_{i,\beta}^{int(V_j)}, u_{m_j,\beta}^{int(V_j)}, -\mathbf{n}_j^i)l_j^i\big],$$

for $1 \leqslant i \leqslant m_j - 1$, and

$$H_{m_j,\beta} = u_{m_j,\beta}^{int(V_j)} - \frac{\omega_\beta \Delta t}{a_{m_j,\beta}^j |V_j|} \Big[\sum_{i=1}^{m_j-1} \widehat{F}(u_{m_j,\beta}^{int(V_j)}, u_{i,\beta}^{int(V_j)}, \mathbf{n}_j^i) l_j^i + \widehat{F}(u_{m_j,\beta}^{int(V_j)}, u_{m_j,\beta}^{ext(V_j)}, \mathbf{n}_j^{m_j}) l_j^{m_j} \Big].$$

We know that $H_{i,\beta}$, $1 \leqslant i \leqslant M_j$ are all formal monotone schemes under the condition

$$a \frac{\omega_\beta \Delta t}{a_{i,\beta}^j |V_j|} \sum_{i=1}^{m_j} l_j^i \leqslant 1.$$

Since $a_{i,\beta}^j = \frac{|V_j^{(r)}|}{|V_j|} \frac{2}{3} \omega_\beta \hat{\omega}_1$ if $\mathbf{x}_{i,\beta}^j \in \partial V_j^{(r)}$, we obtain the CFL condition (44).

Writing the right-hand side of (47) as a function H of $u_{i,\beta}^{int(V_j)}$, $u_{i,\beta}^{ext(V_j)}$ and ξ_j, then H is monotone increasing with respect to each argument. Hence, if all the point values involved here are in the range $[m, M]$, then we have the maximum principle:

$$m = H(m, \cdots, m) \leqslant \bar{u}_j^{n+1} \leqslant H(M, \cdots, M) = M.$$

□

As in [21], we now modify the solution polynomial $u_j^{n,new}$ to get a new function $\tilde{u}_j^{n,new} \in P^k(V)$ which satisfies the conditions in the above theorem. For all V_j, we define the following modified polynomial

$$\tilde{u}_j^{n,new}(\mathbf{x}) = \theta(u_j^{n,new}(\mathbf{x}) - \bar{u}_j) + \bar{u}_j, \qquad \theta = \min\left\{ \left| \frac{m - \bar{u}_j}{\tilde{m}_j - \bar{u}_j} \right|, \left| \frac{M - \bar{u}_j}{\tilde{M}_j - \bar{u}_j} \right|, 1 \right\}, \quad (48)$$

with

$$\tilde{M}_j = \max\{\xi_j, u_j^{n,new}(\mathbf{x}), \mathbf{x} \in S_k^j\}, \qquad \tilde{m}_j = \min\{\xi_j, u_j^{n,new}(\mathbf{x}), \mathbf{x} \in S_k^j\}. \quad (49)$$

Following the proof in [21] for the rectangular mesh, we can prove that this limiter preserves the same high-oder accuracy and the conservation of the original polynomial. After we get $\tilde{u}_j^{n,new}$, we then use it to march to the next time level.

We can also use SSP high order time discretization and it will keep the positivity-preserving property because of the convexity. In this case, we need to perform the procedure above in each stage for a Runge-Kutta method or in each step for a multistep method.

4.3 Positivity-preserving limiter for the Euler equations

Consider the two-dimensional Euler system which is given by Equation (21). We define the set of admissible states as

$$G = \left\{ \mathbf{u} = \begin{pmatrix} \rho \\ m \\ n \\ E \end{pmatrix} \middle| \rho > 0 \ \ and \ \ p(\mathbf{u}) > 0 \right\}. \tag{50}$$

As in the last section, we consider the Euler forward time discretization. We also denote ξ_j as in Equation (34), computed by the solution polynomial $\mathbf{q} = \mathbf{u}_j^{n,new}$. But in the system case, it is a vector. We denote its components as $\xi_j = (\xi_\rho^j, \xi_m^j, \xi_n^j, \xi_E^j)^T$. By replacing the scalar variables with vectors in Equation (42), we obtain the following scheme:

$$\bar{\mathbf{u}}_j^{n+1} = \bar{\mathbf{u}}_j^n - \frac{\Delta t}{|V_j|} \sum_{i=1}^{m_j} \sum_{\beta=1}^{k+1} \widehat{F}(\mathbf{u}_{i,\beta}^{int(V_j)}, \mathbf{u}_{i,\beta}^{ext(V_j)}, \mathbf{n}_j^i) \omega_\beta l_j^i, \tag{51}$$

We are interested in finding solutions within the admissible set G. Note that G is a convex set. Following the same idea as in Theorem 1, we have the following theorem. We omit the proof since it is almost the same as in Theorem 1.

Theorem 2. *For the scheme (51) to satisfy the positivity property*

$$\bar{\mathbf{u}}_j^{n+1} \in G, \tag{52}$$

a sufficient condition is that each $\mathbf{u}_j^{n,new}(\mathbf{x}) \in G, \forall \mathbf{x} \in S_k^j$ *and* $\xi_j \in G$, *where* $S_k^j = \{\mathbf{x}_{i,\beta}^j,\ i = 1, \cdots, m_j,\ \beta = 1, \cdots, k+1\}$, *under the CFL condition*

$$\alpha \frac{\Delta t}{|V_j|} \sum_{i=1}^{m_j} l_j^i \leqslant min_r \frac{|V_j^{(r)}|}{|V_j|} \frac{2}{3} \hat{\omega}_1, \tag{53}$$

where $\hat{\omega}_1$ *is the quadrature weight of the l-point Gauss-Lobatto rule on* $[-\frac{1}{2}, \frac{1}{2}]$ *for the first quadrature point. For* $k = 2, 3$, $\hat{\omega}_1 = \frac{1}{6}$ *and for* $k = 4, 5$, $\hat{\omega}_1 = \frac{1}{12}$.

Given the vector of approximation polynomials $\mathbf{u}_j^{n,new} = (\rho_j, m_j, n_j, E_j)^T$ on cell V_j, with its cell average $\bar{\mathbf{u}}_j^n = (\bar{\rho}_j, \bar{m}_j, \bar{n}_j, \bar{E}_j)^T \in G$, we use the following algorithm to modify $\mathbf{u}_j^{n,new}$ into $\tilde{\mathbf{u}}_j^{n,new} \in G$, and then use it to march to the next time level.

1. In each cell, we enforce the positivity of density first. Set up a small number $\varepsilon > 0$ such that $\bar{\rho}_j \geqslant \varepsilon$ for all V_j. In practice, we can choose $\varepsilon = 10^{-13}$. Replace $\rho_j(\mathbf{x})$ by
$$\hat{\rho}_j(\mathbf{x}) = \theta_1\big(\rho_j(\mathbf{x}) - \bar{\rho}_j\big) + \bar{\rho}_j, \qquad \theta_1 = \min\{\frac{\bar{\rho}_j - \varepsilon}{\bar{\rho}_j - \rho_{min}}, 1\}, \tag{54}$$
with $\rho_{min} = \min\{\rho_j(\mathbf{x}), \mathbf{x} \in S_k^j,\ \xi_\rho^j\}$.

2. The second step is to enforce the positivity of the pressure. Define $\hat{\mathbf{u}}_j(\mathbf{x}) = \big(\hat{\rho}_j(\mathbf{x}), m_j(\mathbf{x}), n_j(\mathbf{x}), E_j(\mathbf{x})\big)^T$. For each $\mathbf{x} \in S_k^j$, if $p(\hat{\mathbf{u}}_j(\mathbf{x})) \geqslant 0$, set $\theta_{\mathbf{x}} = 1$; otherwise, set
$$\theta_{\mathbf{x}} = \frac{p(\bar{\mathbf{u}}_j^n)}{p(\bar{\mathbf{u}}_j^n) - p(\hat{\mathbf{u}}_j(\mathbf{x}))}. \tag{55}$$
We replace $\mathbf{q}$ in (34) with $\hat{\mathbf{u}}_j$ to obtain $\hat{\xi}_j$. If $p(\hat{\xi}_j) \geqslant 0$, set $\theta_\xi = 1$; otherwise, set
$$\theta_\xi = \frac{p(\bar{\mathbf{u}}_j^n)}{p(\bar{\mathbf{u}}_j^n) - p(\hat{\xi}_j)}. \tag{56}$$
Then we get the limited polynomial
$$\tilde{\mathbf{u}}_j^{n,new}(\mathbf{x}) = \theta_2(\hat{\mathbf{u}}_j(\mathbf{x}) - \bar{\mathbf{u}}_j^n) + \bar{\mathbf{u}}_j^n, \qquad \theta_2 = \min\{\theta_{\mathbf{x}}, \mathbf{x} \in S_k^j, \theta_\xi\}. \tag{57}$$

Similar to the proof in [21], we can prove that the new polynomial $\tilde{\mathbf{u}}_j^{n,new}$ satisfies the conditions in Theorem 2. Starting from $\mathbf{u}_j^n$, we first replace it with a new solution polynomial $\mathbf{u}_j^{n,new}$ by using the WENO limiter to control oscillations, then further modify the new solution into $\tilde{\mathbf{u}}_j^{n,new}$ by using the positivity-preserving limiter to ensure that the cell average of the next time level is within the admissible set G.

5 Numerical results

In this section, we provide numerical experiments to demonstrate the performance of the WENO limiter and the positivity preserving limiter.

We use the third order TVD Runge-Kutta method [17] for the time discretization. Second, third and fourth order DG schemes in the space are tested. Since the time discretization is only third order accurate, we take $\Delta t \sim \Delta x^{4/3}$ to obtain the fourth order accurate results for accuracy test examples. For the examples containing discontinuities, the positivity-preserving limiter is used for the third order scheme.

For accuracy test examples, the outmost nodes are set to be uniform as in [9], in order to impose periodic boundary conditions. But all inner points are randomly generated that satisfy a uniform distribution in the computational domain. For examples containing discontinuities, all points are randomly generated. Note that our mesh refinement is unstructured, that is, the generations of random points are independent with the refinement of the number of points N.

In the procedure of grouping Voronoi regions into cells, we need to bound the condition number of the matrix A by a threshold value δ. As in [9], we take δ as 100, 1000 and 3000 for the second, third and fourth order schemes, respectively.

Since the initial data are only given on the point cloud and we care about the values on these points for the later time, we show error tables measuring the numerical error on the points from the point cloud. For the L^∞ norm, we compute the maximum absolute value of the error on these points. For the L^1 norm, we multiply the absolute value of the error on each point with the area of the corresponding Voronoi region, add them together and divide the result by the area of the entire domain. For all figures, we divide the computational domain by a triangulation with the given points as the vertexes and thus plot the values on these points.

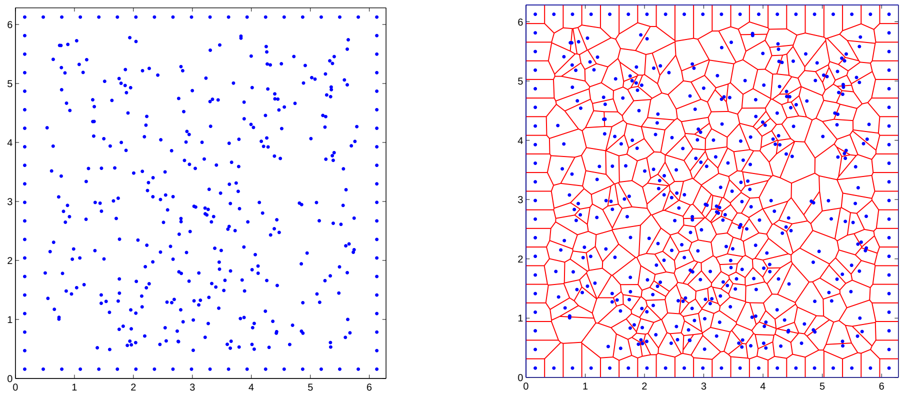
For the purpose of artificially generating a larger percentage of troubled cells in order to test accuracy when the WENO reconstruction procedure is enacted in more cells, we adjust the constant C_k in different examples when using KXRCF technique to identify troubled cells. We list in each table the percentage of troubled cells among all the cells.

Example 1. Let us first consider the two-dimensional linear equation

$$u_t + u_x - 2u_y = 0, \qquad 0 \le x, y \le 2\pi, \tag{58}$$

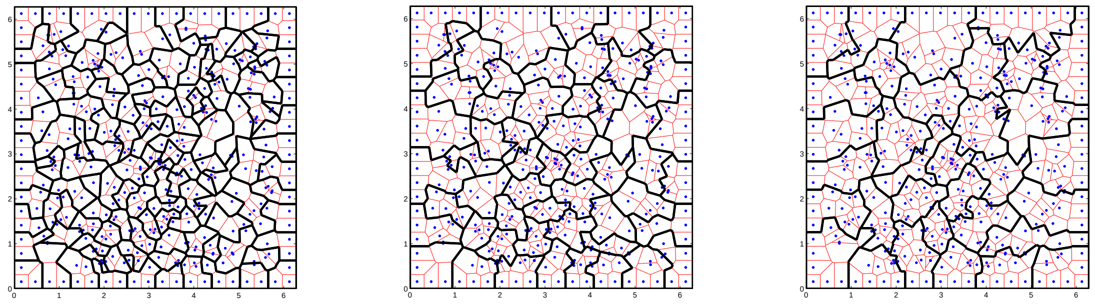
with the initial condition $u(x, y, 0) = \sin(x + y)$ and a 2π-periodic boundary condition.

Figure 4 shows $N = 400$ random points and the corresponding Voronoi diagram. Figure 5 shows mesh subdivisions of these points for different orders of schemes. Here we use red lines to denote Voronoi edges and use black lines to denote cell boundaries of the mesh. Numerical errors and numerical orders of accuracy for the high order method with the WENO limiter comparing with the original high order method without limiter at $t = 2\pi$ are listed in Table 1. We list in the last column of the table the percentage of troubled cells among all the cells. We can see that the WENO limiter keeps the designed order of accuracy.



(a) point cloud (b) Voronoi diagram

Figure 4: **Random points and the corresponding Voronoi diagram in the domain $[0, 2\pi] \times [0, 2\pi]$, $N = 400$.**



(a) P^1 (b) P^2 (c) P^3

Figure 5: **Mesh decomposition of the domain $\Omega = [0, 2\pi] \times [0, 2\pi]$ based on 400 given points.**

Table 1: 2D linear equation with $u(x, y, 0) = \sin(x + y)$ at $t = 2\pi$.

number of	DG without limiter				DG with WENO limiter				
points	L^1 norm	order	L^∞ norm	order	L^1 norm	order	L^∞ norm	order	percentage
	$k = 1$, $C_k = 0.1$								
400	1.32E-01	–	3.55E-01	–	2.14E-01	–	5.01E-01	–	10.81%
1600	1.69E-02	2.97	7.23E-02	2.30	1.87E-02	3.52	7.23E-02	2.79	0.87%
6400	3.19E-03	2.40	2.00E-02	1.85	3.21E-03	2.54	2.00E-02	1.85	0.11%
25600	7.04E-04	2.18	7.84E-03	1.35	7.03E-04	2.19	7.84E-03	1.35	0.00%
102400	1.74E-04	2.02	2.33E-03	1.75	1.74E-04	2.02	2.33E-03	1.75	0.00%
average	–	2.37	–	1.77	–	2.53	–	1.87	–
	$k = 2$, $C_k = 0.01$								
400	2.15E-02	–	8.57E-02	–	4.07E-02	–	1.13E-01	–	37.04%
1600	2.28E-03	3.24	1.84E-02	2.22	3.49E-03	3.55	2.32E-02	2.28	9.40%
6400	2.92E-04	2.96	2.16E-03	3.09	5.36E-04	2.70	4.24E-03	2.45	3.09%
25600	2.92E-05	3.32	3.86E-04	2.49	5.84E-05	3.20	5.60E-04	2.92	0.77%
102400	4.26E-06	2.78	5.90E-05	2.71	5.32E-06	3.46	7.23E-05	2.95	0.13%
average	–	3.09	–	2.66	–	3.17	–	2.66	–
	$k = 3$, $C_k = 0.0005$								
400	7.28E-03	–	2.54E-02	–	1.86E-02	–	7.30E-02	–	81.82%
1600	3.89E-04	4.23	2.05E-03	3.63	6.44E-04	4.85	2.98E-03	4.61	19.70%
6400	2.20E-05	4.14	1.75E-04	3.55	2.30E-05	4.81	1.76E-04	4.08	0.37%
25600	1.66E-06	3.73	1.90E-05	3.21	1.65E-06	3.80	1.90E-05	3.21	0.00%
102400	9.56E-08	4.11	1.33E-06	3.84	9.56E-08	4.11	1.33E-06	3.84	0.00%
average	–	4.03	–	3.52	–	4.37	–	3.88	–

Example 2. Consider the two-dimensional nonlinear scalar Burgers equation

$$u_t + (\frac{u^2}{2})_x + (\frac{u^2}{2})_y = 0, \qquad 0 \le x, y \le 2\pi, \tag{59}$$

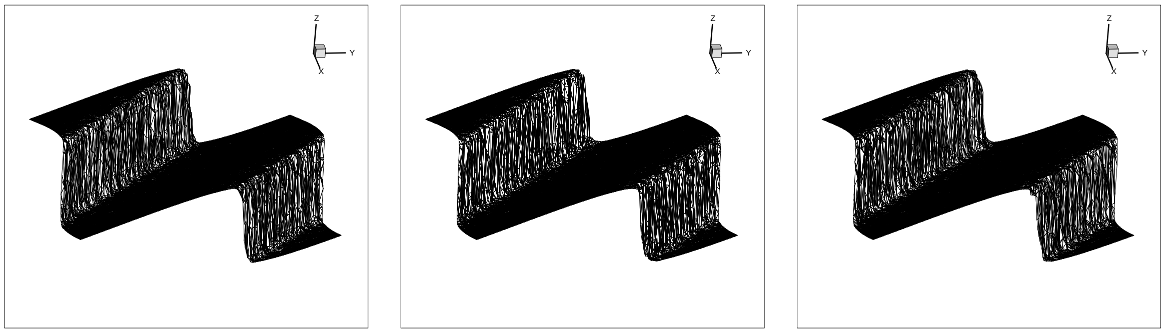
with the initial condition $u(x, y, 0) = 0.5 + \sin(x + y)$ and periodic boundary conditions in both directions. For this test case, we use the same mesh as in Example 1. Table 2 gives the L^1 and L^∞ errors and numerical orders of accuracy at $t = 0.25$ when the solution is smooth. Similar to the previous example, we can see that the WENO limiter keeps the designed order of accuracy, even when a large percentage of good cells are artificially identified as troubled cells.

At $t = 0.5$, a shock begins to appear in the solution. We plot the solution surfaces at $t = 0.75$ with $N = 25600$ points in Figure 6. We can see that the schemes give non-oscillatory shock transitions for this problem.

Table 2: 2D Burgers equation with $u(x, y, 0) = 0.5 + \sin(x + y)$ at $t = 0.25$.

number of points	DG without limiter				DG with WENO limiter				
	L^1 norm	order	L^∞ norm	order	L^1 norm	order	L^∞ norm	order	percentage
	$k = 1$, $C_k = 0.03$								
400	3.80E-02	–	2.30E-01	–	1.09E-01	–	6.68E-01	–	81.08%
1600	9.04E-03	2.07	6.39E-02	1.85	3.97E-02	1.46	5.24E-01	0.35	54.47%
6400	2.74E-03	1.72	3.90E-02	0.71	6.20E-03	2.68	2.10E-01	1.32	19.05%
25600	7.50E-04	1.87	1.61E-02	1.27	7.99E-04	2.96	1.42E-02	3.89	4.18%
102400	1.90E-04	1.98	5.36E-03	1.59	1.89E-04	2.08	5.36E-03	1.40	1.37%
average	–	1.89	–	1.28	–	2.40	–	1.91	–
	$k = 2$, $C_k = 0.01$								
400	1.88E-02	–	1.52E-01	–	3.97E-02	–	6.02E-01	–	37.04%
1600	2.69E-03	2.80	4.70E-02	1.69	2.72E-03	3.87	4.64E-02	3.70	10.26%
6400	4.18E-04	2.69	1.24E-02	1.92	4.09E-04	2.73	1.39E-02	1.74	1.99%
25600	5.62E-05	2.89	2.44E-03	2.34	5.58E-05	2.87	2.17E-03	2.68	0.55%
102400	7.50E-06	2.91	6.51E-04	1.91	7.52E-06	2.89	1.19E-03	0.87	0.05%
average	–	2.82	–	2.00	–	3.03	–	2.24	–
	$k = 3$, $C_k = 0.001$								
400	1.06E-02	–	1.03E-01	–	3.10E-02	–	5.15E-01	–	93.94%
1600	1.16E-03	3.19	3.92E-02	1.40	1.20E-03	4.69	2.16E-02	4.58	23.48%
6400	1.05E-04	3.47	3.55E-03	3.47	1.15E-04	3.39	4.36E-03	2.31	6.34%
25600	7.41E-06	3.82	3.80E-04	3.23	9.13E-06	3.66	4.64E-04	3.23	1.36%
102400	5.62E-07	3.72	6.18E-05	2.62	6.22E-07	3.88	9.08E-05	2.35	0.16%
average	–	3.57	–	2.81	–	3.82	–	3.05	–

Example 3. Let us consider the two-dimensional Euler system which is given by Equation (21). The initial condition is set to be $\rho(x, y, 0) = 1+0.2\sin(x+y)$, $u(x, y, 0) = 0.7$, $v(x, y, 0) = 0.3$ and $p(x, y, 0) = 1$, $0 \le x, y \le 2\pi$. The boundary conditions are periodic. $\gamma = 1.4$ is used in the computation. The exact solution is $\rho(x, y, t) = 1+0.2\sin(x+y-t)$,



(a) P^1, $Ck = 0.03$ (b) P^2, $Ck = 0.01$ (c) P^3, $Ck = 0.001$

Figure 6: **2D Burgers on random point cloud.** $t = 0.75$. $N = 25600$.

$u(x,y,t)=0.7$, $v(x,y,t)=0.3$ and $p(x,y,t)=1$. Table 3 gives the L^1 and L^∞ errors and numerical orders of accuracy of the density at $t=2\pi$. Similar to the previous example, we can see that the WENO limiter keeps the designed order of accuracy.

Table 3: 2D Euler equation with $\rho(x,y,0)=1+0.2\sin(x+y)$, $u(x,y,0)=0.7$, $v(x,y,0)=0.3$ and $p(x,y,0)=1$ at $t=2\pi$.

number of	DG without limiter				DG with WENO limiter				
points	L^1 norm	order	L^∞ norm	order	L^1 norm	order	L^∞ norm	order	percentage
	$k=1$, $C_k=0.01$								
400	2.78E-02	–	8.91E-02	–	5.74E-02	–	1.24E-01	–	14.41%
1600	3.80E-03	2.87	1.91E-02	2.22	4.87E-03	3.56	2.29E-02	2.44	1.31%
6400	6.45E-04	2.56	4.91E-03	1.96	6.49E-04	2.91	4.90E-03	2.22	0.00%
25600	1.42E-04	2.19	1.61E-03	1.60	1.42E-04	2.20	1.61E-03	1.60	0.00%
102400	3.34E-05	2.09	5.26E-04	1.62	3.34E-05	2.09	5.26E-04	1.62	0.00%
average	–	2.41	–	1.84	–	2.66	–	1.96	–
	$k=2$, $C_k=0.0006$								
400	4.42E-03	–	2.74E-02	–	9.23E-03	–	5.95E-02	–	61.11%
1600	4.30E-04	3.36	4.11E-03	2.74	8.57E-04	3.43	4.24E-03	3.81	23.50%
6400	4.35E-05	3.30	5.81E-04	2.82	6.28E-05	3.77	6.43E-04	2.72	1.44%
25600	5.24E-06	3.05	7.67E-05	2.92	5.24E-06	3.58	7.67E-05	3.07	0.00%
102400	6.63E-07	2.98	1.23E-05	2.64	6.63E-07	2.98	1.23E-05	2.64	0.00%
average	–	3.18	–	2.80	–	3.49	–	3.03	–
	$k=3$, $C_k=0.0001$								
400	1.30E-03	–	6.86E-03	–	2.37E-02	–	1.15E-01	–	84.85%
1600	6.45E-05	4.33	5.12E-04	3.74	2.78E-04	6.41	3.61E-03	5.00	12.88%
6400	3.45E-06	4.23	4.41E-05	3.54	3.55E-06	6.29	4.41E-05	6.36	0.00%
25600	2.14E-07	4.01	2.61E-06	4.08	2.14E-07	4.05	2.61E-06	4.08	0.00%
102400	1.39E-08	3.95	2.97E-07	3.14	1.39E-08	3.95	2.97E-07	3.14	0.00%
average	–	4.13	–	3.66	–	5.17	–	4.76	–

Example 4. Consider the two-dimensional vortex evolution problem, which is an idealized problem for the two-dimensional Euler equations. The setup of this problem is: The mean flow is $\rho=1$, $p=1$ and $(u,v)=(1,1)$ (diagonal flow). We add, to this mean flow, an isentropic vortex (perturbation in (u,v) and the temperature $T=\frac{p}{\rho}$, no perturbation in the entropy $S=\frac{p}{\rho^\gamma}$):

$$(\delta u,\delta v)=\frac{\epsilon}{2\pi}e^{0.5(1-t^2)}(-\bar{y},\bar{x}),\qquad \delta T=-\frac{(\gamma-1)\epsilon^2}{8\gamma\pi^2}e^{1-r^2},\qquad \delta S=0, \tag{60}$$

where $(\bar{x},\bar{y})=(x-7,y-7)$, $r^2=\bar{x}^2+\bar{y}^2$, and the vortex strength $\epsilon=5$. The computational domain is taken as $[0, 14] \times [0, 14]$, extended periodically in both directions. It is clear that the exact solution of the Euler equation with the above initial and boundary conditions is just the passive convection of the vortex with the mean velocity. Table 4 gives the L^1 and L^∞ errors and numerical orders of accuracy of the density at $t = 0.2$. We can see that the WENO limiter maintains the designed order of accuracy of the original DG method.

Table 4: 2D Euler system. The smooth vortex evolution problem at $t = 0.2$.

number of	DG without limiter				DG with WENO limiter				
points	L^1 norm	order	L^∞ norm	order	L^1 norm	order	L^∞ norm	order	percentage
	$k = 1$, $C_k = 0.01$								
400	4.18E-03	–	1.13E-01	–	6.41E-03	–	2.41E-01	–	15.04%
1600	1.11E-03	1.92	4.38E-02	1.36	3.32E-03	0.95	1.87E-01	0.37	7.93%
6400	4.20E-04	1.40	2.45E-02	0.84	8.81E-04	1.91	4.55E-02	2.04	3.72%
25600	1.09E-04	1.94	6.24E-03	1.98	1.35E-04	2.71	1.07E-02	2.09	0.92%
102400	3.07E-05	1.83	2.53E-03	1.30	3.12E-05	2.11	3.21E-03	1.73	0.21%
average	–	1.75	–	1.38	–	2.00	–	1.66	–
	$k = 2$, $C_k = 0.001$								
400	3.76E-03	–	7.30E-02	–	7.91E-03	–	3.26E-01	–	30.91%
1600	5.94E-04	2.66	1.88E-02	1.95	2.58E-03	1.62	1.24E-01	1.39	20.89%
6400	9.29E-05	2.68	3.37E-03	2.48	1.43E-04	4.17	7.00E-03	4.15	7.50%
25600	1.56E-05	2.57	1.05E-03	1.68	1.76E-05	3.03	1.11E-03	2.65	2.23%
102400	2.37E-06	2.72	2.46E-04	2.09	2.72E-06	2.69	2.87E-04	1.96	0.45%
average	–	2.65	–	2.06	–	3.02	–	2.71	–
	$k = 3$, $C_k = 0.001$								
400	4.30E-03	–	8.02E-02	–	1.07E-02	–	3.49E-01	–	37.14%
1600	5.06E-04	3.09	1.44E-02	2.48	2.51E-03	2.09	1.14E-01	1.62	14.18%
6400	4.38E-05	3.53	2.15E-03	2.74	5.84E-05	5.42	6.96E-03	4.03	1.87%
25600	2.94E-06	3.90	1.99E-04	3.43	3.02E-06	4.27	2.55E-04	4.77	0.05%
102400	2.38E-07	3.63	2.11E-05	3.24	2.36E-07	3.68	1.82E-05	3.81	0.00%
average	–	3.57	–	3.00	–	4.06	–	3.73	–

Example 5. We consider the double Mach reflection problem. The computational domain is set to be $[0, 4] \times [0, 1]$. The reflection wall lies at the bottom of the computational domain starting from $x = \frac{1}{6}$. Initially a right-moving Mach 10 shock is positioned at $x = \frac{1}{6}, y = 0$ and makes a 60° angle with the x-axis. For the bottom boundary, the exact post-shock condition is imposed for the part from $x = 0$ to $x = \frac{1}{6}$, and a reflective boundary condition is used for the rest. At the top boundary of the computational do-

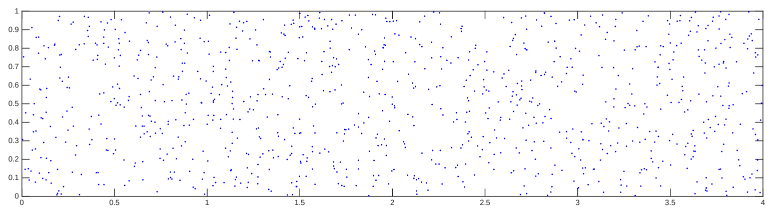
main, the flow values are set to describe the exact motion of the Mach 10 shock. The initial pre-shock condition is

$$(\rho, p, u, v) = (1.4, 1, 0, 0), \tag{61}$$

and the post-shock condition is

$$(\rho, p, u, v) = (8, 116.5, 8.25\cos(30^\circ), -8.25\sin(30^\circ)). \tag{62}$$

We take $C_k = 0.01$ in the troubled cell indicator. For the second order scheme (P^1), we use $N = 400000$ totally random points. For the third order scheme (P^2), we use $N = 1664000$ random points. We show a sample mesh with $N = 1000$ points to illustrate our mesh subdivisions. Figure 7 shows $N = 1000$ random points and the corresponding Voronoi diagram. Figure 8 shows mesh subdivisions of these points for different orders of schemes. The density contours at $t = 0.2$ are plotted in Figure 9. The "zoomed-in" pictures around the double Mach stem to show more details are given in Figure 10. In all the plots, we use 29 contours equally distributed from $\rho = 1.3$ to 23. We can see that the resolution around the double Mach region improves with an increasing k.



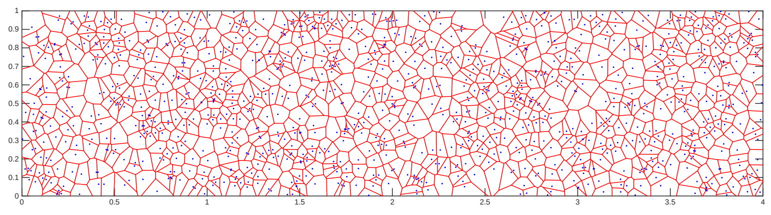


Figure 7: **Double Mach reflection problem.** $N = 1000$ **random points and the corresponding Voronoi diagram.**

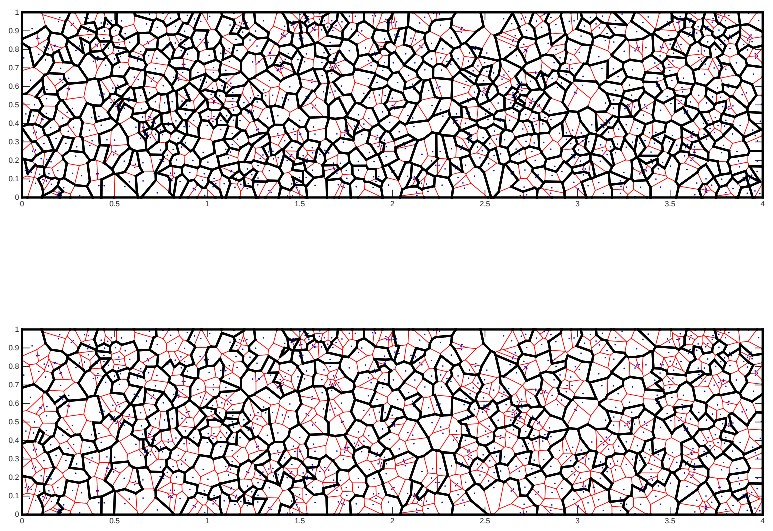


Figure 8: **Double Mach reflection problem. Mesh subdivision. Top: second order ($k = 1$) scheme. Bottom: third order ($k = 2$) scheme.**

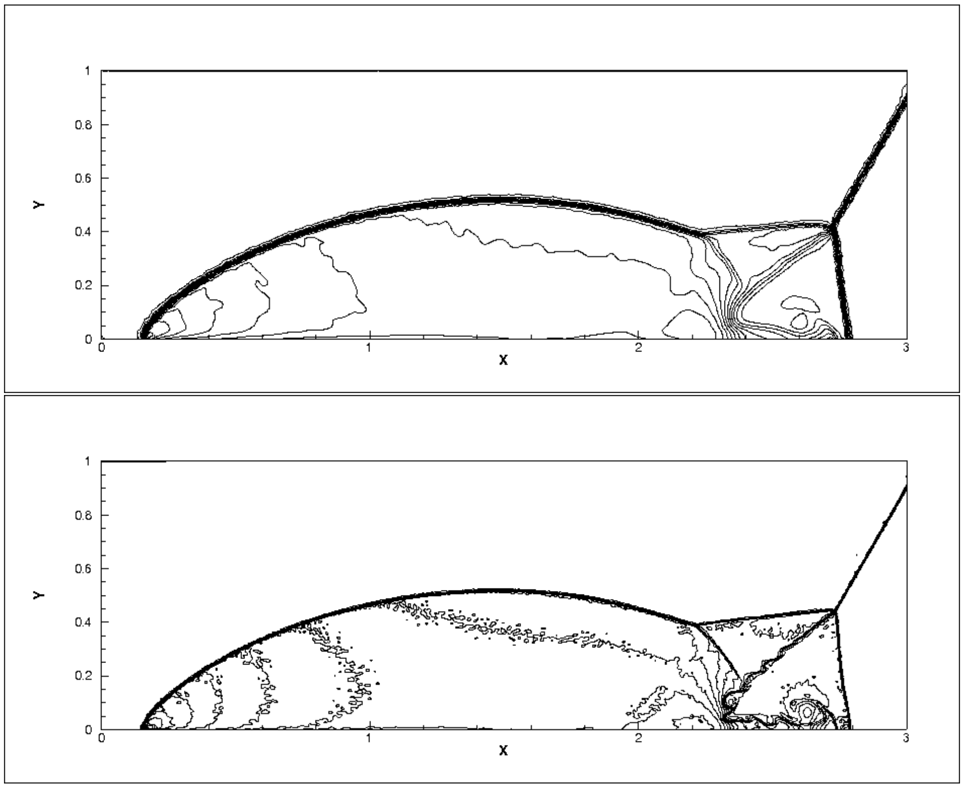
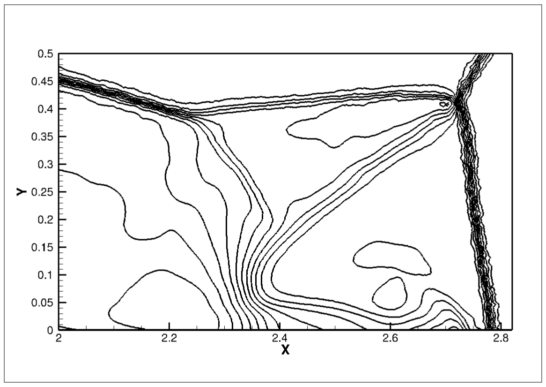


Figure 9: **Double Mach reflection problem. Top: second order ($k = 1$) scheme with** 400000 **random points. Bottom: third order ($k = 2$) scheme with** 1664000 **random points.**



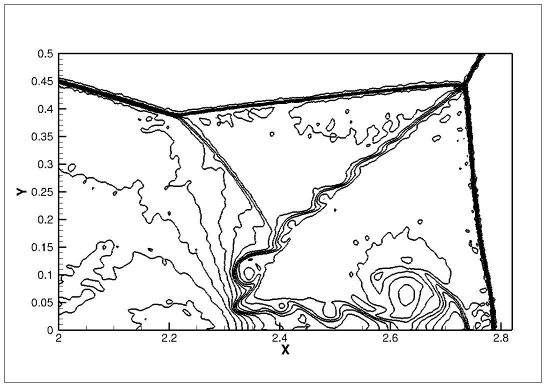


Figure 10: **Zoomed-in figure. Double Mach reflection problem. Left: second order ($k = 1$) scheme with 400000 random points. Right: third order ($k = 2$) scheme with 1664000 random points.**

Example 6. Let us consider the problem of a shock passing a backward facing corner (diffraction). The setup is the following: the computational domain is the union of $[0, 1] \times [6, 11]$ and $[1, 13] \times [0, 11]$; the initial condition is a pure right-moving Mach 5.09 shock, initially located at $x = 0.5$ and $6 \leqslant y \leqslant 11$, moving into undisturbed air ahead of the shock with a density of 1.4 and pressure of 1. The boundary conditions are inflow at $x = 0$, $6 \leqslant y \leqslant 11$, outflow at $x = 13$, $0 \leqslant y \leqslant 11$ and $1 \leqslant x \leqslant 13$, $y = 0$, and reflective at the walls $0 \leqslant x \leqslant 1$, $y = 6$ and $x = 1$, $0 \leqslant y \leqslant 6$. At the top boundary, we use the exact solution of a free-moving Mach 5.09 shock. We choose $C_k = 0.01$ in this example. $N = 219200$ random points are used in the second order scheme (P^1). $N = 493200$ random points are used in the third order scheme (P^2). The density at t = 2.3 is presented in Figure 11. We use 20 equally spaced contour lines from 0.066227 to 7.0668.

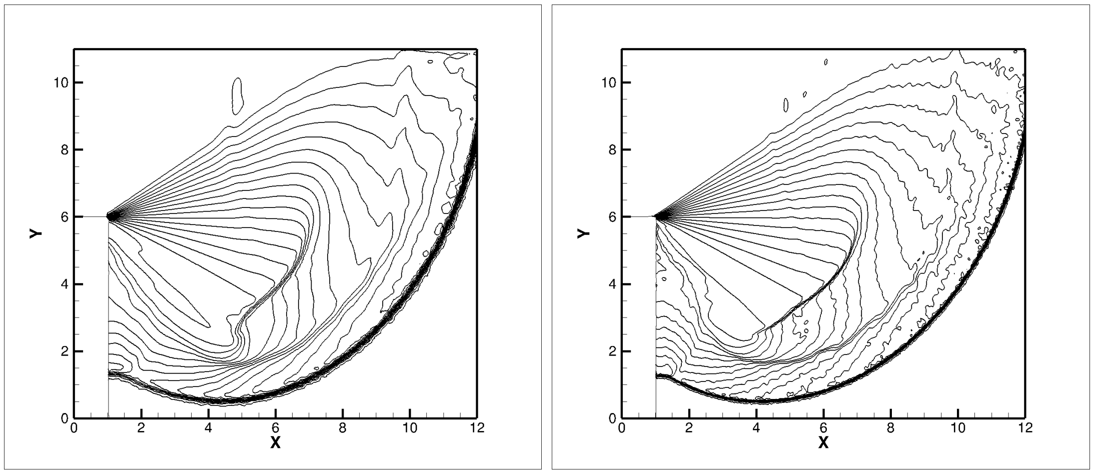


Figure 11: **Shock diffraction problem. Left: second order ($k = 1$) scheme with 219200 random points. Right: third order ($k = 2$) scheme with 493200 random points.**

6 Concluding remarks

In this paper, we adapt a simple WENO limiter [27] original designed for DG method on the triangular mesh to our previous work in [9] for solving hyperbolic conservation laws on a set of two dimensional arbitrarily distributed points. As in [9], a polygonal mesh can be constructed based on the given random points, in which each cell has arbitrary number of edges and can be in any shape. The extended WENO limiter is designed on such a complex polygonal mesh. The goal is to simultaneously maintain uniform high order accuracy of the original method in smooth regions and control spurious numerical oscillations near discontinuities. Also, we extend the maximum-principle-satisfying limiter for the scalar case and the positivity-preserving limiter for the Euler system case in [20, 21, 26, 25] to our method on polygonal mesh. On each time level, we first use the WENO limiter to reconstruct the solutions on those troubled cells, and then use the maximum-principle-satisfying limiter or the positivity-preserving limiter to further modify the solution polynomials in each cell if necessary. Finally, we replace the solution on each cell with the new solution polynomial, and perform the normal DG procedure to march to the next time level. Since the WENO limiter uses information only from immediate neighbors, it is very simple to implement and can maintain the compactness of the original method. Numerical results are also provided to demonstrate the performance of these limiters.

References

[1] I. Babuška and J.M. Melenk, *The partition of unity finite element method*, Technical Note BN-1185, Univ. of Maryland, 1995.

[2] D.S. Balsara and C.-W. Shu, *Monotonicity preserving weighted essentially non-oscillatory schemes with increasingly high order of accuracy*, Journal of Computational Physics, 160 (2000), 405–452.

[3] B. Cockburn, S. Hou and C.-W. Shu, *The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case*, Mathematics of Computation, 54 (1990), 545–581.

[4] B. Cockburn, S.-Y. Lin and C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: one dimensional systems*, Journal of Computational Physics, 84 (1989), 90–113.

[5] B. Cockburn and C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: general framework*, Mathematics of Computation, 52 (1989), 411–435.

[6] B. Cockburn and C.-W. Shu, *The Runge-Kutta local projection P1-discontinuous-Galerkin finite element method for scalar conservation laws*, Mathematical Modelling and Numerical Analysis (M2AN), 25 (1991), 337–361.

[7] B. Cockburn and C.-W. Shu, *The Runge-Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems*, Journal of Computational Physics, 141 (1998), 199–224.

[8] M. Dahleh, M.A. Dahleh and G. Verghese, *Lectures on dynamic systems and control*, A+ A, 4 (2004), 1–100.

[9] J. Du and C.-W. Shu, *A high order stable conservative method for solving hyperbolic conservation laws on arbitrarily distributed point clouds*, SIAM Journal on Scientific Computing, 38 (2016), A3094–A3128.

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

[11] G. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, Journal of Computational Physics, 126 (1995), 202–228.

[12] L. Krivodonova, J. Xin, J.-F. Remacle, N. Chevaugeon and J.E. Flaherty, *Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws*, Applied Numerical Mathematics, 48 (2004), 323–338.

[13] H. Luo, J.D. Baum and R. Lohner, *A Hermite WENO-based limiter for discontinuous Galerkin method on unstructured grids*, Journal of Computational Physics, 225 (2007), 686–713.

[14] S. Li and W.K. Liu, *Meshfree and particle methods and their applications*. Applied Mechanics Reviews, 55 (2002), 1–34.

[15] A. Okabe, B. Boots, and K. Sugihara, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, Wiley, Chichester, UK, 1992.

[16] J. Qiu and C.-W. Shu, *A comparison of troubled-cell indicators for Runge-Kutta discontinuous Galerkin methods using weighted essentially nonoscillatory limiters*, SIAM Journal on Scientific Computing, 27 (2005), 995–1013.

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

[18] Siraj-ul-Islam, R. Vertnik and B. Šarler, *Local radial basis function collocation method along with explicit time stepping for hyperbolic partial differential equations*, Applied Numerical Mathematics, 67 (2013), 136–151.

[19] J. Shi, C. Hu and C.-W. Shu, *A technique of treating negative weights in WENO schemes*, Journal of Computational Physics, 175 (2002), 108–127.

[20] C. Wang, X. Zhang, C.-W. Shu and J. Ning, *Robust high order discontinuous Galerkin schemes for two-dimensional gaseous detonations*, Journal of Computational Physics, 231 (2012), 653–665.

[21] Y. Xing and C.-W. Shu, *High-order finite volume WENO schemes for the shallow water equations with dry states*, Advances in Water Resources, 34 (2011), 1026–1038.

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

[23] X. Zhang and C.-W. Shu, *On maximum-principle-satisfying high order schemes for scalar conservation laws*, Journal of Computational Physics, 229 (2010), 3091–3120.

[24] X. Zhang and C.-W. Shu, *On positivity preserving high order discontinuous Galerkin schemes for compressible Euler equations on rectangular meshes*, Journal of Computational Physics, 229 (2010), 8918–8934.

[25] X. Zhang and C.-W. Shu, *Maximum-principle-satisfying and positivity-preserving high-order schemes for conservation laws: survey and new developments*, Proceedings of the Royal Society A, 467 (2011), 2752–2776.

[26] X. Zhang, Y. Xia and C.-W. Shu, *Maximum-principle-satisfying and positivity-preserving high order discontinuous Galerkin schemes for conservation laws on triangular meshes*, Journal of Scientific Computing, 50 (2012), 29–62.

[27] J. Zhu, X. Zhong, C.-W. Shu and J.-X. Qiu, *Runge-Kutta discontinuous Galerkin method using a new type of WENO limiters on unstructured meshes*, Journal of Computational Physics, 248 (2013), 200–220.