SIAM J. SCI. COMPUT.
Vol. 24, No. 3, pp. 1005–1030


# HIGH-ORDER WENO SCHEMES FOR HAMILTON–JACOBI EQUATIONS ON TRIANGULAR MESHES*

YONG-TAO ZHANG† AND CHI-WANG SHU†

**Abstract.** In this paper we construct high-order weighted essentially nonoscillatory (WENO) schemes for solving the nonlinear Hamilton–Jacobi equations on two-dimensional unstructured meshes. The main ideas are nodal based approximations, the usage of monotone Hamiltonians as building blocks on unstructured meshes, nonlinear weights using smooth indicators of second and higher derivatives, and a strategy to choose diversified smaller stencils to make up the bigger stencil in the WENO procedure. Both third-order and fourth-order WENO schemes using combinations of second-order approximations with nonlinear weights are constructed. Extensive numerical experiments are performed to demonstrate the stability and accuracy of the methods. High-order accuracy in smooth regions, good resolution of derivative singularities, and convergence to viscosity solutions are observed.

**Key words.** weighted essentially nonoscillatory schemes, Hamilton–Jacobi equations, high-order accuracy, unstructured mesh

**AMS subject classification.** 65M99

**PII.** S1064827501396798

## 1. Introduction.

In this paper, we consider the numerical solution of Hamilton–Jacobi (H-J) equations

$$\phi_t + H(\phi_{x_1}, \ldots, \phi_{x_d}) = 0, \qquad \phi(x, 0) = \phi_0(x). \tag{1.1}$$

Such equations appear often in applications, such as in optimal control, differential games, image processing and computer vision, and geometric optics. With the popularity of level set methods [14] the necessity to have good algorithms to solve H-J equations becomes even more obvious.

There have been many papers in the literature developing numerical methods to solve H-J equations. In a certain sense H-J equations are easier to solve than their conservation laws counterparts because the solutions here are smoother: typical solutions are continuous with possibly discontinuous derivatives.

For a structured mesh, finite difference methods similar to those developed for conservation laws could be easily designed. Thus we have the earlier second-order ENO schemes of Osher and Sethian [14], higher-order ENO schemes of Osher and Shu [15], higher-order weighted ENO (WENO) schemes of Jiang and Peng [9], and central high resolution schemes of Lin and Tadmor [12], among many others.

However, for unstructured meshes there are conceptional difficulties in designing schemes such as finite volume schemes and finite element schemes, which are very useful for conservation laws. The problem arises because the H-J equations cannot be written in a "conservation form," suitable for integration by parts, which is the backbone of finite volume and finite element methods. Thus algorithms, especially

*Received by the editors October 22, 2001; accepted for publication (in revised form) June 24, 2002; published electronically January 23, 2003. This research was supported by ARO grant DAAD19-00-1-0405, NSF grants DMS-9804985 and ECS-9906606, NASA Langley grant NCC1-01035 and contract NAS1-97046, while the second author was in residence at ICASE, NASA Langley Research Center, Hampton, VA 23681-2199, and AFOSR grant F49620-99-1-0077.

http://www.siam.org/journals/sisc/24-3/39679.html

†Division of Applied Mathematics, Brown University, Providence, RI 02912 (zyt@cfm.brown.edu, shu@cfm.brown.edu).

high-order algorithms for H-J equations on unstructured meshes, are relatively few. We have the very good first- and second-order finite volume–type schemes of Abgrall [2] and ENO- or limiter-type high-order version of Augoula and Abgrall [4]. The monotone Hamiltonians developed in [2] are used in this paper as building blocks. We also have the continuous finite element methods of Barth and Sethian [6] and the discontinuous Galerkin methods of Hu and Shu [7]. However, the formulation of the finite element methods in [7] actually uses the differentiated version of H-J, which is a system of conservation laws. It would be desirable in many situations to use directly H-J on an unstructured mesh and still obtain high-order, nonoscillatory numerical results. The algorithms developed in this paper fulfills this purpose.

The WENO methodology adopted in this paper can be traced back to the earlier work for conservation laws started in [13] for a third-order finite volume version in one space dimension and in [10] for third- and fifth-order finite difference WENO schemes in multispace dimensions with a general framework for the design of the smoothness indicators and nonlinear weights. The techniques most relevant to the current paper are the third- and fourth-order finite volume WENO schemes for two-dimensional (2D) conservation laws in general triangulations [8] and the finite difference WENO schemes for H-J equations [9]. However, significant new components of the algorithms have been developed in this paper to deal with the additional difficulty caused by the "nonconservation" form of the H-J equations and unstructured meshes. These include nodal based approximations, the usage of monotone Hamiltonians as building blocks on unstructured meshes, nonlinear weights using smooth indicators of second and higher derivatives, and a strategy to choose diversified smaller stencils to make up the bigger stencil in the WENO procedure. Both third-order and fourth-order WENO schemes using combinations of second-order approximations with nonlinear weights are constructed. Extensive numerical experiments are performed to demonstrate the stability and accuracy of the methods. High-order accuracy in smooth regions, good resolution of derivative singularities, and convergence to viscosity solutions are observed.

The algorithm is developed in section 2. Section 3 contains numerical examples verifying the stability, convergence, and accuracy of the algorithms. Concluding remarks are given in section 4.

**2. The WENO algorithm for 2D unstructured mesh.** In this section we develop third- and fourth-order WENO schemes on unstructured meshes in two dimensions for the H-J equations.

**2.1. The framework.** We take $d = 2$ in (1.1), and use $x, y$ instead of $x_1, x_2$:

$$\phi_t + H(\phi_x, \phi_y) = 0, \qquad \phi(x, y, 0) = \phi_0(x, y). \tag{2.1}$$

The equation (2.1) is solved in the domain $\Omega$, which has a triangulation $\mathcal{T}_h$ consisting of triangles. The nodes will be named by their indices $0 \le i \le N$, with a total of $N+1$ nodes. For every node $i$, we define the $k_i+1$ angular sectors $T_0, \ldots, T_{k_i}$ meeting at the point $i$; they are the inner angles at node $i$ of the triangles having $i$ as a vertex. The indexing of the angular sectors is ordered counterclockwise. $\vec{n}_{l+\frac{1}{2}}$ is the unit vector of the half-line $D_{l+\frac{1}{2}} = T_l \bigcap T_{l+1}$, and $\theta_l$ is the inner angle of sector $T_l$, $0 \le l \le k_i$; see Figure 2.1.

We will denote by $\phi_i$ the numerical approximation to the viscosity solution of (2.1) at node $i$. $(\nabla\phi)_0, \ldots, (\nabla\phi)_{k_i}$ will, respectively, represent the numerical approximation of $\nabla\phi$ at node $i$ in each angular sector $T_0, \ldots, T_{k_i}$.

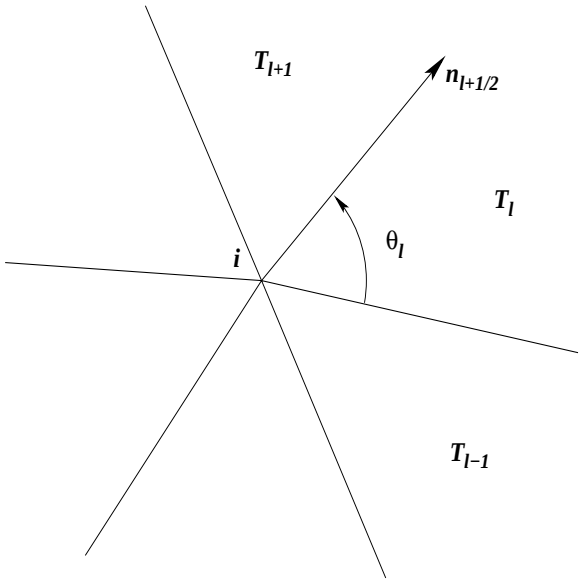


FIG. 2.1. *Node $i$ and its angular sectors.*

An important building block for the algorithms in this paper is the Lax–Friedrichs-type monotone Hamiltonian for arbitrary triangulations developed by Abgrall in [2], which is a generalization of the Lax–Friedrichs monotone Hamiltonian for Cartesian meshes in Osher and Shu [15]. This monotone Hamiltonian is given by

(2.2)
$$\hat{H}((\nabla\phi)_0,\ldots,(\nabla\phi)_{k_i}) = H\left(\frac{\sum_{l=0}^{k_i}\theta_l(\nabla\phi)_l}{2\pi}\right) - \frac{\alpha}{\pi}\sum_{l=0}^{k_i}\beta_{l+\frac{1}{2}}\left(\frac{(\nabla\phi)_l+(\nabla\phi)_{l+1}}{2}\right)\cdot\vec{n}_{l+\frac{1}{2}},$$

where

$$\beta_{l+\frac{1}{2}} = \tan\left(\frac{\theta_l}{2}\right)+\tan\left(\frac{\theta_{l+1}}{2}\right), \qquad \alpha = \max\left\{\max_{\substack{A\le u\le B\\ C\le v\le D}}|H_1(u,v)|, \max_{\substack{A\le u\le B\\ C\le v\le D}}|H_2(u,v)|\right\}.$$

Here $H_1$ and $H_2$ are the partial derivatives of $H$ with respect to $\phi_x$ and $\phi_y$, respectively, or the Lipschitz constants of $H$ with respect to $\phi_x$ and $\phi_y$, if $H$ is not differentiable. $[A,B]$ is the value range for $(\phi_x)_l$, and $[C,D]$ is the value range for $(\phi_y)_l$, over $0\le l\le k_i$ for the local Lax–Friedrichs Hamiltonian, and over $0\le l\le k_i$ and $0\le i\le N$ for the global Lax–Friedrichs Hamiltonian.

The $\hat{H}$ in (2.2) defines a monotone Hamiltonian. It is Lipschitz continuous in all arguments and is consistent with $H$, i.e., $\hat{H}(\nabla\phi,\ldots,\nabla\phi)=H(\nabla\phi)$. Hence if we have high-order approximations to $\nabla\phi$ at node $i$ in every angular sector, the numerical Hamiltonian $\hat{H}$ will be a high-order approximation to $H$.

The semidiscrete scheme is thus given by

(2.3)
$$\frac{d}{dt}\phi_i(t)+\hat{H}((\nabla\phi)_0,\ldots,(\nabla\phi)_{k_i})=0$$

and it is discretized in time by the high order nonlinearly stable Runge–Kutta time discretization in [18]. If the spatial dimension reduces to one, the numerical Hamiltonian (2.2) will reduce to the local or global Lax–Friedrichs Hamiltonian [15]:

$$\hat{H}(u^-, u^+) = H\left(\frac{u^- + u^+}{2}\right) - \frac{1}{2}\alpha(u^+ - u^-) \tag{2.4}$$

with $\alpha = \max_{A \le u \le B} |H'(u)|$. If the maximum for computing $\alpha$ is taken over the range covered by $u^-$ and $u^+$, we get the local Lax–Friedrichs Hamiltonian; if it is also taken over all grid points, we get the global Lax–Friedrichs Hamiltonian.

**2.2. Linear schemes.** Now we discuss how to construct a high-order approximation for $\nabla\phi$ in every angular sector of every node. Let $P^k$ denote the set of 2D polynomials of degree less than or equal to $k$. We use Lagrange interpolations as follows: given a smooth function $\phi$, and a triangulation with triangles $\{\triangle_0, \triangle_1, \ldots, \triangle_M\}$ and nodes $\{0, 1, 2, \ldots, N\}$, we would like to construct, for each triangle $\triangle_i$, a polynomial $p(x, y) \in P^k$ such that $p(x_l, y_l) = \phi(x_l, y_l)$, where $(x_l, y_l)$ are the coordinates of the three nodes of the triangle $\triangle_i$ and a few neighboring nodes. $p(x, y)$ would thus be a $(k+1)$st-order approximation to $\phi$ on the cell $\triangle_i$.

Because $k$th degree polynomial $p(x, y)$ has $K = \frac{(k+1)(k+2)}{2}$ degrees of freedom, we need to use the information of at least $K$ nodes. In addition to the three nodes of the triangle $\triangle_i$, we may take the other $K - 3$ nodes from the neighboring cells around triangle $\triangle_i$. We rename these $K$ nodes as $S_i = \{M_1, M_2, \ldots, M_K\}$; $S_i$ is called a big stencil for the triangle $\triangle_i$. Let $(x_i, y_i)$ be the barycenter of $\triangle_i$. Define $\xi = (x - x_i)/h_i$, $\eta = (y - y_i)/h_i$, where $h_i = \sqrt{|\triangle_i|}$ with $|\triangle_i|$ denoting the area of the triangle $\triangle_i$; then we can write $p(x, y)$ as

$$p(x, y) = \sum_{j=0}^{k} \sum_{s+r=j} a_{sr} \xi^s \eta^r.$$

Using the $K$ interpolation conditions,

$$p(M_l) = \phi(M_l), \qquad l = 1, 2, \ldots, K,$$

we get a $K \times K$ linear system for the $K$ unknowns $a_{sr}$. The normalized variables $\xi, \eta$ are used to make the condition number of the linear system independent of mesh sizes.

It is well known that in two and higher dimensions such interpolation problems is not always well defined. The linear system can be very ill-conditioned or even singular; in such cases we would have to add more nodes to the big stencil $S_i$ from the neighboring cells around triangle $\triangle_i$ to obtain an overdetermined linear system and then use the least-squares method to solve it. We remark that this ill-conditioning comes from both the geometric distribution of the nodes, for which we could do nothing other than changing the mesh, and from the choice of basis functions in the interpolation. For higher-order methods, a closer to orthogonal basis rather than $\xi^s \eta^r$ would be preferred, such as the procedure using barycentric coordinates in [1] and [3]. However, for the third- and fourth-order cases considered in this paper, we have chosen to use $\xi^s \eta^r$ for simplicity.

After we have obtained the approximation polynomial $p(x, y)$ on the triangle $\triangle_i$, $\nabla p$ will be a $k$th-order approximation for $\nabla\phi$ on $\triangle_i$. Hence we get the high-order approximation $\nabla p(x_l, y_l)$ to $\nabla\phi(x_l, y_l)$, for any one of the three vertices $(x_l, y_l)$ of the triangle $\triangle_i$, in the relevant angular sectors.

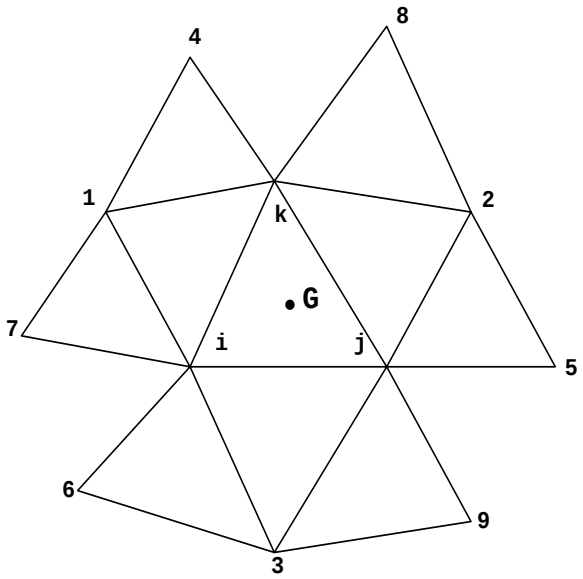


FIG. 2.2. *The nodes used for the big stencil of the third-order scheme.*

**2.2.1. Third-order linear schemes.** A scheme is called linear if it is linear when applied to a linear equation with constant coefficients. We need a third-order approximation for $\nabla\phi$ to construct a third-order linear scheme, hence we need a cubic polynomial interpolation. A cubic polynomial $p^3$ has 10 degrees of freedom. We will use some or all of the nodes shown in Figure 2.2 to form our big stencil. For extremely distorted meshes the number of nodes in Figure 2.2 may be fewer than the required 10. In such extreme cases we would need to expand the choice for the big stencil, following the idea in Figure 2.3 in the next subsection for the fourth-order scheme. For our target triangle $\triangle_0$, which has three vertices $i$, $j$, $k$ and the barycenter G, we need to construct a cubic polynomial $p^3$; then $\nabla p^3$ will be a third-order approximation for $\nabla\phi$ on $\triangle_0$, and the values of $\nabla p^3$ at points $i$, $j$, and $k$ will be third-order approximations for $\nabla\phi$ at the angular sector $\triangle_0$ of nodes $i$, $j$, and $k$. We name the nodes of the neighboring triangles of triangle $\triangle_0$ as follows: nodes 1, 2, 3 are the nodes (other than i, j, k) of neighbors of $\triangle_0$, nodes 4, 5, 6, 7, 8, 9 (other than 1, 2, 3, i, j, k) are the nodes of the neighbors of the three neighboring triangles of $\triangle_0$. Notice that the points 4, 5, 6, 7, 8, 9 do not have to be six distinct points. For example, the points 5 and 9 could be the same point.

Our interpolation points are nodes taken from a sorted node set. An ordering is given in the set so that, when the nodes are chosen sequentially from it to form the big stencil $S_0$, the target triangle $\triangle_0$ remains central to avoid serious downwind bias which could lead to linear instability. Referring to Figure 2.2, our interpolation points for the polynomial $p^3$ include nodes $i, j, k$ and the nodes taken from the sorted set: $W = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. The detailed procedure to determine the big stencil $S_0$ for the target triangle $\triangle_0$ is given below.

*Procedure* 2.1. The choice of the big stencil for the third-order scheme.

1. Referring to Figure 2.2, form a sorted node set: $W = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. In extreme cases when this set does not contain enough distinct points, we may need to add more points from the next layer of neighbors.

2. To start with, we take $S_0 = \{i, j, k, 1, 2, 3, 4, 5, 6, 7\}$. Use this stencil $S_0$ to form the $10 \times 10$ interpolation coefficient matrix $A$.
3. Compute the reciprocal condition number $c$ of $A$. This is provided by most linear solvers. If $c \geq \delta$ for some threshold $\delta$, we have obtained the final stencil $S_0$. Otherwise, add the next node in $W$ (i.e., node 8) to $S_0$. Use the 11 nodes in $S_0$ as interpolation points to get the $11 \times 10$ least square interpolation coefficient matrix $A$. Judge the reciprocal condition number $c$ again. Continue doing this until $c \geq \delta$ is satisfied. In our computation, we have taken $\delta = 10^{-3}$ as a good threshold after extensive numerical experiments. Notice that, since we have normalized the coordinates, this threshold does not change when the mesh is scaled uniformly in all directions. For all the triangulations we have tested, at most 12 nodes are needed in $S_0$ to reach the condition $c \geq \delta$. ∎

*Remark* 2.1. The ordering in the sorted node set $W$ is important to make the target cell $\triangle_0$ sufficiently "central" in the stencil, thus avoiding $S_0$ to be seriously downwind biased, which could lead to linear instability. Linear instability is indeed observed in our numerical experiments when 1, 2, 3 are not forced to be in $S_0$. ∎

We now have obtained the big stencil $S_0$ and its associated cubic polynomial $p^3$. For each node $(x_l, y_l)$ in $\triangle_0$, $\nabla p^3(x_l, y_l)$ is a third-order approximation to $\nabla\phi(x_l, y_l)$. In order to construct a high-order WENO scheme, an important step is to obtain a high-order approximation using a linear combination of lower-order approximations. We will use a linear combination of second-order approximations to get the same third-order approximation to $\nabla\phi(x_l, y_l)$ as $\nabla p^3(x_l, y_l)$, i.e., we require

$$\frac{\partial}{\partial x}p^3(x_l, y_l) = \sum_{s=1}^{q} \gamma_{s,x} \frac{\partial}{\partial x} p_s(x_l, y_l), \qquad \frac{\partial}{\partial y}p^3(x_l, y_l) = \sum_{s=1}^{q} \gamma_{s,y} \frac{\partial}{\partial y} p_s(x_l, y_l), \tag{2.5}$$

where $p_s$ are quadratic interpolation polynomials, and $\gamma_{s,x}$ and $\gamma_{s,y}$ are the linear weights for the $x$-directional derivative and the $y$-directional derivative, respectively, for $s = 1, \ldots, q$. The linear weights are constants depending only on the local geometry of the mesh. The equalities in (2.5) should hold for any choices of the function $\phi$.

Notice that to get a second-order approximation for the derivatives $\nabla\phi(x_l, y_l)$, we need a quadratic interpolation polynomial. According to the argument in [8], the cubic polynomial $p^3(x, y)$ has four more degrees of freedom than each quadratic polynomial $p_s(x, y)$, namely $x^3$, $x^2y$, $xy^2$, $y^3$. For the six degrees of freedom 1, $x$, $y$, $x^2$, $xy$, $y^2$, if we take $\phi = 1$, $\phi = x$, $\phi = y$, $\phi = x^2$, $\phi = xy$, and $\phi = y^2$, the equalities in (2.5) will hold for all these cases under only one constraint each on $\gamma_{s,x}$ and $\gamma_{s,y}$, namely $\sum_{s=1}^{q} \gamma_{s,x} = 1$ and $\sum_{s=1}^{q} \gamma_{s,y} = 1$, because $p^3$ and $p_s$ all reproduce these functions exactly. Hence we should only need $q \geq 5$. We take $q = 5$ in our scheme.

We now need $q = 5$ small stencils $\Gamma_s, s = 1, \ldots, 5$ for the target triangle $\triangle_0$, satisfying $S_0 = \bigcup_{s=1}^{5} \Gamma_s$, and every quadratic polynomial $p_s$ is associated with a small stencil $\Gamma_s$. In our third-order scheme, the small stencils will be the same for both directions $x, y$ and all three nodes $i, j, k$ in $\triangle_0$. However, the linear weights $\gamma_{s,x}, \gamma_{s,y}$ can be different for different nodes $i, j, k$ and different directions $x, y$. Because each quadratic polynomial has six degrees of freedom, the number of nodes in $\Gamma_s$ must be at least six. To build a small stencil $\Gamma_s$, we start from several candidates $\Gamma_s^{(r)}, r = 1, 2, \ldots, n_s$. These candidates are constructed by first taking a point $A_s^{(r)}$ as the "center," then finding at least six nodes from $S_0$ which have the shortest distances from $A_s^{(r)}$ and can generate the interpolation coefficient matrix with a good condition number, using the method of Procedure 2.1. We then choose the best $\Gamma_s$ among $\Gamma_s^{(r)}$,

$r = 1, \ldots, n_s$ for every $s = 1, \ldots, 5$. Here "best" means that by using this group of small stencils, the linear weights $\gamma_{s,x}$, $\gamma_{s,y}$, $s = 1, \ldots, 5$ for all three nodes $i, j, k$, are either all positive or have the smallest possible negative values in magnitude. The details of the algorithm is described in the following procedure.

*Procedure* 2.2. The third-order linear scheme. For every triangle $\triangle_l$, $l = 1, \ldots, N$, do steps 1–5:

1. Follow Procedure 2.1 to obtain the big stencil $S_l$ for $\triangle_l$.
2. For $s = 1, \ldots, 5$, find the set $W_s = \{\Gamma_s^{(r)}, r = 1, 2, \ldots, n_s\}$, which are the candidate small stencils for the $s$th small stencil. We use the following method to find the $\Gamma_s^{(r)}$ in $W_s$: First, nodes $i, j, k$ are always included in every $\Gamma_s^{(r)}$; then, we take a point $A_s^{(r)}$ as the center of $\Gamma_s^{(r)}$, detailed below, and find at least 3 additional nodes other than $i, j, k$ from $S_l$ which satisfy the following two conditions: (1) they have the shortest distances from $A_s^{(r)}$; and (2) taking them and the nodes $i, j, k$ as the interpolation points, we will obtain the interpolation coefficient matrix $A$ with a good condition number, namely the reciprocal condition number $c$ of $A$ satisfies $c \geq \delta$ with the same threshold $\delta = 10^{-3}$. For the triangulations we have tested, at most 8 nodes are used to reach this threshold value. Finally, the center of the candidate stencils $A_s^{(r)}$, $r = 1, \ldots, n_s; s = 1, \ldots, 5$ are taken from the nodes around $\triangle_l$ (see Figure 2.2) as follows:
   - $A_1^{(1)}$ = point G, $n_1 = 1$;
   - $A_2^{(1)}$ = node 1, $A_2^{(2)}$ = node 4, $A_2^{(3)}$ = node 7, $n_2 = 3$;
   - $A_3^{(1)}$ = node 2, $A_3^{(2)}$ = node 5, $A_3^{(3)}$ = node 8, $n_3 = 3$;
   - $A_4^{(1)}$ = node 3, $A_4^{(2)}$ = node 6, $A_4^{(3)}$ = node 9, $n_4 = 3$;
   - $\{A_5^{(r)}\}_{r=1}^{9}$ = nodes $4, 5, 6, 7, 8, 9$ and the middle points of nodes 4 and 8, 5 and 9, 6 and 7. $n_5 \leq 9$.
3. By taking one small stencil $\Gamma_s^{(r_s)}$ from each $W_s, s = 1, \ldots, 5$, to form a group, we obtain $n_1 \times n_2 \times \cdots \times n_5$ groups of small stencils. We eliminate the groups which contain the same small stencils and also eliminate the groups which do not satisfy the condition

$$\bigcup_{s=1}^{5} \Gamma_s^{(r_s)} = S_l.$$

According to every group $\{\Gamma_s^{(r_s)}, \ s = 1, \ldots, 5\}$ of small stencils, we have 5 quadratic polynomials $\{p_s^{(r_s)}\}_{s=1}^{5}$. We evaluate $\frac{\partial}{\partial x} p_s^{(r_s)}$ and $\frac{\partial}{\partial y} p_s^{(r_s)}$ at points $i, j, k$, to obtain second-order approximation values for $\nabla\phi$ at the three vertices of the triangle $\triangle_l$. We remark that for practical implementation, we do not use the polynomial itself, but compute a series of constants $\{a_l\}_{l=1}^{m}$ which depend on the local geometry only, such that

$$\frac{\partial}{\partial x} p_s^{(r_s)}(x_n, y_n) = \sum_{l=1}^{m} a_l \phi_l, \tag{2.6}$$

where every constant $a_l$ corresponds to one node in the stencil $\Gamma_s^{(r_s)}$ and $m$ is the total number of nodes in $\Gamma_s^{(r_s)}$. For every vertex $(x_n, y_n)$ of triangle $\triangle_l$, we obtain a series of such constants. And for the $y$-directional partial derivative, we compute the corresponding constants too.

4. For every group $\{\Gamma_s^{(r_s)},\ s=1,\ldots,5\}$, we form linear systems and solve them to get a series of linear weights $\gamma_{s,x}^{(r_s)}$ and $\gamma_{s,y}^{(r_s)}$ satisfying the equalities (2.5), for the three vertices $i,j,k$. Using the previous argument for combining low-order approximations to get high-order approximations, we form the linear system for $\gamma_{s,x}^{(r_s)}$ at a vertex $(\xi_n,\eta_n)$ as follows (note that we use normalized variables): take $\phi=\xi^3,\xi^2\eta,\xi\eta^2,\eta^3$, respectively; the equalities are

$$\sum_{s=1}^{5}\gamma_{s,x}^{(r_s)}\frac{\partial}{\partial\xi}p_s^{(r_s)}(\xi_n,\eta_n)=\frac{\partial}{\partial\xi}\phi(\xi_n,\eta_n), \tag{2.7}$$

where $p_s^{(r_s)}$ is the quadratic interpolation polynomial for $\phi$, using stencil $\Gamma_s^{(r_s)}$. Again, in practical implementation, we will not use $p_s^{(r_s)}$ itself; instead we use the constants computed in the last step and (2.6) to compute the approximation for the derivatives of $\phi$. Together with the requirement

$$\sum_{s=1}^{5}\gamma_{s,x}^{(r_s)}=1, \tag{2.8}$$

we obtain a $5\times5$ linear system for $\gamma_{s,x}^{(r_s)}$. For $\gamma_{s,y}^{(r_s)}$, the same argument can be applied. Note that we need to compute the reciprocal condition number $c$ for every linear system again. If $c\geq\delta$ for the same threshold $\delta=10^{-3}$, we will accept this group of stencils as one of the remaining candidates. Otherwise, the linear system is considered to be ill-conditioned and its corresponding group of small stencils $\{\Gamma_s^{(r_s)},s=1,\ldots,5\}$ is eliminated from further consideration.

5. For each of the remaining groups $\Lambda_l=\{\Gamma_s^{(r_s)},\ s=1,\ldots,5\}$, find the minimum value $\gamma_l$ of all these linear weights $\gamma_{s,x}^{(r_s)},\gamma_{s,y}^{(r_s)}$ of the three vertices $i,j,k$. Then find the group of small stencils whose $\gamma_l$ is the biggest, and take this group as our final 5 small stencils for triangle $\triangle_l$. Denote them by $\Gamma_s,\ s=1,\ldots,5$. For every final small stencil $\Gamma_s,s=1,2,\ldots,5$, we store the index numbers of the nodes in $\Gamma_s$, the constants in the linear combinations of node values to approximate values of $\nabla\phi$ at points $i,j,k$, and the linear weights $\gamma_{s,x}$, $\gamma_{s,y}$ of the three points $i,j,k$.
6. Now we have set up the necessary constants which depend only on the mesh for all triangles. To form the final linear scheme, we compute the third-order approximations $(\nabla\phi)_0,\ldots,(\nabla\phi)_{k_l}$ for all mesh nodes l, by the linear combinations of second-order approximations, using the prestored constants and linear weights. Then we can form the scheme (2.3). ∎

**2.2.2. Fourth-order linear schemes.** To construct a fourth-order linear scheme, we need a fourth-order approximation to $\nabla\phi$. Hence a fourth degree interpolation polynomial $p^4(x,y)$ is required, which has 15 degrees of freedom. We will use part or all of the nodes shown in Figure 2.3 to construct the big stencil of the fourth-order scheme. If there are fewer than 15 points in Figure 2.3, we will need to add more points from the next layer of neighbors into the candidate nodes, although this has not been necessary in our numerical experiments.

Comparing Figure 2.3 with Figure 2.2, we can see that, in order to get the set of candidate nodes for the big stencil of the fourth-order scheme, we have added nodes $10,11,12,13,14,15$ (Figure 2.3) to the set for the third-order scheme (Figure 2.2). Notice again that some of the added nodes may not be distinct.

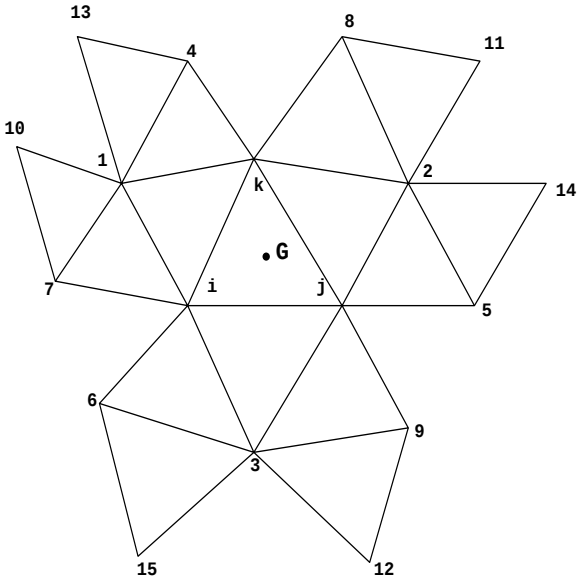


FIG. 2.3. *The nodes used for the big stencil of the fourth-order scheme.*

As in Procedure 2.1, in order to avoid possible linear instability due to serious downwinding, we first give an ordering to the candidate nodes. Referring to Figure 2.3, our interpolation points for the polynomial $p^4$ include nodes $i, j, k$ and the nodes taken from the sorted set: $W = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$. The detailed procedure to determine the big stencil $S_0$ for the target triangle $\triangle_0$ is given below.

*Procedure* 2.3. The big stencil of the fourth-order scheme.

1. Referring to Figure 2.3, form a sorted node set: $W = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$. In extreme cases when this set does not contain enough distinct points, we may need to add more points from the next layer of neighbors.
2. To start with, we take $S_0 = \{i, j, k, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$. Use this stencil $S_0$ to form the $15 \times 15$ interpolation coefficient matrix $A$.
3. Compute the reciprocal condition number $c$ of $A$. If $c \geq \delta$ for the same threshold $\delta = 10^{-3}$, we have obtained the final big stencil $S_0$. Otherwise, we add the next node (i.e., node 13) in $W$ to $S_0$. Then we obtain the $16 \times 15$ interpolation coefficient matrix $A$ associated with this $S_0$, an overdetermined system which can be solved by the least square procedure. Compute the reciprocal condition number of $A$ and check whether $c \geq \delta$ again. Repeat this if necessary until $c \geq \delta$ is satisfied. In our numerical tests, at most 16 nodes are needed in $S_0$ to reach our threshold value in all the cases. ∎

We emphasize again that the ordering for the nodes as in Figure 2.3 is important to guarantee the chosen stencil to yield to linear stability of the scheme.

*Remark* 2.2. Figures 2.2 and 2.3 are just the typical distribution of nodes around a target triangle. If the node distribution is different from that in Figures 2.2 and 2.3 because some nodes coincide, then the next level of neighboring nodes is added into the candidate node set, and these extra nodes are also sorted to make the target cell central. We can always add more nodes into our candidate node set if the nodes are not enough to reach our threshold value of condition number. ∎

Using the big stencil, we obtain the fourth-order approximation $\nabla p^4$ for $\nabla \phi$.

The key step in building our fourth-order WENO scheme is to get a fourth-order approximation for $\nabla\phi$ based on lower-order approximations. We still would like to construct several second-order approximations whose weighted average will give the same result as $\nabla p^4$ at each angular sector of every node, i.e.,

(2.9)
$$\frac{\partial}{\partial x}p^4(x_l,y_l)=\sum_{s=1}^{q}\gamma_{s,x}\frac{\partial}{\partial x}p_{s,x}(x_l,y_l),\qquad \frac{\partial}{\partial y}p^4(x_l,y_l)=\sum_{s=1}^{q}\gamma_{s,y}\frac{\partial}{\partial y}p_{s,y}(x_l,y_l),$$

where $p_{s,x},p_{s,y}$ are the quadratic interpolation polynomials and $\gamma_{s,x},\gamma_{s,y}$ are the linear weights.

*Remark* 2.3. The reason that we still use second-order approximations, not third-order ones, as the lower-order approximations is that each second-order approximation needs fewer nodes in the big stencil; thus it is easier to make the small stencils to be sufficiently diversified, which is important for the WENO technique to function well near discontinuous derivatives in the solutions. We have encountered difficulties in obtaining nonoscillatory results converging to the correct viscosity solutions for some of the more demanding test cases when third-order building blocks are used instead of the second-order ones. □

We now determine the value of $q$, which is the number of small stencils, in the equalities (2.9). Because $p^4$ has nine more degrees of freedom than the quadratic polynomials $p_{s,x},p_{s,y}$, i.e., $x^3,x^2y,xy^2,y^3,x^4,x^3y,x^2y^2,xy^3,y^4$, according to our previous argument, we would need $q\geq 10$. We take $q=10$ in our fourth-order scheme. The procedure to find all the small stencils is described below.

*Procedure* 2.4. The fourth-order linear scheme.

For every triangle $\triangle_l$, $l=1,\ldots,N$, do steps 1–3:

1. Follow Procedure 2.3 to obtain the big stencil $S_l$ for $\triangle_l$.
2. For each of the six approximations ($\phi_x,\phi_y$ at vertices $i,j,k$ of $\triangle_l$), find 10 small stencils, respectively. Let $\{\Gamma_{s,x},\ s=1,\ldots,10\}$ denote the small stencils to approximate $\phi_x$ at vertex $i$. To find them, we first determine 10 preliminary small stencils $\{\Gamma_s^0,s=1,\ldots,10\}$ as follows (see Figure 2.3):
   (1) Include nodes $\{i,j,k,1,2,3\}$ in $\Gamma_1^0$.
   (2) Include nodes $\{i,j,k,1,4,7\}$ in $\Gamma_2^0$.
   (3) Include nodes $\{i,j,k,2,5,8\}$ in $\Gamma_3^0$.
   (4) Include nodes $\{i,j,k,3,6,9\}$ in $\Gamma_4^0$.
   (5) Include nodes $\{i,j,k\}$ in all $\Gamma_s^0$, $s=5,6,\ldots,10$.
   (6) Take points $A_s$, $s=1,\ldots,10$ as the center of $\{\Gamma_s^0\}_{s=1}^{10}$, where $A_1=$ point $G$, $A_2=$ node 1, $A_3=$ node 2, $A_4=$ node 3, $A_5=$ node 10, $A_6=$ node 13, $A_7=$ node 11, $A_8=$ node 14, $A_9=$ node 12, and $A_{10}=$ node 15.
   (7) Obtain 6 nodes for each of $\{\Gamma_s^0\}_{s=1}^{10}$. There are already 6 points in $\{\Gamma_s^0\}_{s=1}^{4}$. For $\{\Gamma_s^0\}_{s=5}^{10}$, we add 3 nodes (other than $i,j,k$) from $S_l$ to each of them, which have the shortest distances from $A_s$. Then use $\Gamma_s^0$ to get the interpolation coefficient matrix, and judge the reciprocal condition number $c$ of the matrix. If $c$ reaches our threshold value $\delta=10^{-3}$, then $\Gamma_s^0$ is found. Otherwise add one different node from $S_l$ to $\Gamma_s^0$, which has the shortest distance from $A_s$. Continue this until $c\geq\delta$ is satisfied.
   (8) All of the 6 approximations (to $\phi_x,\phi_y$ at vertices $i,j,k$ of $\triangle_l$) have the common 10 preliminary small stencils $\{\Gamma_s^0,\ s=1,\ldots,10\}$.

   Now we come to determine $\{\Gamma_{s,x},\ s=1,\ldots,10\}$. Our goal is that the coefficient matrix $\widehat{A}$ of the $10\times 10$ linear system, which is obtained from $\{\Gamma_{s,x}\}_{s=1}^{10}$

and used to compute the linear weights, has a good condition number. For $s = 1, 2, \ldots, 10$, we perform the following steps:

(1) Taking the original small stencil $\Gamma_s^0$ as the interpolation stencil, compute the constants $\{a_l\}_{l=1}^m$, which depend on the mesh only and are the coefficients in the linear combination of function values at the nodes in $\Gamma_s^0$ to get the second-order approximation to $\phi_x$ at vertex $i$.

(2) Form the $s$th column of the matrix $\widehat{A}$. Let $\widehat{a}_{rt}$ be the elements of the matrix $\widehat{A}$. Take the 9 functions $\{\phi^{(r)}\}_{r=2}^{10} = \xi^3, \xi^2\eta, \xi\eta^2, \eta^3, \xi^4, \xi^3\eta, \xi^2\eta^2, \xi\eta^3, \eta^4$, respectively, and let $p_{s,x}^{(r)}$ be the second-order interpolation polynomial for $\phi^{(r)}$, $r = 2, \ldots, 10$; then $\widehat{a}_{1s} = 1$, and

$$\widehat{a}_{rs} = \frac{\partial}{\partial \xi} p_{s,x}^{(r)}(\xi_i, \eta_i), \qquad r = 2, \ldots, 10, \tag{2.10}$$

where $(\xi_i, \eta_i)$ is the coordinate of vertex $i$. (Again note that we use normalized variables.) We use the constants computed at the last step to implement this and will not need to use the polynomials themselves.

(3) Now $\widehat{A}$ has $s$ columns and is a $10 \times s$ matrix. Compute its reciprocal condition number $c$. If $c \geq \delta$, take the current small stencil as our $s$th small stencil $\Gamma_{s,x}$. Otherwise, change one node, which is in the current small stencil and farthest from the center point $A_s$, to another node from $S_l$, which is not in the current small stencil but is nearest to $A_s$. Now we get a new small stencil. This part can be repeated if $c \geq \delta$ is still not satisfied. Then the current small stencil is our $s$th small stencil.

For the small stencils to approximate the $y$-directional derivative at vertex $i$ and $x, y$-directional derivatives at other 2 vertices $j, k$, we use a similar procedure.

3. Now we have obtained the coefficient matrix $\widehat{A}$ with a good condition number. Along with the right hand vector $\vec{b}$, whose components are $b_1 = 1$ and

$$b_r = \frac{\partial}{\partial \xi} \phi^{(r)}(\xi_i, \eta_i), \qquad r = 2, \ldots, 10, \tag{2.11}$$

we obtain the $10 \times 10$ linear system

$$\widehat{A}\vec{\gamma} = \vec{b}. \tag{2.12}$$

We solve it to get the linear weights $\vec{\gamma} = (\gamma_{1,x}, \ldots, \gamma_{10,x})^T$. We use a similar method to get $\{\gamma_{s,y}\}_{s=1}^{10}$. Then for each of small stencils, we store the index numbers of the nodes in it, the constants in the linear combinations of node values to approximate values of the derivatives of $\phi$, and the linear weights.

4. Now we have set up necessary constants which depend only on the mesh for all triangles. To form the final linear scheme, we compute the fourth-order approximation $(\nabla\phi)_0, \ldots, (\nabla\phi)_{k_l}$ for all the mesh nodes l, by the linear combinations of second-order approximations, using the prestored constants and linear weights. We then form the scheme (2.3). ∎

*Remark* 2.4. We notice that in the fourth-order scheme, although the big stencil is the same to approximate $\frac{\partial}{\partial x}\phi$ and $\frac{\partial}{\partial y}\phi$ at all three vertices $i, j, k$ of the target cell $\triangle_0$, the small stencils can be different for the $x$-direction and $y$-direction derivatives at one vertex and also different at the three vertices $i, j, k$ in the same cell $\triangle_0$. This strategy is different from that for the third-order scheme described before. The reason

for this difference is that we need more small stencils than in the third-order case and it is difficult to find a group of common small stencils for all the six cases (three vortices, 2 derivatives each), although the cost using common small stencils is much cheaper. □

*Remark* 2.5. The strategy described above of finding second-order smaller stencils to make up the third-order and fourth-order big stencils has been reached after our extensive numerical experiments. They are found to be robust to different triangulations and equations. However, we are not claiming that these procedures will not fail; the back-up strategy if Procedure 2.2 or 2.4 fails is to use more small stencils (e.g., use 6 small stencils for the third-order case, or use 11 small stencils for the fourth-order case), even though we have not had to use this back-up strategy in our numerical experiments. The computational cost of these procedures is quite high, as many choices and comparisons are needed. Fortunately, at least for problems without adaptive mesh refinements, this procedure is done only once at the beginning and does not need to be repeated during time marching. Because of the prestored constant coefficients for computing the approximation to derivatives, the time evolution part of the scheme is very efficient. □

**2.3. WENO schemes.** In this section, we construct the WENO schemes based on nonlinear weights. The resulting schemes will be suitable to compute the H-J equations whose solutions are not smooth.

**2.3.1. WENO approximation.** We discuss only the case of WENO approximation for the $x$-directional derivative at vertex $i$ of the target cell $\triangle_l$. Other cases are similar. In order to compute the nonlinear weights, we need to compute the smoothness indicators first.

For a polynomial $p(x, y)$ defined on the target cell $\triangle_0$ with degree up to $k$, we take the smoothness indicator $\beta$ as

$$\beta = \sum_{2 \le |\alpha| \le k} \int_{\triangle_0} |\triangle_0|^{|\alpha|-2} \left(D^\alpha p(x, y)\right)^2 dxdy, \tag{2.13}$$

where $\alpha$ is a multi-index and $D$ is the derivative operator. The smoothness indicator measures how smooth the function $p$ is on the triangle $\triangle_0$: the smaller the smoothness indicator, the smoother the function $p$ is on $\triangle_0$. The scaling factor in front of the derivatives renders the smoothness indicator self-similar and invariant under uniform scaling of the mesh in all directions.

*Remark* 2.6. The definition of the smoothness indicator in (2.13) is different from that used in the WENO schemes for conservation laws [10]. In equation (2.13), the range of summation is $2 \le |\alpha| \le k$, but in [10] for conservation laws, it is $1 \le |\alpha| \le k$. An intuitive reason is that H-J equations are in some sense the conservation laws "integrated once"; hence smooth indicators for the former should involve derivatives one order higher than those for the latter. A similar strategy is also used in ENO schemes for H-J equations in [15]. We have found out through numerical experiments that if we still take $1 \le |\alpha| \le k$ to compute the smoothness indicators, we cannot obtain the correct viscosity solutions for some nonconvex H-J equations. □

Now we define the nonlinear weights as

$$\omega_j = \frac{\widetilde{\omega}_j}{\sum_m \widetilde{\omega}_m}, \qquad \widetilde{\omega}_j = \frac{\gamma_j}{(\varepsilon + \beta_j)^2}, \tag{2.14}$$

where $\gamma_j$ is the $j$th linear weight (e.g., the $\gamma_{s,x}$ in our linear schemes), $\beta_j$ is the smoothness indicator for the $j$th interpolation polynomial $p_j(x,y)$ (e.g., the $p_s$ in (2.5) for the third-order case and the $p_{s,x}$ in (2.9) for the fourth-order case) associated with the $j$th small stencil, and $\varepsilon$ is a small positive number to avoid the denominator to become 0. We take $\varepsilon = 10^{-6}$ for all the computations in this paper. The final WENO approximation for the $x$-directional derivative at vertex $i$ of target cell $\triangle_l$ is given by

$$(\phi_x)_i = \sum_{j=1}^{q} \omega_j \frac{\partial}{\partial x} p_j(x_i, y_i), \tag{2.15}$$

where $(x_i, y_i)$ are the coordinates of vertex $i$, $q = 5$ for the third-order schemes, and $q = 10$ for the fourth-order schemes.

In our WENO schemes, the linear weights $\{\gamma_j\}_{j=1}^{q}$ depend on the local geometry of the mesh and can be negative. If $\min(\gamma_1, \ldots, \gamma_q) < 0$, we adopt the splitting technique of treating negative weights in WENO schemes developed by Shi, Hu, and Shu [17]. We omit the details of this technique and refer the readers to [17].

Again, we remark that the smoothness indicator (2.13) is a quadratic function of function values on nodes of the small stencil, so in practical implementation, to compute the smoothness indicator $\beta_j$ for the $j$th small stencil by (2.13), we do not need to use the interpolation polynomial itself; instead we use a series of constants $\{a_{rt},\ r = 1, \ldots, t;\ t = 1, \ldots, m\}$, which can be precomputed and they depend on the mesh only, such that

$$\beta_j = \sum_{t=1}^{m} \phi_t \left( \sum_{r=1}^{t} a_{rt} \phi_r \right), \tag{2.16}$$

where $m$ is the total number of nodes in the $j$th small stencil. These constants for all smoothness indicators should be precomputed and stored once the mesh is generated.

**2.3.2. Algorithm flowchart.** We summarize the algorithm for the third-order and the fourth-order WENO schemes as follows:

*Procedure* 2.5. The third- and fourth-order WENO schemes.

1. Generate a triangular mesh.
2. Compute and store all constants which depend only on the mesh and the accuracy order of the scheme. These constants include the node index numbers of each small stencil, the coefficients in the linear combinations of function values on nodes of small stencils to approximate the derivative values and the linear weights, following the Procedure 2.2 for the third-order case and the Procedure 2.4 for the fourth-order case, and the constants for computing smoothness indicators in (2.16).
3. Using the prestored constants, for each angular sector of every node $i$, compute the low-order approximations for $\nabla\phi$ and nonlinear weights, then compute the third- or fourth-order WENO approximations (2.15). Form the scheme (2.3). Use high-order TVD Runge–Kutta time stepping [18] to evolve in time.

*Remark* 2.7. The storage requirements for the third- and fourth-order methods are proportional to the product of the number of nodes and the maximum number of angular sectors. The constants for this proportion, taking into consideration smooth indicators, linear weights, and coefficients for the derivatives, are about 730 and 1460, respectively. These numbers can be dramatically reduced at the price of

additional calculations for each time step. In our implementation, the fourth-order WENO scheme is about 7 times more costly in CPU time than the third-order version. Whether to use a higher-order or lower-order version of the scheme is problem dependent. Presumably, one would want to use a higher-order scheme if smooth region accuracy is important. □

**3. Numerical examples.** In this section, we apply the WENO schemes developed in the previous section to a set of one-dimensional (1D) and 2D problems. The CFL number is taken as 0.5 in all the cases, except for the accuracy test where it is taken to be smaller if necessary to guarantee that spatial errors dominate. The local Lax–Friedrichs flux is used for all the test cases. For the temporal discretization, we use the third-order TVD Runge–Kutta scheme of Shu and Osher in [18].

We have not described the details of the 1D algorithm in the previous section. The algorithm in 1D is just the same 2D algorithm with only two "angular sectors." The WENO interpolation follows along the lines of [10] and [5]. In all the 1D numerical examples, we use nonuniform meshes. These nonuniform meshes are obtained by randomly shifting the cell boundaries in a uniform mesh in the range $[-0.1h, 0.1h]$, where $h$ is the uniform mesh size. When we perform the accuracy test, the refinement of the meshes is achieved by cutting each cell into two smaller equally sized ones.

*Example* 3.1. A 1D linear equation is

$$\begin{cases} \phi_t + \phi_x = 0, & 0 \le x < 2\pi, \\ \phi(x,0) = \sin(x), \end{cases} \tag{3.1}$$

with periodic boundary conditions. We remark that even though this equation is also a conservation law, the method used here is different from the WENO schemes for conservation laws in [10].

We use third-, fifth-, and seventh-order linear and WENO schemes to compute the problem to $t = 2.0$. The errors and the numerical orders of accuracy are listed in Table 3.1. We can see that the correct orders of accuracy are obtained by both the linear and WENO schemes. In all the accuracy tests, we list only results for $L^1$ errors. Those for $L^2$ or $L^\infty$ errors, which follow the same patterns for the smooth cases, are omitted to save space.

TABLE 3.1
*Accuracy for 1D linear equation, linear and WENO schemes, $t = 2$.*

| Linear | 3rd order | | 5th order | | 7th order | |
|---|---|---|---|---|---|---|
| $N$ | $L^1$ error | order | $L^1$ error | order | $L^1$ error | order |
| 10 | 2.68E-02 | — | 3.10E-03 | — | 7.95E-04 | — |
| 20 | 3.57E-03 | 2.91 | 1.05E-04 | 4.88 | 6.83E-06 | 6.86 |
| 40 | 4.55E-04 | 2.97 | 3.39E-06 | 4.95 | 5.45E-08 | 6.97 |
| 80 | 5.74E-05 | 2.99 | 1.08E-07 | 4.97 | 4.31E-10 | 6.98 |
| 160 | 7.20E-06 | 3.00 | 3.40E-09 | 4.99 | 3.39E-12 | 6.99 |

| WENO | 3rd order | | 5th order | | 7th order | |
|---|---|---|---|---|---|---|
| $N$ | $L^1$ error | order | $L^1$ error | order | $L^1$ error | order |
| 10 | 1.18E-01 | — | 8.60E-03 | — | 2.27E-03 | — |
| 20 | 2.60E-02 | 2.18 | 3.74E-04 | 4.52 | 1.95E-05 | 6.87 |
| 40 | 4.68E-03 | 2.48 | 1.42E-05 | 4.72 | 1.83E-07 | 6.74 |
| 80 | 7.44E-04 | 2.65 | 4.88E-07 | 4.87 | 1.81E-09 | 6.65 |
| 160 | 8.93E-05 | 3.06 | 1.57E-08 | 4.96 | 1.54E-11 | 6.87 |

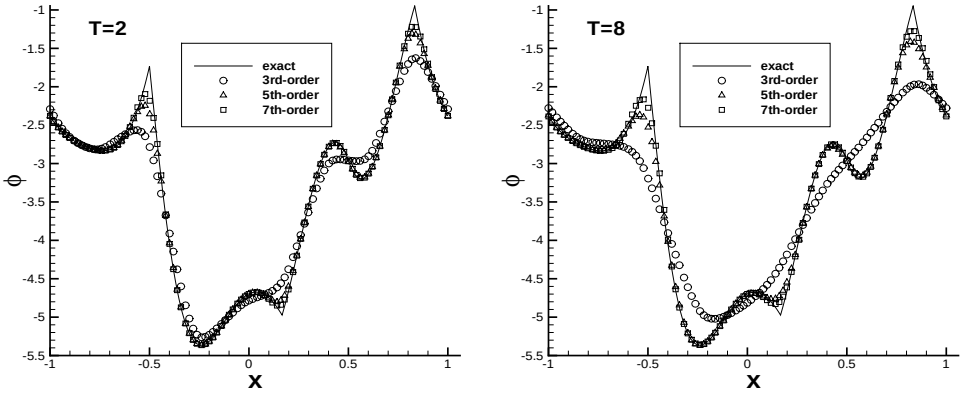


FIG. 3.1. *1D linear equation. Nonuniform mesh with* 101 *points. Solid line is the exact solution; "o" is the third-order WENO solution; "Δ" is the fifth-order WENO solution; and "□" is the seventh-order WENO solution. Left: results at* $t = 2$*; right: results at* $t = 8$*.*

*Example* 3.2. A 1D linear equation is

$$\begin{cases} \phi_t + \phi_x = 0, & -1 \le x < 1, \\ \phi(x,0) = g(x - 0.5), \end{cases} \tag{3.2}$$

with periodic boundary conditions, where

(3.3)

$$g(x) = -\left(\frac{\sqrt{3}}{2} + \frac{9}{2} + \frac{2\pi}{3}\right)(x+1) + \begin{cases} 2\cos(\frac{3\pi x^2}{2}) - \sqrt{3}, & -1 \le x < -\frac{1}{3}; \\ \frac{3}{2} + 3\cos(2\pi x), & -\frac{1}{3} \le x < 0; \\ \frac{15}{2} - 3\cos(2\pi x), & 0 \le x < \frac{1}{3}; \\ \frac{28+4\pi+\cos(3\pi x)}{3} + 6\pi x(x-1), & \frac{1}{3} \le x < 1. \end{cases}$$

This example comes from [9]. We use third-, fifth-, and seventh-order WENO schemes with 101 nonuniformly spaced points and show the results at t=2 and 8 in Figure 3.1. We can clearly observe better resolution with an increased order of accuracy, and the advantage of higher-order schemes is more prominent for longer time.

*Example* 3.3. A 1D Burgers equation is

$$\begin{cases} \phi_t + \frac{(\phi_x+1)^2}{2} = 0, & -1 \le x < 1, \\ \phi(x,0) = -\cos(\pi x), \end{cases} \tag{3.4}$$

with periodic boundary conditions.

At $t = 0.5/\pi^2$, the solution is still smooth. We list the errors and the numerical orders of accuracy in Table 3.2, using linear and WENO schemes. We see that the correct orders of accuracy are obtained. At $t = 3.5/\pi^2$, the solution has developed a discontinuous derivative. We list the $L^1$ errors and the numerical orders of accuracy in Table 3.3 using WENO schemes, both globally and in the smooth regions 0.1 away from the derivative singularity. We can see that globally the solution is a bit higher than second order because of the lower accuracy at the singularity, although higher-order schemes have lower magnitudes of errors for the same mesh. In smooth regions

TABLE 3.2
*Accuracy for 1D Burgers equation, linear and WENO schemes, $t = 0.5/\pi^2$.*

| Linear | 3rd order | | 5th order | | 7th order | |
|---|---|---|---|---|---|---|
| $N$ | $L^1$ error | order | $L^1$ error | order | $L^1$ error | order |
| 10 | 4.26E-03 | — | 7.84E-04 | — | 5.81E-04 | — |
| 20 | 5.51E-04 | 2.95 | 4.73E-05 | 4.05 | 2.28E-05 | 4.67 |
| 40 | 7.54E-05 | 2.87 | 2.39E-06 | 4.31 | 4.28E-07 | 5.74 |
| 80 | 9.86E-06 | 2.93 | 9.07E-08 | 4.72 | 4.90E-09 | 6.45 |
| 160 | 1.26E-06 | 2.97 | 3.08E-09 | 4.88 | 4.55E-11 | 6.75 |
| 320 | 1.59E-07 | 2.98 | 9.91E-11 | 4.96 | 4.18E-13 | 6.77 |

| WENO | 3rd order | | 5th order | | 7th order | |
|---|---|---|---|---|---|---|
| $N$ | $L^1$ error | order | $L^1$ error | order | $L^1$ error | order |
| 10 | 1.94E-02 | — | 1.81E-03 | — | 7.04E-04 | — |
| 20 | 4.77E-03 | 2.02 | 8.58E-05 | 4.39 | 2.92E-05 | 4.59 |
| 40 | 1.02E-03 | 2.22 | 4.93E-06 | 4.12 | 4.28E-07 | 6.09 |
| 80 | 1.77E-04 | 2.53 | 2.04E-07 | 4.60 | 4.32E-09 | 6.63 |
| 160 | 2.62E-05 | 2.76 | 7.98E-09 | 4.67 | 3.83E-11 | 6.81 |
| 320 | 2.98E-06 | 3.13 | 2.58E-10 | 4.95 | 3.55E-13 | 6.75 |

TABLE 3.3
*Accuracy for 1D Burgers equation, WENO schemes, $t = 3.5/\pi^2$. Global $L^1$ errors and $L^1$ errors in smooth regions 0.1 away from the derivative singularity.*

| | 3rd order | | 5th order | | 7th order | |
|---|---|---|---|---|---|---|
| $N$ | $L^1$ error | order | $L^1$ error | order | $L^1$ error | order |
| Global region | | | | | | |
| 40 | 2.30E-03 | | 1.50E-03 | | 1.23E-03 | |
| 80 | 4.96E-04 | 2.21 | 3.44E-04 | 2.12 | 2.78E-04 | 2.15 |
| 160 | 1.01E-04 | 2.29 | 7.11E-05 | 2.27 | 5.55E-05 | 2.32 |
| 320 | 1.87E-05 | 2.44 | 1.17E-05 | 2.61 | 8.46E-06 | 2.71 |
| Smooth region 0.1 away from the derivative singularity | | | | | | |
| 40 | 5.60E-05 | | 1.45E-06 | | 1.17E-06 | |
| 80 | 6.76E-06 | 3.05 | 3.16E-08 | 5.52 | 1.79E-08 | 6.03 |
| 160 | 8.87E-07 | 2.93 | 4.20E-10 | 6.23 | 1.55E-10 | 6.86 |
| 320 | 1.14E-07 | 2.97 | 1.28E-11 | 5.03 | 4.22E-13 | 8.52 |

away from the derivative singularity, the schemes maintain their correct high-order accuracy, thus justifying the usage of higher-order schemes if accuracy in smooth regions is a major concern. In Figure 3.2, we show the sharp corner-like numerical solution with 41 points using the fifth- and seventh-order WENO schemes. From now on, the solid line denotes the exact solution, and the circles denote the numerical solutions.

*Example* 3.4. The 1D Riemann problem with a nonconvex flux is as follows:

$$\begin{cases} \phi_t + \frac{1}{4}(\phi_x^2 - 1)(\phi_x^2 - 4) = 0, & -1 < x < 1, \\ \phi(x, 0) = -2|x|. \end{cases} \tag{3.5}$$

We remark that this is a demanding test case. Many schemes have poor resolutions or could even converge to a nonviscosity solution for this case. Numerical results at $t = 1$ with 81 nonuniform grid points are shown in Figure 3.3. Third-, fifth-, and seventh-order WENO schemes are used.

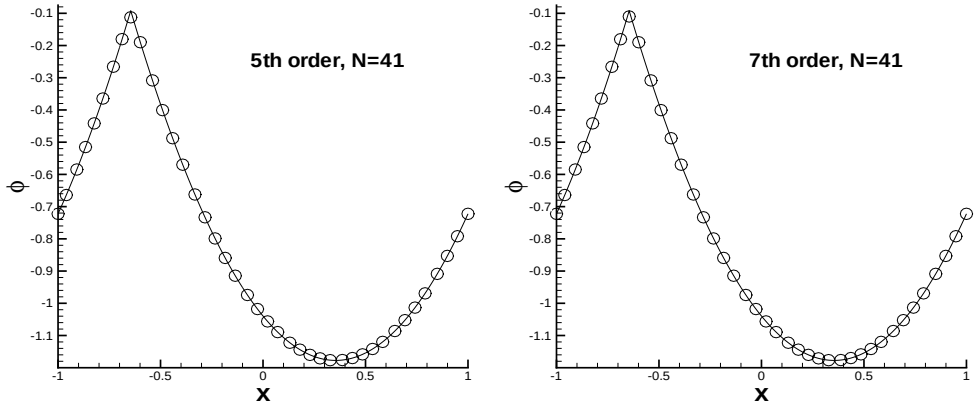


FIG. 3.2. *1D Burgers equation, $t = 3.5/\pi^2$. Solid line: the exact solution; circles: numerical solutions. Nonuniform mesh with* 41 *points. Left: fifth-order WENO scheme; right: seventh-order WENO scheme.*

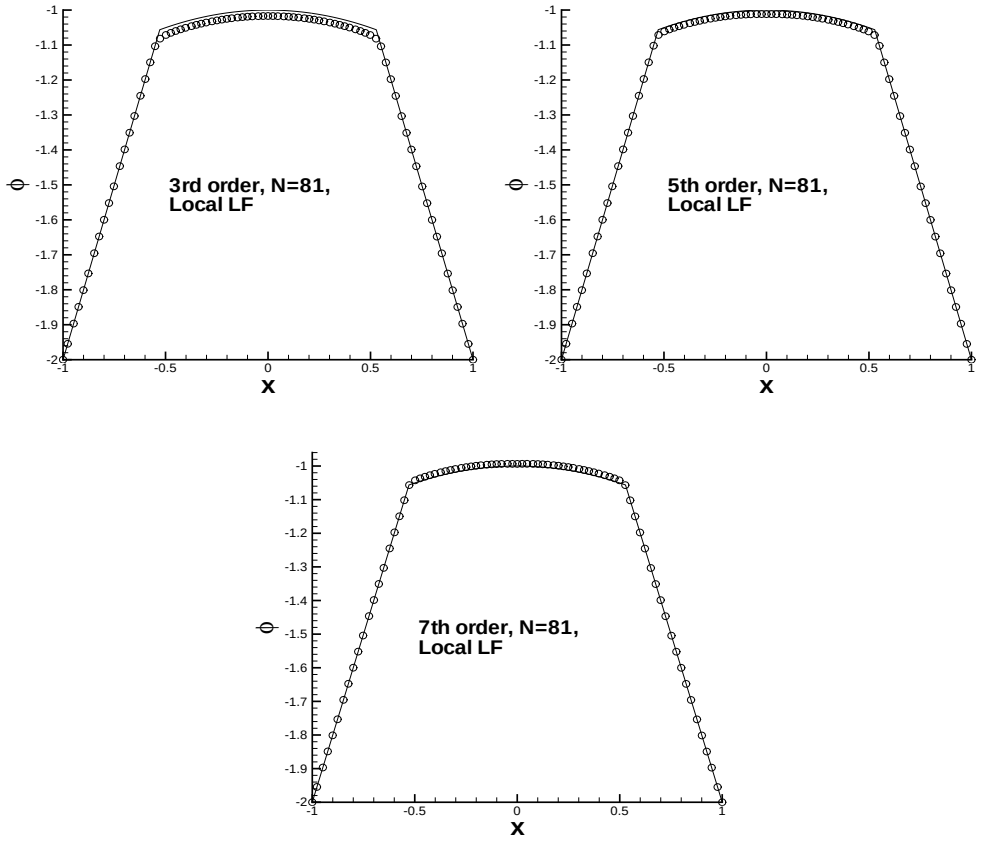


FIG. 3.3. *1D Riemann problem, $H(u) = \frac{1}{4}(u^2-1)(u^2-4)$, $t = 1$. Solid line: the exact solution; circles: numerical solutions. Nonuniform mesh with* 81 *points. Top left: third-order WENO scheme; top right: fifth-order WENO scheme; bottom: seventh-order WENO scheme.*

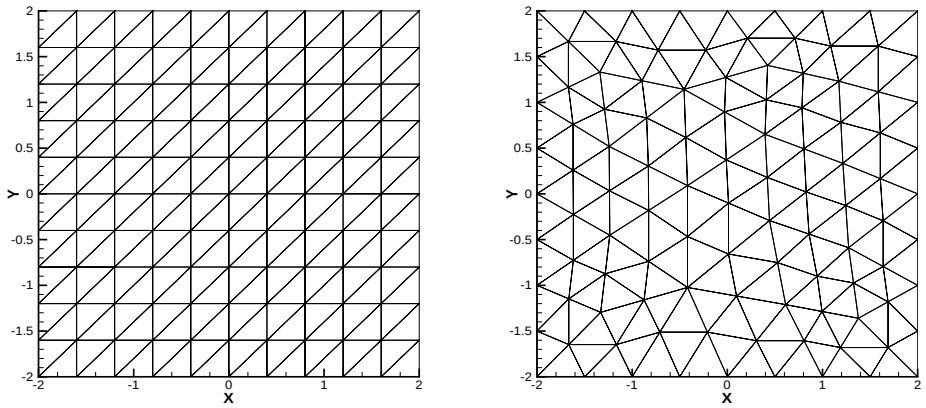


FIG. 3.4. *Left: coarsest uniform mesh with* $h = \frac{2}{5}$*; right: coarsest nonuniform mesh with* $N = 105$ *nodes.*

TABLE 3.4
*Accuracy for 2D linear equation, nonuniform meshes, linear and WENO schemes,* $t = 2$*.*

| | 3rd order | | | | 4th order | | | |
|---|---|---|---|---|---|---|---|---|
| | Linear | | WENO | | Linear | | WENO | |
| $N$ | $L^1$ error | order | $L^1$ error | order | $L^1$ error | order | $L^1$ error | order |
| 105 | 1.89E-01 | — | 4.51E-01 | — | 4.81E-02 | — | 4.24E-01 | — |
| 385 | 2.93E-02 | 2.69 | 1.61E-01 | 1.49 | 2.17E-03 | 4.47 | 1.36E-01 | 1.64 |
| 1473 | 3.88E-03 | 2.92 | 3.13E-02 | 2.37 | 9.99E-05 | 4.44 | 1.89E-02 | 2.84 |
| 5761 | 4.95E-04 | 2.97 | 4.81E-03 | 2.70 | 5.33E-06 | 4.23 | 1.64E-03 | 3.52 |
| 22785 | 6.22E-05 | 2.99 | 5.51E-04 | 3.13 | 3.15E-07 | 4.08 | 1.08E-04 | 3.93 |

*Example* 3.5. A 2D linear equation is

$$\begin{cases} \phi_t + \phi_x + \phi_y = 0, & -2 \le x < 2, -2 \le y < 2, \\ \phi(x, y, 0) = \sin(\frac{\pi}{2}(x + y)), \end{cases} \tag{3.6}$$

with periodic boundary conditions.

We first use uniform triangular meshes, shown in Figure 3.4, left, for the coarsest case $h = \frac{2}{5}$, where $h$ is the length of the right angled side, to test the accuracy for the third- and fourth-order linear schemes and WENO schemes. We then use nonuniform meshes, shown in Figure 3.4, right, for the coarsest case $N = 105$, where $N$ is the total number of nodes in the mesh. The refinement of the nonuniform meshes is done in a uniform way, namely by cutting each triangle into four smaller similar ones. The accuracy results are shown only for the nonuniform meshes, to save space, in Table 3.4. The expected orders of accuracy are obtained.

*Example* 3.6. A 2D Burgers equation is

$$\begin{cases} \phi_t + \dfrac{(\phi_x + \phi_y + 1)^2}{2} = 0, & -2 \le x < 2, -2 \le y < 2, \\ \phi(x, y, 0) = -\cos(\frac{\pi(x+y)}{2}), \end{cases} \tag{3.7}$$

with periodic boundary conditions.

At $t = 0.5/\pi^2$, the solution is still smooth. We use both the uniform meshes (Figure 3.4, left) and the nonuniform meshes (Figure 3.4, right) to test the accuracy, but errors and orders of accuracy for the third- and fourth-order linear and WENO

TABLE 3.5
*Accuracy for 2D Burgers equation, nonuniform meshes, linear and WENO schemes, $t = 0.5/\pi^2$.*

| | 3rd order | | | | 4th order | | | |
|---|---|---|---|---|---|---|---|---|
| | Linear | | WENO | | Linear | | WENO | |
| $N$ | $L^1$ error | order | $L^1$ error | order | $L^1$ error | order | $L^1$ error | order |
| 105 | 1.09E-02 | — | 3.76E-02 | — | 5.97E-03 | — | 4.00E-02 | — |
| 385 | 1.58E-03 | 2.78 | 8.98E-03 | 2.07 | 4.72E-04 | 3.66 | 8.50E-03 | 2.23 |
| 1473 | 2.18E-04 | 2.85 | 2.05E-03 | 2.13 | 2.69E-05 | 4.14 | 1.61E-03 | 2.40 |
| 5761 | 2.90E-05 | 2.92 | 3.66E-04 | 2.49 | 1.30E-06 | 4.37 | 1.72E-04 | 3.22 |
| 22785 | 3.74E-06 | 2.95 | 4.35E-05 | 3.07 | 6.06E-08 | 4.42 | 1.31E-05 | 3.72 |

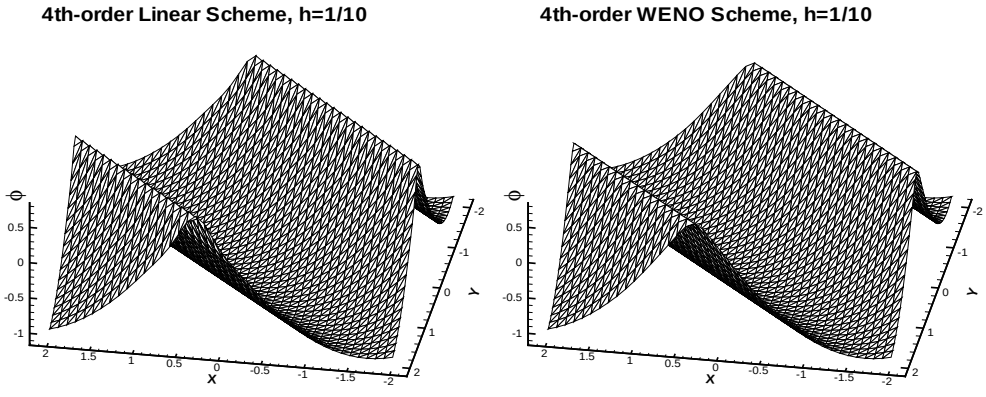


FIG. 3.5. *2D Burgers equation, $t = 1.5/\pi^2$. Uniform mesh with $h = \frac{1}{10}$. Left: fourth-order linear scheme; right: fourth-order WENO scheme.*

schemes only for the nonuniform meshes are listed in Table 3.5, to save space. We can see that the correct orders of accuracy are obtained. At $t = 1.5/\pi^2$, the solution has developed discontinuous derivatives. We use the third- and fourth-order linear and WENO schemes to compute, with uniform mesh of $h = \frac{1}{10}$. The results for the fourth-order case are shown in Figure 3.5. We can see that for this example, which is not very demanding, the linear schemes which do not use WENO nonlinear weights can also obtain nonoscillatory results. It seems that the nonlinear WENO strategy is needed only for the more demanding cases such as those for some nonconvex fluxes. The $L^1$ errors after the appearance of the singularity behave similarly as in the 1D case in Table 3.3 of Example 3.3; we thus omit the details.

*Example* 3.7. The 2D methods are applied to the 1D nonconvex Riemann problem in Example 3.4. This example is more demanding than the previous two examples, and we will see that the linear scheme will not work. We solve the problem in Example 3.4 in the domain $[-1, 1] \times [-0.2, 0.2]$ with the triangulation shown in Figure 3.6. The periodic boundary condition is applied in the $y$-direction. We plot the solutions along the central cut line $y = 0$.

First we use the third-order linear scheme to compute this problem and refine the mesh to test convergence. The results are plotted in Figure 3.7, with $h = \frac{1}{20}, \frac{1}{40}, \frac{1}{80}$. We can see that the solutions of the linear scheme with a mesh refinement does not converge to the correct viscosity solution. This is similar to the conservation law case of convergence to a entropy violating weak solution.

Next we use the third-order and fourth-order WENO schemes. From Figure 3.8,

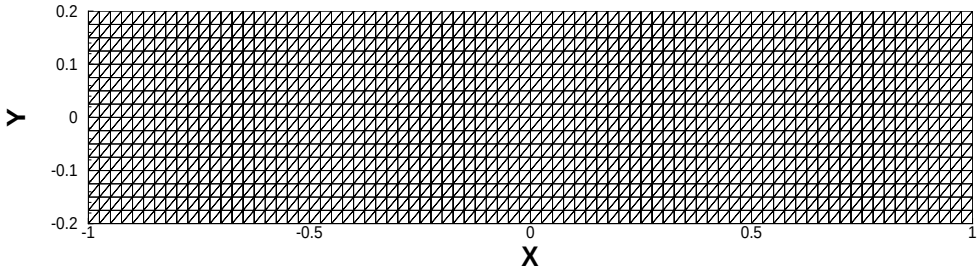


FIG. 3.6. *Mesh for Example* 3.7 *with* $h = \frac{1}{40}$.

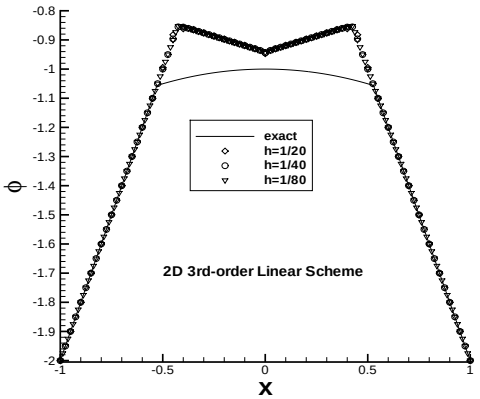


FIG. 3.7. *Convergence study for Example* 3.7*, linear scheme, central line cut at* $y = 0$*. Solid line is the exact solution; diamonds are for the coarsest mesh with* $h = \frac{1}{20}$*, circles are for the next refined mesh with* $h = \frac{1}{40}$*; reverse triangles are for the most refined mesh with* $h = \frac{1}{80}$.

we can see that the solutions of WENO schemes converge to the correct viscosity solution when the mesh is refined. And obviously, the fourth-order scheme has a better resolution than the third-order scheme.

At last we use the first-order monotone scheme with the monotone Lax–Friedrichs Hamiltonian (2.2) to compute the problem. The results are shown in Figure 3.9. It converges to the correct viscosity solution, as expected. But comparing with the results of the third- and fourth-order WENO schemes, the first-order monotone scheme needs a much more refined mesh to achieve the same resolution.

*Example* 3.8. A 2D Riemann problem is

$$\begin{cases} \phi_t + \sin(\phi_x + \phi_y) = 0, & -1 < x < 1, -1 < y < 1, \\ \phi(x, y, 0) = \pi(|y| - |x|). \end{cases} \tag{3.8}$$

For this example, we use the uniform triangular mesh with $40 \times 40 \times 2$ elements. Third- and fourth-order WENO schemes are used. We show the numerical results at $t = 1$ in Figure 3.10.

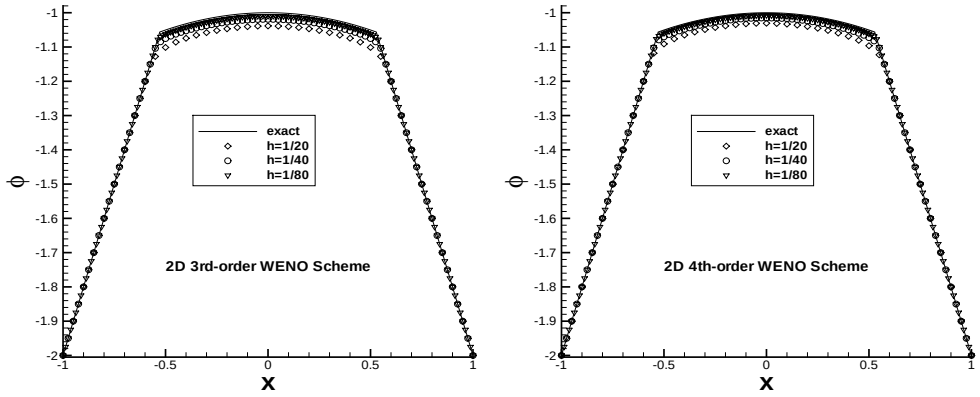


FIG. 3.8. *Convergence study for Example* 3.7, *WENO schemes, central line cut at* $y = 0$. *Solid line is the exact solution; diamonds are for the coarsest mesh with* $h = \frac{1}{20}$; *circles are for the next refined mesh with* $h = \frac{1}{40}$; *reverse triangles are for the most refined mesh with* $h = \frac{1}{80}$. *Left: third-order WENO schemes; right: fourth-order WENO schemes.*

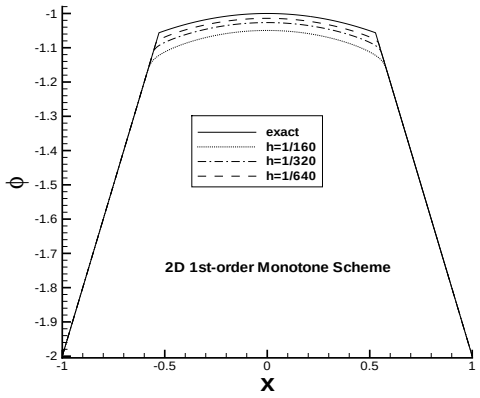


FIG. 3.9. *Convergence study for Example* 3.7, *first-order monotone scheme, central line cut at* $y = 0$. *Solid line is the exact solution; dotted line is for the coarsest mesh with* $h = \frac{1}{160}$; *dash-dotted line is for the next refined mesh with* $h = \frac{1}{320}$; *dashed line is for the most refined mesh with* $h = \frac{1}{640}$.

*Example* 3.9. A problem from optimal control is

(3.9)
$$\begin{cases} \phi_t + (\sin y)\phi_x + (\sin x + sign(\phi_y))\phi_y - \frac{1}{2}\sin^2 y - (1 - \cos x) = 0, \\ \qquad\qquad - \pi < x < \pi, -\pi < y < \pi, \\ \phi(x, y, 0) = 0, \end{cases}$$

with periodic boundary conditions; see [15].

We use the uniform triangle mesh with $60 \times 60 \times 2$ elements. Third- and fourth-order WENO schemes are used but only the result with the fourth-order WENO schemes are shown to save space. The solution at $t = 1$ is shown in Figure 3.11, left, and the optimal control $\omega = sign(\phi_y)$ is shown in Figure 3.11, right.

*Example* 3.10. A 2D eikonal equation with a nonconvex Hamiltonian, which

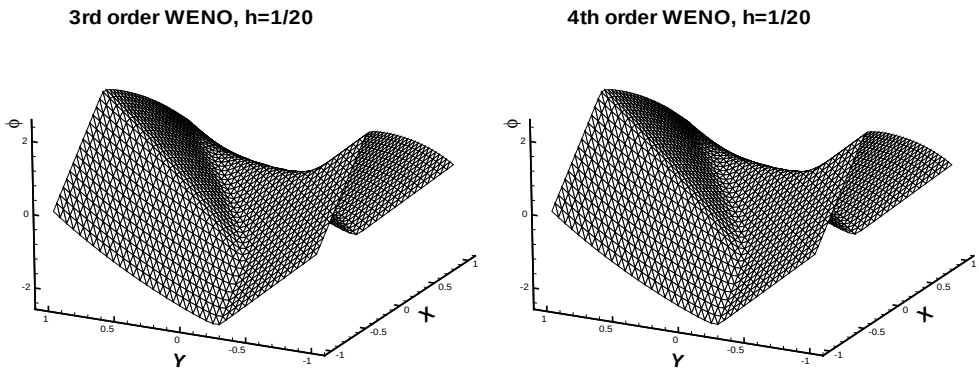


FIG. 3.10. *2D Riemann problem, $H(u,v) = \sin(u+v)$, $t = 1$. Uniform triangle mesh with $h = \frac{1}{20}$. Left: third-order WENO scheme; right: fourth-order WENO scheme.*

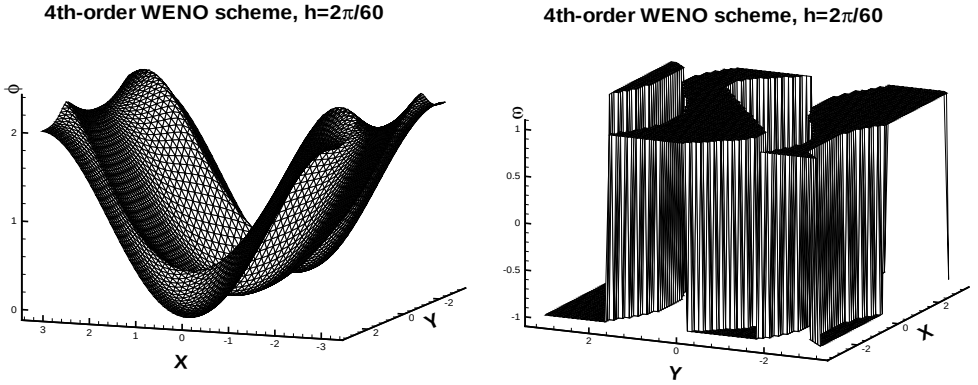


FIG. 3.11. *Control problem, $t = 1$. Uniform triangle mesh with $h = \frac{2\pi}{60}$. Fourth-order WENO scheme. Left: the solution; right: the optimal control $\omega = sign(\phi_y)$.*

arises in geometric optics [11], is given by

$$\begin{cases} \phi_t + \sqrt{\phi_x^2 + \phi_y^2 + 1} = 0, & 0 \le x < 1,\ 0 \le y < 1, \\ \phi(x,y,0) = 0.25(\cos(2\pi x) - 1)(\cos(2\pi y) - 1) - 1. \end{cases} \tag{3.10}$$

We use the third-order and fourth-order WENO schemes but show only the results with the fourth-order WENO scheme. We use both the uniform meshes and the nonuniform mesh shown in Figure 3.12, left, and present the result in Figure 3.12, right, for the nonuniform mesh case at $t = 0.6$.

*Example* 3.11. The level set equation in a domain with a hole is

$$\begin{cases} \phi_t + sign(\phi_0)\left(\sqrt{\phi_x^2 + \phi_y^2} - 1\right) = 0, & \frac{1}{2} < \sqrt{x^2 + y^2} < 1, \\ \phi(x,y,0) = \phi_0(x,y). \end{cases} \tag{3.11}$$

This problem comes from [19]. The solution $\phi$ to (3.11) has the same zero level set as $\phi_0$, and the steady state solution is the distance function to that zero level curve.

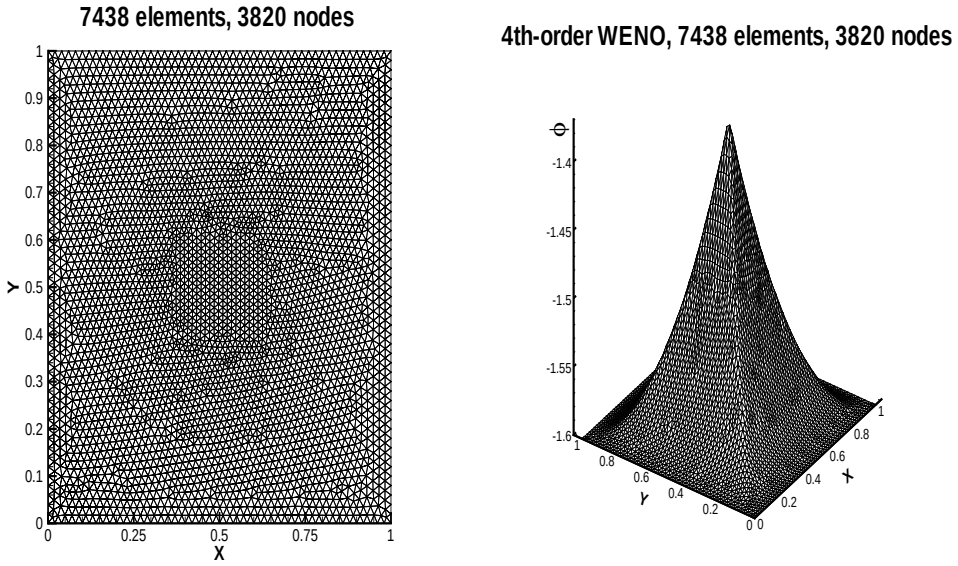


FIG. 3.12. *The 2D eikonal equation,* $H(u,v) = \sqrt{u^2+v^2+1}$, $t = 0.6$, *Left: the nonuniform mesh for the 2D eikonal equation; right: solution of the fourth-order WENO scheme.*

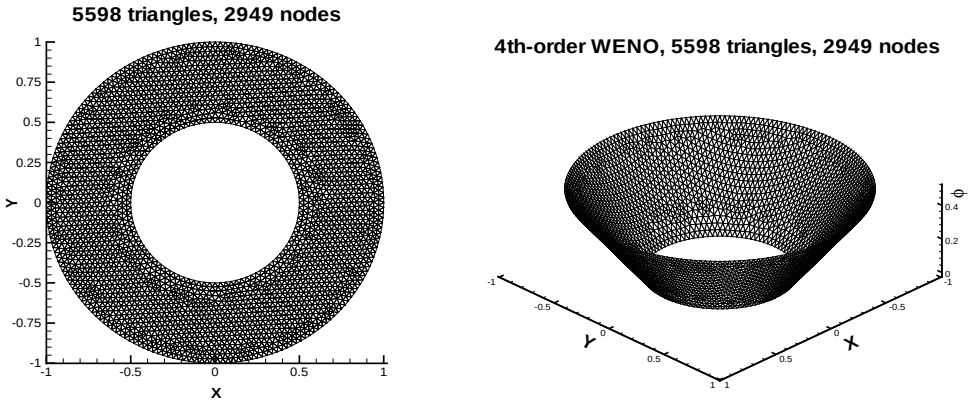


FIG. 3.13. *The level set equation. Left: the mesh; right: the steady state solution of the fourth-order WENO scheme.*

In this example, the exact steady state solution is the distance function to the inner boundary of the domain. We compute the time-dependent problem to reach a steady state solution, using the exact solution of the steady state as the boundary condition. The mesh is shown in Figure 3.13, left, and the result using the fourth-order WENO scheme is shown in Figure 3.13, right.

*Example* 3.12. The problem of a propagating surface is

$$\begin{cases} \phi_t - (1-\varepsilon K)\sqrt{1+\phi_x^2+\phi_y^2} = 0, \qquad 0 \le x < 1,\ 0 \le y < 1, \\ \phi(x,y,0) = 1 - \frac{1}{4}(\cos 2\pi x - 1)(\cos 2\pi y - 1), \end{cases} \tag{3.12}$$

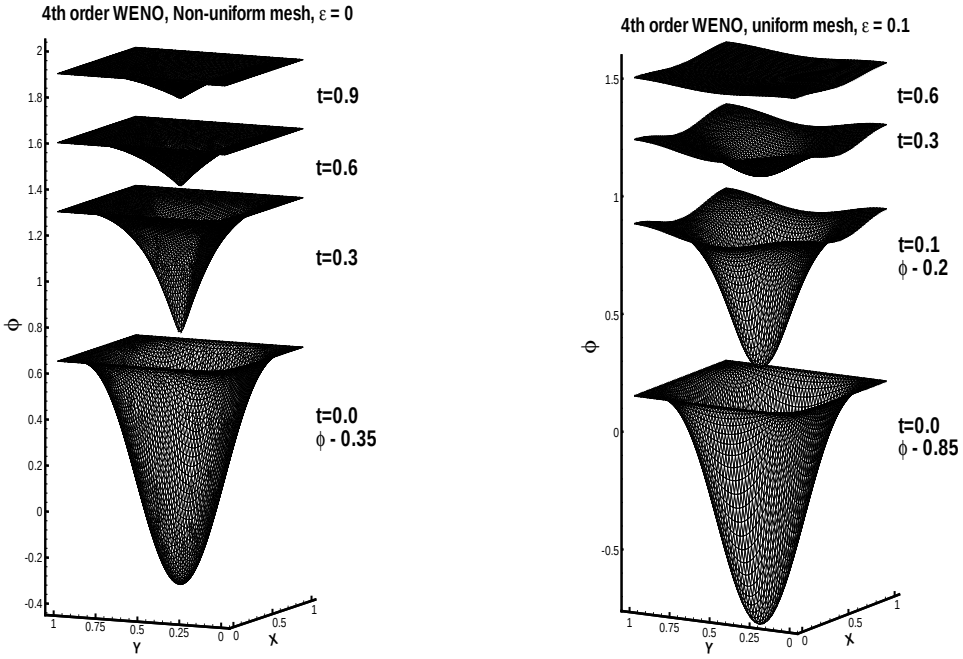


FIG. 3.14. *Propagating surfaces. Fourth-order WENO. Left:* $\varepsilon = 0$ *with a nonuniform mesh; right:* $\varepsilon = 0.1$ *with a uniform mesh.*

where $K$ is the mean curvature defined by

$$K = -\frac{\phi_{xx}(1+\phi_y^2) - 2\phi_{xy}\phi_x\phi_y + \phi_{yy}(1+\phi_x^2)}{(1+\phi_x^2+\phi_y^2)^{\frac{3}{2}}}, \tag{3.13}$$

and $\varepsilon$ is a small constant. A periodic boundary condition is used.

This problem came from [14]. We use the fourth-order WENO scheme, and the second derivative terms are approximated by those of the interpolating polynomial using the stable big stencil of our fourth-order scheme discussed in section 2. For $\varepsilon = 0$ (pure convection), whose solution will develop a discontinuous derivative the center of the domain, we use the nonuniform mesh shown in Figure 3.12, and for $\varepsilon = 0.1$, we use the uniform mesh with $50 \times 50 \times 2$ elements since the solution is smooth. The results are shown in Figure 3.14. Notice that the surfaces at $t = 0$ and $t = 0.1$ (for $\varepsilon = 0.1$) are shifted downward in order to show the detail of the solution at later time.

*Example* 3.13. A problem from computer vision is

$$\begin{cases} \phi_t + I(x,y)\sqrt{1+\phi_x^2+\phi_y^2} - 1 = 0, & -1 < x < 1, -1 < y < 1, \\ \phi(x,y,0) = 0, \end{cases} \tag{3.14}$$

where $I$ is defined by

$$I(x,y) = \frac{1}{\sqrt{1+(1-|x|)^2+(1-|y|)^2}} \tag{3.15}$$

with $\phi = 0$ as the boundary condition.

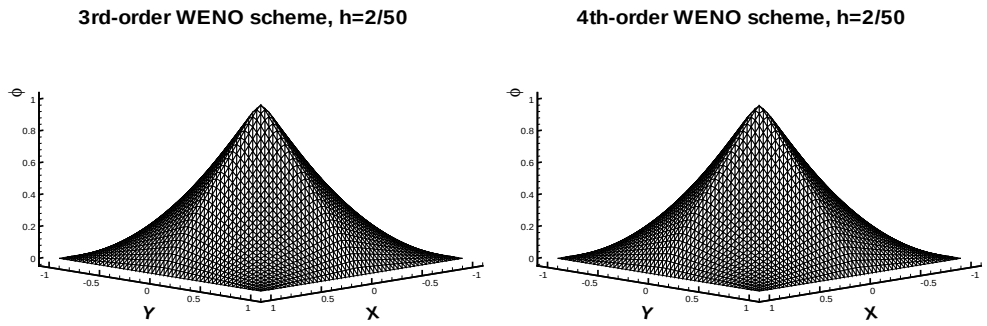


FIG. 3.15. *Computer vision problem. Left: third-order WENO; right: fourth-order WENO.*

This problem comes from [16]. The steady state solution of this problem is the shape lighted by a source located at infinity with vertical direction. The solution is not unique if there are points at which $I(x, y) = 1$. Conditions must be prescribed at one of those points where $I(x, y) = 1$. The exact steady solution is

$$\phi(x, y, \infty) = (1 - |x|)(1 - |y|). \tag{3.16}$$

We use the uniform mesh with $50 \times 50 \times 2$ elements and the third-order and fourth-order WENO schemes to compute the steady state solution. The boundary conditions are imposed exactly from the above exact steady solution. The results are shown in Figure 3.15.

**4. Concluding remarks.** We have constructed third-order and fourth-order WENO schemes for H-J equation based on 2D triangular meshes. Extensive numerical experiments have been performed to find the most robust strategies in the choice and grouping of stencils for the WENO schemes. These methods have high-order accuracy for smooth problems and high resolution for singularity of derivatives. No parameters have to be tuned in the algorithms. Numerical examples are shown to illustrate the capability of the methods. These methods should be useful if geometry or adaptivity requires unstructured meshes for the solution of H-J equations. The methods are efficient if the mesh does not change often, for example, for fixed mesh calculations or for adaptive steady state calculations via time marching.

REFERENCES

[1] R. ABGRALL, *On essentially non-oscillatory schemes on unstructured meshes: Analysis and implementation*, J. Comput. Phys., 114 (1994), pp. 45–54.

[2] R. ABGRALL, *Numerical discretization of the first-order Hamilton-Jacobi equation on triangular meshes*, Comm. Pure Appl. Math., 49 (1996), pp. 1339–1373.

[3] R. ABGRALL AND TH. SONAR, *On the use of Muehlbach expansions in the recovery step of ENO methods*, Numer. Math., 76 (1997), pp. 1–25.

[4] S. AUGOULA AND R. ABGRALL, *High order numerical discretization for Hamilton-Jacobi equations on triangular meshes*, J. Sci. Comput., 15 (2000), pp. 197–229.

[5] D. BALSARA AND C.-W. SHU, *Monotonicity preserving weighted essentially non-oscillatory schemes with increasingly high order of accuracy*, J. Comput. Phys., 160 (2000), pp. 405–452.

[6] T. Barth and J. Sethian, *Numerical schemes for the Hamilton-Jacobi and level set equations on triangulated domains*, J. Comput. Phys., 145 (1998), pp. 1–40.
[7] C. Hu and C.-W. Shu, *A discontinuous Galerkin finite element method for Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 21 (1999), pp. 666–690.
[8] C. Hu and C.-W. Shu, *Weighted essentially non-oscillatory schemes on triangular meshes*, J. Comput. Phys., 150 (1999), pp. 97–127.
[9] G.-S. Jiang and D. Peng, *Weighted ENO schemes for Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 21 (2000), pp. 2126–2143.
[10] G.-S. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, J. Comput. Phys., 126 (1996), pp. 202–228.
[11] S. Jin and Z. Xin, *Numerical passage from systems of conservation laws to Hamilton–Jacobi equations, and relaxation schemes*, SIAM J. Numer. Anal., 35 (1998), pp. 2385–2404.
[12] C.-T. Lin and E. Tadmor, *High-resolution nonoscillatory central schemes for Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 21 (2000), pp. 2163–2186.
[13] X.-D. Liu, S. Osher, and T. Chan, *Weighted essentially non-oscillatory schemes*, J. Comput. Phys., 115 (1994), pp. 200–212.
[14] S. Osher and J. Sethian, *Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.
[15] S. Osher and C.-W. Shu, *High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 28 (1991), pp. 907–922.
[16] E. Rouy and A. Tourin, *A viscosity solutions approach to shape-from-shading*, SIAM J. Numer. Anal., 29 (1992), pp. 867–884.
[17] J. Shi, C. Hu, and C.-W. Shu, *A technique of treating negative weights in WENO schemes*, J. Comput. Phys., 175 (2002), pp. 108–127.
[18] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes*, J. Comput. Phys., 77 (1988), pp. 439–471.
[19] M. Sussman, P. Smereka, and S. Osher, *A level set approach for computing solution to incompressible two-phase flow*, J. Comput. Phys., 114 (1994), pp. 146–159.