**TBD manuscript No.**
(will be inserted by the editor)

# An Efficient and Accurate Approximation to the Classic Polynomial Smoothness Indicator

**Todd Arbogast · Chieh-Sen Huang · Chenyu Tian · Gabriel M. Gray**

Received: date / Accepted: date

**Abstract** For a stencil polynomial in any number of dimensions, its classic smoothness indicator $\sigma_{\mathrm{JS}}$ can be expressed as a double sum over the coefficients. A single sum approximation is presented leading to the polynomial coefficient squared smoothness indicator $\sigma_{\mathrm{P}}$. It is proven to satisfy the appropriate asymptotic behavior for a smoothness indicator. Theoretical estimates of the computational costs are given for implementations appropriate for both explicit and implicit timestepping. Computational tests involving single stencils show that $\sigma_{\mathrm{P}}$ is a remarkably accurate approximation of $\sigma_{\mathrm{JS}}$, especially for finer stencil meshes, and much more efficient to compute. Application to solving hyperbolic conservation laws show solutions using $\sigma_{\mathrm{JS}}$ and $\sigma_{\mathrm{P}}$ that are very comparable, with the latter being significantly more efficient to compute.

**Keywords** polynomial coefficient squared smoothness indicator · weighted essentially non-oscillatory (WENO) · indicator of smoothness (IS) · finite volume · hyperbolic conservation law

**Mathematics Subject Classification (2010)** 65D15 · 65M08 · 76M12

The first author was funded in part by the U.S. National Science Foundation grant DMS-1912735. The second author was funded by the Taiwan Ministry of Science and Technology grant MOST 109-2115-M-110-003-MY3 and the National Center for Theoretical Sciences, Taiwan.

T. Arbogast
Department of Mathematics C1200; University of Texas; Austin, TX 78712–1202; USA and Oden Institute for Computational Engineering and Sciences C0200; University of Texas; Austin, TX 78712–1229; USA E-mail: arbogast@oden.utexas.edu

C.-S. Huang
Department of Applied Mathematics, National Sun Yat-sen University, Kaohsiung 804, Taiwan, R.O.C. E-mail: huangcs@math.nsysu.edu.tw

Chenyu Tian
Oden Institute for Computational Engineering and Sciences C0200; University of Texas; Austin, TX 78712–1229; USA E-mail: chenyu@ices.utexas.edu

Gabriel M. Gray
University of Texas; Austin, TX 78712; USA E-mail: gabriel.m.gray@gmail.com

## 1 Introduction

Weighted essentially non-oscillatory (WENO) schemes [17,30,27,29] are often used to solve hyperbolic conservation laws and advection-diffusion equations [28,1,6]. Finite difference and finite volume WENO schemes rely on WENO reconstructions of the discrete solution $u(\mathbf{x})$. These reconstructions are a weighted sum of stencil polynomials $P_\ell(\mathbf{x})$ for $\ell$ in some index set, each of which approximates $u(\mathbf{x})$. The goal of the weighting is to effectively remove the polynomials that are defined on stencils that contain shocks or steep fronts in the solution.

There are several ways to define the (nonlinear) weights in the reconstruction (see, e.g., [20,29,9,5]), but all of them require a good measure of the smoothness of the stencil polynomial. This *indicator of smoothness* (IS), or more simply *smoothness indicator*, will be denoted as $\sigma$. In particular, if the size of the stencil elements or cells is $h>0$, then as $h\to 0^+$, there must be some $D\geq 0$ such that

$$\sigma=\begin{cases} Dh^2+\mathscr{O}(h^3) & \text{if } u \text{ is smooth on the stencil,}\\ \mathscr{O}(1) & \text{if } u \text{ has a jump discontinuity on the stencil.}\end{cases} \tag{1.1}$$

Moreover, $\sigma\geq 0$ is required to use it effectively in practice.

We remark that we actually want $\sigma=\Theta(1)$ in the case that $u$ has a jump discontinuity (see [5] for more details). Fortunately, the difference between $\sigma=\mathscr{O}(1)$ and $\sigma=\Theta(1)$ does not cause problems when solving conservation laws, since $\sigma$ not being order 1 near a jump discontinuity may occur only occasionally in space and time. It is common practice in the literature to justify (1.1) only in the case that $u$ is smooth.

The most common and perhaps the most reliable smoothness indicator is the classic one defined by Jiang and Shu [20], $\sigma_{\text{JS}}$, and generalized to multiple dimensions by O. Friedrich [14]. It does a good job of identifying whether or not a shock appears in the solution over the stencil. This property is critical to the success of WENO schemes.

The evaluation of $\sigma_{\text{JS}}$ is a significant computational cost in dimensions greater than one over general stencils, and especially for implicit WENO schemes. This has motivated recent research on simplified smoothness indicators that do not require derivatives of $P_\ell$ [18,23,7,19]. Simplified smoothness indicators generally do not preserve all the good properties of the classic $\sigma_{\text{JS}}$.

In this paper we propose an *approximation* of the classic smoothness indicator, denoted $\sigma_{\text{P}}$. It is a special weighted sum of the squares of the stencil polynomial coefficients, and so we say that it is the *polynomial coefficients squared* smoothness indicator. The idea of using a weighted sum of the squares of the stencil polynomial coefficients first appeared in [7]. However, in that work, a simple weighting was used. Here, we give a more involved weighting (see (4.1)) defined so as to make $\sigma_{\text{P}}$ a true approximation of $\sigma_{\text{JS}}$.

To preview the results, $\bar{u}_E$ will denote the average of the solution $u(\mathbf{x})$ over the mesh element $E$ and, for the multi-index $\alpha$, $c_\alpha$ will denote the $\alpha$th coefficient of the stencil polynomial $P(\mathbf{x})$. We will show that one can compute $\sigma_{\text{JS}}$ by either of two

formulas, at least the first of which is commonly used (see [3]). These are

$$\sigma_{\text{JS}} = \sum_E \sum_F \sigma_{E,F} \bar{u}_E \bar{u}_F = \sum_\alpha \sum_\beta \eta_{\alpha,\beta} c_\alpha c_\beta, \tag{1.2}$$

where each sum in the first formula is over the elements in the stencil, and in the second formula over the polynomial coefficient multi-indices. Since the number of elements in the stencil is greater than (or at best equal to) the number of coefficients, the later formula turns out to be more efficient. The new smoothness indicator will be computed by a sum of the form

$$\sigma_{\text{P}} = \sum_\alpha \tilde{\eta}_\alpha c_\alpha^2, \tag{1.3}$$

where $\tilde{\eta}_\alpha = \eta_{\alpha,\alpha}$. The use of a single sum over the multi-indices makes it more efficient.

In practice, the computational efficiency of computing the smoothness indicator depends on the needs of the overall WENO scheme, allowing for the possibility of reusing some computations. In addition to evaluation of the smoothness indicator, one always computes $P(\mathbf{x})$ at specific points. If implicit methods are used, solution of a nonlinear problem by Newton's method also requires computation of the Jacobian terms, i.e., the derivatives $\partial P/\partial \bar{\mathbf{u}}$ and $\partial \sigma/\partial \bar{\mathbf{u}}$. We will discuss the computational efficiency of computing the smoothness indicators in this broader context.

In the next section, Section 2, we fix notation and discuss finite volume stencil polynomials arising from general meshes in multiple dimensions. The classic smoothness indicator is discussed in Section 3. The new polynomial coefficient squared smoothness indicator is developed and analyzed in Section 4. Section 5 compares the costs of computing the smoothness indicators and Jacobian derivative terms. Section 6 is devoted to a numerical comparison of the classic and polynomial coefficient squared smoothness indicators in two dimensions, where it is shown that the approximation is accurate and computationally efficient (at least for the chosen test problems). Application of the smoothness indicators to the solution of conservation laws exhibiting hyperbolic behavior in two space dimensions is given in Section 7, where the computational efficiency and accuracy is discussed for three test problems. We close the paper with a summary of our results and conclusions in Section 8.

## 2 Finite Volume Stencil Polynomials

We follow the notation used in [4]. Let $d$ be the dimension of space and let a stencil $S = \{E_0, E_1, \ldots, E_{N_S-1}\}$ be a set of $N_S$ nonoverlapping (except on their boundaries) and contiguous mesh elements $E_k$. The stencil domain is $\Omega_S = \bigcup_{E\in S} E \subset \mathbb{R}^d$, $h_E = \text{diam}(E)$, and $h = h_S = \max\{h_E : E \in S\}$. We tacitly assume that the stencil is quasiuniform, so all stencil elements are of comparable size.

In finite volume schemes, the data available at the start of the time step, or at the start of the current Newton iteration, are the element averages of the solution $u(\mathbf{x})$ over the stencil, which are

$$\bar{u}_E = \frac{1}{|E|} \int_E u(\mathbf{x})\, d\mathbf{x} \quad \forall E \in S. \tag{2.1}$$

Denote the $N_S$-vector $\bar{\mathbf{u}} = (\bar{u}_E)$.

Let $\mathbb{P}_r$ denote the set of polynomials of degree up to $r$, and represent the stencil polynomial $P(\mathbf{x}) \in \mathbb{P}_r$ as

$$P(\mathbf{x}) = \sum_{|\alpha|\le r} c_\alpha \left( \frac{\mathbf{x} - \mathbf{x}_S}{h_S} \right)^\alpha, \tag{2.2}$$

where $\mathbf{x}_S$ is the center of $\Omega_S$ and $\alpha$ is a $d$-dimensional multi-index. The stencil polynomial $P(\mathbf{x})$ is supposed to satisfy

$$\frac{1}{|E|} \int_E P(\mathbf{x})\, d\mathbf{x} = \bar{u}_E, \quad \text{for each } E \in S. \tag{2.3}$$

In one dimension, (2.3) can be set directly by choosing $r+1 = N_S$. However, in multiple dimensions, the number of polynomial coefficients $N_C = \binom{r+d}{d}$ will not normally agree with the number of stencil elements $N_S$, and so a least-squares polynomial fitting is (usually) required [26]. Since we want to evaluate $P(\mathbf{x})$ on a target element, say $E_0$, we use a constrained least squares fitting with the constraint that (2.3) holds for the target element. As discussed in [4], one can compute the polynomial coefficients, written as an $N_C$-vector $\mathbf{c} = (c_\alpha)$, as a linear transformation of $\bar{\mathbf{u}}$. That is,

$$\mathbf{c} = A\bar{\mathbf{u}}, \tag{2.4}$$

where the matrix $A$ is $N_C \times N_S$.

There is an alternative to evaluation of $P(\mathbf{x})$ using (2.2) and (2.4). It involves a special basis for the stencil polynomials [3,6]. For the given stencil $S$ and target element $E_0$, let $P_E(\mathbf{x})$ be the base polynomial of degree $r$ that is the constrained least squares solution of (2.3) modified so that

$$\frac{1}{|F|} \int_F P_E(\mathbf{x})\, d\mathbf{x} = \begin{cases} 1 & \text{if } F = E, \\ 0 & \text{otherwise.} \end{cases} \tag{2.5}$$

The stencil polynomial is then evaluated simply as

$$P(\mathbf{x}) = \sum_{E\in S} P_E(\mathbf{x})\, \bar{u}_E. \tag{2.6}$$

Under reasonable hypotheses [4], the stencil polynomial $P \in \mathbb{P}_r$ approximates $u(\mathbf{x})$ for all $\mathbf{x} \in E_0 \in S$ as

$$|\mathscr{D}^\alpha u(\mathbf{x}) - \mathscr{D}^\alpha P(\mathbf{x})| \le \begin{cases} C h^{r+1-|\alpha|} & \text{if } u \text{ is smooth on the stencil,} \\ C & \text{otherwise,} \end{cases} \tag{2.7}$$

for some constant $C > 0$ as $h \to 0^+$, where $\mathscr{D}^\alpha$ is partial differentiation of order $\alpha$, $|\alpha| \le r+1$.

## 3 The Classic Smoothness Indicator $\sigma_{\mathrm{JS}}$

The classic smoothness indicator [20, 14] is

$$\sigma_{\mathrm{JS}} = \sum_{1\le|\alpha|\le r} \frac{h_0^{2|\alpha|}}{|E_0|}\int_{E_0} |\mathscr{D}^\alpha P(\mathbf{x})|^2\, d\mathbf{x}, \tag{3.1}$$

where $h_0 = h_{E_0} = \mathrm{diam}(E_0)$.

When $u$ is smooth on the stencil, the estimate $\sigma_{\mathrm{JS}} = Dh^2 + \mathscr{O}(h^3)$ in (1.1) is known to hold in one dimension [20], where $D \ge 0$ is related to the derivative of $u$. The result is also known in multiple dimensions, but to be complete and precise, we provide the following lemma.

**Lemma 3.1** *If $u$ is approximated on the target element $E_0$ over the stencil $S$ by the polynomial $P \in \mathbb{P}_r$, then there is some constant $D \ge 0$ such that as $h = h_S \to 0^+$,*

$$\sigma_{JS} = Dh_0^2 + \mathscr{O}(h^3), \quad \textit{if } u \textit{ is smooth on the stencil.} \tag{3.2}$$

*Moreover, when $r \ge 1$,*

$$D = \frac{1}{|E_0|}\int_{E_0} |\nabla u(\mathbf{x})|^2\, d\mathbf{x} \ge 0. \tag{3.3}$$

*Proof* The result is trivial when $r = 0$, since $\sigma_{\mathrm{JS}} = 0$ and we can take $D = 0$. For $r \ge 1$, we start from (3.1) and note that

$$\sigma_{\mathrm{JS}} = \frac{h_0^2}{|E_0|}\int_{E_0} |\nabla P(\mathbf{x})|^2\, d\mathbf{x} + \sum_{2\le|\alpha|\le r} \frac{h_0^{2|\alpha|}}{|E_0|}\int_{E_0} |\mathscr{D}^\alpha P(\mathbf{x})|^2\, d\mathbf{x}.$$

By the trivial inequality $P^2 \le 2((u-P)^2 + u^2)$, the latter sum is

$$\begin{aligned}
&\sum_{2\le|\alpha|\le r} \frac{h_0^{2|\alpha|}}{|E_0|}\int_{E_0} |\mathscr{D}^\alpha P(\mathbf{x})|^2\, d\mathbf{x} \\
&\quad\le 2\sum_{2\le|\alpha|\le r} \frac{h_0^{2|\alpha|}}{|E_0|}\int_{E_0} \left(|\mathscr{D}^\alpha(u(\mathbf{x}) - P(\mathbf{x}))|^2 + |\mathscr{D}^\alpha u(\mathbf{x})|^2\right) d\mathbf{x} \\
&\quad\le C\sum_{2\le|\alpha|\le r} h_0^{2|\alpha|}\left(h^{2(r+1-|\alpha|)} + 1\right) \\
&\quad\le Ch^4,
\end{aligned}$$

wherein we used (2.7). Moreover,

$$\begin{aligned}
&\frac{h_0^2}{|E_0|}\int_{E_0} |\nabla P(\mathbf{x})|^2\, d\mathbf{x} \\
&\quad= \frac{h_0^2}{|E_0|}\left(\int_{E_0} |\nabla u(\mathbf{x})|^2\, d\mathbf{x} + \int_{E_0} \left(\nabla P(\mathbf{x}) + \nabla u(\mathbf{x})\right)\cdot\left(\nabla P(\mathbf{x}) - \nabla u(\mathbf{x})\right) d\mathbf{x}\right) \\
&\quad= Dh_0^2 + \frac{h_0^2}{|E_0|}\int_{E_0} \left(\nabla P(\mathbf{x}) - \nabla u(\mathbf{x}) + 2\nabla u(\mathbf{x})\right)\cdot\left(\nabla P(\mathbf{x}) - \nabla u(\mathbf{x})\right) d\mathbf{x}.
\end{aligned}$$

Finally, the last term above, using (2.7) again, is bounded by

$$\frac{h_0^2}{|E_0|}\int_{E_0}\big(|\nabla P(\mathbf{x})-\nabla u(\mathbf{x})|+2|\nabla u(\mathbf{x})|\big)|\nabla P(\mathbf{x})-\nabla u(\mathbf{x})|\,d\mathbf{x}$$
$$\le Ch_0^2\big[(h_0^r+1)\,h_0^r\big]\le Ch_0^{r+2}\le Ch^3,$$

since $r\ge 1$. The result follows. □

The classic smoothness indicator can be implemented using either the representation of $P(\mathbf{x})$ from base polynomials or directly after computing the polynomial coefficients.

### 3.1 Implementation using base polynomials

It is natural to compute $\sigma_{\text{JS}}$ directly from $\bar{\mathbf{u}}$. An efficient implementation that is commonly used involves the special base polynomials (2.5) [3,6]. The smoothness indicator (3.1) can be computed using (2.6) as

$$\begin{aligned}\sigma_{\text{JS}} &= \sum_{1\le|\alpha|\le r}\frac{h_0^{2|\alpha|}}{|E_0|}\int_{E_0}\Big(\sum_{E\in S}\bar{u}_E\mathscr{D}^\alpha P_E(\mathbf{x})\Big)^2 d\mathbf{x}\\ &= \sum_{1\le|\alpha|\le r}\frac{h_0^{2|\alpha|}}{|E_0|}\int_{E_0}\sum_{E_1\in S}\sum_{E_2\in S}\bar{u}_{E_1}\bar{u}_{E_2}\mathscr{D}^\alpha P_{E_1}(\mathbf{x})\mathscr{D}^\alpha P_{E_2}(\mathbf{x})\,d\mathbf{x}\\ &= \sum_{E_1\in S}\sum_{E_2\in S}\Bigg(\sum_{1\le|\alpha|\le r}\frac{h_0^{2|\alpha|}}{|E_0|}\int_{E_0}\mathscr{D}^\alpha P_{E_1}(\mathbf{x})\mathscr{D}^\alpha P_{E_2}(\mathbf{x})\,d\mathbf{x}\Bigg)\bar{u}_{E_1}\bar{u}_{E_2}\\ &= \sum_{E_1\in S}\sum_{E_2\in S}\sigma_{E_1,E_2}\bar{u}_{E_1}\bar{u}_{E_2},\end{aligned}\tag{3.4}$$

where

$$\sigma_{E_1,E_2}=\sum_{1\le|\alpha|\le r}\frac{h_0^{2|\alpha|}}{|E_0|}\int_{E_0}\mathscr{D}^\alpha P_{E_1}(\mathbf{x})\mathscr{D}^\alpha P_{E_2}(\mathbf{x})\,d\mathbf{x}\tag{3.5}$$

is independent of the data $\bar{\mathbf{u}}$. It can be precomputed and reused for multiple reconstructions as $\bar{\mathbf{u}}$ changes.

In matrix form, $\sigma=(\sigma_{E_1,E_2})$ is a symmetric, positive semidefinite $N_S\times N_S$ matrix, and

$$\sigma_{\text{JS}}=\bar{\mathbf{u}}^T\sigma\bar{\mathbf{u}}.\tag{3.6}$$

Symmetry shows that

$$\sigma_{\text{JS}}=\sum_{i=0}^{N_S-1}\sum_{j=0}^{N_S-1}\sigma_{E_i,E_j}\bar{u}_{E_i}\bar{u}_{E_j}=\sum_{i=0}^{N_S-1}\sigma_{E_i,E_i}\bar{u}_{E_i}^2+2\sum_{i=1}^{N_S-1}\sum_{j=0}^{i-1}\sigma_{E_i,E_j}\bar{u}_{E_i}\bar{u}_{E_j}.\tag{3.7}$$

### 3.2 Implementation using the polynomial coefficients

Starting from $\bar{\mathbf{u}}$, recall from (2.4) that $\mathbf{c} = A\bar{\mathbf{u}}$, where the matrix $A$ is $N_C \times N_S$. Once one has computed the polynomial coefficients, the smoothness indicator can be easily computed. A combination of (3.1) and (2.2) gives

$$\begin{aligned}
\sigma_{\text{JS}} &= \sum_{1\le|\gamma|\le r} \frac{h_0^{2|\gamma|}}{|E_0|} \int_{E_0} \left( \sum_{|\alpha|\le r} c_\alpha \mathscr{D}^\gamma \left( \frac{\mathbf{x}-\mathbf{x}_S}{h_S} \right)^\alpha \right)^2 d\mathbf{x} \\
&= \sum_{1\le|\gamma|\le r} \frac{h_0^{2|\gamma|}}{|E_0|} \int_{E_0} \sum_{|\alpha|\le r} \sum_{|\beta|\le r} c_\alpha c_\beta \mathscr{D}^\gamma \left( \frac{\mathbf{x}-\mathbf{x}_S}{h_S} \right)^\alpha \mathscr{D}^\gamma \left( \frac{\mathbf{x}-\mathbf{x}_S}{h_S} \right)^\beta d\mathbf{x} \\
&= \sum_{|\alpha|\le r} \sum_{|\beta|\le r} \left( \sum_{1\le|\gamma|\le r} \frac{h_0^{2|\gamma|}}{|E_0|} \int_{E_0} \mathscr{D}^\gamma \left( \frac{\mathbf{x}-\mathbf{x}_S}{h_S} \right)^\alpha \mathscr{D}^\gamma \left( \frac{\mathbf{x}-\mathbf{x}_S}{h_S} \right)^\beta d\mathbf{x} \right) c_\alpha c_\beta \\
&= \sum_{|\alpha|\le r} \sum_{|\beta|\le r} \eta_{\alpha,\beta} c_\alpha c_\beta, \qquad (3.8)
\end{aligned}$$

where

$$\begin{aligned}
\eta_{\alpha,\beta} &= \sum_{1\le|\gamma|\le r} \frac{h_0^{2|\gamma|}}{|E_0|} \int_{E_0} \mathscr{D}^\gamma \left( \frac{\mathbf{x}-\mathbf{x}_S}{h_S} \right)^\alpha \mathscr{D}^\gamma \left( \frac{\mathbf{x}-\mathbf{x}_S}{h_S} \right)^\beta d\mathbf{x} \\
&= \sum_{\substack{1\le|\gamma|\le r, \\ \gamma\le\alpha,\gamma\le\beta}} \left( \frac{h_0}{h_S} \right)^{2|\gamma|} \frac{\alpha!\,\beta!}{(\alpha-\gamma)!\,(\beta-\gamma)!} \frac{1}{|E_0|} \int_{E_0} \left( \frac{\mathbf{x}-\mathbf{x}_S}{h_S} \right)^{\alpha+\beta-2\gamma} d\mathbf{x} \qquad (3.9)
\end{aligned}$$

is independent of the data $\bar{\mathbf{u}}$ and $\mathbf{c}$ and can be precomputed. The symmetric $N_C \times N_C$ matrix $\eta = (\eta_{\alpha,\beta})$ is positive semidefinite and gives

$$\sigma_{\text{JS}} = \mathbf{c}^T \eta \mathbf{c}. \qquad (3.10)$$

## 4 A New Polynomial Coefficient Squared Smoothness Indicator

We now present our approximation to $\sigma_{\text{JS}}$ on the stencil $S$ for target element $E_0 \in S$. Return to (3.8), where $\sigma_{\text{JS}}$ was expanded in terms of the polynomial coefficients. We conjecture that the cross terms in the expansion are uncorrelated and can be dropped, so we define

$$\begin{aligned}
\sigma_{\text{P}} &= \sum_{1\le|\gamma|\le r} \frac{h_0^{2|\gamma|}}{|E_0|} \int_{E_0} \sum_{|\alpha|\le r} \left( c_\alpha \mathscr{D}^\gamma \left( \frac{\mathbf{x}-\mathbf{x}_S}{h_S} \right)^\alpha \right)^2 d\mathbf{x} \\
&= \sum_{|\alpha|\le r} \left( \sum_{1\le|\gamma|\le r} \frac{h_0^{2|\gamma|}}{|E_0|} \int_{E_0} \left( \mathscr{D}^\gamma \left( \frac{\mathbf{x}-\mathbf{x}_S}{h_S} \right)^\alpha \right)^2 d\mathbf{x} \right) c_\alpha^2. \qquad (4.1)
\end{aligned}$$

That is,

$$\sigma_{\text{P}} = \sum_\alpha \tilde{\eta}_\alpha c_\alpha^2 = \mathbf{c}^T \operatorname{diag}(\tilde{\eta})\, \mathbf{c}, \qquad (4.2)$$

where the $N_C$-vector $\tilde{\eta} = (\eta_{\alpha,\alpha})$ is

$$\tilde{\eta}_\alpha = \eta_{\alpha,\alpha} = \sum_{\substack{1\le|\gamma|\le r,\\ \gamma\le\alpha}} \left(\frac{h_0}{h_S}\right)^{2|\gamma|} \left(\frac{\alpha!}{(\alpha-\gamma)!}\right)^2 \frac{1}{|E_0|} \int_{E_0} \left(\frac{\mathbf{x}-\mathbf{x}_S}{h_S}\right)^{2(\alpha-\gamma)} d\mathbf{x}, \quad (4.3)$$

which can be precomputed, being independent of $\bar{\mathbf{u}}$ and $\mathbf{c}$. Moreover, $\operatorname{diag}(\tilde{\eta})$ is positive semidefinite, so $\sigma_{\mathrm{P}} \ge 0$.

The following lemma provides an estimate of the asymptotic behavior of $\sigma_{\mathrm{P}}$ in the smooth case.

**Lemma 4.1** *If $u$ is approximated on the target element $E_0$ over the stencil $S$ by the polynomial $P \in \mathbb{P}_r$, then there is some constant $D \ge 0$ such that as $h = h_S \to 0^+$,*

$$\sigma_P = Dh_0^2 + \mathscr{O}(h^3), \quad \text{if } u \text{ is smooth on the stencil.} \quad (4.4)$$

*Moreover, when $r \ge 1$, $D$ is given by* (3.3).

*Proof* The result is trivial when $r = 0$, so suppose that $r \ge 1$. When evaluated at the base point $\mathbf{x} = \mathbf{x}_S$, we have that

$$c_\alpha = \frac{h_S^{|\alpha|}}{\alpha!} \mathscr{D}^\alpha P(\mathbf{x}_S). \quad (4.5)$$

Note from (4.3) that in general $\tilde{\eta}_0 = 0$ and $\tilde{\eta}_\alpha = \mathscr{O}(1)$, and that when $|\alpha| = 1$ (say $\alpha = \mathbf{e}_m \in \mathbb{R}^d$), we have $\tilde{\eta}_\alpha = (h_0/h_S)^2$. Therefore,

$$\sigma_{\mathrm{P}} = \sum_{|\alpha|\le r} \tilde{\eta}_\alpha \Big(\frac{h_S^{|\alpha|}}{\alpha!} \mathscr{D}^\alpha P(\mathbf{x}_S)\Big)^2 = 0 + h_0^2 |\nabla P(\mathbf{x}_S)|^2 + \sum_{1<|\alpha|\le r} \mathscr{O}(h^{2|\alpha|}).$$

Invoke (2.7) to see that

$$|\nabla P(\mathbf{x}_S)| \le |\nabla P(\mathbf{x}_S) - \nabla u(\mathbf{x}_S)| + |\nabla u(\mathbf{x}_S)| \le Ch^r + |\nabla u(\mathbf{x}_S)|,$$

and the result follows with the wrong value for $D$, i.e., $|\nabla u(\mathbf{x}_S)|^2$. However,

$$|\nabla u(\mathbf{x}_S)|^2 = \frac{1}{|E_0|} \int_{E_0} \big(|\nabla u(\mathbf{x}_S)|^2 - |\nabla u(\mathbf{x})|^2\big)\, d\mathbf{x} + D \le Ch + D,$$

by the mean value theorem. The proof is complete. □

## 5 Computational Costs

WENO schemes are generally applied to time dependent problems. We tacitly assume that the smoothness indicators are being used in a time stepping loop on a fixed computational mesh, or perhaps a mesh that is updated only infrequently. Various quantities can be precomputed before the time stepping loop begins. The main computational effort required is that required at each time step. We thus consider only the cost for evaluation in a time step (or Newton iteration), assuming that the matrices $A$, $\sigma$, $\eta$, and the vector $\tilde{\eta}$ have been precomputed. At the start of the time step, assume only that the $N_S$-vector $\bar{\mathbf{u}} = (\bar{u}_E)$ is given.

The computational costs depend strongly on the needs of the overall WENO scheme. We consider efficiency in the context of an unstructured mesh in multiple space dimensions. In any application, the stencil polynomial $P(\mathbf{x})$ will be evaluated at, say, $N_Q$ points (such as quadrature points). Explicit and implicit methods have very different needs, so we will consider each separately. If implicit methods are used, solution by Newton's method of the nonlinear problem also requires evaluation of the Jacobian derivatives $\partial P/\partial \bar{\mathbf{u}}$ and $\partial \sigma/\partial \bar{\mathbf{u}}$. It is evaluation of these additional derivatives that motivates the use of the base polynomials [3].

For the computations, we consider both the number of floating point operations (FLOPs) required as well as the number of floating point (FP) number transfers from the main memory.

### 5.1 Evaluations needed in explicit schemes

Perhaps Horner's method is the most efficient algorithm to compute a polynomial in one dimension, and recursion over the variables can be used to handle polynomials in $d > 1$ dimensions. Horner's method is discussed in Appendix A. The work required to evaluate a single polynomial at a single point is $2(N_C - 1)$ (see (A.5)), about twice the number of coefficients.

Since the coefficients of the base polynomials $P_E$ can be precomputed, we simply need to transfer these $N_S N_C$ coefficients from the main memory to access them (i.e., $N_S N_C$ FP transfers). Evaluation of $\mathbf{c} = A\bar{\mathbf{u}}$ requires $N_C(2N_S - 1)$ FLOPs with the same number of FP transfers for $A$. In fact, the coefficients of $P_E$ are exactly the columns of $A$, so these are simply transfered from menory for either case. For reference, these results are tabulated in Table 5.1.

Consider first the case of evaluating the stencil polynomial using the base polynomials (2.6). In this approach, $N_S$ base polynomials must be evaluated and then combined using $2N_S - 1$ FLOPs, so the total work required is $N_Q(2N_S N_C - 1)$ FLOPs for evaluation at $N_Q$ points. Given the coefficients of $P(\mathbf{x})$, the second case needs the evaluation of a single polynomial at each point, which is $2N_Q(N_C - 1)$ FLOPs. These results appear in Table 5.1.

Each time step, the smoothness indicator $\sigma_{\text{JS}}$ can be computed from $\bar{\mathbf{u}} = (\bar{u}_E)$ directly using (3.7), which will take

$$(3N_S - 1) + \sum_{i=1}^{N_S - 1} (3i - 1) + 2 = \tfrac{3}{2}N_S^2 + \tfrac{1}{2}N_S + 2 \quad \text{FLOPs.} \tag{5.1}$$

Table 5.1: For explicit schemes, FLOP and FP transfer costs per time step related to evaluation of the stencil polynomial $P(\mathbf{x}_Q)$ at $N_Q$ points and the smoothness indicator $\sigma_{\text{JS}}$ or $\sigma_{\text{P}}$.

| to evaluate | using | requiring from memory | FLOPs | FP transfers |
|---|---|---|---|---|
| $A$ | — | $A$ | — | $N_S N_C$ |
| $P_E$ (coef) | $A$ | — | — | — |
| $\mathbf{c}$ | $\bar{\mathbf{u}}$, $A$ | — | $N_C(2N_S-1)$ | — |
| $P(\mathbf{x}_Q)$ | $\bar{\mathbf{u}}$, $P_E$ (coef) | — | $N_Q(2N_S N_C-1)$ | — |
| $P(\mathbf{x}_Q)$ | $\mathbf{c}$ | — | $2N_Q(N_C-1)$ | — |
| $\sigma_{\text{JS}}$ | $\bar{\mathbf{u}}$ | $\sigma$ | $\frac{3}{2}N_S^2+\frac{1}{2}N_S+2$ | $\frac{1}{2}N_S(N_S+1)$ |
| $\sigma_{\text{JS}}$ | $\mathbf{c}$ | $\eta$ | $\frac{3}{2}N_C^2+\frac{1}{2}N_C+2$ | $\frac{1}{2}N_C(N_C+1)$ |
| $\sigma_{\text{P}}$ | $\mathbf{c}$ | $\tilde{\eta}$ | $3N_C-1$ | $N_C$ |

The computation also requires the additional transfer of the floating point (FP) numbers in the symmetric matrix $\sigma$ from the main memory, which is $\frac{1}{2}N_S(N_S+1)$ FP transfers. On the other hand, given the polynomial coefficients, $\sigma_{\text{JS}}$ can be computed via the symmetric version of (3.10) with a cost of $\frac{3}{2}N_C^2+\frac{1}{2}N_C+2$ FLOPs and $\frac{1}{2}N_C(N_C+1)$ FP transfers, since the matrix $\eta$ is symmetric. This is the more efficient way to compute $\sigma_{\text{JS}}$ in isolation. Finally, the smoothness indicator $\sigma_{\text{P}}$ can be computed in $3N_C-1$ FLOPs and $N_C$ FP transfers, which is by far the most efficient computation of a smoothness indicator alone. These results appear in Table 5.1.

Which combination of computations is most efficient overall depends on the costs of computing both $P(\mathbf{x}_Q)$ and the smoothness indicator, which in turn depends on the values of $N_C \geq 1$, $N_S \geq N_C \geq 1$, and $N_Q \geq 1$. Assuming that $N_Q \geq 2$, it is always more efficient to compute the polynomial coefficients $\mathbf{c}$ and then $P(\mathbf{x}_Q)$ from them than to use the base polynomials. To see this, simply check that

$$N_C(2N_S-1)+2N_Q(N_C-1) \stackrel{?}{\leq} N_Q(2N_S N_C-1).$$

Notice that

$$2N_Q(N_C-1) \leq N_Q(2N_S N_C-1),$$

so the minimal value of $N_Q$ is the harder case and we merely check if

$$N_C(2N_S-1)+4(N_C-1) \stackrel{?}{\leq} 2(2N_S N_C-1) \quad \Longleftrightarrow \quad 3N_C \stackrel{?}{\leq} 2N_S N_C+2.$$

The minimal value of $N_S = N_C$ is the harder case, and easily seen to hold true since the polynomial $2x^2-3x+2$ has no real roots.

We conclude that for explicit methods, the use of base polynomials is the most expensive route in terms of FLOPs and FP transfers. The polynomial coefficients $\mathbf{c}$ should first be found using (2.4) and then $P(\mathbf{x})$ evaluated from (2.2) directly using $\mathbf{c}$. Given $\mathbf{c}$, $\sigma_{\text{JS}}$ should be computed using $\eta$ in (3.10). But it is much more efficient for both FLOPs and FP transfers to instead compute $\sigma_{\text{P}}$ using $\tilde{\eta}$ in (4.2). The total computational costs are summarized in Table 5.2.

Table 5.2: For explicit schemes, total FLOP and FP transfer costs per time step related to evaluation of the stencil polynomial $P(\mathbf{x}_Q)$ at $N_Q$ points and the smoothness indicator $\sigma_{\text{JS}}$ or $\sigma_{\text{P}}$.

| to evaluate | requiring from memory | FLOPs | FP transfers |
|---|---|---|---|
| $P(\mathbf{x}_Q)$, $\sigma_{\text{JS}}$ | $P_E$ (coef), $\sigma$ | $N_Q(2N_SN_C-1)+\frac{3}{2}N_S^2+\frac{1}{2}N_S+2$ | $\frac{1}{2}N_S(2N_C+N_S+1)$ |
| $\mathbf{c}$, $P(\mathbf{x}_Q)$, $\sigma_{\text{JS}}$ | $A$, $\eta$ | $2N_Q(N_C-1)+N_C(2N_S+\frac{3}{2}N_C-\frac{1}{2})+2$ | $\frac{1}{2}N_C(2N_S+N_C+1)$ |
| $\mathbf{c}$, $P(\mathbf{x}_Q)$, $\sigma_{\text{P}}$ | $A$, $\tilde{\eta}$ | $2N_Q(N_C-1)+2N_C(N_S+1)-1$ | $N_C(N_S+1)$ |

## 5.2 Evaluations needed in implicit schemes

Consider first evaluation of the Jacobian derivative $\partial P/\partial\bar{\mathbf{u}}$. It is easy to compute from the base polynomials (2.6), and it is in fact

$$\frac{\partial P}{\partial\bar{u}_E}=P_E(\mathbf{x}). \tag{5.2}$$

It is therefore necessary to compute $P_E(\mathbf{x}_Q)$ at each point. Direct computation has a workload of $2N_QN_S(N_C-1)$ FLOPs. On the other hand, if one has the polynomial coefficients $\mathbf{c}$, the Jacobian derivative requires evaluation of

$$\frac{\partial P(\mathbf{x})}{\partial\bar{\mathbf{u}}}=\sum_{|\alpha|\le r}\frac{\partial c_\alpha}{\partial\bar{\mathbf{u}}}\left(\frac{\mathbf{x}-\mathbf{x}_S}{h_S}\right)^\alpha. \tag{5.3}$$

This is exactly evaluation of $N_S$ polynomials with coefficients drawn from the columns of $\frac{\partial\mathbf{c}}{\partial\bar{\mathbf{u}}}=A$. These polynomials are the base polynomials. Therefore, (5.2) is the only way to compute the Jacobian derivatives of $P$. These results are tabulated in Table 5.3, along with the costs of evaluating $P(\mathbf{x}_Q)$.

The Jacobian derivatives of the classic smoothness indicator, expressed as $\sigma_{\text{JS}}=\bar{\mathbf{u}}^T\sigma\bar{\mathbf{u}}$ (3.6), is easily computed to be

$$\frac{\partial\sigma_{\text{JS}}}{\partial\bar{\mathbf{u}}}=2\sigma\bar{\mathbf{u}}. \tag{5.4}$$

For efficiency, one should compute $\sigma\bar{\mathbf{u}}$ in $2N_S-1$ FLOPs, $\sigma_{\text{JS}}=\bar{\mathbf{u}}^T\sigma\bar{\mathbf{u}}$ in $2N_S-1$ additional FLOPs, and finally $\partial\sigma_{\text{JS}}/\partial\bar{\mathbf{u}}$ by doubling the values of $\sigma\bar{\mathbf{u}}$ for $N_S$ FLOPs. These results are given in Table 5.3.

Using that $\sigma_{\text{JS}}=\mathbf{c}^T\eta\mathbf{c}$ (3.10), evaluation of the Jacobian derivatives requires including the partial derivative $\partial\mathbf{c}/\partial\bar{\mathbf{u}}=A$, so

$$\frac{\partial\sigma_{\text{JS}}}{\partial\bar{\mathbf{u}}}=2A^T\eta\mathbf{c}. \tag{5.5}$$

Again for efficiency, one should compute $\mathbf{c}$ first, $\eta\mathbf{c}$ in $N_C(2N_C-1)$ FLOPs, $\sigma_{\text{JS}}=\mathbf{c}^T\eta\mathbf{c}$ in $2N_C-1$ additional FLOPs, and finally $\partial\sigma_{\text{JS}}/\partial\bar{\mathbf{u}}$ in $N_C+N_S(2N_C-1)$ FLOPs. Using $\sigma_{\text{P}}=\mathbf{c}^T\text{diag}(\tilde{\eta})\mathbf{c}$ (4.2) gives a similar computation. These results appear in Table 5.3.

Table 5.3: For implicit schemes, FLOP and FP transfer costs per time step related to evaluation of the stencil polynomial $P(\mathbf{x}_Q)$ and Jacobian derivative $P(\mathbf{x}_Q)/\partial\bar{\mathbf{u}}$ at $N_Q$ points and the smoothness indicator $\sigma_{\text{JS}}$ or $\sigma_{\text{P}}$ and its Jacobian derivative.

| to evaluate | using | requiring from memory | FLOPs | FP transfers |
|---|---|---|---|---|
| $A$ | — | $A$ | — | $N_S N_C$ |
| $P_E$ (coef) | $A$ | — | — | — |
| $\mathbf{c}$ | $\bar{\mathbf{u}}$, $A$ | — | $N_C(2N_S-1)$ | — |
| $P_E(\mathbf{x}_Q)$ | $P_E$ (coef) | — | $2N_Q N_S(N_C-1)$ | — |
| $P(\mathbf{x}_Q)$ | $\bar{\mathbf{u}}$, $P_E(\mathbf{x}_Q)$ | — | $N_Q(2N_S-1)$ | — |
| $P(\mathbf{x}_Q)$ | $\mathbf{c}$ | — | $2N_Q(N_C-1)$ | — |
| $\partial P(\mathbf{x}_Q)/\partial\bar{\mathbf{u}}$ | $P_E(\mathbf{x}_Q)$ | — | — | — |
| $\sigma\bar{\mathbf{u}}$ | $\bar{\mathbf{u}}$ | $\sigma$ | $N_S(2N_S-1)$ | $\frac{1}{2}N_S(N_S+1)$ |
| $\sigma_{\text{JS}}$ | $\bar{\mathbf{u}}$, $\sigma\bar{\mathbf{u}}$ | — | $2N_S-1$ | — |
| $\partial\sigma_{\text{JS}}/\partial\bar{\mathbf{u}}$ | $\sigma\bar{\mathbf{u}}$ | — | $N_S$ | — |
| $\eta\mathbf{c}$ | $\mathbf{c}$ | $\eta$ | $N_C(2N_C-1)$ | $\frac{1}{2}N_C(N_C+1)$ |
| $\sigma_{\text{JS}}$ | $\mathbf{c}$, $\eta\mathbf{c}$ | — | $2N_C-1$ | — |
| $\partial\sigma_{\text{JS}}/\partial\bar{\mathbf{u}}$ | $A$, $\eta\mathbf{c}$ | — | $N_C+N_S(2N_C-1)$ | — |
| $\text{diag}(\tilde{\eta})\mathbf{c}$ | $\mathbf{c}$ | $\tilde{\eta}$ | $N_C$ | $N_C$ |
| $\sigma_{\text{P}}$ | $\mathbf{c}$, $\text{diag}(\tilde{\eta})\mathbf{c}$ | — | $2N_C-1$ | — |
| $\partial\sigma_{\text{P}}/\partial\bar{\mathbf{u}}$ | $A$, $\text{diag}(\tilde{\eta})\mathbf{c}$ | — | $N_C+N_S(2N_C-1)$ | — |

The total computational costs are summarized in Table 5.4. In terms of FP transfers, computing $\sigma_{\text{JS}}$ from the base polynomials and $P(\mathbf{x}_Q)$ from these or from $\mathbf{c}$ are the most costly approaches, followed by computing $\mathbf{c}$ and then $P(\mathbf{x}_Q)$ and $\sigma_{\text{JS}}$ using the polynomial coefficients. Computing $\mathbf{c}$ and then $P(\mathbf{x}_Q)$ and $\sigma_{\text{P}}$ is the most efficient.

Table 5.4: For implicit schemes, total FLOP and FP transfer costs per time step related to evaluation of the stencil polynomial $P(\mathbf{x}_Q)$ at $N_Q$ points and the smoothness indicator $\sigma_{\text{JS}}$ or $\sigma_{\text{P}}$, and their Jacobian derivatives.

| to evaluate with der. | requiring from memory | FLOPs | FP transfers |
|---|---|---|---|
| $P(\mathbf{x}_Q)$, $\sigma_{\text{JS}}$ | $P_E$ (coef), $\sigma$ | $N_Q(2N_S N_C-1)+2N_S^2+2N_S-1$ | $N_S N_C+\frac{1}{2}N_S(N_S+1)$ |
| $\mathbf{c}$, $P(\mathbf{x}_Q)$, $\sigma_{\text{JS}}$ | $A$, $\sigma$ | $2N_Q(N_S+1)(N_C-1)$ $+2N_S(N_S+N_C)+2N_S-N_C-1$ | $N_S N_C+\frac{1}{2}N_S(N_S+1)$ |
| $\mathbf{c}$, $P(\mathbf{x}_Q)$, $\sigma_{\text{JS}}$ | $A$, $\eta$ | $2N_Q(N_S+1)(N_C-1)$ $+4N_S N_C+2N_C^2+N_C-N_S-1$ | $N_S N_C+\frac{1}{2}N_C(N_C+1)$ |
| $\mathbf{c}$, $P(\mathbf{x}_Q)$, $\sigma_{\text{P}}$ | $A$, $\tilde{\eta}$ | $2N_Q(N_S+1)(N_C-1)$ $+4N_S N_C+3N_C-N_S-1$ | $N_S N_C+N_C$ |

In terms of FLOPs, it is less clear which approach is best, although given $\mathbf{c}$, computing $\sigma_{\text{JS}}$ is more costly than computing $\sigma_{\text{P}}$. The number of FLOPs required to eval-

uate $P(\mathbf{x}_Q)$ and $\sigma_{\text{JS}}$ (and derivatives) from the base polynomials minus the number for evaluating $\mathbf{c}$ and then $P(\mathbf{x}_Q)$ and $\sigma_{\text{P}}$ is

$$\begin{aligned}&\left[N_Q(2N_SN_C-1)+2N_S^2+2N_S-1\right]\\&\quad-\left[2N_Q(N_S+1)(N_C-1)+4N_SN_C+3N_C-N_S-1\right]\\&\quad=(N_S-N_C)(2N_Q+2N_S+3)+N_Q-2N_SN_C.\end{aligned}\tag{5.6}$$

If this number is positive, then evaluating $\mathbf{c}$ and then $P(\mathbf{x}_Q)$ and $\sigma_{\text{P}}$ is more efficient in terms of FLOPs. In one space dimension, $N_S = N_C$ and $N_Q$ is typically 2, so the number is negative. However, in two dimensions, typically $N_S \sim 3N_C/2$, and the sign of the number depends on the specific values of the parameters. We conclude that for implicit methods, our new approach of computing the coefficients $\mathbf{c}$ and then $P(\mathbf{x}_Q)$, $\sigma_{\text{P}}$, and their Jacobian derivatives is the most efficient in terms of FP transfers, and is at least competitive in terms of FLOPs.

## 6 Comparison of the Performance of the Smoothness Indicators

In this section, we present numerical tests in two dimensions of the classic and the new smoothness indicators to assess the quality of our approximation. The critical property (1.1) of a smoothness indicator is that it can identify a shock or contact discontinuity in the solution. That is, when the solution is smooth, its smoothness indicator should be small and tend to zero as $Dh^2 + \mathcal{O}(h^3)$ as $h \to 0$. When the solution has a jump discontinuity, the smoothness indicator should be relatively large and remain essentially constant as $h \to 0$. In our case, we already know that $\sigma_{\text{JS}}$ has these properties, so we are merely checking that its approximation, $\sigma_{\text{P}}$, maintains the properties.

The test solution is taken to be $u(x,y) = 2(1+\cos(2\pi x))\exp(xy-y)+J(x)$, where the term $J(x)$ can be used to impose a jump. It is a multiple of the Heaviside function (0 for $x < 0$ and 1 for $x > 0$), and the multiple is taken to be 0 for a smooth solution, and two discontinuous solutions use 1.0 for a full jump and 0.1 for a mild jump.

The test solution $u(x,y)$ is approximated on 19 different stencils. The first set of 9 stencils are based on polygonal meshes (of triangular and quadrilateral elements) and shown in Figure 6.1. StencilPoly19 has 19 elements and supports polynomial approximation of degree 4. We also consider 7 substencils. The larger substencils are identified as StencilPoly19-9, StencilPoly19-7a, StencilPoly19-7b, and StencilPoly19-6. These have 9, 7, 7, and 6 elements, respectively, and each supports quadratic approximations. The smallest stencils, StencilPoly19-3a, StencilPoly19-3b, and StencilPoly19-3c, all have 3 elements and support linear approximation. Finally, StencilPoly17 is a distortion of StencilPoly19, but it has only 17 elements and still supports polynomial approximation of degree 4. The unrefined maximal element diameter of these stencils is about $h_0 = 0.06$.

The second set of 4 stencils are based on the notion of a sectorial stencil [16], as depicted in Figure 6.2. The top element defines the sector, and normally one would target that element, although we will give results for all the elements as possible target

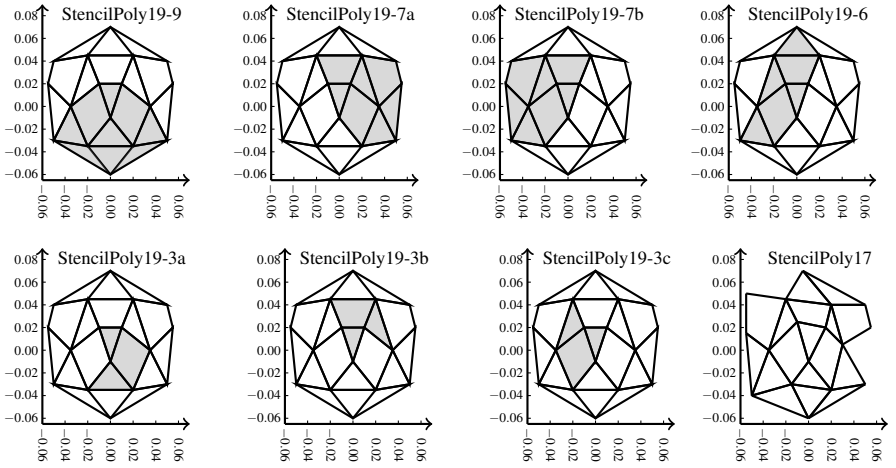



Fig. 6.1: Unrefined StencilPoly19 with its substencils (shaded in gray) and StencilPoly17. The maximal element diameter is about $h_0 = 0.06$.

elements. StencilSect16 is the full stencil, and it has 16 elements which support polynomial approximation of degree 4. Its substencils are shaded in gray. The substencils are identified as StencilSect10, StencilSect6, and StencilSect3. They have 10, 6, and 3 elements and support polynomial approximation of degree 3, 2, and 1, respectively. The maximal element diameter is about $h_0 = 0.075$.

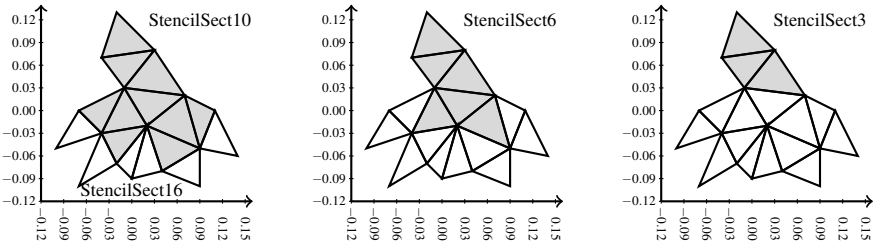



Fig. 6.2: Unrefined StencilSect16 and its substencils (shaded in gray). The maximal element diameter is about $h_0 = 0.075$.

The last 6 stencils are based on simple rectangular meshes. Stencil3x3, Stencil3x5, and Stencil5x5 are meshes of $3 \times 3$, $3 \times 5$, and $5 \times 5$ square elements centered about $(0,0)$ with side length $h_0 = 0.1$ for the first two and $h_0 = 0.04$ for the third stencil. These 3 stencils support approximation by polynomials of degree 2, 2, and 4, respectively. The final 3 stencils are simply the previous three crosshatched from the lower left corner to the upper right corner forming two triangles for each square. These triangular stencils support approximation by polynomials of degree 4, 5, and 8,

respectively. These meshes do not resolve the jump, because it appears at $x = 0$, in the middle of some elements and outside the rest.

Each of the 19 stencils is fairly coarse, but we rescale them. For each rescaled refinement level $L = 0,1,2,3,4$, we divide each stencil point by a factor of 2, so $h_L = h_0/2^L$ and the stencil shrinks about the center point $(0,0)$. The jump at $x = 0$ appears fixed under refinement.

For each of the 19 stencils, we approximate the test solution $u(x,y)$, with or without a jump, by its element averages (2.1), and then compute its stencil polynomial for a select target element. This then gives $\sigma_{\text{JS}}$ and $\sigma_{\text{P}}$ for the target element. This process is repeated on each of the 4 refinement levels, for each possible target element in the stencil, and for each of the 3 values for the jump. In total, we have 7680 tests, so we give detailed results for only a few cases and summarize the rest of the results.

## 6.1 Overview of the results.

In this section, we give an overview of our results. We begin by showing a small fraction of the raw data. Table 6.1, shows the results for the StencilPoly19 and StencilPoly17 tests using the single target element defined as the most central one depicted in Figure 6.1. We see good agreement between the two smoothness indicators. When $u$ is smooth, the smoothness indicators converge to zero as $\mathscr{O}(h^2)$ for $h \to 0$ (i.e., as the refinement level increases). When $u$ has a jump, the smoothness indicators remain approximately constant as $h \to 0$.

Ideally, the ratios $\sigma_{\text{P}}/\sigma_{\text{JS}}$ would be one, and we see that they are approximately one. When $u$ is smooth, the ratios converge to one as $h \to 0$. When $u$ is discontinuous, a ratio $\sigma_{\text{P}}/\sigma_{\text{JS}} > 1$ might be advantageous, as the larger $\sigma_{\text{P}}$ will tend to bias a WENO reconstruction away from the stencil better than $\sigma_{\text{JS}}$.

Table 6.1: The smoothness indicators for StencilPoly19 and StencilPoly17 using the center element as the target. Shown are the results for the smooth and the two discontinuous test solutions, on each refinement level.

| | | StencilPoly19, center target | | | StencilPoly17, center target | | |
|---|---|---|---|---|---|---|---|
| $u$ | level | $\sigma_{\text{JS}}$ | $\sigma_{\text{P}}$ | $\sigma_{\text{P}}/\sigma_{\text{JS}}$ | $\sigma_{\text{JS}}$ | $\sigma_{\text{P}}$ | $\sigma_{\text{P}}/\sigma_{\text{JS}}$ |
| Smooth | 0 | 2.194e-02 | 2.217e-02 | 1.011 | 3.164e-02 | 3.238e-02 | 1.023 |
| | 1 | 4.354e-03 | 4.377e-03 | 1.005 | 5.923e-03 | 5.993e-03 | 1.012 |
| | 2 | 1.020e-03 | 1.022e-03 | 1.003 | 1.360e-03 | 1.368e-03 | 1.006 |
| | 3 | 2.509e-04 | 2.512e-04 | 1.001 | 3.329e-04 | 3.338e-04 | 1.003 |
| | 4 | 6.252e-05 | 6.256e-05 | 1.001 | 8.285e-05 | 8.296e-05 | 1.001 |
| Jump | 0 | 3.951e+00 | 4.028e+00 | 1.019 | 2.662e+01 | 3.071e+01 | 1.154 |
| 1.0 | 1 | 3.909e+00 | 3.986e+00 | 1.020 | 2.655e+01 | 3.064e+01 | 1.154 |
| | 2 | 3.900e+00 | 3.977e+00 | 1.020 | 2.652e+01 | 3.062e+01 | 1.155 |
| | 3 | 3.897e+00 | 3.975e+00 | 1.020 | 2.652e+01 | 3.062e+01 | 1.155 |
| | 4 | 3.896e+00 | 3.974e+00 | 1.020 | 2.652e+01 | 3.062e+01 | 1.155 |
| Jump | 0 | 6.421e-02 | 6.509e-02 | 1.014 | 3.044e-01 | 3.449e-01 | 1.133 |
| 0.1 | 1 | 4.421e-02 | 4.494e-02 | 1.017 | 2.737e-01 | 3.145e-01 | 1.149 |
| | 2 | 4.027e-02 | 4.102e-02 | 1.018 | 2.674e-01 | 3.083e-01 | 1.153 |
| | 3 | 3.932e-02 | 4.008e-02 | 1.019 | 2.658e-01 | 3.068e-01 | 1.154 |
| | 4 | 3.907e-02 | 3.983e-02 | 1.020 | 2.654e-01 | 3.063e-01 | 1.154 |

In Table 6.2, we show the results for the StencilSect16 and StencilSect6 tests using the single target element defined as the topmost one depicted in Figure 6.2. We see generally good agreement between the two smoothness indicators, but not as good as in the previous tests. When $u$ is smooth, the smoothness indicators converge to zero as $\mathscr{O}(h^2)$ for $h \to 0$. The ratios $\sigma_P/\sigma_{JS}$ are approximately one, but show some deviation for the coarsest stencil of the StencilSect16 test.

When $u$ has a jump, the smoothness indicators remain approximately constant as $h \to 0$. The ratios $\sigma_P/\sigma_{JS}$ are less than but close to one. However, the value of $\sigma_P$ is significantly greater than its value when $u$ is smooth (especially for the finer stencils).

Table 6.2: The smoothness indicators for StencilSect16 and StencilSect6 using the top element as the target. Shown are the results for the smooth and the two discontinuous test solutions, on each refinement level.

| | | StencilSect16, top target | | | StencilSect6, top target | | |
|---|---|---|---|---|---|---|---|
| $u$ | level | $\sigma_{JS}$ | $\sigma_P$ | $\sigma_P/\sigma_{JS}$ | $\sigma_{JS}$ | $\sigma_P$ | $\sigma_P/\sigma_{JS}$ |
| Smooth | 0 | 2.153e-01 | 2.911e-01 | 1.352 | 2.399e-01 | 2.609e-01 | 1.088 |
| | 1 | 2.806e-02 | 3.250e-02 | 1.158 | 2.918e-02 | 3.092e-02 | 1.060 |
| | 2 | 5.428e-03 | 5.772e-03 | 1.063 | 5.464e-03 | 5.623e-03 | 1.029 |
| | 3 | 1.265e-03 | 1.298e-03 | 1.026 | 1.266e-03 | 1.282e-03 | 1.013 |
| | 4 | 3.118e-04 | 3.155e-04 | 1.012 | 3.118e-04 | 3.136e-04 | 1.006 |
| Jump | 0 | 6.604e+03 | 5.196e+03 | 0.787 | 2.263e+01 | 1.946e+01 | 0.860 |
| 1.0 | 1 | 6.620e+03 | 5.187e+03 | 0.783 | 1.999e+01 | 1.690e+01 | 0.846 |
| | 2 | 6.626e+03 | 5.185e+03 | 0.783 | 1.938e+01 | 1.629e+01 | 0.840 |
| | 3 | 6.627e+03 | 5.185e+03 | 0.782 | 1.925e+01 | 1.614e+01 | 0.838 |
| | 4 | 6.628e+03 | 5.185e+03 | 0.782 | 1.922e+01 | 1.610e+01 | 0.838 |
| Jump | 0 | 6.408e+01 | 5.321e+01 | 0.830 | 7.496e-01 | 7.333e-01 | 0.978 |
| 0.1 | 1 | 6.555e+01 | 5.205e+01 | 0.794 | 2.957e-01 | 2.705e-01 | 0.915 |
| | 2 | 6.609e+01 | 5.189e+01 | 0.785 | 2.136e-01 | 1.864e-01 | 0.873 |
| | 3 | 6.624e+01 | 5.186e+01 | 0.783 | 1.962e-01 | 1.673e-01 | 0.853 |
| | 4 | 6.627e+01 | 5.185e+01 | 0.782 | 1.926e-01 | 1.625e-01 | 0.844 |

Perhaps it is easier to understand the results in a log-plot. Figure 6.3 shows the results for a possible WENO application, which might combine the stencil polynomials on StencilPoly19 and its substencils targeting the center element in Figure 6.1. The bottom sets of data (i.e., the bottom two lines) show the case when $u$ is smooth. We see a near perfect agreement between the two smoothness indicators in the plots. Moreover, (1.1), i.e., $\sigma = Dh^2 + \mathscr{O}(h^3)$, with $D$ given by (3.3) is seen to hold, since the $y$-intercepts (or $x$-intercepts) agree between the tests and the second order rate of convergence is evidenced by the data following a line of slope $-2$. The top and middle sets of data are for the cases where $u$ has jump 1.0 and 0.1, respectively. For either jump discontinuity, the two indicators remain unchanged as the stencil is refined, i.e., $\sigma = \mathscr{O}(1)$, at least beyond refinement level 0. The two smoothness indicators are also in excellent agreement. Interestingly, some of the results show an inability to detect the mild shock on refinement level 0, and the results for StencilPoly19-6 show that $\sigma_{JS}$ does not detect the mild shock, while $\sigma_P$ does detect it. Overall, one should expect WENO schemes using either of the two smoothness indicators to perform similarly.

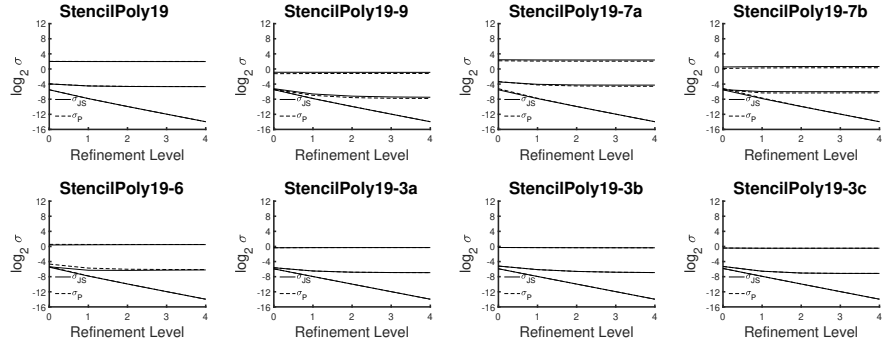



Fig. 6.3: StencilPoly19 and its substencils smoothness indicators for the smooth (lower line) and the two jump solutions (top line jump 1.0, middle line jump 0.1) for the center target element.

Similar results appear in Figure 6.4 for StencilSect16 and its substencils. The target element here is the topmost element in Figure 6.2. We note that neither smoothness indicator detects the mild shock on StencilSect3 at refinement levels 0 and 1. However, $\sigma_{\mathrm{JS}}$ and $\sigma_{\mathrm{P}}$ continue to agree in these cases, and (1.1), (3.3) hold.

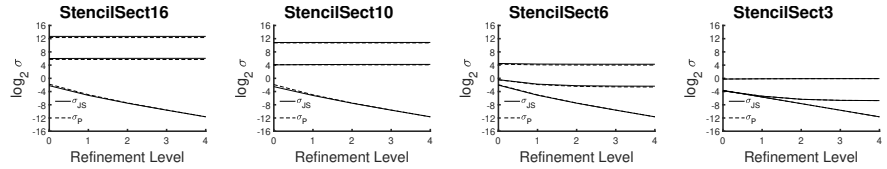



Fig. 6.4: StencilSect16 and its substencils smoothness indicators for the smooth (lower line) and the two jump solutions (top line jump 1.0, middle line jump 0.1) for the center target element.

The results in Figures 6.3–6.4 are typical of what one observes in the data set. In Figure 6.5, we show some of the more extreme results. The test using StencilPoly19-9 shows a case where the two smoothness indicators differ in their asymptotic values for the jump tests. Moreover, there is no detection of the mild shock for refinement levels 0 and 1. The StencilPoly19-7b test actually shows that the smoothness indicators (which agree) suggest that the case of the mild jump is smoother than the smooth solution at refinement level 0. The two StencilPoly17 tests again show that the two smoothness indicators can differ in their asymptotic values for the jump tests. The first shows $\sigma_{\mathrm{P}}/\sigma_{\mathrm{JS}} < 1$ while the second shows $\sigma_{\mathrm{P}}/\sigma_{\mathrm{JS}} > 1$.

The test using Stencil3x5Triangles shows some deviation between the two smoothness indicators for the smooth solution on refinement level 0. The Stencil3x3Triangles test shows the two smoothness indicators differing in their asymptotic values for the jump cases, but agreeing on the coarser levels. The Stencil5x5 test shows a case where the classic smoothness indicator considers the mild jump as smoother than the smooth

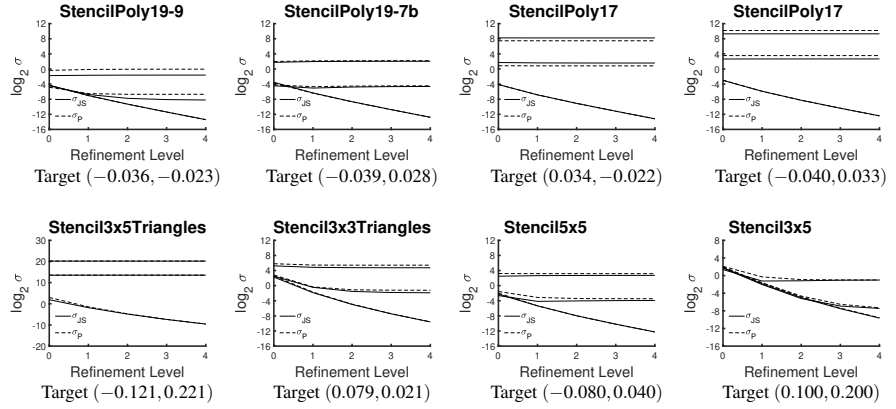



Fig. 6.5: Smoothness indicators for the smooth (lower line) and the two jump solutions (top line jump 1.0, middle line jump 0.1). These are some of the more extreme results. The stencil and center of the unrefined target element is given for each plot.

solution, but $\sigma_{\mathrm{P}}$ gives the correct ordering. Finally, the test using Stencil3x5 shows a case where the smoothness indicators have difficulty in detecting the jumps. The mild jump is not detected by either smoothness indicator on refinement levels 0–3, and the full jump is not detected on refinement level 0. Moreover, the classic smoothness indicator fails to detect the jump on refinement level 1.

These results show that the the classic smoothness indicator does not always detect the jumps as well as we might have hoped. However, the collective years of experience of researchers in WENO methods shows that the classic smoothness indicator works well in practice. What we have seen here is that the approximate smoothness indicator $\sigma_{\mathrm{P}}$ behaves very similarly to the classic one, and so is in fact a very good approximation.

## 6.2 Relative errors.

Let us now show the entire data set in summary form. The results presented are the maximum (or worst case) of the relative errors between $\sigma_{\mathrm{JS}}$ and $\sigma_{\mathrm{P}}$ over all possible target elements in the stencil, i.e.,

$$\text{relative error } = \max_E \frac{|\sigma_{\mathrm{JS}}^E - \sigma_{\mathrm{P}}^E|}{\sigma_{\mathrm{JS}}^E},$$

where $\sigma^E$ is the smoothness indicator with target element $E$ in the stencil.

The results for the polygonal stencils StencilPoly19, its substencils, and StencilPoly17 appear in Table 6.3. The exception is that StencilPoly19-3(a,b,c) has no error and so is not shown. That there is no error follows from (3.9). When $r = 1$, $|\alpha| = |\beta| = |\gamma| = 1$ and so $\eta_{\alpha,\beta} = 0$ when $\alpha \neq \beta$. There are no cross terms in the sum, and so $\sigma_{\mathrm{JS}} = \sigma_{\mathrm{P}}$.

Table 6.3: Maximum relative errors over all possible target elements for StencilPoly19 and its substencils and for StencilPoly17. The results for StencilPoly19-3(a,b,c) are exact (no errors), and so not shown.

| | StencilPoly19, Jump | | | StencilPoly19-9, Jump | | | StencilPoly19-7a, Jump | | |
|---|---|---|---|---|---|---|---|---|---|
| level | 0.0 | 1.0 | 0.1 | 0.0 | 1.0 | 0.1 | 0.0 | 1.0 | 0.1 |
| 0 | 0.155 | 1.433 | 1.314 | 0.036 | 1.607 | 0.404 | 0.230 | 0.170 | 0.079 |
| 1 | 0.078 | 1.513 | 1.239 | 0.026 | 1.888 | 0.368 | 0.082 | 0.190 | 0.158 |
| 2 | 0.038 | 1.532 | 1.466 | 0.017 | 1.940 | 1.095 | 0.023 | 0.196 | 0.185 |
| 3 | 0.018 | 1.536 | 1.525 | 0.010 | 1.960 | 1.638 | 0.009 | 0.198 | 0.188 |
| 4 | 0.009 | 1.538 | 1.538 | 0.005 | 1.968 | 1.847 | 0.003 | 0.199 | 0.195 |
| | StencilPoly19-7b, Jump | | | StencilPoly19-6, Jump | | | StencilPoly17, Jump | | |
| level | 0.0 | 1.0 | 0.1 | 0.0 | 1.0 | 0.1 | 0.0 | 1.0 | 0.1 |
| 0 | 0.298 | 0.238 | 0.263 | 0.135 | 1.124 | 0.790 | 0.115 | 0.272 | 0.064 |
| 1 | 0.126 | 0.210 | 0.293 | 0.078 | 1.012 | 1.260 | 0.071 | 0.320 | 0.168 |
| 2 | 0.046 | 0.203 | 0.210 | 0.037 | 0.975 | 1.133 | 0.029 | 0.336 | 0.270 |
| 3 | 0.018 | 0.201 | 0.203 | 0.017 | 0.961 | 1.038 | 0.013 | 0.340 | 0.315 |
| 4 | 0.008 | 0.200 | 0.201 | 0.008 | 0.956 | 0.993 | 0.007 | 0.342 | 0.332 |

We see low relative errors when $u$ is smooth, at least for the finer levels of refinement. In fact, the convergence is $\mathcal{O}(h)$, as we should expect from (1.1) and (3.3). When $u$ is not smooth, the relative errors do not vary much, indicating that the smoothness indicators do not change value as the mesh is refined.

The results for the sectorial stencils StencilSect16 and its substencils appear in Table 6.4, except StencilSect3 is exact and not shown. We see behavior similar to that for the polygonal stencils.

Table 6.4: Maximum relative errors over all possible target elements for StencilSect16 and its substencils, except StencilSect3.

| | StencilSect16, Jump | | | StencilSect10, Jump | | | StencilSect6, Jump | | |
|---|---|---|---|---|---|---|---|---|---|
| level | 0.0 | 1.0 | 0.1 | 0.0 | 1.0 | 0.1 | 0.0 | 1.0 | 0.1 |
| 0 | 0.352 | 0.590 | 0.658 | 0.531 | 0.216 | 0.159 | 0.115 | 0.272 | 0.064 |
| 1 | 0.158 | 0.582 | 0.605 | 0.148 | 0.231 | 0.191 | 0.071 | 0.320 | 0.168 |
| 2 | 0.063 | 0.581 | 0.585 | 0.051 | 0.234 | 0.225 | 0.029 | 0.336 | 0.270 |
| 3 | 0.026 | 0.579 | 0.581 | 0.020 | 0.235 | 0.233 | 0.013 | 0.340 | 0.315 |
| 4 | 0.012 | 0.579 | 0.580 | 0.009 | 0.235 | 0.235 | 0.007 | 0.342 | 0.332 |

The results for the rectangular and crosshatched triangular stencils appear in Tables 6.5–6.6. These results are consistent with the previous. However, the triangular meshes show more error for the coarsest stencils.

### 6.3 Comparison for smooth solutions.

Both smoothness indicators have a large and nearly constant value as $h \to 0$ when there is a jump in the solution. Although the value of $\sigma_{\text{JS}}$ and $\sigma_{\text{P}}$ may not agree, the jump has been identified and WENO methods should work well with either the classic smoothness indicator or its approximation.

Table 6.5: Maximum relative errors over all possible target elements for the rectangular stencils.

| | Stencil3x3, Jump | | | Stencil3x5, Jump | | | Stencil5x5, Jump | | |
|---|---|---|---|---|---|---|---|---|---|
| level | 0.0 | 1.0 | 0.1 | 0.0 | 1.0 | 0.1 | 0.0 | 1.0 | 0.1 |
| 0 | 0.065 | 0.632 | 0.109 | 0.112 | 0.675 | 0.157 | 0.476 | 0.684 | 1.210 |
| 1 | 0.082 | 0.991 | 0.236 | 0.139 | 1.021 | 0.298 | 0.207 | 0.452 | 1.070 |
| 2 | 0.078 | 0.239 | 0.423 | 0.128 | 0.251 | 0.471 | 0.098 | 0.393 | 0.554 |
| 3 | 0.050 | 0.054 | 0.315 | 0.079 | 0.057 | 0.319 | 0.046 | 0.379 | 0.418 |
| 4 | 0.026 | 0.013 | 0.114 | 0.041 | 0.014 | 0.117 | 0.022 | 0.375 | 0.385 |

Table 6.6: Maximum relative errors over all possible target elements for the triangular stencils.

| | Stencil3x3tri, Jump | | | Stencil3x5tri, Jump | | | Stencil5x5tri, Jump | | |
|---|---|---|---|---|---|---|---|---|---|
| level | 0.0 | 1.0 | 0.1 | 0.0 | 1.0 | 0.1 | 0.0 | 1.0 | 0.1 |
| 0 | 0.510 | 0.633 | 0.458 | 1.087 | 0.036 | 0.054 | 0.474 | 1.550 | 1.550 |
| 1 | 0.249 | 0.703 | 0.899 | 0.418 | 0.033 | 0.034 | 0.210 | 1.550 | 1.550 |
| 2 | 0.115 | 0.636 | 0.821 | 0.190 | 0.033 | 0.033 | 0.098 | 1.550 | 1.550 |
| 3 | 0.056 | 0.615 | 0.668 | 0.090 | 0.033 | 0.033 | 0.046 | 1.550 | 1.550 |
| 4 | 0.028 | 0.610 | 0.623 | 0.042 | 0.033 | 0.033 | 0.022 | 1.550 | 1.550 |

In this section we concentrate on the way the smoothness indicators approximate a smooth solution by plotting the ratio $\sigma_P/\sigma_{JS}$ versus the refinement level. Shown in Figures 6.6–6.7 are results for every possible target element.

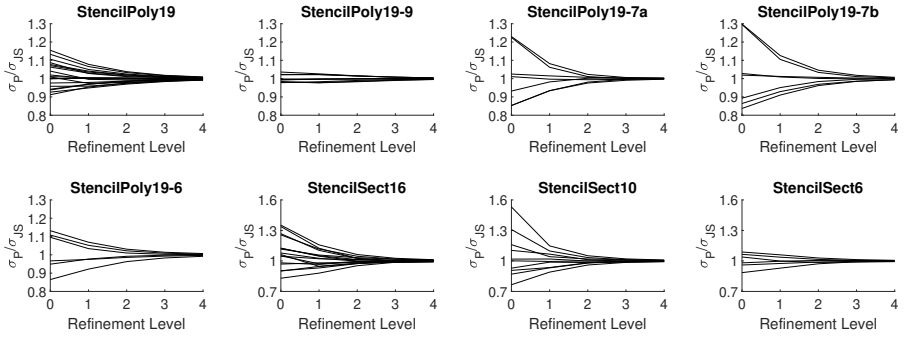



Fig. 6.6: StencilPoly19 and StencilSect16 and their substencils (except the linear polynomial substencils). Smoothness indicator ratio $\sigma_P/\sigma_{JS}$ plotted versus the refinement level for the smooth solution (no jump). Shown are results for every possible target element.

Overall, the results show that there is some scatter in the ratios, but only for coarser levels of refinement.

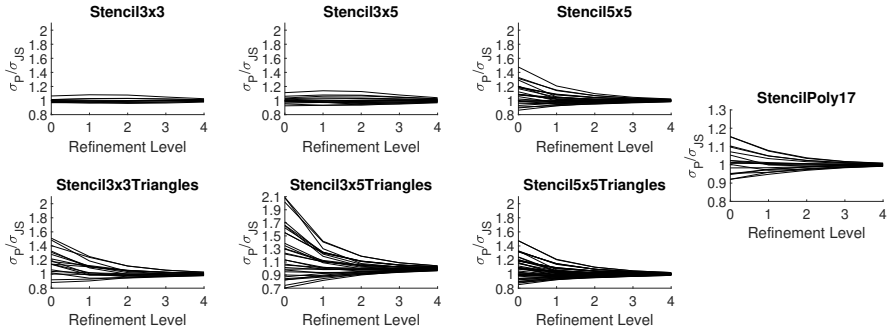



Fig. 6.7: Rectangular and triangular stencils (left) and StencilPoly17 (right). Smoothness indicator ratio $\sigma_P/\sigma_{JS}$ plotted versus the refinement level for the smooth solution (no jump). Shown are results for every possible target element.

## 6.4 Comparison of the run-time.

We close our direct comparison of the two smoothness indicators by giving as assessment of the computational costs. In Table 6.7, we report the average CPU time of the classic smoothness indicator, implemented using base polynomials ($\sigma_{JS}$-base computed using (3.7)) and polynomial coefficients ($\sigma_{JS}$-coef computed using (3.10), but in symmetric form), and its approximation ($\sigma_P$ computed using (4.2)). We also show the speedup (ratio of the time for the minimum of the two classic smoothness indicator times versus the polynomial smoothness indicator time). We report these numbers for each of our stencils. Each number is the average of 10 runs. Each run totals the time for 3 evaluations (no, full, and mild jumps), 5 levels of mesh refinement, and the use of each stencil element as a target element.

Except for the evaluation of the stencils of 3 elements, the classic smoothness indicator is more efficiently computed using $\eta_{\alpha,\beta}$ and the polynomial coefficients. We find a significant improvement in CPU time with our new smoothness indicator for all but one of the tests (where it is equivalent to evaluating $\sigma_{JS}$ using the base polynomials). Overall, the cost of computing our new smoothness indicator based on the square of the polynomial coefficients is on average almost 6 (5.771) times faster than the classic smoothness indicator in these tests. Moreover, if we omit the results for the linear polynomials, the average increases to about 7 (6.926).

To put the timing results into context, the tests of this section and the next one are run on an 8 core Apple Mac mini with an M2 chip (maximum clock speed of 3.495 GHz). The code itself is written in C++ and compiled using the GNU compiler with optimization level 2.

Table 6.7: CPU time for computing the smoothness indicators (microseconds).

| Stencil | Number of elements | Polynomial degree | Time for $\sigma_{\text{JS}}$-base | Time for $\sigma_{\text{JS}}$-coef | Time for $\sigma_{\text{P}}$ | Speedup |
|---|---|---|---|---|---|---|
| Poly17 | 17 | 4 | 0.537 | 0.419 | 0.051 | 8.216 |
| Poly19 | 19 | 4 | 0.664 | 0.421 | 0.054 | 7.796 |
| Poly19-9 | 9 | 2 | 0.264 | 0.121 | 0.040 | 3.025 |
| Poly19-7a | 7 | 2 | 0.241 | 0.184 | 0.053 | 3.472 |
| Poly19-7b | 7 | 2 | 0.249 | 0.194 | 0.057 | 3.404 |
| Poly19-6 | 6 | 2 | 0.223 | 0.226 | 0.066 | 3.379 |
| Poly19-3a | 3 | 1 | 0.079 | 0.082 | 0.050 | 1.580 |
| Poly19-3b | 3 | 1 | 0.067 | 0.068 | 0.041 | 1.634 |
| Poly19-3c | 3 | 1 | 0.079 | 0.083 | 0.079 | 1.000 |
| Sect16 | 16 | 4 | 0.484 | 0.425 | 0.054 | 7.870 |
| Sect10 | 10 | 3 | 0.371 | 0.365 | 0.072 | 5.069 |
| Sect6 | 6 | 2 | 0.227 | 0.226 | 0.070 | 3.229 |
| Sect3 | 3 | 1 | 0.077 | 0.081 | 0.050 | 1.540 |
| 3x3 | 9 | 2 | 0.353 | 0.169 | 0.054 | 3.130 |
| 3x5 | 15 | 2 | 0.600 | 0.107 | 0.031 | 3.452 |
| 5x5 | 25 | 4 | 1.126 | 0.414 | 0.050 | 8.280 |
| 3x3Triangles | 18 | 4 | 0.599 | 0.424 | 0.052 | 8.154 |
| 3x5Triangles | 30 | 5 | 1.601 | 0.796 | 0.072 | 11.056 |
| 5x5Triangles | 50 | 8 | 5.922 | 4.873 | 0.200 | 24.365 |

## 7 Application to Hyperbolic Conservation Laws

We consider the computational efficiency of the smoothness indicators when applied to solving scalar conservation laws in two space dimensions, i.e., to

$$\begin{cases} u_t(\mathbf{x},t) + \nabla \cdot \mathbf{f}(u;\mathbf{x},t) = 0, & t > 0,\ \mathbf{x} \in \Omega, \\ u(\mathbf{x},0) = u_0(\mathbf{x}), & \mathbf{x} \in \Omega, \end{cases} \tag{7.1}$$

where the domain $\Omega \subset \mathbb{R}^2$, $u : \Omega \times (0,\infty) \to \mathbb{R}$ is the solution, and $\mathbf{f} : \mathbb{R} \times \Omega \times (0,\infty) \to \mathbb{R}^2$ is the flux function. For simplicity, we use periodic boundary conditions. We choose three relatively standard test problems. We give the computational time required to compute the solutions, with the only difference in the runs being the choice of smoothness indicator and whether an explicit or implicit time stepping is used. We also show that the two smoothness indicators give comparable results. Recall that the tests are run on an Apple Mac mini with an M2 chip.

Our stencils are defined by rings (see [2,31,22,32,4]). For each target element $E$ in the mesh, our ML-WENO reconstructions uses a *large stencil* which is a ring 1.5 stencil. It is the union of $E$ and every element sharing an edge or vertex with $E$, and every element sharing an edge with these elements. We also use many ring 1 *small stencils*. A small stencil about element $F$ consists of the union of $F$ and every element sharing an edge or vertex with $F$. The reconstruction for $E$ uses the large stencil about $E$ and every small stencil that contains $E$.

In the tests, the multilevel WENO (ML-WENO) reconstruction [4] is used because of its flexibility in solving problems in multiple space dimensions. The actual number of elements in each stencil varies depending on the mesh. As an illustration, a uniform mesh of hexagons would have a ring 1.5 large stencil of 19 cells, and the ring 1 small stencils would each have 7 cells. A uniform mesh of quadrilaterals would have a ring 1.5 large stencil of 21 cells, and the ring 1 small stencils would each have 9 cells. These large and small stencils would be expected to support stencil polynomials of degree 4 and 2, respectively. In the notation of WENO schemes, we would denote these as ML-WENO(5,3) reconstructions. However, as discussed in [4], a lower order approximation may be obtained for some stencils, either because they are actually smaller than expected, or they simply do not support stencil polynomials that give accurate approximation. Hence, technically, we use ML-WENO(5,4,3,2) reconstructions in these tests, although the reconstructions are ML-WENO(5,3) over most of the domain.

To be specific about the computations performed, we define the ML-WENO reconstruction $R(\mathbf{x})$ for the target element $E$. For its $j$th stencil, denote by $P_j(\mathbf{x})$ the stencil polynomial of degree $r_j$, $\sigma_j$ the smoothness indicator, and $\tilde{\omega}_j$ the nonlinear weight. Then

$$\hat{\omega}_j = \frac{1}{(\sigma_j + 0.01h^2)^{r_j}}, \qquad \tilde{\omega}_j = \frac{\hat{\omega}_j}{\sum_k \hat{\omega}_k}, \qquad R(\mathbf{x}) = \sum_j \tilde{\omega}_j P_j(\mathbf{x}). \tag{7.2}$$

As shown in [4], for all $\mathbf{x} \in E$,

$$|u(\mathbf{x}) - R(\mathbf{x})| \leq Ch^{r_{\max}}, \tag{7.3}$$

where $r_{\max} = \max_\ell \{r_\ell : u \text{ is smooth on the } \ell\text{th stencil}\}$. That is, $r_{\max}$ is the highest order of accuracy obtained by stencil polynomials for which their stencils do not contain a shock in the solution.

Timing results would be similar using multilevel reconstructions of type WENO with Adaptive Order (WENO-AO) [8,5,7], and also using reconstructions of type Central WENO (CWENO), [25], which is a two-level WENO-AO reconstruction. We use the classic weighting procedure [20], but other weightings such as Z-weighting [12] would give similar results.

Time integration is accomplished using a third order, strong stability preserving Runge-Kutta method. We give results for both an explicit method, using the standard SSP3 Runge-Kutta time integrator, and an implicit method, using the standard DirkSSP23 (diagonally implicit 2 stage, third order accurate) Runge-Kutta time integrator.

### 7.1 Linear Advection

The first test problem is due to LeVeque [24] and uses the linear flux function $\mathbf{f}(x,y) = \big((0.5-y)u, (x-0.5)u\big)$, which gives a rotating flow on $\Omega = (0,1)^2$. The initial con-

dition consists of a slotted disk, a cone, and a smooth hump, defined by

$$\begin{aligned} r_0 &= 0.15, \quad r_H = \sqrt{(x-0.25)^2+(y-0.50)^2}, \\ r_C &= \sqrt{(x-0.50)^2+(y-0.25)^2}, \quad r_D = \sqrt{(x-0.50)^2+(y-0.75)^2} \\ u_0(x,y) &= (r_D \le r_0)\big[1-(y<0.75)(0.45<x)(x<0.55)\big] \\ &\quad +(r_C \le r_0)(1-r_C/r_0)+(r_H \le r_0)0.25(1+\cos(\pi r_H/r_0)), \end{aligned} \tag{7.4}$$

wherein we use the convention that a true logical comparison evaluates to 1, and a false one evaluates to 0. This problem is extremely difficult to solve accurately, due especially to the sharp contact discontinuities of the slotted disk.

A polygonal mesh is used that has 18,858 vertices and 10,000 elements. Some elements have as many as 10 edges. It can be seen in [4]. The solution is computed up to the time of one revolution, which is $t = 2\pi$. The explicit method, SSP3, uses 2080 timesteps ($\Delta t \approx 0.003021$), for a CFL number close to 1. The implicit method, DirkSSP23, uses a nominal 520 timesteps ($\Delta t \approx 0.012083$), although the time step is cut when Newton's method has difficulty solving the nonlinear Runge-Kutta equations. This happens only rarely, so the CFL number is close to 4 most of the time.

Table 7.1: Timings for the rotating flow, linear advection problem. Given are the CPU total run time, the time for the time steps themselves, and the time to compute the smoothness indicators, for both explicit and implicit timestepping.

| | SSP3, 2080 steps time (sec) | | | DirkSSP23, 520 steps time (sec) | | |
|---|---|---|---|---|---|---|
| | total | time steps | $\sigma$ | total | time steps | $\sigma$ |
| $\sigma_{\text{JS}}$-base | 453.90 | 380.53 | 98.07 | 1665.50 | 1592.18 | 60.01 |
| $\sigma_{\text{JS}}$-coef | 350.06 | 344.11 | 60.94 | 2358.25 | 2352.27 | 115.50 |
| $\sigma_{\text{P}}$ | 302.97 | 299.23 | 13.83 | 1474.12 | 1470.37 | 15.76 |

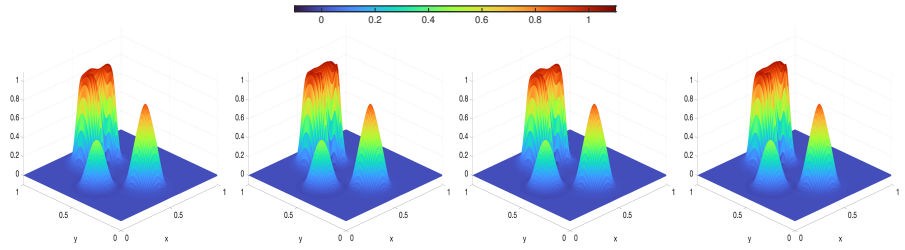
The timing results appear in Table 7.1. For the explicit method, the use of $\sigma_{\text{P}}$ is clearly more efficient than using $\sigma_{\text{JS}}$-coef ($\sigma_{\text{JS}}$ computed using the polynomial coefficients), and both these are much more efficient than using $\sigma_{\text{JS}}$-base ($\sigma_{\text{JS}}$ computed using the base polynomials). For the implicit method, the use of $\sigma_{\text{P}}$ is more efficient than using $\sigma_{\text{JS}}$-base, and both these are much more efficient than using $\sigma_{\text{JS}}$-coef. These observations are consistent with the computational cost estimates of Sections 5.1–5.2.

The exact solution is known for this problem, since it returns to the initial condition after one revolution. The discrete $L_h^1$-errors and $L_h^\infty$-errors are given in Table 7.2 ($\sigma_{\text{JS}}$-base and $\sigma_{\text{JS}}$-coef give the same results up to rounding error). Interestingly, $\sigma_{\text{P}}$ gives a solution that is a bit more accurate. The norm of the difference between the discrete solutions is also given. These show that there is very little difference in the discrete solutions, and significantly less than the approximation error. That is, it makes little difference whether $\sigma_{\text{JS}}$ or $\sigma_{\text{P}}$ is used in the computations in terms of the quality of the solution.

The final solutions for the four runs are depicted in Figure 7.1.

Table 7.2: Assessment of the errors for the rotating flow, linear advection problem.

| | explicit | | implicit | |
|---|---|---|---|---|
| | $L_h^1$-norm | $L_h^\infty$-norm | $L_h^1$-norm | $L_h^\infty$-norm |
| Solution error using $\sigma_{JS}$ | 1.97e-02 | 6.45e-01 | 1.97e-02 | 6.45e-01 |
| Solution error using $\sigma_P$ | 1.87e-02 | 6.42e-01 | 1.87e-02 | 6.42e-01 |
| Difference of the discrete solutions | 2.15e-03 | 1.34e-01 | 2.15e-03 | 1.34e-01 |



$\sigma_{JS}$, explicit $\sigma_P$, explicit $\sigma_{JS}$, implicit $\sigma_P$, implicit

Fig. 7.1: Final solution for the rotating flow, linear advection problem.

## 7.2 Burgers Equation

The second test problem uses the nonlinear Burgers flux $\mathbf{f}(u) = (u^2/2, u^2/2)$ on the domain $\Omega = (0,1)^2$. The smooth initial condition is $u_0(x,y) = \sin^2(\pi x)\sin^2(\pi y)$. In this test, shocks form in the solution.

A $100 \times 100$ uniform mesh is distorted by moving the internal vertices randomly up to a factor of 0.2 times the unperturbed mesh spacing. Because of the regularity of the mesh, the reconstructions are of type ML-WENO(5,3). Numerical solutions are computed to time $t = 0.4$, well after the shock has formed. The explicit timestepping uses 200 steps at $\Delta t = 0.002$, for a CFL number close to 1, and the implicit timestepping uses 40 steps at $\Delta t = 0.01$, for a CFL number close to 5.

The timing results appear in Table 7.3. They are fully consistent with the theoretical estimates of Sections 5.1–5.2.

Table 7.3: Timings for the Burgers problem. Given are the CPU total run time, the time for the time steps themselves, and the time to compute the smoothness indicators, for both explicit and implicit timestepping.

| | SSP3, 200 steps time (sec) | | | DirkSSP23, 40 steps time (sec) | | |
|---|---|---|---|---|---|---|
| | total | time steps | $\sigma$ | total | time steps | $\sigma$ |
| $\sigma_{JS}$-base | 74.42 | 31.47 | 13.41 | 132.82 | 89.86 | 6.28 |
| $\sigma_{JS}$-coef | 27.68 | 24.75 | 6.66 | 96.71 | 93.76 | 10.05 |
| $\sigma_P$ | 21.68 | 19.93 | 1.53 | 87.54 | 85.84 | 1.49 |

The norm of the difference between the discrete solutions is given in Table 7.4. The results on the left show that there is very little difference between the discrete

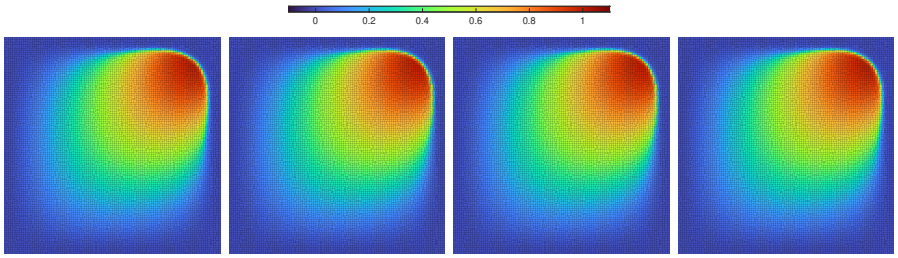
solutions whether $\sigma_{\mathrm{JS}}$ or $\sigma_{\mathrm{P}}$ is used in the computations, whichever timestepping is used. Moreover, these differences are comparable to other discretization errors. As seen on the right of Table 7.4, similar values are observed for the difference between using explicit and implicit timestepping, for a fixed choice of $\sigma$.

Table 7.4: Solution differences for the Burgers problem. Comparisons are made between the solutions computed using $\sigma_{\mathrm{JS}}$ and $\sigma_{\mathrm{P}}$ for explicit and implicit timestepping (results on the left), and comparisons between the explicit and implicit timestepping for the given smoothness indicator (results on the right).

| quantity | $L_h^1$-norm | $L_h^\infty$-norm |
|---|---|---|
| $\|u^{\text{Explicit}}_{\sigma_{\mathrm{JS}}} - u^{\text{Explicit}}_{\sigma_{\mathrm{P}}}\|$ | 9.62e-05 | 2.58e-02 |
| $\|u^{\text{Implicit}}_{\sigma_{\mathrm{JS}}} - u^{\text{Implicit}}_{\sigma_{\mathrm{P}}}\|$ | 9.85e-05 | 2.61e-02 |

| quantity | $L_h^1$-norm | $L_h^\infty$-norm |
|---|---|---|
| $\|u^{\text{Explicit}}_{\sigma_{\mathrm{JS}}} - u^{\text{Implicit}}_{\sigma_{\mathrm{JS}}}\|$ | 2.88e-05 | 1.02e-02 |
| $\|u^{\text{Explicit}}_{\sigma_{\mathrm{P}}} - u^{\text{Implicit}}_{\sigma_{\mathrm{P}}}\|$ | 2.98e-05 | 1.02e-02 |

The final solutions for the four runs are depicted in Figure 7.2. No discernable difference appears.




$\sigma_{\mathrm{JS}}$, explicit $\sigma_{\mathrm{P}}$, explicit $\sigma_{\mathrm{JS}}$, implicit $\sigma_{\mathrm{P}}$, implicit

Fig. 7.2: Final solution for the Burgers problem.

## 7.3 Buckley-Leverett Flux with Gravity

For the final test problem, we consider the nonconvex Buckley-Leverett flux, modified to incorporate gravity, so $\mathbf{f}(u) = \dfrac{u^2}{u^2+(1-u)^2}\begin{pmatrix}1\\ 1-5(1-u)^2\end{pmatrix}$. In this test [21, 13, 15], the domain $\Omega = [-1.5, 1.5]^2$ and

$$u_0(x,y) = \begin{cases} 1 & \text{if } x^2+y^2 \le 0.5, \\ 0 & \text{otherwise.} \end{cases} \tag{7.5}$$

Shocks and rarefactions form and propagate across the domain.

Time is advanced to $t = 0.5$. For the explicit time stepping, we used a $200 \times 200$ element square mesh and 250 time steps $\Delta t = 0.002$, which gives CFL number 1. For

the implicit time stepping, we used a $120 \times 120$ element square mesh and 50 time steps $\Delta t = 0.01$, which gives CFL number 3. These meshes give ML-WENO(5,3) reconstructions.

The timing results appear in Table 7.5. They are fully consistent with the theoretical estimates of Sections 5.1–5.2.

Table 7.5: Timings for the Buckley-Leverett with gravity problem. Given are the CPU total run time, the time for the time steps themselves, and the time to compute the smoothness indicators, for both explicit and implicit timestepping.

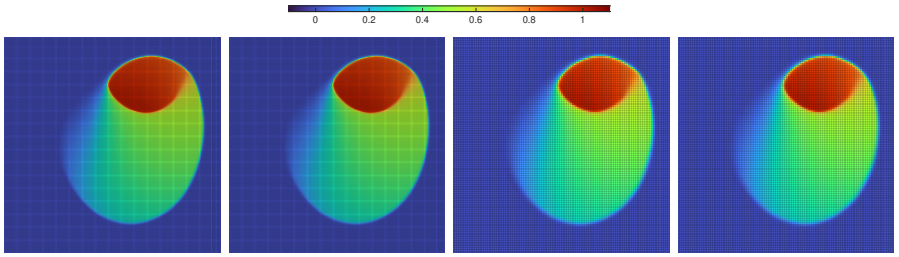
| | SSP3, 250 steps, $200^2$ mesh time (sec) | | | DirkSSP23, 50 steps, $120^2$ mesh time (sec) | | |
|---|---|---|---|---|---|---|
| | total | time steps | $\sigma$ | total | time steps | $\sigma$ |
| $\sigma_{\text{JS}}$-base | 373.68 | 194.39 | 69.49 | 313.78 | 252.20 | 15.45 |
| $\sigma_{\text{JS}}$-coef | 169.84 | 156.13 | 33.79 | 256.49 | 252.44 | 23.79 |
| $\sigma_{\text{P}}$ | 140.50 | 131.41 | 7.76 | 237.10 | 234.77 | 3.59 |

The norm of the difference between the discrete solutions is given in Table 7.6. There is only a small difference between the discrete solutions using $\sigma_{\text{JS}}$ or $\sigma_{\text{P}}$, whether explicit or implicit timestepping is used.

Table 7.6: Solution differences for the Buckley-Leverett with gravity problem, computed using $\sigma_{\text{JS}}$ and $\sigma_{\text{P}}$, for both explicit and implicit timestepping.

| quantity | $L_h^1$-norm | $L_h^\infty$-norm |
|---|---|---|
| $\|u_{\sigma_{\text{JS}}}^{\text{Explicit}} - u_{\sigma_{\text{P}}}^{\text{Explicit}}\|$ | 4.10e-03 | 8.85e-02 |
| $\|u_{\sigma_{\text{JS}}}^{\text{Implicit}} - u_{\sigma_{\text{P}}}^{\text{Implicit}}\|$ | 6.30e-03 | 7.12e-02 |

The final solutions for the four runs are depicted in Figure 7.3. No discernable difference appears.




$\sigma_{\text{JS}}$, explicit $\sigma_{\text{P}}$, explicit $\sigma_{\text{JS}}$, implicit $\sigma_{\text{P}}$, implicit

Fig. 7.3: Final solution for the Buckley-Leverett with gravity problem.

## 8 Summary and Conclusions

The classic smoothness indicator $\sigma_{\mathrm{JS}}$ can be implemented using either the base polynomials (3.4) or directly using the polynomial coefficients (3.8). Both require a double sum over either the elements in the stencil of the polynomial coefficients. The latter implementation was exploited to define an approximate smoothness indicator $\sigma_{\mathrm{P}}$ (4.2), described as the smoothness indicator based on the polynomial coefficients squared, requiring only a single sum over the coefficients.

The smoothness indicators were proven in Lemmas 3.1 and 4.1 to satisfy the appropriate asymptotic relation (1.1) when the solution is smooth. Moreover, the constant $D$ was shown to be fixed according to (3.3).

Theoretical estimates were presented for the full set of computational costs required in implementation for solving temporal partial differential equations in multiple space dimensions. For explicit timestepping methods, computations based on first computing the polynomial coefficients and then using $\sigma_{\mathrm{P}}$ was clearly the most efficient, followed by computing the polynomial coefficients and using $\sigma_{\mathrm{JS}}$ computed using these coefficients. A distant third was the case of computing using $\sigma_{\mathrm{JS}}$ from the base polynomials.

Theoretical computational cost estimates for implicit methods, however, showed a different trend. It was not completely clear which was most efficient, using $\sigma_{\mathrm{P}}$ or $\sigma_{\mathrm{JS}}$ computed from the base polynomials. The least efficient was the case of using $\sigma_{\mathrm{JS}}$ computed from the polynomial coefficients.

A large number of computational tests involving single stencils were preformed. Analysis of the results were perhaps complicated by the fact that $\sigma_{\mathrm{JS}}$ does not always detect a shock as well as we might have wished, at least for coarser stencil meshes. Nevertheless, the test results showed that $\sigma_{\mathrm{P}}$ is a remarkably accurate approximation of $\sigma_{\mathrm{JS}}$, especially for finer stencil meshes. The two smoothness indicators track each other quite well in the case where the test solution is smooth (at least when the mesh is not too coarse). In the cases where the test solution has a full jump discontinuity of 1.0 and a mild jump discontinuity of 0.1, the two smoothness indicators did not always agree closely. However, they both had large values and remained essentially constant under stencil mesh refinement, as required by the asymptotics (1.1). In terms of computational time, $\sigma_{\mathrm{P}}$ was significantly more efficient to compute.

Application was made to solving hyperbolic conservation laws for three standard test problems using both explicit and implicit timestepping. The CPU times for the runs agreed with the theoretical cost estimates. In fact, we found that in these tests, computing the polynomial coefficients and then using $\sigma_{\mathrm{P}}$ was always the most efficient, and sometimes significantly so, whether using explicit and implicit timestepping. The quality of the solutions was also assessed, and it was seen that results using $\sigma_{\mathrm{JS}}$ or $\sigma_{\mathrm{P}}$ were very comparable.

We conclude that $\sigma_{\mathrm{P}}$ is a good approximation to $\sigma_{\mathrm{JS}}$ in isolation. Within WENO schemes, these two smoothness indicators perform similarly in terms of the quality of the solution. However, $\sigma_{\mathrm{P}}$ is significantly more computationally efficient.

## A Some Remarks on Horner's Method for Evaluation of a Polynomial.

Perhaps Horner's method is the most efficient algorithm to compute the value of a polynomial and possibly some or all of its derivatives. Consider the polynomial $p(t) = \sum_{i=0}^{n} a_i t^i$ in one dimension.

**Algorithm A.1** *Horner's method for evaluation of a polynomial*

**Input** $x \in \mathbb{R}$, polynomial $p(t) = \sum_{i=0}^{n} a_i t^i$

$q = a_n$
**for** $i = n-1, n-2, ..., 0$ **do**
  $q = a_i + qx$
**end for**
**Output** $q = p(x)$

The algorithm takes only the coefficients as input, so the memory transfer requirements are small. The number of floating point operations is given by $2n$.

Generalization to higher order derivatives is straightforward, and given in Algorithm A.2 (see also [11, 10]).

**Algorithm A.2** *Horner's method for evaluation of a polynomial and all its derivatives*

**Input** $x \in \mathbb{R}$, polynomial $p(t) = \sum_{i=0}^{n} a_i t^i$

$P(d) = 0$ for $d = 0, 1, ..., n$
**for** $i = n, n-1, ..., 0$ **do**
  **for** $d = n-i, n-i-1, ..., 1$ **do**
    $P(d) = P(d)\,x + d\,P(d-1)$
  **end for**
  $P(0) = P(0)\,x + a_i$
**end for**
**Output** $P(d) = p^{(d)}(x)$ for $d = 0, 1, ..., n$

The number of floating point operations is given by

$$\sum_{i=0}^{n} \left( \sum_{i=0}^{n-i} 3 + 2 \right) = \sum_{i=0}^{n} \big(3(n-i)+2\big) = \big(\tfrac{3}{2}n + 2\big)(n+1). \tag{A.1}$$

For completeness, we provide a proof that indeed $P(d) = p^{(d)}(x)$.

*Proof* Proceed by induction on $n$. The case $n = 0$ is trivial, giving only $P(0) = a_0 = p(x)$.

If the result holds for $n-1$, one can show that it holds for $n \geq 1$ using the factorization $p(t) = q(t)\,t + a_0$, where $q(t) = \sum_{i=0}^{n-1} a_{i+1} t^i$. The algorithm applied to $q(t)$ results in the derivatives $P(d) = q^{(d)}(x)$. For clarity, we rewrite the algorithm for $q \in \mathbb{P}_{n-1}$, with a change in index $j = i+1$ as follows.

$P(d) = 0$ for $d = 0, 1, ..., n-1$
**for** $j = n, ..., 1$ **do**

```
    for d = n − j, n − j − 1, ..., 1 do
      P(d) = P(d) x + d P(d − 1)
    end for
    P(0) = P(0) x + a_j
  end for
```

To treat $p(x)$, the algorithm continues by setting $P(n) = 0$ and taking $j$ down to 0. The extra loop for $j = 0$ results in $P(0) = p(x)$ by the usual Horner's algorithm and

$$P(d) = P(d)\,x + d\,P(d-1) = q^{(d)}(x)\,x + d\,q^{(d-1)}(x) = p^{(d)}(x),$$

for $d = 1, \ldots, n$. □

Recursion over the variables can be used to handle polynomials in $d > 1$ dimensions. Specifically, for

$$p(\mathbf{x}) = \sum_{|\alpha| \le n} c_\alpha \mathbf{x}^\alpha, \tag{A.2}$$

we rearrange the sum in terms of the last component. Let $\alpha = (\alpha', \alpha_d)$ and $\mathbf{x} = (\mathbf{x}', x_d)$. Then

$$p(\mathbf{x}) = \sum_{\alpha_d = 0}^{n} \left( \sum_{|\alpha'| \le n} c_\alpha (\mathbf{x}')^{\alpha'} \right) x_d^{\alpha_d}. \tag{A.3}$$

and recursion commences on the inner sum.

The operation counts can be computed recursively as well. Let $F(n,d)$ denote the number of FLOPs to compute a polynomial (and possibly its derivatives) of degree $n$ in $d$ dimensions. Then

$$F(n,d) = F(n,1) + \sum_{k=0}^{n} F(k, d-1). \tag{A.4}$$

For polynomial evaluation alone, $F(n,1) = 2n$. Induction and the Hockey-Stick combinatorial identity can be used to show that

$$F(n,d) = 2\big(N(n,d) - 1\big), \tag{A.5}$$

where $N(n,d) = \binom{n+d}{d}$ is the number of coefficients.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## References


1. Abedian, R., Adibi, H., Dehghan, M.: A high-order weighted essentially non-oscillatory (WENO) finite difference scheme for nonlinear degenerate parabolic equations. Computer Physics Communications **184**, 1874–1888 (2013)
2. Abgrall, R.: On essentially non-oscillatory schemes on unstructured meshes: analysis and implementation. J. Comput. Phys. **114**, 45–58 (1994)
3. Aràndiga, F., Baeza, A., Belda, A.M., Mulet, P.: Analysis of WENO schemes for full and global accuracy. SIAM J. Numer. Anal. **49**(2), 893–915 (2011)
4. Arbogast, T., Huang, C.S., Tian, C.: A finite volume multilevel WENO scheme for multidimensional scalar conservation laws. Comput. Methods Appl. Mech. Engrg. **421**(116818) (2024). DOI 10.1016/j.cma.2024.116818
5. Arbogast, T., Huang, C.S., Zhao, X.: Accuracy of WENO and adaptive order WENO reconstructions for solving conservation laws. SIAM J. Numer. Anal. **56**(3), 1818–1847 (2018). DOI 10.1137/17M1154758
6. Arbogast, T., Huang, C.S., Zhao, X.: Finite volume WENO schemes for nonlinear parabolic problems with degenerate diffusion on non-uniform meshes. J. Comput. Phys. **399**(108921) (2019). DOI 10.1016/j.jcp.2019.108921
7. Balsara, D.S., Garain, S., Florinski, V., Boscheri, W.: An efficient class of WENO schemes with adaptive order for unstructured meshes. J. Comput. Phys. **404**, 109062 (2020). DOI 10.1016/j.jcp.2019.109062
8. Balsara, D.S., Garain, S., Shu, C.W.: An efficient class of WENO schemes with adaptive order. J. Comput. Phys. **326**, 780–804 (2016)
9. Borges, R., Carmona, M., Costa, B., Don, W.: An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws. J. Comput. Phys. **227**, 3191–3211 (2008)
10. Burrus, C.S., Fox, J.W., Sitton, G.A., Treitel, S.: Horner's method for evaluating and deflating polynomials. DSP Software Notes 26, Rice University (2003)
11. Carnicer, J., Gasca, M.: Evaluation of multivariate polynomials and their derivatives. Math. Comp. **54**(189), 231–243 (1990)
12. Castro, M., Costa, B., Don, W.: High order weighted essentially non-oscillatory WENO-Z schemes for hyperbolic conservation laws. J. Comput. Phys. **230**, 1766–1792 (2011)
13. Christov, I., Popov, B.: New non-oscillatory central schemes on unstructured triangulations for hyperbolic systems of conservation laws. J. Computational Physics **227**, 5736–5757 (2008)
14. Friedrich, O.: Weighted essentially non-oscillatory schemes for the interpolation of mean values on unstructured grids. J. Comput. Phys. **144**, 194–212 (1998)
15. Guermond, J.L., Pasquetti, R., Popov, B.: Entropy viscosity method for nonlinear conservation laws. Journal of Computational Physics **230**, 4248–4267 (2011). DOI https://doi.org/10.1016/j.jcp.2010.11.043
16. Harten, A., Chakravarthy, S.R.: Multi-dimensional ENO schemes for general geometries. Tech. Rep. 91–76, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, Virginia (1991). Contract No. NAS1-18605
17. Harten, A., Osher, S.: Uniformly high-order accurate nonoscillatory schemes I. SIAM J. Numer. Anal. **24**(2), 279–309 (1987)
18. Huang, C., Chen, L.L.: A simple smoothness indicator for the WENO scheme with adaptive order. J. Comput. Phys. **352**, 498–515 (2018)
19. Huang, C.S., Arbogast, T., Tian, C.: Multidimensional WENO-AO reconstructions using a simplified smoothness indicator and applications to conservation laws. J. Sci. Comput. **97**(8) (2023). DOI 10.1007/s10915-023-02319-x
20. Jiang, G.S., Shu, C.W.: Efficient implementation of weighted ENO schemes. J. Comput. Phys. **126**, 202–228 (1996)
21. Karlsen, K.H., Brusdal, K., Dahle, H., Evje, S., Lie, K.A.: The corrected operator splitting approach applied to a nonlinear advection-diffusion problem. Comput. Methods Appl. Mech. Eng. **167**(3–4), 239–260 (1998)
22. Kašer, M., Iske, A.: ADER schemes on adaptive triangular meshes for scalar conservation laws. Journal of Computational Physics **205**, 486–508 (2005)
23. Kumar, R., Chandrashekar, P.: Simple smoothness indicator and multi-level adaptive order WENO scheme for hyperbolic conservation laws. J. Comput. Phys. **375**, 1059–1090 (2018)

24. LeVeque, R.J.: High-resolution conservative algorithms for advection in incompressible flow. SIAM. J. Numer. Anal. **33**(2), 627–665 (1996)
25. Levy, D., Puppo, G., Russo, G.: Central WENO schemes for hyperbolic systems of conservation laws. Math. Model. Numer. Anal. **33**, 547–571 (1999)
26. Liu, H., Jiao, X.: WLS-ENO: Weighted-least-squares based essentially non-oscillatory schemes for finite volume methods on unstructured meshes. J. Comput. Phys. **314**, 749–773 (2016)
27. Liu, X.D., Osher, S., Chan, T.: Weighted essentially non-oscillatory schemes. J. Comput. Phys. **115**, 200–212 (1994)
28. Liu, Y., Shu, C.W., Zhang, M.: High order finite difference WENO schemes for nonlinear degenerate parabolic equations. SIAM Journal on Scientific Computing **33**(2), 939–965 (2011)
29. Shu, C.W.: Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. Tech. Rep. ICASE Report no. 97–65, National Aeronautics and Space Administration, Langley Research Center, Hampton, Virginia (1997)
30. Shu, C.W., Osher, S.: Efficient implementation of essentially non-oscillatory shock capturing schemes. J. Comput. Phys. **77**, 439–471 (1988)
31. Sonar, T.: On the construction of essentially non-oscillatory finite volume approximations to hyperbolic conservation laws on general triangulations: polynomial recovery, accuracy and stencil selection. Comput. Methods Appl. Mech. Engrg. **140**, 157–181 (1997)
32. Tsoutsanis, P.: Stencil selection algorithms for WENO schemes on unstructured meshes. J. Comput. Phys. **475**, 108840 (2023)