Proceedings of ALGORITMY 2016
pp. 155–164

SMOOTHNESS INDICATORS FOR WENO SCHEME USING UNDIVIDED DIFFERENCES*

TAMER H. M. A. KASEM† AND FRANÇOIS G. SCHMITT‡

Abstract. The weighted essentially non-oscillatory method (WENO) has been used widely in numerical solutions during the last two decades. This method relies on Smoothness Indicators (SI) to produce smooth solutions near discontinuities. It was concluded before that evaluating SI based on undivided differences (UD) is inefficient, and the $L2$ norm of the interpolation polynomials was used instead. In the current study the idea of using UD is revisited with the key feature of careful selection of the stencil. Improvement in terms of the accuracy and the number of arithmetic operations is illustrated by numerical simulations.

Key words. WENO, high gradient, smoothness indicator, hyperbolic system, undivided difference

AMS subject classifications. 65M06, 76N15, 35L65, 35L67, 35L60

1. Introduction. Systems of hyperbolic partial differential equations are used to model various phenomena including compressible gas flow, shallow water flow, and traffic waves [1]. The existence of discontinuities complicates the task of solving them numerically. High order numerical methods may produce spurious oscillations near discontinuities due to Gibbs phenomenon [2]. Although this problem is avoided upon using low order numerical methods, the resulting solutions suffer from extra diffusion. About two decades ago, the fifth order accurate, weighted essentially non-oscillatory method (WENO) was developed [3, 4]. WENO provides high order numerical solutions while avoiding spurious oscillations. An extensive description of WENO algorithm and its applications is provided by [2]. WENO relies on Smoothness Indicators (SI) to produce smooth solutions near discontinuities. A stencil which induces spurious oscillations is detected (hence avoided) based on its SI. The first version of WENO [3] used undivided differences (UD) to estimate SI. This was a natural choice due to the close relation between UD and the presence of discontinuities. However an improved version (WENO-JS) was introduced shortly [4]. It was proved that UD formula of [3] reduced the formal accuracy from 5th to 4th order. Instead SI were calculated in [4] using the $L2$ norm of the interpolation polynomials. The $L2$ SI formulas have been dominant during the last two decades. Major improvements for WENO which were developed later [5, 6], adopted the $L2$ SI formulas.
In order to detect discontinuities (edges) for signals and image processing applications, Archibald et al. [7] clarified the direct relation between UD and discontinuities. In the current work a new formula for SI based on UD is introduced, based on the theory developed by [7]. The new method is termed as WENO-edge. The key difference between WENO-edge and the version of [3] is the stencil used to calculate UD. In addition the 5^{th} order accuracy of WENO-edge is proved theoretically and verified numerically. Finally the advantages of WENO-edge compared to WENO-JS are clarified.

*This work was partially funded by the French institute in Cairo (Institut Français d'Egypte).
†Department of Engineering Mathematics and Physics, Faculty of Engineering, Cairo University, 12613, Egypt (tamerkasem@eng.cu.edu.eg).
‡Laboratory of Oceanology and Geosciences (LOG), 62930 Wimereux, France.

2. Methodology. A brief review of WENO is presented. The numerical algorithm for solving a scalar hyperbolic equation will be given. However extending the algorithm to hyperbolic systems is straightforward. The interested reader can refer to [2] for more details. Considering the scalar hyperbolic equation:

$$\partial u/\partial t + \partial f(u)/\partial x = 0. \tag{2.1}$$

Here u is a scalar dependent variable, governed by the non-linear flux $f(u)$, with variation in the space and time variables x and t, respectively. The space derivative term $\partial f(u)/\partial x$ can be approximated using up-winded or down-winded stencils. Only the details of the up-wind algorithm will be presented. The downwind algorithm can be deduced trivially. The task of calculation of $\partial f(u)/\partial x$ is shifted to finding the flux function $\hat{f}(\zeta)$. This function is defined implicitly as:

$$f\left(u(x)\right) = \frac{1}{\Delta x}\int_{x-\Delta x/2}^{x+\Delta x/2} \hat{f}(\zeta)d\zeta. \tag{2.2}$$

$\hat{f}(\zeta)$ can be used to write $\partial f(u)/\partial x$

$$\frac{\partial f(u)}{\partial x} = \frac{\hat{f}(x+\Delta x/2) - \hat{f}(x-\Delta x/2)}{\Delta x}. \tag{2.3}$$

A third order approximation for $\hat{f}(x - \Delta x/2)$ is termed as the numerical flux $\hat{f}^k_{i-1/2}$. The index k takes the values 0, 1 and 2, depending on the subinterval used to evaluate $\hat{f}^k_{i-1/2}$. $f(u)$ is interpolated using a Lagrange polynomial at the nodes of the subinterval $S^k = \{x_{i-3+k}, x_{i-2+k}, x_{i-1+k}\}$, and $\hat{f}^k_{i-1/2}$ is equated to the derivative of this polynomial. A third order approximation of the space derivative is written as:

$$\left.\frac{\partial f(u)}{\partial x}\right|_i = \frac{\hat{f}^k_{i+1/2} - \hat{f}^k_{i-1/2}}{\Delta x} + O(\Delta x^3). \tag{2.4}$$

Although $\hat{f}^k_{i+1/2}$ and $\hat{f}^k_{i-1/2}$ are $O(x^3)$, their difference produces a term $O(x^4)$ [2, 6]. Three choices for $\hat{f}^k_{i-1/2}$ are possible based on the value of k, i.e. the chosen stencil:

$$\hat{f}^k_{i-1/2} = \begin{cases} \hat{f}^0_{i-1/2} = \frac{1}{3}f_{i-3} - \frac{7}{6}f_{i-2} + \frac{11}{6}f_{i-1} \\ \hat{f}^1_{i-1/2} = -\frac{1}{6}f_{i-2} + \frac{5}{6}f_{i-1} + \frac{1}{3}f_i \\ \hat{f}^2_{i-1/2} = -\frac{1}{6}f_{i-2} + \frac{5}{6}f_{i-1} + \frac{1}{3}f_i \end{cases} \tag{2.5}$$

The three available stencils are combined to get a fifth order accurate approximation for the flux, using the following constants [2, 4].

$$d_0 = 0.1, d_1 = 0.6, d_2 = 0.3. \tag{2.6}$$

The fifth order accurate numerical flux $\bar{f}_{i-1/2}$ can be calculated as:

$$\bar{f}_{i-1/2} = \sum_{k=0}^{k=2} d_k \hat{f}^k_{i-1/2}. \tag{2.7}$$

In order to avoid spurious oscillations near high gradients, (2.7) is modified as:

$$\tilde{f}_{i-1/2} = \sum_{k=0}^{k=2} \omega_k \hat{f}^k_{i-1/2}. \tag{2.8}$$

The weights ω_k should be higher for stencils that yield smooth solutions and vice versa. If all stencils yield smooth solutions ω_k and $\tilde{f}_{i-1/2}$ should converge to d_k and $\bar{f}_{i-1/2}$, respectively. "Smoothness indicators" β_k are used to calculate ω_k. Calculation algorithms for β_k will be detailed on the next section. Let us assume for the moment that β_k are given. In this case the weights are calculated as:

$$\alpha_k = \frac{d_k}{(\epsilon + \beta_k)^2}. \tag{2.9}$$

$$\omega_k = \frac{\alpha_k}{\sum_{s=0}^{s=2} \alpha_s}. \tag{2.10}$$

Here ϵ is a very small arbitrary parameter (assigned to 10^{-8}) introduced to avoid division by zero, α_k is an intermediate weight used to calculate ω_k. (2.10) is adopted to enforce the condition $\sum_{k=0}^{k=2} \omega_k = 1$.

3. Smoothness Indicators. The $L2$ norm was used to calculate β_k in [4] using the following formula:

$$\beta_k^{JS} = \sum_{m=1}^{2} \int_{x_{i-1/2}}^{x_{i+1/2}} \Delta x^{2m-1} \left(q_k^m\right)^2 dx. \tag{3.1}$$

Here q_k^m is the m^{th} derivative of the polynomial used to evaluate $\hat{f}^k_{i-1/2}$. The superscript JS is used in β_k^{JS} since the formula was introduced by Jiang and Shu [4]. It can be proved that β_k^{JS} is equal to:

$$\beta_k^{JS} = \begin{cases} \beta_0^{JS} = \frac{13}{12}\left(f_{i-3} - 2f_{i-2} + f_{i-1}\right)^2 + \frac{1}{4}\left(f_{i-3} - 4f_{i-2} + 3f_{i-1}\right)^2 \\ \beta_1^{JS} = \frac{13}{12}\left(f_{i-2} - 2f_{i-1} + f_i\right)^2 + \frac{1}{4}\left(f_{i-2} - f_i\right)^2 \\ \beta_2^{JS} = \frac{13}{12}\left(f_{i-1} - 2f_i + f_{i+1}\right)^2 + \frac{1}{4}\left(3f_{i-1} - 4f_i + f_{i+1}\right)^2 \end{cases} \tag{3.2}$$

The numerical results based on (3.1) are satisfactory. However the motivation of this choice is vague. It will be shown shortly, that better results can be obtained using the undivided differences (UD). The undivided differences of order $n = 1, 2$ evaluated about the point $i - 1/2$, termed as $[f]^n_{i-1/2}$ are defined as:

$$[f]^1_{i-1/2} = f_i - f_{i-1} \tag{3.3}$$

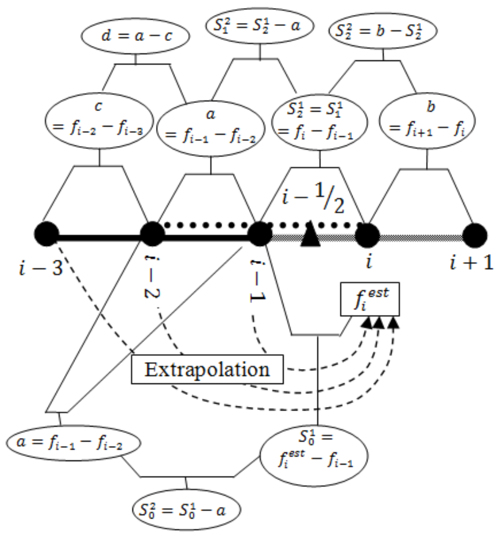

FIG. 3.1. *Undivided differences used to estimate smoothness indicators.*

$$[f]^2_{i-1/2} = \begin{cases} [f]^{2-}_{i-1/2} = [f]^1_{i-1/2} - [f]^1_{i-3/2} \\ [f]^{2+}_{i-1/2} = [f]^1_{i+1/2} - [f]^1_{i-1/2} \end{cases} \tag{3.4}$$

Although $[f]^1_{i-1/2}$ is uniquely defined, two alternatives exist for $[f]^2_{i-1/2}$; left biased $[f]^{2-}_{i-1/2}$, and right biased $[f]^{2+}_{i-1/2}$. It should be noted that $[f]^{2-}_{i-1/2} = f_i - 2f_{i-1} + f_{i-2}$ and $[f]^{2+}_{i-1/2} = f_{i+1} - 2f_i + f_{i-1}$. Referring to Fig. 3.1 the following parameters are defined using first order UDs: $a = [f]^1_{i-3/2}, b = [f]^1_{i+1/2}, c = [f]^1_{i-5/2}, S_2^1 = S_1^1 = [f]^1_{i-1/2}$, and second order UDs: $d = [f]^{2-}_{i-3/2}, S_1^2 = [f]^{2-}_{i-1/2}, S_2^2 = [f]^{2+}_{i-1/2}$. Special symbols (S_k^n) are assigned to n^{th} order UD evaluated using substencil k, about $i-1/2$. In [3] β_k^{LOC} was calculated using the summation of averages of square values of *all* the same order UDs in sub-stencil k as:

$$\beta_k^{LOC} = \begin{cases} \beta_0^{LOC} = \frac{1}{2}(a^2 + c^2) + d^2 \\ \beta_1^{LOC} = \frac{1}{2}(a^2 + \left(S_1^1\right)^2) + \left(S_1^2\right)^2 \\ \beta_2^{LOC} = \frac{1}{2}(b^2 + \left(S_2^1\right)^2) + \left(S_2^2\right)^2 \end{cases} \tag{3.5}$$

Jiang and Shu [4] compared β_k^{LOC} and β_k^{JS}. They proved that β_k^{LOC} reduced the formal accuracy from 5^{th} to 4^{th} order and concluded β_k^{JS} is better. UD are introduced here again with a key modification; β_k is based on UD evaluated about the point $i-1/2$, since this is the point at which $\tilde{f}_{i-1/2}$ is computed. The UD denoted by a, b and c, are used indirectly (via S_k^2) due to their indirect influence on $i-1/2$. Using three up-winded stencils, S_k^n can be evaluated as (Fig. 3.1):

$$S_k^1 = \begin{cases} S_0^1 = f_i^{est} - f_{i-1} \\ S_1^1 = f_i - f_{i-1} \\ S_2^1 = f_i - f_{i-1} \end{cases} \tag{3.6}$$

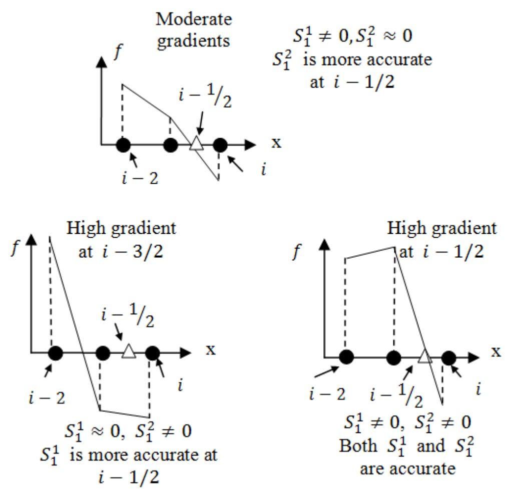
Moderate gradients
f
$i - 1/2$
x
$i - 2$
i
$S_1^1 \neq 0, S_1^2 \approx 0$
S_1^2 is more accurate at $i - 1/2$
High gradient at $i - 3/2$
f
$i - 1/2$
x
$i - 2$
i
$S_1^1 \approx 0$, $S_1^2 \neq 0$
S_1^1 is more accurate at $i - 1/2$
High gradient at $i - 1/2$
f
x
$i - 2$
$i - 1/2$
i
$S_1^1 \neq 0$, $S_1^2 \neq 0$
Both S_1^1 and S_1^2 are accurate

FIG. 3.2. *Behavior of S_1^1 and S_1^2 in the neighborhood of large gradients.*

$$S_k^2 = \begin{cases} S_0^2 = f_i^{est} - 2f_{i-1} + f_{i-2} \\ S_1^2 = f_i - 2f_{i-1} + f_{i-2} \\ S_2^2 = f_{i+1} - 2f_i + f_{i-1} \end{cases} \tag{3.7}$$

The point of interest $(i - 1/2)$ is totally outside the leftmost substencil $(k = 0)$. To deal with this situation an estimate of f_i (termed as f_i^{est}) based on extrapolation is used:

$$f_i^{est} = f_{i-3} - 3f_{i-2} + 3f_{i-1}. \tag{3.8}$$

Using (3.8), we can rewrite (3.6) and (3.7):

$$S_k^1 = \begin{cases} S_0^1 = f_{i-3} - 3f_{i-2} + 2f_{i-1} \\ S_1^1 = f_i - f_{i-1} \\ S_2^1 = f_i - f_{i-1} \end{cases} \tag{3.9}$$

$$S_k^2 = \begin{cases} S_0^2 = f_{i-3} - 2f_{i-2} + f_{i-1} \\ S_1^2 = f_i - 2f_{i-1} + f_{i-2} \\ S_2^2 = f_{i+1} - 2f_i + f_{i-1} \end{cases} \tag{3.10}$$

Archibald et al. [7] estimated jumps in f in the interval $[i-1, i]$ using UD. A jump (edge) in f is termed as $\overline{\overline{f}}_{i-1/2}$. They arrived at two conclusions (Fig. 3.2).

- $S_k^1 \approx [f]_{i-1/2}$. However S_k^1 is proportional to the local gradient at i-1/2 whether an edge exists or not.
- S_k^2 is a more accurate estimate for $[f]_{i-1/2}$. However S_k^2 produces fictitious oscillations in the neighborhood of steep gradients whether an edge exists or not.

To make benefit from both S_k^1 and S_k^2, Archibald et al. [7] calculated $\overline{\overline{f}}_{i-1/2}$ according to

$$\overline{\overline{f}}_{i-1/2} = minmod(S_k^1, S_k^2). \tag{3.11}$$

Archibald et al. [7] illustrated and proved that the *minmod* filter returns the closer estimate of $\overline{\overline{f}}_{i-1/2}$ among S_k^1 and S_k^2 and excludes the other. Unlike [7] we are interested in $\tilde{f}_{i-1/2}$ instead of $\overline{\overline{f}}_{i-1/2}$. The sensitivity of S_k^2 to neighbour high gradients (present in S_k^2) should never be excluded. In addition the magnitude of local gradient at $i - 1/2$ (present in S_k^1) is always important. Instead of the minmod filter, the following formula is introduced for β_k:

$$\beta_k^{edge} = abs(S_k^1) + abs(S_k^2). \tag{3.12}$$

The superscript *edge* is used in β_k^{edge} since the idea is based on the edge detection algorithm presented by Archibald et al. [7]. The *abs* function is used since the sign of $S_k^{1,2}$ is irrelevant.

Other forms of (3.12) are possible. For example $\beta_k = W_1[abs(S_k^1)]^h + W_2[abs(S_k^2)]^h$, where $h > 0$ is a suitable exponent, and $W_{1,2}$ are suitable weights. However, numerical experiments illustrated that (3.12) yields good results. In order to clarify the analogy with β_k^{JS}, (3.12) is expanded and rewritten as:

$$\beta_k^{edge} = \begin{cases} \beta_0^{edge} = abs(f_{i-3} - 3f_{i-2} + 2f_{i-1}) + abs(f_{i-3} - 2f_{i-2} + f_{i-1}) \\ \beta_1^{edge} = abs(f_i - f_{i-1}) + abs(f_i - 2f_{i-1} + f_{i-2}) \\ \beta_2^{edge} = abs(f_i - f_{i-1}) + abs(f_{i+1} - 2f_i + f_{i-1}) \end{cases} \tag{3.13}$$

Calculating β_k^{edge} consumes less computer time than the original β_k^{JS}. The arithmetic operations for β_k^{JS} includes four extra multiplications (squaring brackets and multiplication by 13/12 and 1/4 factors). On the other hand, the *abs* operation needed by β_k^{edge} consumes negligible time.

3.1. Formal Order of accuracy. The following necessary and sufficient conditions to retain the 5^{th} order accuracy were provided and derived by [6]:

$$3\omega_0^+ - 3\omega_0^- - \omega_1^+ + \omega_1^- + \omega_2^+ - \omega_2^- = O(\Delta x^3). \tag{3.14}$$

$$\omega_k - d_k = O(\Delta x^2). \tag{3.15}$$

Here the superscripts (+) or (-) on ω_k indicate their use in $\tilde{f}_{i+1/2}$ or $\tilde{f}_{i-1/2}$, respectively. (3.14) and (3.15) are checked by substituting β_k^{edge} in (2.9), then Taylor series expansion is used. Performing this task manually requires huge time. The results are obtained using the free symbolic manipulator SAGE available online (https://cloud.sagemath.com).

$$3\omega_0^+ - 3\omega_0^- - \omega_1^+ + \omega_1^- + \omega_2^+ - \omega_2^- = -\frac{3}{5}\frac{f''_{i-1/2}f'''_{i-1/2}}{\left(f'_{i-1/2}\right)^2}\Delta x^3 + H.O.T. \tag{3.16}$$

$$\omega_k - d_k = \begin{cases} \omega_0 - d_0 = \frac{21}{50}\frac{f'''_{i-1/2}}{f'_{i-1/2}}\left(\Delta x^2\right) + H.O.T \\ \omega_1 - d_1 = \frac{6}{50}\frac{f'''_{i-1/2}}{f'_{i-1/2}}\left(\Delta x^2\right) + H.O.T \\ \omega_2 - d_2 = -\frac{27}{50}\frac{f'''_{i-1/2}}{f'_{i-1/2}}\left(\Delta x^2\right) + H.O.T \end{cases} \tag{3.17}$$

It is clear that β_k^{edge} satisfies all the formal accuracy conditions.

4. Results and discussion. In order to illustrate the advantage of the new scheme, numerical solutions of WENO will be presented. The solutions based on (3.1) will be termed as WENO-JS . The results based on (3.12) will be termed as WENO-edge. For all cases explicit 3^{rd} order TVD time integration is adopted [2].

4.1. Passive convection. In this section, WENO-JS and WENO-edge are applied to the linear advection equation $\partial u/\partial t = -\partial u/\partial x$. The signal propagation speed is a positive constant equal to 1, implying that information always comes from the left. Consequently the up-winded algorithm is adopted. A more general problem is discussed in §4.2. For a linear advection problem the exact solution is available in terms of the initial condition $u(x, t = 0) = F(x)$. The exact solution is given as $u_{exact} = F(x - t)$.

4.1.1. Smooth sine. The algorithms are applied to transport of a smooth continuous function. The initial condition is given by $u(x, t = 0) = sin(\pi x)$. The equation is solved until the final time $t = 2$ on the space interval $x \in [-1, 1]$, with periodic space boundary conditions. Various grids are adopted to investigate convergence. The number of grid points N is assigned $N = 10 \times 2^i$ where i takes the integer values between 0 and 6. Since the time integration method produces $O(\Delta t^3)$ errors, the numerical time step is equated to $\Delta t = 2(\Delta x^{5/3})$ in order that the overall error of the scheme is a measure of the spatial convergence only [6, 5]. For each grid the $L1$ norm of the error is calculated as [8]:

$$L_1(N) = \frac{\sum_{i=1}^{N} abs\left(u_i^{exact} - u_i^{WENO}\right)}{N}, \tag{4.1}$$

The order of the numerical error (p) can be estimated using $L1(N)$ computed from two consecutive grids. This is done based on the assumption $L1 = aN^{-p}$. The ratio between consecutive grids is assigned to 2.0 in the current work. Hence the order is estimated as

$$p = log_2\left(\frac{L1(N)}{L1(2N)}\right). \tag{4.2}$$

The results are shown in table 4.1. Both versions (WENO-edge and WENO-JS) produce 5^{th} order accurate solutions as expected. However the error of WENO-edge is considerably smaller than that of WENO-JS. In fact the error of WENO-edge is at least 15% lower than WENO-JS. The computation times in milli seconds are shown on the same table (Intel Core i3 M330 2.13GHz processor, 3.67 GB memory). It should be noted that the measured times varied randomly by $\pm 15ms$ for each run. This random variation may be attributed to the various operating system tasks. In addition the developed codes have not been optimized for speed. However the results show that the WENO-edge is faster (at least not slower) than WENO-JS.

Table 4.1

L1 norm, estimated order of discretization error p and computation time for a smooth initial condition $u(x, t = 0) = sin(\pi x)$. Values are shown for various grids.

N	WENO JS			WENO edge		
	Error $L1$	p	computation time ms	Error $L1$	p	computation time ms
10	3.67E-02	-	16	2.72E-02	-	15
20	1.80E-03	4.348927	16	1.44E-03	4.236559	0
40	5.64E-05	4.998263	16	4.96E-05	4.86288	15
80	1.78E-06	4.987103	46	1.57E-06	4.979707	47
160	5.59E-08	4.992303	187	4.93E-08	4.99632	172
320	1.75E-09	4.995802	1138	1.54E-09	5.000627	1061
640	5.52E-11	4.988147	6692	4.79E-11	5.005998	6630

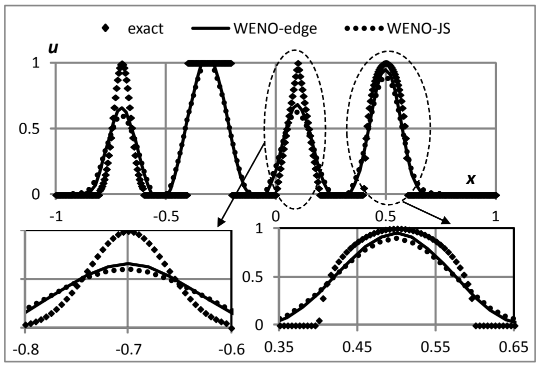

Fig. 4.1. *Numerical and exact solutions of the passive convection problem ($\Delta x = 1/50$) at $t = 8$. Magnified plots of the triangular and semi-ellipse waves are shown on the same figure.*

4.1.2. Discontinuous waves. WENO-JS and WENO-edge are applied to transport of discontinuous functions extracted from [5]. The advection equation is solved until the final time $t = 8$ on the interval $x \in [-1, 1]$, with periodic space boundary conditions. The numerical time step is assigned to $\Delta t = 8(\Delta x^{5/3})$. The initial condition consists of; a Gaussian function, square wave, triangular wave and a semi ellipse. Simulation results are shown in Fig. 4.1 for grid spacing $\Delta x = 1/50$. For this relatively coarse mesh the advantage of WENO-edge is clear. To clarify the improvement a finer mesh is adopted; $\Delta x = 1/200$ in Fig. 4.2. The solution is magnified near the edges of the square and the Gaussian waves to clarify the advantage of WENO-edge. Convergence is further clarified by calculating errors for various grids (table 4.2). The rate of convergence drops to be close to first order. This is common among high order methods in the presence of discontinuities [6]. The error of WENO-edge is at least 15% lower than WENO-JS for this case also.

4.2. Euler system. The results of WENO-edge and WENO-JS are presented for Euler non-linear system of equations, which describes one-dimensional compressible flow. Euler Equations can be written in the following compact form:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} = \mathbf{0} \tag{4.3}$$

Where the vectors $\mathbf{U}$ and $\mathbf{F}(\mathbf{U})$ are the vectors of conserved variables and flux,

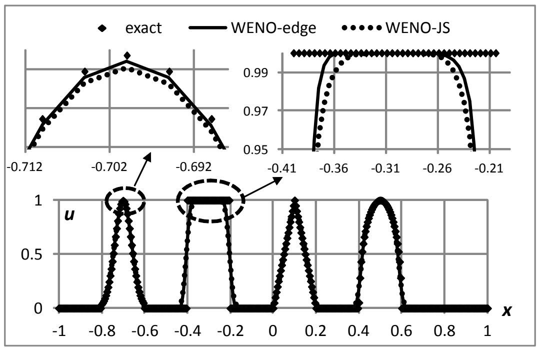

FIG. 4.2. *Numerical and exact solutions of the passive convection problem ($\Delta x = 1/200$) at $t = 8$.*

TABLE 4.2
Error $L1$ norm and the estimated order of error p for a non-smooth initial condition (final time $t = 2$). Values are shown for various grids.

N	WENO JS		WENO edge	
	Error $L1$	p	Error $L1$	p
80	9.29E-02	-	7.73E-02	-
160	3.71E-02	1.322884	3.12E-02	1.307421
320	1.81E-02	1.034656	1.53E-02	1.029718
640	9.12E-03	0.989733	7.78E-03	0.976826

respectively. They are defined as

$$\mathbf{U} = [\rho\ \rho u E]^T = [U_1\ U_2\ U_3]^T \tag{4.4}$$

$$\mathbf{F(U)} = \left[\rho u\ \rho u^2 + P\ (E+P)u\right]^T = [F_1\ F_2\ F_3]^T \tag{4.5}$$

Here ρ is the fluid density, $E = \rho(C_vT + u^2/2)$ is the total energy defined in terms of the temperature T, the specific heat capacity C_v, and the velocity u. In addition the pressure P appears in the flux of ρu and E. The procedure of applying WENO to systems of hyperbolic equations is fully described in [2]. Briefly, the Jacobian of the flux is used to transform the coupled system of (4.3) to a diagonal uncoupled form. The uncoupled system is obtained by a characteristic transformation using the eigen vectors of the Jacobian. Lax-Friedrichs flux-splitting is applied to account for signals coming from left and right directions.

Riemann Problems are useful for the development and testing of numerical algorithms for the one-dimensional Euler equations [9]. Initially, two different states of the flow exist. For $x < 0.5$, the flow variables ρ_1, u_1, P_1 are imposed, while ρ_4, u_4, P_4 are imposed for $x > 0.5$. A Riemann problem contains the fundamental physical and mathematical character of Euler equations, in spite of its simple initial condition [10].

The following initial condition is specified:$(\rho_1, u_1, P_1) = (1.0, 0, 1.0)$,$(\rho_4, u_4, P_4) = (0.125, 0, 0.1)$. This problem is often termed as Sod's problem in the literature [4]. The values of grid spacing and time step are assigned to $\Delta x = 0.1$ and $\Delta t = 0.012$, respectively. The exact solution (left rarefaction, a contact discontinuity and a right shock) is detailed in [9, 10]. The numerical and exact solutions for the density are shown in Fig. 4.3 at the instant $t = 1.2$. To illustrate improvement, the plot is

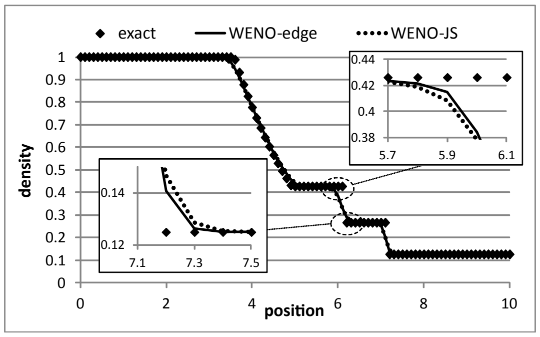

FIG. 4.3. *Sod problem numerical and exact density solutions at $t = 1.2$.*

magnified at two corners near the sharp gradient of the contact discontinuity. The results of WENO-edge are closer than WENO-JS to the exact solution as shown.

5. Conclusions. A new algorithm based on undivided differences (UD) for computing smoothness indicators used in WENO method is presented. The novel idea of computing the UD about the flux point is introduced. The new smoothness indicator formulas (β_k^{edge}) consume fewer arithmetic operations. The fifth order accuracy of β_k^{edge} is verified theoretically and numerically. They also yield better results, compared to the original formula based on the $L2$ norms of the derivatives (${\beta_k}^{JS}$). Improvement was illustrated for scalar advection of smooth and sharp waves and a Riemann problem of compressible flow. One possible extension of the current work is implementing the improvements described in [5, 6] using β_k^{edge} instead of β_k^{JS}.

6. Acknowledgements. The author thanks the anonymous reviewer whose comments largely improved the quality of the paper.

REFERENCES

[1] G. B. WHITHAM *Linear and nonlinear waves* John Wiley and sons, 1974.
[2] C. W. SHU, *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws*, Lect. Notes Math., 1697 (1998), pp. 325–432.
[3] X.-D. LIU, S. OSHER, AND T. CHAN, *Weighted Essentially Non-oscillatory Schemes*, J. Comput. Phys., 115 (1994), pp. 200–212.
[4] G.-S. JIANG, C.-W. SHU, *Efficient Implementation of Weighted ENO Schemes*, J. Comput. Phys., 126 (1996), pp. 202–228.
[5] R. BORGES, M. CARMONA, B. COSTA, AND W.S. DON, *An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws*, J. Comput. Phys., 227 (2008), pp. 3191-3211.
[6] A.K. HENRICK, T.D. ASLAM, AND J.M. POWERS, *Mapped weighted essentially non-oscillatory schemes: Achieving optimal order near critical points*, J. Comput. Phys., 207 (2005), pp. 542-567.
[7] R. ARCHIBALD, A. GELB, AND J. YOON, *Polynomial Fitting for Edge Detection in Irregularly Sampled Signals and Images*, SIAM J. Numer. Anal., 43 (2005), pp. 259–279.
[8] JOHN A. TRANGENSTEIN, *Numerical Solution of Hyperbolic Partial Differential Equations*, Cambridge University Press, 2009.
[9] D.D. KNIGHT, *Elements of Numerical Methods for Compressible Flows*, Cambridge University Press, 2006.
[10] E.F. TORO, *Riemann Solvers and Numerical Methods for Fluid Dynamics*, Springer, 2009.