AN ADAPTIVE MULTIRESOLUTION DISCONTINUOUS GALERKIN METHOD WITH ARTIFICIAL VISCOSITY FOR SCALAR HYPERBOLIC CONSERVATION LAWS IN MULTIDIMENSIONS*

JUNTAO HUANG† AND YINGDA CHENG‡

Abstract. In this paper, we develop an adaptive multiresolution discontinuous Galerkin (DG) scheme for scalar hyperbolic conservation laws in multidimensions. Compared with previous work for linear hyperbolic equations [W. Guo and Y. Cheng, *SIAM J. Sci. Comput.*, 38 (2016), pp. A3381–A3409, W. Guo and Y. Cheng, *SIAM J. Sci. Comput.*, 39 (2017), pp. A2962–A2992], a class of interpolatory multiwavelets are applied to efficiently compute the nonlinear integrals over elements and edges in DG schemes. The resulting algorithm, therefore, can achieve similar computational complexity as the sparse grid DG method for smooth solutions. Theoretical and numerical studies are performed taking into consideration the accuracy and stability with regard to the choice of the interpolatory multiwavelets. Artificial viscosity is added to capture the shock and only acts on the leaf elements taking advantage of the multiresolution representation. Adaptivity is realized by auto error thresholding based on hierarchical surplus. Accuracy and robustness are demonstrated by several numerical tests.

Key words. discontinuous Galerkin methods, multiresolution analysis, sparse grids, hyperbolic conservation laws, artificial viscosity

AMS subject classification. 65M60

DOI. 10.1137/19M126565X

1. Introduction. In this paper, we develop an adaptive multiresolution discontinuous Galerkin (DG) method for scalar nonlinear conservation laws in the multidimensional case:

$$(1.1) \qquad \partial_t u + \nabla \cdot \mathbf{f}(u) = 0, \quad (\mathbf{x}, t) \in \Omega \times (0, T],$$

with appropriate initial and boundary conditions. Here $\Omega \subset \mathbb{R}^d$, $u = u(\mathbf{x}, t)$ is the unknown function, and $\mathbf{f}(u) = (f_1(u), f_2(u), \ldots, f_d(u))$ is the physical flux. We assume $\Omega = [0, 1]^d$ in this paper, but the discussion can be easily generalized to arbitrary box-shaped domains.

The DG method is a class of finite element methods using discontinuous approximation space for the numerical solutions and the test functions. The Runge–Kutta DG (RK-DG) scheme for hyperbolic equations became very popular due to its provable stability and convergence, excellent conservation properties, and accommodation for adaptivity and parallel implementations. We refer readers to the review papers [16, 14] for details. To adapt the degrees of freedom according to the local behavior of the numerical solution, many kinds of a posteriori error estimates have been designed for the DG schemes for hyperbolic equations; see, e.g., [6, 1, 48, 32, 34, 33]. On the

†Department of Mathematics, Michigan State University, East Lansing, MI 48824 (huangj75@msu.edu).

‡Department of Mathematics, Department of Computational Mathematics, Science and Engineering, Michigan State University, East Lansing, MI 48824 (ycheng@msu.edu).

other hand, by using multiresolution analysis (MRA), automatic adaptivity can be achieved and no additional a posteriori error indicator is needed. Such ideas have been used to accelerate the computations for conservation laws under finite difference or finite volume frameworks [31, 7, 19, 3, 18, 12] and were used as trouble cell indicators for DG methods [54]. In recent years, there has been interest in developing adaptive multiresolution DG schemes [11]. In particular, multiresolution-based adaptive DG schemes for solving one-dimensional scalar conservation laws were proposed by Hovhannisyan, Müller, and Schäfer in [35] and further extended to multidimensional cases [25, 22], compressible flows [37, 24], and shallow water equations [23, 38]. The key idea is to perform MRA using multiwavelets on a hierarchy of nested grids for the data given on a uniformly refined mesh. With such an approach, the scheme on a uniformly refined mesh is computed on a locally refined adapted subgrid while preserving the accuracy.

Another idea to utilize the computational advantages of the MRA framework is called the sparse grid method [10], which is a well-known tool to compute high-dimensional PDEs and stochastic differential equations. Based on the attractive features of DG methods for solving convection-dominated problems, in recent years, we initiated a line of research developing the (adaptive) sparse grid DG methods, including the work for elliptic equations [55], transport equations [28], reaction-diffusion equations [43], and Vlasov–Maxwell equations [52]. For smooth solutions, the schemes we constructed can successfully reduce the number of degrees of freedom (DoF) of the unknowns from $\mathcal{O}(h^{-d})$ to $\mathcal{O}(h^{-1}|\log_2 h|^{d-1})$ for d-dimensional problems, where h is the uniform mesh size in each dimension. Stability and conservation of standard DG methods can be maintained. Errors are only slightly deteriorated for smooth solutions. Adaptivity can be incorporated naturally to treat solutions with less smoothness or local structures. This is the line of research we continue in this paper.

However, the main bottleneck for the sparse grid DG scheme developed so far is that it is mainly for *linear* equations. Here, *linear* refers to either linear variable coefficient equations with given coefficients or coefficients that have some specified dependence on the unknowns, e.g., Vlasov systems through self-consistent field. There remain significant challenges to extend the methods to truly nonlinear problems in an efficient manner. For example, previous work in the literature on adaptive multiresolution DG schemes resort to the finest scale for the actual time evolution for the nonlinear terms. Therefore, the computational cost is proportional to the number of cells on the finest level, i.e., $\mathcal{O}(h^{-d})$ operations, and the reduced DoF in the solution representation is not realized in the actual computation. For nonlinear equations, there is only limited literature on collocation or finite difference based sparse grid methods [26], and the order of accuracy of the schemes is low. Sparse grid combination methods work for nonlinear problems, but they are less flexible in terms of adaptivity [42].

This work serves as a proof-of-concept for a systematic approach for the adaptive sparse grid DG method to solve nonlinear PDEs in high dimensions. We construct a scheme that can recover the computational complexity of the sparse grid method for smooth solutions. This is different from previous approaches for adaptive multiresolution DG methods [25, 4, 49]. The evolution in our scheme is carried out by the multiresolution basis functions, and it can recover the computational efficiency of sparse grid approaches for smooth solutions in high dimensions. In particular, our methods never convert the multiscale coefficients to single scale coefficients. This also raises many computational challenges as mentioned before. To compute nonlinear terms, we use sparse grid collocation methods introduced in [53], where the

idea is to design new multiwavelets associated with interpolation on nested grids and gives a framework to design adaptive sparse grid collocation onto arbitrary high order piecewise polynomial spaces. Moreover, the algorithm converting between the point values and the derivatives to the coefficients of the hierarchical wavelets can be performed efficiently by the fast wavelet transform [53]. In this paper, we approximate the nonlinear integral terms in the semidiscrete DG scheme by integrating a linear combination of collocation wavelets up to the desired order of accuracy. We analyze the truncation error of the DG scheme with the interpolation following the approach in [13, 36]. It is shown that for the interpolation we need polynomials of one degree higher than in the original DG function space if one would like to preserve the order of accuracy for the original (standard or sparse grid) DG scheme. Then, we compare different ways of sparse grid collocation methods and find that there exists some *instability* for Lagrange interpolation when solving (1.1) especially for DG space with higher degrees of polynomial (see Tables 2 and 3 in section 4). This motivates us to apply Hermite interpolation, which is more stable than the Lagrange interpolation [30]. Because the Alpert's multiwavelets and the interpolation multiwavelet bases are both global, the evaluation of the residual yields a denser matrix than those obtained by standard local bases. We use fast matrix-vector multiplications, e.g., those developed for sparse grid methods [50, 56], to recover efficient computational scaling for such calculations.

Another challenge we address in the paper is how to capture the shock and entropy solutions to (1.1). There are two approaches in the literature. The first one is to apply limiters to control spurious oscillations and at the same time maintain accuracy in smooth regions, e.g., the minmod-type limiter [15], the moment-based limiter [8], and the WENO limiter [46]. However, it is quite difficult to impose limiters in the sparse grid DG methods, due to the global feature of the basis functions. Also a preliminary calculation from us shows that the piecewise constant sparse grid DG method in multidimensions is not monotone. This motivates us to use the second approach, which is to add artificial viscosity; see, e.g., [5, 45, 27, 39, 40]. The idea is to add a diffusion term in the equation where the diffusion coefficient vanishes in the smooth region and becomes nonzero near the shock. This can be achieved by techniques such as entropy production [27] or local smoothness indicator [45]. We add an artificial viscosity term following the approach in [5]. Based on the estimate of the magnitudes of coefficients of hierarchical basis functions in [28, 29], we propose a smoothness indicator, which is built upon the inherent MRA and can automatically pick out the discontinuous regions. To improve the computational efficiency of our scheme, the implicit-explicit (IMEX) Runge–Kutta time integration is applied, where the nonlinear convection term is treated explicitly and the linear diffusion term is computed implicitly.

The rest of this paper is organized as follows. In section 2, we review MRA associated with two sets of basis functions, i.e., the Alpert's multiwavelets [2] and the interpolatory multiwavelets [53]. The adaptive multiresolution DG scheme is constructed in section 3 using both sets of multiwavelets. The numerical performance is validated by linear advection equations, Burgers' equations, and Kurganov, Petrova, and Popov (KPP) problems [41] in section 4. We conclude this paper in section 5. Appendices A and B collect the explicit formulas of the interpolatory multiwavelets used in this paper.

2. MRA and multiwavelets. In this section, we review MRA associated with piecewise polynomial space. We will start with Alpert's multiwavelets [2] and then

review the interpolatory multiwavelets [53].

2.1. Alpert's multiwavelets. We first review MRA achieved by Alpert's basis functions in one dimension [2]. We define a set of nested grids, where the nth level grid Ω_n consists of 2^n uniform cells

$$I_n^j = (2^{-n}j, 2^{-n}(j+1)], \quad j = 0, \ldots, 2^n - 1,$$

for $n \geq 0$. For notational convenience, we also denote $I_{-1} = [0, 1]$. The usual piecewise polynomial space of degree at most $k \geq 1$ on the nth level grid Ω_n for $n \geq 0$ is denoted by

$$(2.1) \qquad V_n^k := \{v : v \in P^k(I_n^j) \quad \forall j = 0, \ldots, 2^n - 1\}.$$

Then, we have the nested structure

$$V_0^k \subset V_1^k \subset V_2^k \subset V_3^k \subset \cdots.$$

We can now define the multiwavelet subspace W_n^k, $n = 1, 2, \ldots$, as the orthogonal complement of V_{n-1}^k in V_n^k with respect to the L^2 inner product on $[0, 1]$, i.e.,

$$V_{n-1}^k \oplus W_n^k = V_n^k, \quad W_n^k \perp V_{n-1}^k.$$

For notational convenience, we let $W_0^k := V_0^k$, which is the standard polynomial space of degree up to k on $[0, 1]$. Therefore, we have $V_n^k = \bigoplus_{0 \leq l \leq n} W_l^k$.

Now we define a set of orthonormal basis associated with the space W_l^k. The case of mesh level $l = 0$ is trivial. We use the normalized shifted Legendre polynomials in $[0, 1]$ and denote the basis by $v_{i,0}^0(x)$ for $i = 0, \ldots, k$. When $l > 0$, the orthonormal bases in W_l^k are presented in [2] and denoted by

$$v_{i,l}^j(x), \quad i = 0, \ldots, k, \quad j = 0, \ldots, 2^{l-1} - 1.$$

The construction follows a repeated Gram–Schmidt process, and the explicit expression of the multiwavelet basis functions is provided in Table 1 in [2]. Note that such multiwavelet bases retain the orthonormal property of wavelet bases for different mesh levels, i.e.,

$$(2.2) \qquad \int_0^1 v_{i,l}^j(x) v_{i',l'}^{j'}(x)\, dx = \delta_{ii'} \delta_{ii'} \delta_{jj'},$$

and the support of $v_{i,l}^j$ is in I_{l-1}^j.

Multidimensional case when $d > 1$ follows from a tensor-product approach. First, we recall some basic notation. For a multi-index $\alpha = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}_0^d$, where $\mathbb{N}_0$ denotes the set of nonnegative integers, the l^1 and l^∞ norms are defined as

$$|\boldsymbol{\alpha}|_1 := \sum_{m=1}^d \alpha_m, \qquad |\boldsymbol{\alpha}|_\infty := \max_{1 \leq m \leq d} \alpha_m.$$

The componentwise arithmetic operations and relational operations are defined as

$$\boldsymbol{\alpha} \cdot \boldsymbol{\beta} := (\alpha_1 \beta_1, \ldots, \alpha_d \beta_d), \qquad c \cdot \boldsymbol{\alpha} := (c\alpha_1, \ldots, c\alpha_d), \qquad 2^{\boldsymbol{\alpha}} := (2^{\alpha_1}, \ldots, 2^{\alpha_d}),$$

$$\boldsymbol{\alpha} \leq \boldsymbol{\beta} \Leftrightarrow \alpha_m \leq \beta_m \quad \forall m, \quad \boldsymbol{\alpha} < \boldsymbol{\beta} \Leftrightarrow \boldsymbol{\alpha} \leq \boldsymbol{\beta} \text{ and } \boldsymbol{\alpha} \neq \boldsymbol{\beta}.$$

By making use of the multi-index notation, we denote by $\mathbf{l} = (l_1, \ldots, l_d) \in \mathbb{N}_0^d$ the mesh level in a multivariate sense. We define the tensor-product mesh grid $\Omega_{\mathbf{l}} = \Omega_{l_1} \otimes \cdots \otimes \Omega_{l_d}$ and the corresponding mesh size $h_{\mathbf{l}} = (h_{l_1}, \ldots, h_{l_d})$. Based on the grid $\Omega_{\mathbf{l}}$, we denote $I_{\mathbf{l}}^{\mathbf{j}} = \{\mathbf{x} : x_m \in (h_m j_m, h_m(j_m + 1)), m = 1, \ldots, d\}$ as an elementary cell, and

$$\mathbf{V}_{\mathbf{l}}^k := \{\mathbf{v} : \mathbf{v} \in Q^k(I_{\mathbf{l}}^{\mathbf{j}}), \ \mathbf{0} \leq \mathbf{j} \leq 2^{\mathbf{l}} - \mathbf{1}\} = V_{l_1, x_1}^k \times \cdots \times V_{l_d, x_d}^k$$

as the tensor-product piecewise polynomial space, where $Q^k(I_{\mathbf{l}}^{\mathbf{j}})$ represents the collection of polynomials of degree up to k in each dimension on cell $I_{\mathbf{l}}^{\mathbf{j}}$. If we use equal mesh refinement of size $h_N = 2^{-N}$ in each coordinate direction, the grid and space will be denoted by Ω_N and $\mathbf{V}_N^k$, respectively.

Based on a tensor-product construction, the multidimensional increment space can be defined as

$$\mathbf{W}_{\mathbf{l}}^k = W_{l_1, x_1}^k \times \cdots \times W_{l_d, x_d}^k.$$

Therefore, the standard tensor-product piecewise polynomial space on Ω_N can be written as

$$(2.3) \qquad \mathbf{V}_N^k = \bigoplus_{\substack{|\mathbf{l}|_\infty \leq N \\ \mathbf{l} \in \mathbb{N}_0^d}} \mathbf{W}_{\mathbf{l}}^k,$$

while the sparse grid approximation space in [55] is

$$(2.4) \qquad \hat{\mathbf{V}}_N^k := \bigoplus_{\substack{|\mathbf{l}|_1 \leq N \\ \mathbf{l} \in \mathbb{N}_0^d}} \mathbf{W}_{\mathbf{l}}^k \subset \mathbf{V}_N^k.$$

The dimension of $\hat{\mathbf{V}}_N^k$ scales as $O((k+1)^d 2^N N^{d-1})$ [55], which is significantly less than that of $\mathbf{V}_N^k$ with exponential dependence on Nd. The approximation results for $\hat{\mathbf{V}}_N^k$, which has a stronger smoothness requirement than the traditional $\mathbf{V}_N^k$ space, are discussed in [55, 28]. In this paper, we will not require the numerical solution to be in $\hat{\mathbf{V}}_N^k$, but rather in $\mathbf{V}_N^k$ and to be chosen adaptively similar to [29].

The basis functions in multidimensions are defined as

$$(2.5) \qquad v_{\mathbf{i}, \mathbf{l}}^{\mathbf{j}}(\mathbf{x}) := \prod_{m=1}^d v_{i_m, l_m}^{j_m}(x_m)$$

for $\mathbf{l} \in \mathbb{N}_0^d$, $\mathbf{j} \in B_{\mathbf{l}} := \{\mathbf{j} \in \mathbb{N}_0^d : \mathbf{0} \leq \mathbf{j} \leq \max(2^{\mathbf{l}-\mathbf{1}} - \mathbf{1}, \mathbf{0})\}$, and $\mathbf{1} \leq \mathbf{i} \leq \mathbf{k} + \mathbf{1}$. The orthonormality of the bases can be established by (2.2).

There exists the estimate of the coefficients of hierarchical basis functions in [28, 29], which is also a classical result from wavelet theory and holds in a more general context [17]. It is shown in [28, 29] that for a function $u \in H^{p+1}(\Omega)$,

$$(2.6) \qquad \left(\sum_{\substack{\mathbf{0} \leq \mathbf{j} \leq \max(2^{\mathbf{l}-\mathbf{1}} - \mathbf{1}, \mathbf{0}) \\ \mathbf{1} \leq \mathbf{i} \leq \mathbf{k}+\mathbf{1}}} |u_{\mathbf{i}, \mathbf{l}}^{\mathbf{j}}|^2 \right)^{1/2} \leq C 2^{-(q+1)|\mathbf{l}|_1} |u|_{H^{p+1}(\Omega)},$$

where $q = \min\{p, k\}$, and C is a constant independent of mesh level $\mathbf{l}$. Therefore, by assuming that $u \in W^{p+1, \infty}(I_{\mathbf{l}}^{\mathbf{j}})$, we can obtain a local estimate in each element on

mesh level $\mathbf{l}$: for any index $\mathbf{j}$,

$$(2.7) \qquad \left(\sum_{\mathbf{1}\leq\mathbf{i}\leq\mathbf{k+1}} |u_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}|^2\right)^{1/2} \leq C 2^{-(q+\frac{1}{2})|\mathbf{l}|_1} |u|_{W^{p+1,\infty}(\Omega)}.$$

Therefore, for sufficiently smooth functions, the coefficients should decay like

$$(2.8) \qquad \left(\sum_{\mathbf{1}\leq\mathbf{i}\leq\mathbf{k+1}} |u_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}|^2\right)^{1/2} \sim 2^{-(k+\frac{1}{2})|\mathbf{l}|_1}.$$

This will be used in the construction of the smoothness indicator in section 3.

2.2. Interpolatory multiwavelets. Alpert's multiwavelets and the space W_l^k are constructed so that they correspond to the difference of the L^2 projection on adjacent levels. The idea of the sparse grid collocation basis proposed in [53] is to switch the operator to be interpolation on nested grids. Below, we will outline the construction. Denote the set of interpolation points in the interval $I = [0,1]$ at mesh level 0 by $X_0 = \{x_i\}_{i=0}^P \subset I$. Here, the number of points in X_0 is $(P+1)$. Then the interpolation points at mesh level $n \geq 1$, X_n, can be obtained correspondingly as

$$X_n = \{x_{i,n}^j := 2^{-n}(x_i + j), \quad i = 0,\ldots,P, \quad j = 0,\ldots,2^n - 1\}.$$

We require the points to be nested, i.e.,

$$(2.9) \qquad X_0 \subset X_1 \subset X_2 \subset X_3 \subset \cdots,$$

to save computational cost. This can be achieved by requiring $X_0 \subset X_1$, and then one can deduce (2.9) easily.

Given the nodes, we define the basis functions on the 0th level grid as Lagrange ($K = 0$) or Hermite ($K \geq 1$) interpolation polynomials of degree $\leq M := (P+1)(K+1) - 1$ which satisfy the property

$$\phi_{i,l}^{(l')}(x_{i'}) = \delta_{ii'}\delta_{ll'}$$

for $i, i' = 0,\ldots,P$ and $l, l' = 0,\ldots,K$. It is easy to see that $\mathrm{span}\{\phi_{i,l}, i = 0,\ldots,P, l = 0,\ldots,K\} = V_0^M$. The constants P, K, M will be specified later in this paper. With the basis function at mesh level 0, we can define basis function at mesh level $n \geq 1$:

$$\phi_{i,l,n}^j := 2^{-nl}\phi_{i,l}(2^n x - j), \quad i = 0,\ldots,P, \quad l = 0,\ldots,K, \quad j = 0,\ldots,2^n - 1,$$

which is a complete basis set for V_n^M.

Next, we introduce the hierarchical representations. Define $\tilde{X}_0 := X_0$ and $\tilde{X}_n := X_n \backslash X_{n-1}$ for $n \geq 1$; then we have the decomposition

$$X_n = \tilde{X}_0 \cup \tilde{X}_1 \cup \cdots \cup \tilde{X}_n.$$

Denote the points in $\tilde{X}_1$ by $\tilde{X}_1 = \{\tilde{x}_i\}_{i=0}^P$. Then the points in $\tilde{X}_n$ for $n \geq 1$ can be represented by

$$\tilde{X}_n = \{\tilde{x}_{i,n}^j := 2^{-(n-1)}(\tilde{x}_i + j), \quad i = 0,\ldots,P, \quad j = 0,\ldots,2^{n-1} - 1\}.$$

For notational convenience, we let $\tilde{W}_0^M := V_0^M$. The increment function space $\tilde{W}_n^M$ for $n \geq 1$ is introduced as a function space that satisfies

$$(2.10) \qquad\qquad V_n^M = V_{n-1}^M \oplus \tilde{W}_n^M,$$

and is defined through the multiwavelets $\psi_{i,l} \in V_1^M$ that satisfy

$$\psi_{i,l}^{(l')}(x_{i'}) = 0, \quad \psi_{i,l}^{(l')}(\tilde{x}_{i'}) = \delta_{i,i'}\delta_{l,l'}$$

for $i, i' = 0, \ldots, P$ and $l, l' = 0, \ldots, K$. Here the superscript (l') denotes the l'th order derivative. Then W_n^M is given by

$$\tilde{W}_n^M = \mathrm{span}\{\psi_{i,l,n}^j, \quad i = 0, \ldots, P, \quad l = 0, \ldots, K, \quad j = 0, \ldots, 2^{n-1} - 1\},$$

where $\psi_{i,l,n}^j(x) := 2^{-(n-1)l}\psi_{i,l}(2^{n-1}x - j)$. For completeness, we list the basis functions used in this paper in Appendices A and B .

The construction above has close connection with interpolation operators. For a given function $f(x) \in C^{K+1}(I)$, we define $\mathcal{I}_N^{P,K}[f]$ as the standard Hermite interpolation on V_N^M, and we have the representation

$$\mathcal{I}_N^{P,K}[f](x) = \sum_{n=0}^{N} \sum_{j=0}^{\max(2^{n-1}-1,0)} \sum_{l=0}^{K} \sum_{i=0}^{P} b_{i,l,n}^j \psi_{i,l,n}^j(x).$$

Clearly, $(\mathcal{I}_n^{P,K} - \mathcal{I}_{n-1}^{P,K})[f](x) \in \tilde{W}_n^M$. The algorithm converting between the point values and the derivatives $\{f^{(l)}(x_{i,n}^j)\}$ to hierarchical coefficients $\{b_{i,l,n}^j\}$ is given in [53], and by a standard argument in fast wavelet transform, can be performed in $O(M2^n)$ flops.

The multidimensional construction follows similar lines as in section 2.1. We let

$$\tilde{\mathbf{W}}_{\mathbf{l}}^M = \tilde{W}_{l_1,x_1}^M \times \cdots \times \tilde{W}_{l_d,x_d}^M.$$

Therefore,

$$\mathbf{V}_N^M = \bigoplus_{\substack{|\mathbf{l}|_\infty \leq N \\ \mathbf{l} \in \mathbb{N}_0^d}} \tilde{\mathbf{W}}_{\mathbf{l}}^M,$$

while the sparse grid approximation space is

$$\hat{\mathbf{V}}_N^M = \bigoplus_{\substack{|\mathbf{l}|_1 \leq N \\ \mathbf{l} \in \mathbb{N}_0^d}} \tilde{\mathbf{W}}_{\mathbf{l}}^M.$$

Note that the construction by Alpert's multiwavelet and the interpolatory multiwavelet gives the same sparse grid space. Finally, the interpolation operator in multidimension $\mathcal{I}_N^{P,K} : C^{K+1}(\Omega) \to \mathbf{V}_N^M$ is

$$\mathcal{I}_N^{P,K}[f](\mathbf{x}) = \sum_{\substack{|\mathbf{n}|_\infty \leq N \\ \mathbf{0} \leq \mathbf{j} \leq \max(2^{\mathbf{n}-1}-\mathbf{1},\mathbf{0}) \\ \mathbf{0} \leq \mathbf{l} \leq \mathbf{K} \\ \mathbf{0} \leq \mathbf{i} \leq \mathbf{P}}} b_{\mathbf{i},\mathbf{l},\mathbf{n}}^{\mathbf{j}} \psi_{\mathbf{i},\mathbf{l},\mathbf{n}}^{\mathbf{j}}(\mathbf{x}),$$

where the multidimensional basis functions $\psi_{\mathbf{i,l,n}}^{\mathbf{j}}(\mathbf{x})$ are defined in the same approach as (2.5) by tensor products:

$$(2.11) \qquad \psi_{\mathbf{i,l,n}}^{\mathbf{j}}(\mathbf{x}) := \prod_{m=1}^{d} \psi_{i_m,l_m,n_m}^{j_m}(x_m).$$

If the space is switched from $\mathbf{V}_N^M$ to some subset of $\mathbf{V}_N^M$, e.g., the sparse grid space $\hat{\mathbf{V}}_N^M$ or some other subset of $\mathbf{V}_N^M$ that is dynamically chosen, the interpolation operator can be defined accordingly, taking only multiwavelet basis functions that belong to that space.

3. Adaptive multiresolution DG evolution algorithm. In this section, we will describe the adaptive multiresolution DG scheme for (1.1). We will first introduce the DG scheme with multiresolution interpolation. The accuracy requirement for the interpolation operator is studied by local truncation error analysis. We then describe the adaptive strategy. Finally, the artificial viscosity is introduced based on the estimate of the coefficients of the hierarchical basis functions.

3.1. DG scheme with multiresolution interpolation. First, we review some basis notations about meshes. Let N be the maximum mesh level, and let T_h be the collection of all elementary cells $I_N^{\mathbf{j}}$, $0 \leq j_m \leq 2^N - 1 \ \forall m = 1, \ldots, d$. Define $\Gamma_h := \bigcup_{K \in T_h} \partial K$ to be the union of all the interfaces for all the elements in T_h. Here, for simplicity, we formulate the scheme with periodic boundary conditions, while we keep in mind other boundary conditions can be treated in the DG framework as well.

The semidiscrete DG scheme for the scalar conservation law reads as [15]

$$(3.1) \qquad \sum_{K \in T_h} \int_K (u_h)_t v_h dx - \sum_{K \in T_h} \int_K f(u_h) \cdot \nabla v_h dx + \sum_{e \in \Gamma_h} \int_e \widehat{f \cdot n_K}(u_h) v_h ds = 0.$$

Here, u_h is the numerical solution and v_h is the test function. The numerical flux $\widehat{f \cdot n_K}(u_h) \equiv \widehat{f \cdot n_K}(u_h^{int}, u_h^{ext})$ is taken to be the global Lax–Friedrichs flux:

$$(3.2) \qquad \widehat{f \cdot n_K}(a, b) = \frac{1}{2} (f(a) + f(b)) \cdot n_K - \frac{1}{2} C(b - a),$$

where $C = \max_u |n_K \cdot f'(u)|$ and the maximum is taken over the whole domain. Note that the local Lax–Friedrichs flux can also be used with additional efforts in numerical interpolation. u_h, v_h belong to the same function space $\mathbf{V}^k$. If $\mathbf{V}^k = \mathbf{V}_N^k$, we recover the standard (or full grid) DG method. If $\mathbf{V}^k = \hat{\mathbf{V}}_N^k$, we obtain the sparse grid DG method. In this paper, we will take $\mathbf{V}^k$ as a subset of $\mathbf{V}_N^k$ that is chosen adaptively as outlined in section 3.2.

In DG methods, the integrals over elements and edges are often approximated by numerical quadrature rules on each cell [13]. However, in the sparse grid DG method, this naive approach would result in computational cost that is proportional to the number of fundamental elements, i.e., $\mathcal{O}(h^{-d})$, and is still subject to the curse of dimensionality. To evaluate the integrals over elements and edges more efficiently with a cost proportional to the DoF of the underlying finite element space, we interpolate the nonlinear function $f(u_h)$ by using the multiresolution Lagrange (or Hermite) interpolation basis functions introduced in section 2.2. Therefore, the semidiscrete DG scheme with interpolation is

$$(3.3) \qquad \sum_{K \in T_h} \int_K (u_h)_t v_h dx - \sum_{K \in T_h} \int_K \mathcal{I}[f(u_h)] \cdot \nabla v_h dx + \sum_{e \in \Gamma_h} \int_e \mathcal{I}[\widehat{f \cdot n_K}(u_h)] v_h ds = 0,$$

where $\mathcal{I}[\cdot]$ is a multiresolution interpolation operator onto some finite element space with the same multiresolution structure as $\mathbf{V}^k$, but of polynomial degree M. The choice of $\mathcal{I}[\cdot]$ will be specified later, which plays important roles in numerical stability and accuracy. Note that the numerical flux $\widehat{f \cdot n_K}(u_h)$ is only defined at edges; thus it remains to clarify the meaning of the interpolation $\mathcal{I}[\widehat{f \cdot n_K}(u_h)]$. Since we use the global Lax–Friedrichs flux (3.2), we have

$$
\begin{aligned}
\mathcal{I}[\widehat{f \cdot n_K}(u_h^{int}, u_h^{ext})] &= \frac{1}{2}\left(\mathcal{I}[f(u_h^{int})] + \mathcal{I}[f(u_h^{ext})]\right) \cdot n_K - \frac{1}{2}C(\mathcal{I}[u_h^{ext}] - \mathcal{I}[u_h^{int}]) \\
&= \frac{1}{2}\left(\mathcal{I}[f(u_h^{int})] + \mathcal{I}[f(u_h^{ext})]\right) \cdot n_K - \frac{1}{2}C(u_h^{ext} - u_h^{int})
\end{aligned}
$$

due to the linearity of the interpolation operator $\mathcal{I}[\cdot]$. Therefore, we need only obtain the interpolation $\mathcal{I}[f(u_h)]$ and then read the value on two sides of the edges to obtain $\mathcal{I}[f(u_h^{int})]$ and $\mathcal{I}[f(u_h^{ext})]$. Now, we discuss numerical implementation. First, we read the (derivative) values of u_h, which is a linear combination of Alpert's basis functions at the chosen interpolation points. Second, we calculate the (derivative) values of $f(u_h)$ at these interpolation points. Last, we transfer the (derivative) values to coefficients of interpolation basis, by using the algorithm introduced in [53]. At this point, the numerical integrations can be performed through a fast matrix-vector product as in [50]. We remark that the computational cost does not increase too much compared to the multiresolution DG schemes for linear equations introduced in [29]. The cost of the transformation from the (derivative) values to hierarchical coefficients is only *linearly* dependent on the dimension d [53].

Now we discuss the choice of $\mathcal{I}[\cdot]$. To preserve the accuracy of the original DG scheme (3.1), it is required that the interpolation operator $\mathcal{I}[\cdot]$ reaches certain accuracy. Following [13], we rewrite the weak formulation (3.3) in the ODE form as

$$
(3.4) \qquad \frac{du_h}{dt} = L_h(u_h),
$$

where $L_h(u)$ is an operator onto $\mathbf{V}^k$, which is a discrete approximation of $-\nabla \cdot f(u)$ and satisfies

$$
(3.5) \quad \sum_{K \in T_h} \int_K L_h(u_h)v_h dx = \sum_{K \in T_h} \int_K \mathcal{I}[f(u_h)] \cdot \nabla v_h dx - \sum_{e \in \Gamma_h} \int_e \mathcal{I}[\widehat{f \cdot n_K}(u_h)]v_h ds.
$$

To illustrate the ideas, we only consider the full grid or sparse grid DG methods, i.e., $\mathbf{V}^k = \mathbf{V}_N^k$ or $\mathbf{V}^k = \hat{\mathbf{V}}_N^k$. For adaptive methods, similar intuitive arguments can be made, but a rigorous proof is much harder. Using similar error estimates techniques in [13, 36], we have the following proposition on local truncation error.

PROPOSITION 3.1 (accuracy of semidiscrete DG scheme with interpolation). *Assume that the DG finite element space (standard or sparse) has polynomials up to degree k. If the interpolation operator in (3.3) has the accuracy of h^{k+2} (standard) or $|\log_2 h|^d h^{k+2}$ (sparse) for sufficiently smooth functions, then the truncation error of the semidiscrete DG scheme with interpolation (3.3) is of order h^{k+1} (standard) or $|\log_2 h|^d h^{k+1}$ (sparse). To be more precise, for sufficiently smooth function u, the standard DG with interpolation (3.3) has the truncation error*

$$
(3.6) \qquad \|L_h(u) + \nabla \cdot f(u)\|_{L^2(\Omega)} \leq Ch^{k+1},
$$

and the sparse grid DG with interpolation (3.3) *has the truncation error*

$$(3.7) \qquad \|L_h(u) + \nabla \cdot f(u)\|_{L^2(\Omega)} \le C \left|\log_2 h\right|^d h^{k+1}.$$

Here, the constant C may depend on the solution, but does not depend on h.

Proof. To save space, we only show the proof for full grid DG space $\mathbf{V}^k = \mathbf{V}_N^k$. A similar technique also applies to the sparse grid DG space using projection error estimates in [28].

We denote the standard L^2 projection operator onto the standard DG finite element space by $\mathbb{P}$; then

$$(3.8) \qquad \|L_h(u) + \nabla \cdot f(u)\|_{L^2(\Omega)} \le e_1 + e_2,$$

where

$$e_1 := \|L_h(u) + \mathbb{P}(\nabla \cdot f(u))\|_{L^2(\Omega)}$$

and

$$e_2 := \|\mathbb{P}(\nabla \cdot f(u)) - \nabla \cdot f(u)\|_{L^2(\Omega)}.$$

The estimate for e_2 is trivial using projection properties:

$$(3.9) \qquad e_2 \le Ch^{k+1} \|\nabla \cdot f(u)\|_{H^{k+1}(\Omega)}.$$

To estimate e_1, we consider any test function v_h in DG space, and obtain

$$
\begin{aligned}
\sum_{K \in T_h} & \int_K (L_h(u) + \mathbb{P}(\nabla \cdot f(u)))v_h = \sum_{K \in T_h} \int_K (L_h(u) + \nabla \cdot f(u))v_h \\
&= \sum_{K \in T_h} \int_K \mathcal{I}[f(u)] \cdot \nabla v_h - \sum_{e \in \Gamma_h} \int_e \mathcal{I}[\widehat{f \cdot n_K}(u)] \cdot n_K v_h ds \\
&\quad - \sum_{K \in T_h} \int_K (f(u) \cdot \nabla v_h) + \sum_{e \in \Gamma_h} \int_e f(u) \cdot n_K v_h ds \\
&= \sum_{K \in T_h} \int_K (\mathcal{I}[f(u)] - f(u)) \cdot \nabla v_h - \sum_{e \in \Gamma_h} \int_e (\mathcal{I}[f(u) \cdot n_K] - f(u) \cdot n_K)v_h ds \\
&\le \|\mathcal{I}[f(u)] - f(u)\|_{L^2(\Omega)} \|\nabla v_h\|_{L^2(\Omega)} + \|\mathcal{I}[f(u)] - f(u)\|_{L^2(\Gamma_h)} \|v_h\|_{L^2(\Gamma_h)} \\
&\le Ch^{k+2}h^{-1} \|v_h\|_{L^2(\Omega)} + Ch^{-\frac{1}{2}}h^{k+2}h^{-\frac{1}{2}} \|v_h\|_{L^2(\Omega)} \\
&= Ch^{k+1} \|v_h\|_{L^2(\Omega)}.
\end{aligned}
$$

Here we use the multiplicative trace inequality and the inverse inequality; see, e.g., Lemmas 2.1 and 2.3 in [36]. We take v_h to be $(L_h(u) + \mathbb{P}(\nabla \cdot f(u)))$ in the inequality above and have

$$\|L_h(u) + \mathbb{P}(\nabla \cdot f(u))\|_{L^2(\Omega)}^2 \le Ch^{k+1} \|L_h(u) + \mathbb{P}(\nabla \cdot f(u))\|_{L^2(\Omega)}$$

and eventually arrive at

$$(3.10) \qquad e_1 = \|L_h(u) + \mathbb{P}(\nabla \cdot f(u))\|_{L^2(\Omega)} \le Ch^{k+1}.$$

Combining (3.10) and (3.9), we have the estimate for the truncation error (3.6). $\quad\square$

Remark 3.2. From the proposition above, we find that for the interpolation we need polynomials of one degree higher than in the original DG function space, if one would like to preserve the order of accuracy for the original (standard or sparse grid) DG scheme, i.e., we shall require $M \geq k + 1$. For example, if we take quadratic polynomials for the DG space, then it is required to apply a cubic interpolation operator (Lagrange or Hermite interpolation) to treat the nonlinear terms. From our numerical test, it seems that it is not a necessary condition for the standard DG method, but it is necessary for the sparse grid DG method.

Remark 3.3. For the standard DG method, the collocation in our scheme for the volume integrals should be equivalent to some quadrature formula depending on which interpolation operator is used. However, our method is not standard for the evaluation of the interfacial terms, which use point values (or derivatives) that may be *inside* the elements and not just on the interface.

In Proposition 3.1, we only estimate the truncation error, which is far from a rigorous error estimate that takes into account stability. In numerical experiments, we observe that the standard DG is stable with the Lagrange interpolation. However, the sparse grid DG with Lagrange interpolation is *unstable* and will blow up with very fine mesh for polynomials of high degrees (see the numerical results in Tables 2 and 3 in section 4). With Hermite interpolation, the sparse grid DG scheme is more stable and produces satisfactory convergence rate (see Table 4 in section 4). Actually, for standard DG with quadrature rules applied in each element, if the truncation error satisfies the required order of accuracy, then the convergence and error estimate can be guaranteed [36]. However, it is not true for the sparse grid DG method from our numerical experiments. This indicates that the standard DG method is more stable than the sparse grid DG method in this sense. We also remark that, since the interpolation operator introduced here is global but not local, the approach in [36] would probably fail to obtain the rigorous error estimate here. We will leave the detailed analysis for future work.

3.2. Adaptivity. In this section, we review the adaptive procedure introduced in [9, 29] to determine the space $\mathbf{V}^k$. The method is very similar to those in [9, 29], except that two sets of basis functions are involved and they are adaptively chosen at the same time.

In the adaptive DG algorithm, we specify the maximum mesh level N and an accuracy threshold $\epsilon > 0$. The same adaptive multiresolution projection method in [29] is applied here as the numerical initial condition for DG schemes. The error indicator using an L^2 norm is used. The details are omitted, and we refer readers to Algorithm 1 in [29].

The scheme is implemented by hash table as the underlying data structure. We now introduce the concepts of child, parent, and leaf elements. If an element $V_{\mathbf{l}'}^{\mathbf{j}'}$ with $|\mathbf{l}'|_\infty \leq N$ satisfies the condition that there exists an integer m such that $1 \leq m \leq d$ and $\mathbf{l}' = \mathbf{l} + \mathbf{e}_m$, where $\mathbf{e}_m$ denotes the unit vector in the x_m direction, and the support of $V_{\mathbf{l}'}^{\mathbf{j}'}$ is within that of $V_{\mathbf{l}}^{\mathbf{j}}$, then $V_{\mathbf{l}'}^{\mathbf{j}'}$ is called a child element of $V_{\mathbf{l}}^{\mathbf{j}}$. Accordingly, element $V_{\mathbf{l}}^{\mathbf{j}}$ is called a parent element of $V_{\mathbf{l}'}^{\mathbf{j}'}$. If an element does not have its child element in the hash table, then we call it a leaf element.

The time evolution consists of four steps. The first step is the prediction step, which means given the hash table H that stores the numerical solution u_h at time step t^n and the associated leaf table L, we need to predict the location where the details becomes significant at the next time step t^{n+1}, then add more elements in

order to capture the fine structures. We solve for $u_h \in \mathbf{V}_{N,H}^k$ from t^n to t^{n+1} using a cheap solver, e.g., the forward Euler discretization. Here, the interpolation operator $\mathcal{I}[\cdot]$ is determined by the accuracy requirement, and has the same multiresolution structure as determined by the hash table H corresponding to the numerical solution u_h. The predicted solution at t^{n+1} is denoted by $u_h^{(p)}$. Note that to save cost, the artificial viscosity term as introduced in section 3.4 does not need to be included in the prediction step.

The second step is the refinement step according to the predicted solution $u_h^{(p)}$. We traverse the hash table H and if an element $V_{\mathbf{l}}^{\mathbf{j}}$ satisfies the refinement criteria

$$(3.11) \qquad \left(\sum_{\mathbf{1} \leq \mathbf{i} \leq \mathbf{k+1}} |u_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}|^2 \right)^{1/2} \geq \epsilon,$$

where $u_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}$ denotes the hierarchical coefficient corresponding to the basis $v_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}(\mathbf{x})$, i.e., $u_{\mathbf{i},\mathbf{l}}^{\mathbf{j}} = \int_\Omega u(\mathbf{x}) v_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}(\mathbf{x}) d\mathbf{x}$. Equation (3.11) indicates that such an element becomes significant at the next time step; then we need to refine the mesh by adding its child elements to H. The detailed procedure is described as follows. For a child element $V_{\mathbf{l}'}^{\mathbf{j}'}$ of $V_{\mathbf{l}}^{\mathbf{j}}$, if it has been already added to H, i.e., $V_{\mathbf{l}'}^{\mathbf{j}'} \in H$, we do nothing; if not, we add the element $V_{\mathbf{l}'}^{\mathbf{j}'}$ to H and set the associated detail coefficients $u_{\mathbf{i},\mathbf{l}'}^{\mathbf{j}'} = 0$, $\mathbf{1} \leq \mathbf{i} \leq \mathbf{k+1}$. Moreover, we need to increase the number of children by one for every element that has $V_{\mathbf{l}'}^{\mathbf{j}'}$ as its child element and remove the parent elements of $V_{\mathbf{l}'}^{\mathbf{j}'}$ from the leaf table if they have been added. Finally, we obtain a larger hash table $H^{(p)}$ and the associated approximation space $\mathbf{V}_{N,H^{(p)}}^k$ and the leaf table $L^{(p)}$.

Then, based on the updated hash table $H^{(p)}$, we evolve the numerical solution by the DG formulation with space $\mathbf{V}_{N,H^{(p)}}^k$. Namely, we solve for $\mathbf{V}_{N,H^{(p)}}^k$ from t^n to t^{n+1}, to generate the precoarsened solution $\tilde{u}_h^{n+1}$, by using the the accurate solver with artificial viscosity in section 3.4. Here, the interpolation operator should be determined by the updated hash table $H^{(p)}$. Note that in the artificial viscosity $\nu = \nu(u_h)$ we fix u_h to be u_h^n such that the matrix for the diffusion term need only be resembled once in each time step.

The last step is to coarsen by removing elements that become insignificant at time level t^{n+1}. The hash table $H^{(p)}$ that stores the numerical solution $\tilde{u}_h^{n+1}$ is recursively coarsened by the following procedure. The leaf table $L^{(p)}$ is traversed, and if an element $V_{\mathbf{l}}^{\mathbf{j}} \in L^{(p)}$ satisfies the coarsening criterion

$$(3.12) \qquad \left(\sum_{\mathbf{1} \leq \mathbf{i} \leq \mathbf{k+1}} |u_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}|^2 \right)^{\frac{1}{2}} < \eta,$$

where η is a prescribed error constant, then we remove the element from both tables $L^{(p)}$ and $H^{(p)}$, and set the associated coefficients $u_{\mathbf{i},\mathbf{l}'}^{\mathbf{j}'} = 0$, $\mathbf{1} \leq \mathbf{i} \leq \mathbf{k+1}$. For each of its parent elements in table $H^{(p)}$, we decrease the number of children by one. If the number becomes zero, i.e., the element has no child any longer, then it is added to the leaf table $L^{(p)}$ accordingly. Repeat the coarsening procedure until no element can be removed from the table $L^{(p)}$. By removing only the leaf element at each time, we avoid generating "holes" in the hash table. The output of this coarsening procedure are the updated hash table and leaf table, denoted by H and L, respectively, and the

compressed numerical solution $u_h^{n+1} \in \mathbf{V}_{N,H}^k$. In practice, η is chosen to be smaller than ε for safety. In the simulations presented in this paper, we use $\eta = \varepsilon/10$.

For smooth solutions, the adaptive grids automatically reduce to the sparse grid methods as shown in [29]. We also comment that the adaptive time evolution procedure is stable if the high order RK-DG procedure is stable. This is because (1) in the prediction step, no change to numerical solution is made; (2) in the refinement step, we only add "zero" to the solution at t^n, so it does not add energy; (3) in the evolution step, we use a stable RK-DG method; (4) in the coarsening step, we remove the coefficients when measured in orthogonal multiwavelet bases, so the L^2 energy is guaranteed to decay.

Remark 3.4. The purpose of adaptive refinement is to recover the sparse grid methods when the dimension is bigger than one, hence "breaking the curse of dimensionality" as the best one can under this framework. In other words, we are not trying to match the performance of the scheme on the full grid, but rather the one on the sparse grid (which has larger errors than the full grid method). Therefore, we do not need to pick up an optimal value of ϵ which depends on the refinement level N to match the full grid.

3.3. Fast computations of the residual. We now describe the details of the computation of the right-hand side of DG weak formulation (3.3). This is important because the multiwavelet bases are global, and the evaluation of the residual yields a denser matrix than those obtained by standard local bases. Our work extends the fast matrix-vector multiplication in [50, 56] to an adaptive index set. Consider matrix-vector multiplication in multidimensions in the following form:

$$(3.13) \qquad f_{\boldsymbol{n}} = \sum_{H(\boldsymbol{n}') \leq 0} f'_{\boldsymbol{n}'} t^{(1)}_{n_1', n_1} t^{(2)}_{n_2', n_2} \cdots t^{(d)}_{n_d', n_d}, \quad H(\boldsymbol{n}) \leq 0,$$

where $\boldsymbol{n} = (n_1, n_2, \ldots, n_d)$ and $\boldsymbol{n}' = (n_1', n_2', \ldots, n_d')$ can be thought of as the level of the mesh, and $t^{(i)}_{n_1', n_1} = T^{(i)}_{n_1', n_1}$ represents the calculations in the ith dimension. It is assumed that the constraint function $H = H(\boldsymbol{n}') = H(n_1', n_2', \ldots, n_d')$ is nondecreasing with respect to each variable. This holds true for sparse grid (by taking $H(\boldsymbol{n}') = |\boldsymbol{n}'|_1$) and also for adaptive grid considered in this work.

One can compute the sum (3.13) dimension-by-dimension, i.e., first perform the transformation in the x_1 dimension:

$$(3.14) \qquad g^{(1)}_{(n_1, n_2', \ldots, n_d')} = \sum_{H(n_1', n_2', \ldots, n_d') \leq 0} f'_{(n_1', n_2', \ldots, n_d')} t^{(1)}_{n_1', n_1},$$

then in the x_2 dimension:

$$(3.15) \qquad g^{(2)}_{(n_1, n_2, n_3' \ldots, n_d')} = \sum_{H(n_1, n_2', \ldots, n_d') \leq 0} g^{(1)}_{(n_1, n_2', \ldots, n_d')} t^{(2)}_{n_2', n_2},$$

and all the way up to the x_d dimension:

$$(3.16) \qquad f_{(n_1, n_2, n_3 \ldots, n_d)} = \sum_{H(n_1, n_2, \ldots, n_{d-1}, n_d') \leq 0} g^{(d-1)}_{(n_1, n_2, \ldots, n_{d-1}, n_d')} t^{(d)}_{n_d', n_d}.$$

It can be proved that (3.14)–(3.16) is equivalent to the original summation (3.13), if assuming that, for some integer $1 \leq k \leq d$, $T^{(i)}$ for $i = 1, \ldots, k-1$ are strictly block

lower triangular and $T^{(i)}$ for $i = k + 1, \ldots, d$ are block upper triangular (or $T^{(i)}$ for $i = 1, \ldots, k - 1$ are block lower triangular and $T^{(i)}$ for $i = k + 1, \ldots, d$ are strictly block upper triangular) [50].

When such properties for T matrices are not true, one can perform an $L + U$ split and (3.13) becomes

(3.17)
$$f_{\boldsymbol{n}} = \sum_{H(\boldsymbol{n}') \leq 0} f'_{\boldsymbol{n}'} (l_{n'_1, n_1}^{(1)} + u_{n'_1, n_1}^{(1)})(l_{n'_1, n_1}^{(1)} + u_{n'_1, n_1}^{(1)}) \cdots (l_{n'_{d-1}, n_{d-1}}^{(d-1)} + u_{n'_{d-1}, n_{d-1}}^{(d-1)}) t_{n'_d, n_d}^{(d)},$$

where there are totally 2^{d-1} terms. For each term, we can perform the dimension-by-dimension matrix-vector product. The overall computational cost is $\mathcal{O}(2^{d-1} \cdot DoF \cdot N)$ if the cost of one-dimensional transform is log-linear, i.e., $\mathcal{O}(\mathcal{N} \log \mathcal{N})$ where $\mathcal{N}$ denotes the DoF in one dimension [50]. This assumption holds true for our scheme.

Now, we return to the description of the implementation of (3.3). The computations are done using the following steps with repeated application of the fast matrix-vector product described above. We denote the adaptive numerical solution by

(3.18)
$$u_h(\mathbf{x}) = \sum_{(\mathbf{l},\mathbf{j}) \in G, \mathbf{1} \leq \mathbf{i} \leq \mathbf{k+1}} c_{\mathbf{i},\mathbf{l}}^{\mathbf{j}} v_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}(\mathbf{x}),$$

where $v_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}(\mathbf{x})$ is the Alpert's multiwavelets in multidimensions defined in (2.5) and the set G collects the index of all active elements. The active index set of the interpolatory multiwavelets is also equal to G. In particular, the adaptive interpolation function space is denoted by

(3.19)
$$\mathbf{V}^M = \{\psi_{\mathbf{i},\mathbf{l},\mathbf{n}}^{\mathbf{j}} : (\mathbf{l},\mathbf{j}) \in G, \mathbf{1} \leq \mathbf{i} \leq \mathbf{k+1}, \mathbf{1} \leq \mathbf{n} \leq \mathbf{P+1}\}.$$

The corresponding interpolation points are

(3.20)
$$Q = \{\tilde{\mathbf{x}}_{\mathbf{i},\mathbf{l}}^{\mathbf{j}} = (\tilde{x}_{i_1, l_1}^{j_1}, \tilde{x}_{i_2, l_2}^{j_2}, \ldots, \tilde{x}_{i_d, l_d}^{j_d}) : \mathbf{1} \leq \mathbf{i} \leq \mathbf{k+1}, (\mathbf{l},\mathbf{j}) \in G\}.$$

The first step is to obtain the function and its derivative value at the interpolation points (3.20) from the coefficients $c_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}$. We denote

(3.21)
$$(u_p)_{\mathbf{i},\mathbf{l},\mathbf{n}}^{\mathbf{j}} := \frac{\partial u_h}{\partial \mathbf{x}^{\mathbf{n}}}(\tilde{\mathbf{x}}_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}) = \sum_{(\mathbf{l},\mathbf{j}) \in G, \mathbf{1} \leq \mathbf{i} \leq \mathbf{k+1}} c_{\mathbf{i},\mathbf{l}}^{\mathbf{j}} \frac{\partial v_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}}{\partial \mathbf{x}^{\mathbf{n}}}(\tilde{\mathbf{x}}_{\mathbf{i},\mathbf{l}}^{\mathbf{j}})$$
$$= \sum_{(\mathbf{l},\mathbf{j}) \in G, \mathbf{1} \leq \mathbf{i} \leq \mathbf{k+1}} c_{\mathbf{i},\mathbf{l}}^{\mathbf{j}} \prod_{m=1}^{d} \frac{dv_{i_m, l_m}^{j_m}}{dx^{n_m}}(\tilde{x}_{i_m, l_m}^{j_m}).$$

Here, the values of Alpert's basis functions in one dimension (1D) and their derivatives at all the interpolation points in 1D should be computed and stored before the time evolution. The fast matrix-vector multiplication described above is applied to evaluate this summation (3.21).

The second step is to calculate the value of $f(u_h)$ and its derivative at all the interpolation points (3.20), which is denoted by $\{(f_p)_{\mathbf{i},\mathbf{l},\mathbf{n}}^{\mathbf{j}}\}$, by using $\{(u_p)_{\mathbf{i},\mathbf{l},\mathbf{n}}^{\mathbf{j}}\}$:

$$(f_p)_{\mathbf{i},\mathbf{l},\mathbf{n}}^{\mathbf{j}} := \frac{\partial f(u_h)}{\partial \mathbf{x}^{\mathbf{n}}}(\tilde{\mathbf{x}}_{\mathbf{i},\mathbf{l}}^{\mathbf{j}}).$$

This can be obtained by simply using the chain rule. Afterwards, we can transform the point values $\{(f_p)^{\mathbf{j}}_{\mathbf{i},\mathbf{l},\mathbf{n}}\}$ to the coefficient of the interpolation basis $\tilde{c}^{\mathbf{j}}_{\mathbf{i},\mathbf{l},\mathbf{n}}$ in

$$(3.22) \qquad \mathcal{I}[f(u_h)] = \sum_{\substack{(\mathbf{l},\mathbf{j})\in G, \\ \mathbf{1}\leq\mathbf{i}\leq\mathbf{k}+\mathbf{1}, \\ \mathbf{1}\leq\mathbf{n}\leq\mathbf{P}+\mathbf{1}}} \tilde{c}^{\mathbf{j}}_{\mathbf{i},\mathbf{l},\mathbf{n}} \psi^{\mathbf{j}}_{\mathbf{i},\mathbf{l},\mathbf{n}}(\mathbf{x})$$

by applying the fast algorithm in [53].

Now, the terms in (3.3) can be readily computed. For example, for the volume integral $\sum_{K\in T_h}\int_K \mathcal{I}[f(u_h)]\cdot\nabla v_h d\mathbf{x}$, let the test function be $v_h = v^{\mathbf{j}'}_{\mathbf{i}',\mathbf{l}'}(\mathbf{x})$, then we have

$$
\begin{aligned}
\sum_{K\in T_h}\int_K \mathcal{I}[f(u_h)]\cdot\nabla v_h d\mathbf{x} &= \int_\Omega \mathcal{I}[f(u_h)]\cdot\nabla v_h d\mathbf{x} \\
&= \int_\Omega \sum_{\substack{(\mathbf{l},\mathbf{j})\in G, \\ \mathbf{1}\leq\mathbf{i}\leq\mathbf{k}+\mathbf{1}, \\ \mathbf{1}\leq\mathbf{n}\leq\mathbf{P}+\mathbf{1}}} \tilde{c}^{\mathbf{j}}_{\mathbf{i},\mathbf{l},\mathbf{n}} \psi^{\mathbf{j}}_{\mathbf{i},\mathbf{l},\mathbf{n}}\cdot\nabla v^{\mathbf{j}'}_{\mathbf{i}',\mathbf{l}'} d\mathbf{x} \\
(3.23) \qquad &= \sum_{\substack{(\mathbf{l},\mathbf{j})\in G, \\ \mathbf{1}\leq\mathbf{i}\leq\mathbf{k}+\mathbf{1}, \\ \mathbf{1}\leq\mathbf{n}\leq\mathbf{P}+\mathbf{1}}} \tilde{c}^{\mathbf{j}}_{\mathbf{i},\mathbf{l},\mathbf{n}} \int_\Omega \psi^{\mathbf{j}}_{\mathbf{i},\mathbf{l},\mathbf{n}}\cdot\nabla v^{\mathbf{j}'}_{\mathbf{i}',\mathbf{l}'} d\mathbf{x} \\
&= \sum_{\substack{(\mathbf{l},\mathbf{j})\in G, \\ \mathbf{1}\leq\mathbf{i}\leq\mathbf{k}+\mathbf{1}, \\ \mathbf{1}\leq\mathbf{n}\leq\mathbf{P}+\mathbf{1}}} \tilde{c}^{\mathbf{j}}_{\mathbf{i},\mathbf{l},\mathbf{n}} \prod_{m=1}^{d} \int_0^1 \psi^{j_m}_{i_m,l_m,n_m} \frac{d}{dx_m} v^{j'_m}_{i'_m,l'_m} dx.
\end{aligned}
$$

This can be again treated by the fast matrix-vector multiplication algorithm. The one-dimensional values $\int_0^1 \psi^{j_m}_{i_m,l_m,n_m} \frac{d}{dx} v^{j'_m}_{i'_m,l'_m} dx$ should be precomputed and stored before the time evolution starts. The computation of the edge integrals $\sum_{e\in\Gamma_h}\int_e \mathcal{I}$ $\widehat{[f\cdot n_K(u_h)]}v_h ds$ also follows the same approach as that over the volume. The details are omitted here for brevity.

3.4. Artificial viscosity. For capturing shocks, we add artificial viscosity following the approach in [5] and arrive at the semidiscrete formulation

$$
\begin{aligned}
\sum_{K\in T_h}\int_K (u_h)_t v_h d\mathbf{x} &- \sum_{K\in T_h}\int_K \mathcal{I}[f(u_h)]\cdot\nabla v_h d\mathbf{x} + \sum_{e\in\Gamma_h}\int_e \mathcal{I}[\widehat{f\cdot n_K}(u_h)]v_h ds \\
(3.24) \qquad &- \sum_{K\in T_h}\int_K \nu(u_h)\nabla u_h\cdot\nabla v_h d\mathbf{x} = 0,
\end{aligned}
$$

where $\nu = \nu(u_h) \geq 0$ is the artificial viscosity. The artificial viscosity is piecewise constant in each element and depends on u_h. Since the sharp gradient and shock will only appear in the leaf element [35], the artificial viscosity is only imposed in the leaf element and determined in the following approach:

$$
\nu = \begin{cases} 0 & \text{if} \quad s_e \leq s_0 + \kappa, \\ \nu_0 h & \text{otherwise,} \end{cases}
$$

where $\nu_0 > 0$ and κ are constants chosen empirically; see [20, 21] for discussions on standard DG methods. In this paper, we use $\nu_0 = 2$ and $\kappa = 0$. s_e and s_0 are defined as

$$(3.25) \qquad s_e = \log_{10} \left(\sum_{\mathbf{1} \leq \mathbf{i} \leq \mathbf{k+1}} |u_{\mathbf{i,l}}^{\mathbf{j}}|^2 \right)^{\frac{1}{2}}, \quad s_0 = \log_{10}(2^{-(k+\frac{1}{2})|\mathbf{l}|_1}).$$

In the regions where the solutions are smooth, s_e should be the same order as s_0 by the estimate (2.8). If the solution is discontinuous, s_e should be much larger than s_0.

Remark 3.5. There are still many problems to be explored on the artificial viscosity. The first one is the specific form of the artificial viscosity term. Here, for simplicity, we only add an artificial viscosity term $\int \nu(u_h) \nabla u_h \cdot \nabla v_h$ in (3.24). One may also add a physical diffusion term $\nabla \cdot (\nabla \nu(u) u)$ and then discretize it using local DG [45] or interior penalty DG [39]. The second issue is how to choose the optimal parameters κ and ν_0 in the artificial viscosity to obtain a sharp shock profile. The artificial neural network introduced in [47, 21] might be helpful for this problem. We will explore these subjects in future work.

The diffusion coefficient $\nu(u_h)$ is of order $\mathcal{O}(h)$ for trouble cells and zero for normal cells. Thus, the explicit time integration in both convection and diffusion terms in (3.24) will yield CFL condition $\Delta t = \mathcal{O}(h)$. For hyperbolic problems $u_t = u_x$ with DG discretizations using polynomials of degree k and upwind numerical flux and a $(k+1)$ stage explicit RK method of order $(k+1)$, the CFL constant is around $\frac{1}{2k+1}$ [16]. However, for solving diffusion equation $u_t = u_{xx}$ with local DG discretization with polynomials and alternating numerical flux, the CFL constant is around 0.0555 for $k = 1$, 0.0169 for $k = 2$, 0.0063 for $k = 3$, and 0.003 for $k = 4$, if coupled with explicit Runge–Kutta methods of the corresponding order,[1] which is much smaller than the CFL constant for convection terms, especially for polynomials of high degrees. If the alternating numerical flux is replaced by the central flux for the diffusion equation, the CFL constant is slightly larger but still much smaller than the CFL constant for the convection part: 0.125 for $k = 1$, 0.0384 for $k = 2$, 0.0158 for $k = 3$, and 0.0083 for $k = 4$.

To obtain better computational efficiency, we avoid explicit time integrations and apply the IMEX time discretizations where the convection term is treated explicitly and the diffusion term implicitly. Here, we only present the third-order IMEX method introduced in [44], which will be coupled with the DG space of quadratic polynomials. The explicit part is the same with the explicit third-order strong stability preserving (SSP) Runge–Kutta method [51], and the implicit part has four stages. To be precise, for the ODE systems

$$(3.26) \qquad \frac{dU}{dt} = F(U) + G(U),$$

where $F(U)$ denotes the nonstiff term (convection parts) and $G(U)$ the stiff term

---

[1] The CFL constants are provided by Chi-Wang Shu from Brown University in personal communications.

(diffusion parts). The IMEX scheme for (3.26) reads as

$$(3.27a) \qquad U^{(i)} = U^n + \Delta t \sum_{j=1}^{i-1} \tilde{a}_{ij} F(U^{(j)}) + \Delta t \sum_{j=1}^{i} a_{ij} G(u^{(j)}), \quad i = 1, \ldots, \nu,$$

$$(3.27b) \qquad U^{n+1} = U^n + \Delta t \sum_{i=1}^{\nu} \tilde{w}_i F(U^{(i)}) + \Delta t \sum_{i=1}^{\nu} w_i G(u^{(i)}),$$

with the stage $\nu = 4$ and the parameters

$$\tilde{a}_{32} = 1, \quad \tilde{a}_{42} = \tilde{a}_{43} = \frac{1}{4}, \quad a_{11} = a_{22} = a_{33} = a_{44} = \alpha, \quad a_{21} = -\alpha, \, a_{32} = 1 - \alpha,$$

$$a_{41} = \beta, \, a_{42} = \eta, \, a_{43} = \frac{1}{2} - \beta - \eta - \alpha, \, \tilde{w}_2 = \tilde{w}_3 = w_2 = w_3 = \frac{1}{6}, \, \tilde{w}_4 = w_4 = \frac{2}{3},$$

and

$$\alpha = 0.24169426078821, \quad \beta = 0.06042356519705, \quad \eta = 0.12915286960590.$$

The other parameters not listed above are zero.

By using the IMEX time integrator, the time step restriction remains the same as determined by the convection term. Note that the artificial viscosity $\nu = \nu(u)$ is determined by u_h^n and will remain unchanged in the middle stages of time evolution from t^n to t^{n+1}. Therefore, the matrix for the diffusion term only needs to be assembled once in each time step. Also, we need only solve a linear system in which the coefficient matrix is symmetric positive definite and also sparse (there exist only a small portion of elements with nonzero viscosity). In the computation, we apply the conjugate gradient method to solve this linear system. We also remark that, for smooth solutions, this scheme will reduce to the explicit time integrations when coupled with the semidiscrete DG scheme with artificial viscosity (3.24), since the artificial viscosity will automatically vanish and then IMEX scheme (3.27) reduces to the third-order SSP RK method. For each element V_l^j that satisfies the criteria, we compute the matrix corresponding to the term $\nu_0 h \int_{V_l^j} u_x v_x$ by using the undirectional approach. Then, by making a summation over all the elements that have artificial viscosity, we obtain the corresponding matrix term.

4. Numerical results. In this section, we perform numerical experiments to validate the accuracy and robustness of our scheme. The computational domain is $[0, 1]$ for 1D and $[0, 1]^2$ for two dimensions (2D). A periodic boundary condition is imposed. When testing accuracy for smooth solutions, we apply the TVD Runge–Kutta (RK) time discretizations [51]: second-order RK method for the piecewise linear finite element space ($k = 1$) and third-order RK method for the quadratic ($k = 2$) and cubic ($k = 3$) finite element space. When testing the capability for capturing discontinuous solutions, we use the quadratic finite element space ($k = 2$) coupled with the third-order IMEX time discretizations (3.27). The DoF refers to the number of Alpert's multiwavelet basis functions in the adaptive grids, i.e., $\dim(\mathbf{V}^k)$.

Example 4.1 (one-dimensional linear advection with constant coefficient). In this example, we consider the one-dimensional linear advection equation with constant coefficient:

$$(4.1) \qquad\qquad\qquad u_t + u_x = 0,$$

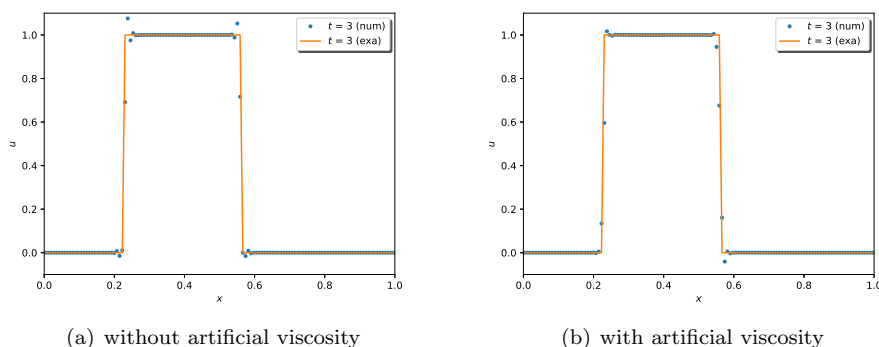(a) without artificial viscosity   (b) with artificial viscosity

FIG. 1. *Example* 4.1*: One-dimensional linear advection with constant coefficient at $t = 3$. Left: without artificial viscosity. Right: with artificial viscosity. The solid lines are the exact solution, and the symbols are the numerical solutions.*

with the initial value $u(x, 0) = u_0(x)$. Since the equation is linear, the interpolation operator is not needed. We focus on a nonsmooth initial condition to test the effectiveness of adaptivity and the artificial viscosity. The initial condition is chosen as

$$(4.2) \qquad u_0(x) = \begin{cases} 1, & 0.23 < x < 0.56, \\ 0 & \text{otherwise.} \end{cases}$$

We set $N = 8$ and $\epsilon = 10^{-5}$. The solutions with and without artificial viscosity at $t = 3$ are presented in Figure 1. We observe that the multiresolution DG method without artificial viscosity as in [29] can also produce a well-resolved solution. With the artificial viscosity, the oscillations are suppressed.

In Figure 2, the DoF and the errors for scheme with and without artificial viscosity are presented. Since the artificial viscosity (diffusion term) can smoothen the solution, the method has fewer DoFs and thus less computational cost. It is also observed that the error with artificial viscosity is comparable to that without artificial viscosity. Note that for this example, the full grid method offers excellent accuracy in the smooth region because the solution is piecewise constant. This is, in general, not true; see, e.g., Figure 6.

Example 4.2 (one-dimensional Burgers' equation). In this example, we focus on the one-dimensional Burgers' equation:

$$u_t + \left(\frac{u^2}{2} \right)_x = 0,$$

with the initial value $u(x, 0) = u_0(x) = \sin(2\pi x) + \frac{1}{2}$. The shock begins to develop at $t = \frac{1}{2\pi} \approx 0.159$. For this example, we only focus on the nonsmooth solution, and defer the accuracy study for the smooth solution to the next example in 2D. The adaptive multiresolution DG scheme without artificial viscosity will blow up when the shock develops. We run the code up to time $t = 0.2$, with the maximum mesh level of $N = 8$. The solutions obtained with artificial viscosity are shown in Figure 3 with $\epsilon = 10^{-3}$ and Figure 4 with $\epsilon = 10^{-4}$. Our scheme can capture the shock very well. We also observe that the leaf elements concentrate near the shock. The artificial viscosity is only imposed on elements near the shock.

(a) time history of degrees of freedom

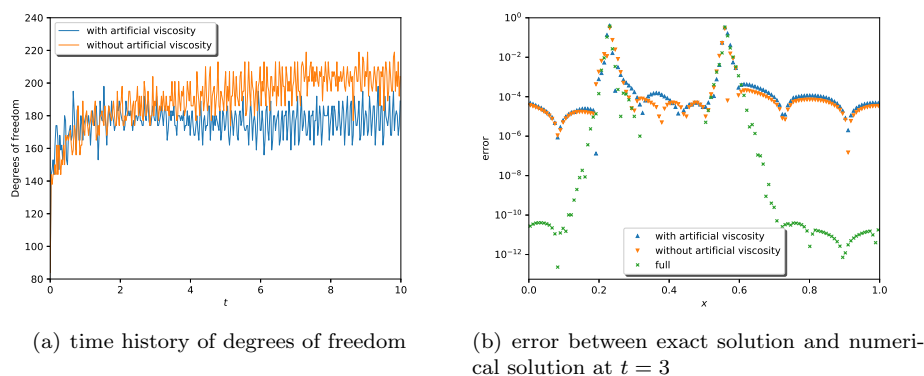(b) error between exact solution and numerical solution at $t = 3$

FIG. 2. *Example 4.1: One-dimensional linear advection with constant coefficient at $t = 3$. Left: time histories of the number of active degrees of freedom with and without artificial viscosity. Right: error between exact solution and numerical solution at $t = 3$ with and without artificial viscosity.*
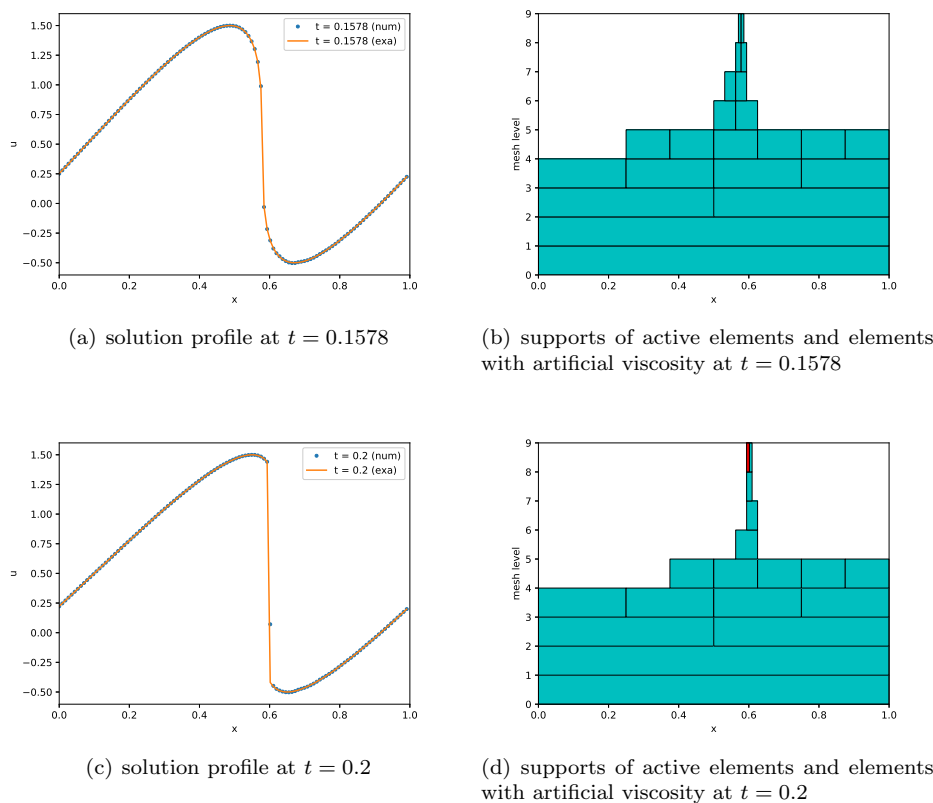

(a) solution profile at $t = 0.1578$

(b) supports of active elements and elements with artificial viscosity at $t = 0.1578$

(c) solution profile at $t = 0.2$

(d) supports of active elements and elements with artificial viscosity at $t = 0.2$

FIG. 3. *Example 4.2: One-dimensional Burgers' equation at $t = 0.1578$ and 0.2. $N = 8$ and $\epsilon = 10^{-3}$. Left: solution profile. Right: blue color denotes supports of active elements, and red color denotes elements with nonzero artificial viscosity.*
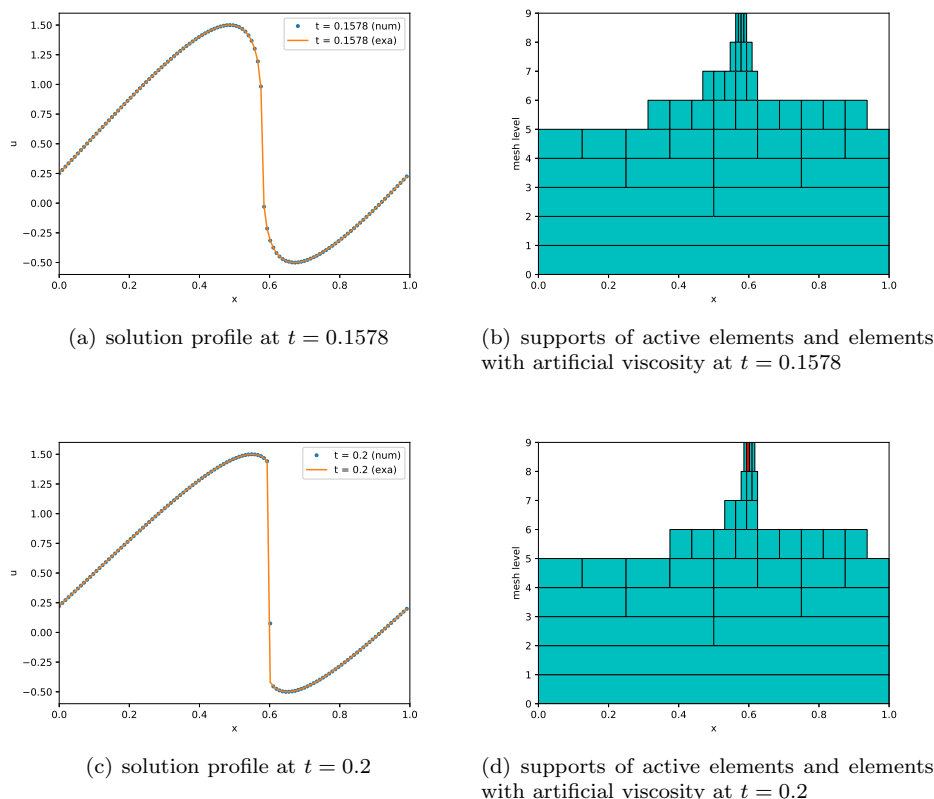
(a) solution profile at $t = 0.1578$

(b) supports of active elements and elements with artificial viscosity at $t = 0.1578$

(c) solution profile at $t = 0.2$

(d) supports of active elements and elements with artificial viscosity at $t = 0.2$

FIG. 4. *Example* 4.2: *One-dimensional Burgers' equation at* $t = 0.1578$ *and* 0.2. *N* = 8 *and* $\epsilon = 10^{-4}$. *Left: solution profile. Right: blue color denotes supports of active elements, and red color denotes elements with nonzero artificial viscosity.*

Example 4.3 (two-dimensional Burgers' equation). In this example, we consider the two-dimensional Burgers' equation:

$$u_t + \left(\frac{u^2}{2}\right)_x + \left(\frac{u^2}{2}\right)_y = 0,$$

with the initial value $u = u_0(x, y) = \sin(2\pi(x + y))$.

To study the effect of the interpolation operator, we first test the convergence rates for smooth solutions without adaptivity and artificial viscosity. The code is run up to $T = 0.01$. Table 1 shows the convergence rate of the standard DG with integrals over elements and edges calculated by Lagrange interpolation techniques. The interpolation points are imposed in the inner domain; see the interpolation points and basis functions in Appendix A.1. Recall the degree of polynomials for the DG finite element space is denoted by k and the degree of interpolation operator is denoted by M. Table 1 shows almost full convergence rate, except some order reduction when $k = 2$ and $M = 2$, similar to previous results in [36]. Actually, for $k = 2$ and $M = 4$, the numerical scheme is the same as the DG scheme in which the integrals are evaluated exactly, since the physical flux for Burgers' equation is a quadratic function. It seems that the convergence rate is almost full order with $k = 1$ and

$M = 1$. Therefore, the condition assumed in Proposition 3.1 may not be necessary for the standard DG method. We also test the accuracy using the Lagrange interpolation in which the interpolation points are at the interface (see the interpolation points and basis functions in Appendix A.2). The results are similar to those in Table 1, and thus they are not presented here for space considerations. The Hermite interpolations with $k = 1, M = 3$ and $k = 2, M = 3$ and $k = 3, M = 5$ (see the interpolation points and basis functions in Appendix B) are also tested. The same results are observed and omitted for brevity.

TABLE 1
Two-dimensional Burgers' equation at $T = 0.01$, $\Delta t = 0.1h$, standard DG, and Lagrange interpolation with inner interpolation points.

Poly degrees	N	L^1-error	Order	L^2-error	Order	L^∞-error	Order
	3	1.88e-02	-	2.51e-02	-	6.40e-02	-
$k = 1,$	4	5.75e-03	1.71	7.13e-03	1.81	2.09e-02	1.61
$M = 1$	5	1.67e-03	1.78	2.02e-03	1.82	6.23e-03	1.75
	6	4.41e-04	1.92	5.33e-04	1.92	1.68e-03	1.89
	3	1.81e-02	-	2.53e-02	-	7.09e-02	-
$k = 1,$	4	5.15e-03	1.81	6.98e-03	1.86	2.18e-02	1.70
$M = 2$	5	1.45e-03	1.82	1.92e-03	1.86	6.44e-03	1.76
	6	3.80e-04	1.94	4.99e-04	1.94	1.71e-03	1.91
	3	2.13e-03	-	2.66e-03	-	1.01e-02	-
$k = 2,$	4	5.11e-04	2.06	6.64e-04	2.00	2.77e-03	1.87
$M = 2$	5	1.08e-04	2.25	1.43e-04	2.21	5.42e-04	2.36
	6	1.71e-05	2.66	2.38e-05	2.59	9.55e-05	2.50
	3	7.70e-04	-	1.03e-03	-	3.77e-03	-
$k = 2,$	4	1.56e-04	2.31	2.05e-04	2.33	7.58e-04	2.31
$M = 3$	5	2.79e-05	2.48	3.63e-05	2.50	1.38e-04	2.45
	6	4.37e-06	2.67	5.99e-06	2.60	2.21e-05	2.65
	3	7.98e-04	-	1.05e-03	-	3.61e-03	-
$k = 2,$	4	1.62e-04	2.30	2.06e-04	2.34	7.53e-04	2.26
$M = 4$	5	2.83e-05	2.52	3.65e-05	2.50	1.38e-04	2.45
	6	4.40e-06	2.69	6.00e-06	2.60	2.21e-05	2.64

Next, we test the convergence rate of the sparse grid DG method. We apply three different types of interpolation. The first one is the Lagrange interpolation with the interpolation points at the inner points of elements. The results are shown in Table 2; some instability is observed for very fine mesh with $k = 2$. The second type is the Lagrange interpolation with the interpolation points at the interface. The results are shown in Table 3. The results are better than the first type. For $k = 1$ and $M = 2$, the convergence order is around 1.5, as predicted. For $k = 2$, the convergence order is around 2 with $M = 2$. This indicates that the condition in Proposition 3.1 is necessary here. For $k = 2$ and $M = 3$, there still exists some instability for very fine mesh. This motivates us to apply the Hermite interpolation in which we only use the end points. As shown in Table 4, the scheme with the Hermite interpolation is stable with predicted accuracy. This numerical experiment reveals that the Hermite interpolation is more stable than the Lagrange interpolation and should be the method of choice.

Next, we discuss the convergence rate with adaptivity. Following [29], two types of convergence rates are calculated. The first one is the convergence rate with respect to the error threshold:

$$R_{\epsilon_l} = \frac{\log(e_{l-1}/e_l)}{\log(\epsilon_{l-1}/\epsilon_l)}.$$

TABLE 2
Two-dimensional Burgers' equation at $T = 0.01$, $\Delta t = 0.1h$, sparse grid DG, Lagrange interpolation with inner interpolation points.

Poly degrees	N	L^1-error	Order	L^2-error	Order	L^∞-error	Order
$k=1$, $M=1$	4	3.61e-02	-	4.90e-02	-	2.45e-01	-
	5	1.60e-02	1.17	2.18e-02	1.17	1.29e-01	0.93
	6	7.61e-03	1.07	1.02e-02	1.09	7.15e-02	0.85
	7	3.48e-03	1.13	4.71e-03	1.12	3.33e-02	1.10
$k=1$, $M=2$	4	2.89e-02	-	4.07e-02	-	2.47e-01	-
	5	8.70e-03	1.73	1.21e-02	1.75	8.07e-02	1.61
	6	2.98e-03	1.55	4.21e-03	1.52	3.03e-02	1.41
	7	1.13e-03	1.40	1.61e-03	1.38	1.10e-02	1.46
$k=2$, $M=2$	4	1.62e-02	-	2.61e-02	-	2.22e-01	-
	5	5.13e-02	-1.67	1.13e-01	-2.11	1.05e+00	-2.23
	6	1.38e+06	-24.68	4.53e+06	-25.26	1.12e+08	-26.67
$k=2$, $M=3$	4	3.37e-03	-	4.78e-03	-	4.77e-02	-
	5	7.12e-04	2.24	1.03e-03	2.21	9.49e-03	2.33
	6	3.71e-04	0.94	6.44e-04	0.68	7.64e-03	0.31
	7	2.13e-03	-2.52	4.31e-03	-2.74	6.56e-02	-3.10

TABLE 3
Two-dimensional Burgers' equation at $T = 0.01$, $\Delta t = 0.1h$, sparse grid DG, Lagrange interpolation with interface interpolation points.

Poly degrees	N	L^1-error	Order	L^2-error	Order	L^∞-error	Order
$k=1$, $M=1$	4	3.58e-02	-	4.87e-02	-	2.45e-01	-
	5	1.58e-02	1.18	2.15e-02	1.18	1.27e-01	0.95
	6	7.47e-03	1.08	1.00e-02	1.10	7.05e-02	0.85
	7	3.40e-03	1.14	4.60e-03	1.13	3.30e-02	1.09
$k=1$, $M=2$	4	2.84e-02	-	4.01e-02	-	2.38e-01	-
	5	8.14e-03	1.80	1.13e-02	1.83	7.49e-02	1.67
	6	2.70e-03	1.59	3.88e-03	1.54	2.73e-02	1.46
	7	8.07e-04	1.74	1.17e-03	1.73	1.10e-02	1.31
$k=2$, $M=2$	4	5.74e-03	-	7.71e-03	-	2.79e-02	-
	5	2.68e-03	1.10	3.35e-03	1.20	1.05e-02	1.41
	6	6.14e-04	2.13	7.70e-04	2.12	3.51e-03	1.58
	7	1.59e-04	1.95	2.03e-04	1.92	1.21e-03	1.54
$k=2$, $M=3$	4	3.37e-03	-	4.78e-03	-	4.72e-02	-
	5	7.09e-04	2.25	1.02e-03	2.22	9.25e-03	2.35
	6	3.49e-04	1.02	6.03e-04	0.77	7.12e-03	0.38
	7	1.70e-03	-2.29	3.44e-03	-2.51	5.21e-02	-2.87

The second one is the convergence rate with respect to DoF:

$$R_{\mathrm{DoF}_l} = \frac{\log(e_{l-1}/e_l)}{\log(\mathrm{DoF}_{l-1}/\mathrm{DoF}_l)}.$$

We run the simulations with a fixed maximum mesh level $N = 8$ and different ϵ values, and we report the L^2-errors and the number of active DoF at $T = 0.01$ in Table 5. We observe similar convergence rates as in Table 1 in [29]: R_ϵ is slightly smaller than 1, and R_{DoF} is much larger than $(k+1)/2$ (R_{DoF} for the standard adaptive DG scheme for two-dimensional problems) but still smaller than $(k+1)$. This demonstrates the effectiveness of the multiresolution adaptive algorithm. Sparsity is indeed achieved for smooth solutions.

Next, we test the ability of our scheme on capturing nonsmooth solutions. We also take the same initial value. The shock begins to develop at $t = \frac{1}{4\pi} \approx 0.07958$. The numerical results at $t = 0.2$ are shown in Figure 5. We observe the numerical solution

TABLE 4
Two-dimensional Burgers' equation at $T = 0.01$, $\Delta t = 0.1h$ for $k = 1, 2$, and $\Delta t = 0.1h^{4/3}$ for $k = 3$, sparse grid DG, Hermite interpolation.

Poly degrees	N	L^1-error	Order	L^2-error	Order	L^∞-error	Order
	5	8.30e-03	-	1.15e-02	-	7.36e-02	-
$k = 1$,	6	2.85e-03	1.54	3.99e-03	1.52	2.59e-02	1.51
$M = 3$	7	8.81e-04	1.70	1.23e-03	1.70	1.03e-02	1.33
	8	2.66e-04	1.73	3.72e-04	1.73	3.69e-03	1.48
	5	1.10e-03	-	1.47e-03	-	5.30e-03	-
$k = 2$,	6	1.60e-04	2.78	2.11e-04	2.80	9.04e-04	2.55
$M = 3$	7	2.75e-05	2.54	3.67e-05	2.52	2.23e-04	2.02
	8	6.25e-06	2.14	8.41e-06	2.13	4.98e-05	2.16
	5	3.85e-04	-	5.45e-04	-	3.07e-03	-
$k = 2$,	6	8.18e-05	2.23	1.25e-04	2.12	1.12e-03	1.45
$M = 5$	7	1.37e-05	2.58	1.99e-05	2.65	2.26e-04	2.31
	8	2.30e-06	2.57	3.41e-06	2.55	2.90e-05	2.97
	5	4.37e-05	-	6.85e-05	-	2.86e-04	-
$k = 3$,	6	3.75e-06	3.54	6.19e-06	3.47	6.26e-05	2.19
$M = 5$	7	2.81e-07	3.74	4.46e-07	3.80	3.16e-06	4.31
	8	2.74e-08	3.35	4.44e-08	3.33	6.30e-07	2.33

TABLE 5
Two-dimensional Burgers' equation at $T = 0.01$. Convergence rates with respect to the error threshold and DoF.

	ϵ	DoF	L^2-error	R_{DoF}	R_ϵ
	1e-04	3488	3.89e-04	-	-
$k = 1$	5e-05	4608	2.95e-04	0.98	0.40
	1e-05	9408	9.29e-05	1.62	0.72
	5e-06	12272	4.79e-05	2.49	0.95
	1e-04	1656	2.02e-04	-	-
$k = 2$	5e-05	1908	1.20e-04	3.71	0.76
	1e-05	3600	3.83e-05	1.79	0.71
	5e-06	4068	2.33e-05	4.09	0.72
	1e-04	1152	5.52e-05	-	-
$k = 3$	5e-05	1472	2.49e-05	3.25	1.15
	1e-05	1920	1.16e-05	2.89	0.48
	5e-06	2624	5.62e-06	2.31	1.04

coincides with the exact solution very well. The elements with nonzero artificial viscosity are also concentrated near the shock.

Next, we compare the efficiency of our adaptive method with the nonadaptive (full grid) DG scheme for nonsmooth solutions in Figure 6. Here, the full grid scheme with $N = 6$ has 36864 DoF. The adaptive DG method has on average 19086 DoF and at most 22000 DoF, which is fewer than the full grid method. However, the adaptive is more accurate than the full grid as can be seen in Figure 6.

We also investigate the efficiency of the adaptive method for smooth solutions in Figure 7. To reach the same accuracy, the adaptive method has much less DoF than the nonadaptive method. This is because of the sparse grid nature of our method.

Example 4.4 (two-dimensional KPP rotating wave problem). In the last example, we considered the two-dimensional KPP rotating wave problem with the nonconvex physical flux:
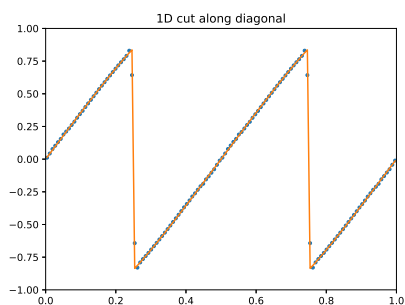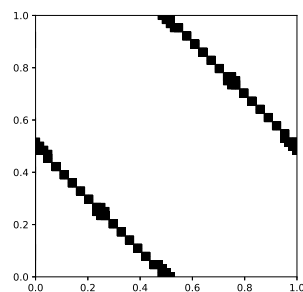
$$u_t + \sin(u)_x + \cos(u)_y = 0.$$
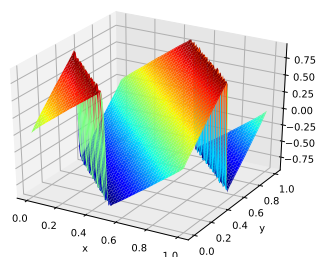
(a) numerical solution (surface plot)
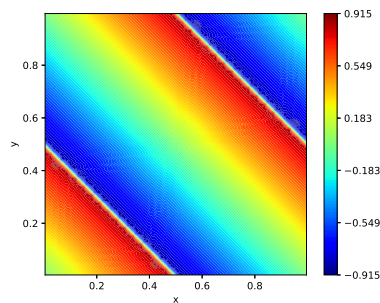
(b) numerical solution (contour plot)

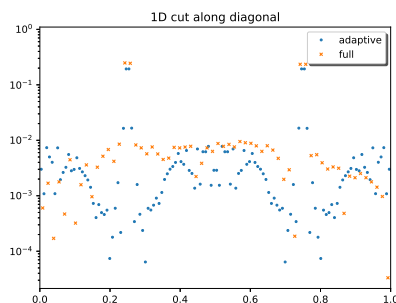(c) numerical solution in a one-dimensional cut along diagonal

(d) elements with artificial viscosity

FIG. 5. *Example 4.3: Two-dimensional Burgers' equation at $t = 0.2$. $N = 7$ and $\epsilon = 5e-4$. (a) Numerical solution (surface plot). (b) Numerical solution (contour plot). (c) Numerical solution in a one-dimensional cut along the diagonal. (d) Elements with artificial viscosity.*
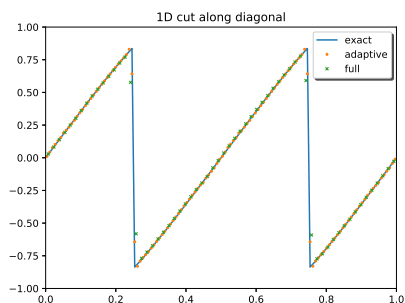
(a) profile in one-dimensional cut along the diagonal

(b) error between numerical solution and exact solution

FIG. 6. *Example 4.3: Two-dimensional Burgers' equation at $t = 0.2$. Full grid (nonadaptive) DG with $N = 6$ versus adaptive DG with $\epsilon = 5e - 4$. Left: profile in one-dimensional cut along the diagonal. Right: error between numerical solution and exact solution.*
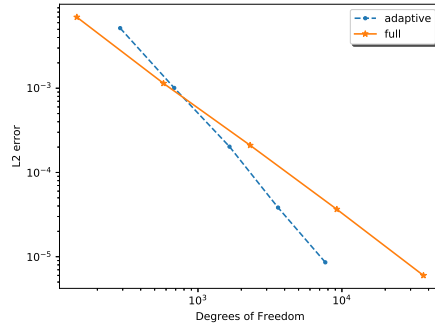
FIG. 7. *Example* 4.3: *Two-dimensional Burgers' equation at* $t = 0.01$. *Full grid (nonadaptive) DG with* $N = 2, 3, 4, 5, 6$ *versus adaptive DG with* $\epsilon = 10^{-2}, 10^{-3}, \ldots, 10^{-6}$ *and fixed* $N = 8$.

The initial condition is

$$u_0(x, y) = \begin{cases} 3.5\pi, & \left(x - \dfrac{1}{2}\right)^2 + \left(y - \dfrac{1}{2}\right)^2 \leq \dfrac{1}{16}, \\ 0.25\pi & \text{otherwise.} \end{cases}$$

This is a rather challenging test case proposed in [41], since the flux is nonconvex and a two-dimensional composite wave structure is present. The code is run up to $t = 0.4$. The maximum mesh level is $N = 7$, and the error threshold is $\epsilon = 5 \times 10^{-4}$. The numerical solutions and elements with nonzero artificial viscosity at $t = 0.2$ and $t = 0.4$ are shown in Figure 8. Our numerical scheme can capture the wave structure very well.

5. Concluding remarks. In this paper, we propose an adaptive multiresolution DG scheme for scalar hyperbolic conservation laws in multidimensions. Besides the Alpert's multiwavelets, the interpolatory multiwavelets are applied to treat the nonlinear integrals over elements and edges in DG schemes. From numerical study, we find that the multiresolution Hermite interpolation is the most stable. Because of the coherence of the multiresolution interpolation with the MRA of the numerical solution, our method can achieve similar computational complexity as the sparse grid DG method for smooth solutions [28, 29]. Artificial viscosity and adaptivity are activated for nonsmooth solutions. The required DoF corresponds to the intrinsic complexity of the solution structure, and artificial viscosity is only added at locations near the shock maintaining sharpness of the solution profile. We also emphasize that the motivation of this work is for efficient computations of nonlinear PDE problems in high dimensions. The approach we take may not be efficient compared to previous work of adaptive multiresolution DG methods in low-dimensional settings. In future work, we will extend this approach to nonlinear high-dimensional PDEs such as the Hamilton–Jacobi equation and kinetic equations. In an effort to promote reproducible research, the code generating the results in this paper can be found at the GitHub link https://github.com/JuntaoHuang/adaptive-multiresolution-DG.

Appendix A. Lagrange interpolation basis functions. For the completeness of our paper, we present the multiresolution interpolation basis functions, which are first introduced in [53]. In this part, we focus on the Lagrange interpolantion, i.e., $K = 0$. The case in which the interpolation points are imposed in the inner domain,

(a) numerical solution in 2D at $t = 0.2$

(b) elements with artificial viscosity at $t = 0.2$

(c) numerical solution in 2D at $t = 0.4$

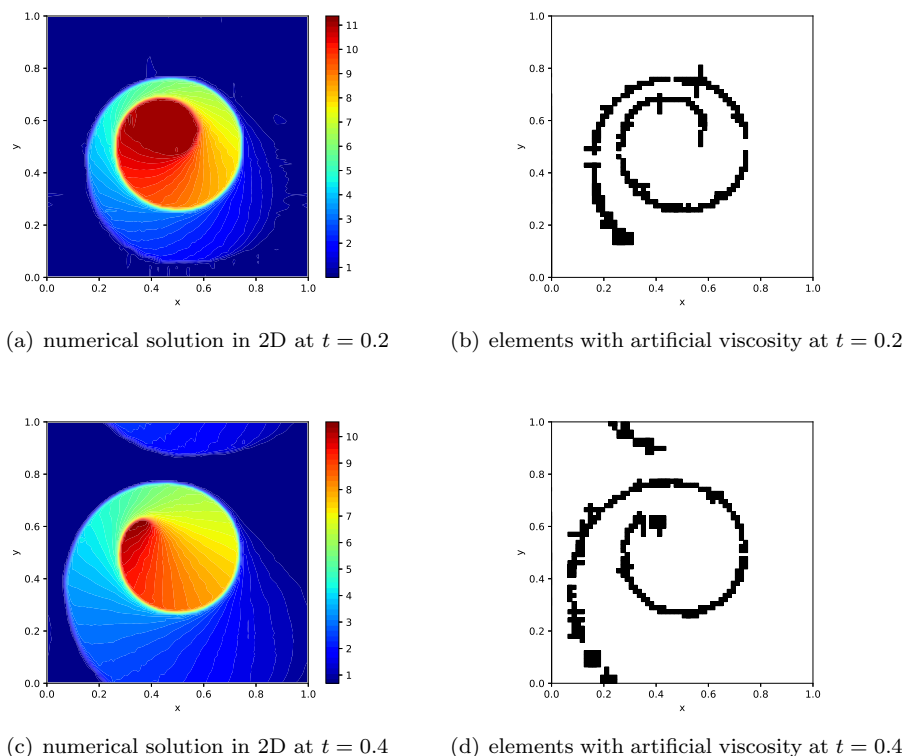(d) elements with artificial viscosity at $t = 0.4$

FIG. 8. *Example 4.4: Two-dimensional KPP rotating wave problem at $t = 0.2$ and $t = 0.4$. $N = 7$ and $\epsilon = 5e - 4$. Left: numerical solution in 2D. Right: elements with artificial viscosity.*

as implemented in Table 2, is discussed first, followed by the case in which the points are imposed at the cell interface; see the corresponding numerical results in Table 3.

The basis functions in $\tilde{W}_1$ are piecewise polynomials on $I_l := (0, \frac{1}{2})$ and $I_r := (\frac{1}{2}, 1)$. Note that the functions may be discontinuous at the interface $x = 1/2$; thus I_l and I_r are both defined to be open intervals. The basis functions in $\tilde{W}_1$ in this paper are all supported on one half interval I_l or I_r and vanish on the other half. For simplicity, we will only declare the function on its support. For example, $\psi_0(x)|_{I_r}$ gives the definition of ψ_0 on I_r and indicates that ψ_0 vanishes on I_l.

A.1. Interpolation points in the inner domain.

A.1.1. $P = 1$ and $K = 0$. The interpolation points are

$$\tilde{X}_0 = \left\{\frac{1}{3}, \frac{2}{3}\right\}, \quad \tilde{X}_1 = \left\{\frac{1}{6}, \frac{5}{6}\right\}.$$

The basis functions in $\tilde{W}_0^1$ and $\tilde{W}_1^1$ are

$$\phi_0(x) = -3x + 2, \qquad \phi_1(x) = 3x - 1.$$
$$\psi_0(x)|_{I_l} = -6x + 2, \quad \psi_1(x)|_{I_r} = 6x - 4.$$

A.1.2. $P = 2$ and $K = 0$. The interpolation points are

$$\tilde{X}_0 = \left\{\frac{1}{6}, \frac{1}{3}, \frac{2}{3}\right\}, \quad \tilde{X}_1 = \left\{\frac{1}{12}, \frac{7}{12}, \frac{5}{6}\right\}.$$

The basis functions in $\tilde{W}_0^2$ and $\tilde{W}_1^2$ are

$$\phi_0(x) = \tfrac{4}{3}(3x-2)(3x-1), \quad \phi_1(x) = -(3x-2)(6x-1),$$
$$\phi_2(x) = \tfrac{1}{3}(3x-1)(6x-1)$$

and

$$\psi_0(x)|_{I_l} = \tfrac{8}{3}(3x-1)(6x-1), \quad \psi_1(x)|_{I_r} = \tfrac{8}{3}(3x-2)(6x-5),$$
$$\psi_2(x)|_{I_r} = \tfrac{2}{3}(3x-2)(12x-7).$$

A.1.3. $P = 3$ and $K = 0$. The interpolation points are

$$\tilde{X}_0 = \Big\{ \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5} \Big\}, \quad \tilde{X}_1 = \Big\{ \frac{1}{10}, \frac{3}{10}, \frac{7}{10}, \frac{9}{10} \Big\}.$$

The basis functions in $\tilde{W}_0^3$ and $\tilde{W}_1^3$ are

$$\phi_0(x) = -\tfrac{1}{6}(5x-4)(5x-3)(5x-2), \quad \phi_1(x) = \tfrac{1}{2}(5x-4)(5x-3)(5x-1),$$
$$\phi_2(x) = -\tfrac{1}{2}(5x-4)(5x-2)(5x-1), \quad \phi_3(x) = \tfrac{1}{6}(5x-3)(5x-2)(5x-1)$$

and

$$\psi_0(x)|_{I_l} = -\tfrac{2}{3}(5x-2)(5x-1)(10x-3),$$
$$\psi_1(x)|_{I_l} = -2(5x-2)(5x-1)(10x-1),$$
$$\psi_2(x)|_{I_r} = 2(5x-4)(5x-3)(10x-9),$$
$$\psi_3(x)|_{I_r} = \tfrac{2}{3}(5x-4)(5x-3)(10x-7).$$

A.2. Interpolation points at the interface.

A.2.1. $P = 1$ and $K = 0$. The interpolation points are

$$\tilde{X}_0 = \{0^+, 1^-\}, \quad \tilde{X}_1 = \Big\{ \Big(\frac{1}{2}\Big)^-, \Big(\frac{1}{2}\Big)^+ \Big\}.$$

Here and below, we use superscripts $+, -$ to emphasize the left and right limits of a function at that point. This is a feature of the discontinuous piecewise polynomial space.

The basis functions in $\tilde{W}_0^1$ and $\tilde{W}_1^1$ are

$$\phi_1(x) = -x+1, \quad \phi_2(x) = x,$$
$$\psi_1(x)|_{I_l} = 2x, \quad \psi_2(x)|_{I_r} = -2x+2.$$

A.2.2. $P = 2$ and $K = 0$. The interpolation points are

$$\tilde{X}_0 = \Big\{ 0^+, \Big(\frac{1}{2}\Big)^-, 1^- \Big\}, \quad \tilde{X}_1 = \Big\{ \Big(\frac{1}{4}\Big)^-, \Big(\frac{1}{2}\Big)^+, \Big(\frac{3}{4}\Big)^- \Big\}.$$

The basis functions in $\tilde{W}_0^2$ and $\tilde{W}_1^2$ are

$$\phi_1(x) = 2(x-\tfrac{1}{2})(x-1), \quad \phi_2(x) = -4x(x-1), \quad \phi_3(x) = 2x(x-\tfrac{1}{2})$$

and

$$\psi_0(x)|_{I_l} = -16x(x-\tfrac{1}{2}), \quad \psi_1(x)|_{I_r} = 8(x-\tfrac{3}{4})(x-1),$$
$$\psi_2(x)|_{I_r} = -16(x-\tfrac{1}{2})(x-1).$$

A.2.3. $P = 3$ and $K = 0$. The interpolation points are

$$\tilde{X}_0 = \left\{ 0^+, \left(\frac{1}{4}\right)^-, \left(\frac{1}{2}\right)^-, 1^- \right\}, \quad \tilde{X}_1 = \left\{ \left(\frac{1}{8}\right)^-, \left(\frac{1}{2}\right)^+, \left(\frac{5}{8}\right)^-, \left(\frac{3}{4}\right)^- \right\}.$$

The basis functions in $\tilde{W}_0^3$ and $\tilde{W}_1^3$ are

$$\phi_0(x) = -(x-1)(2x-1)(4x-1), \quad \phi_1(x) = \frac{32}{3}x(x-1)(2x-1),$$
$$\phi_2(x) = -4x(x-1)(4x-1), \qquad \phi_3(x) = \frac{1}{3}x(2x-1)(4x-1)$$

and

$$\psi_0(x)|_{I_l} = \frac{64}{3}x(2x-1)(4x-1), \qquad \psi_1(x)|_{I_r} = -2(x-1)(4x-3)(8x-5),$$
$$\psi_2(x)|_{I_r} = \frac{64}{3}(x-1)(2x-1)(4x-3), \quad \psi_3(x)|_{I_r} = -8(x-1)(2x-1)(8x-5).$$

Appendix B. Hermite interpolation basis functions. The Hermite interpolation basis functions are presented here. The interpolation points are put at the cell interface:

$$\tilde{X}_0 = \{0^+, 1^-\}, \quad \tilde{X}_1 = \left\{ \left(\frac{1}{2}\right)^-, \left(\frac{1}{2}\right)^+ \right\}.$$

B.1. $P = 1$ and $K = 1$. The basis functions in $\tilde{W}_0^3$ and $\tilde{W}_1^3$ are

$$\phi_{0,0}(x) = (x-1)^2(2x+1), \quad \phi_{1,0}(x) = -x^2(2x-3),$$
$$\phi_{0,1}(x) = x(x-1)^2, \qquad \phi_{1,1}(x) = x^2(x-1)$$

and

$$\psi_{0,0}(x)|_{I_l} = -4x^2(4x-3), \quad \psi_{1,0}(x)|_{I_r} = 4(x-1)^2(4x-1),$$
$$\psi_{0,1}(x)|_{I_l} = 2x^2(2x-1), \qquad \psi_{1,1}(x)|_{I_r} = 2(x-1)^2(2x-1).$$

B.2. $P = 1$ and $K = 2$. The basis functions in $\tilde{W}_0^5$ and $\tilde{W}_1^5$ are

$$\phi_{0,0}(x) = -(x-1)^3(6x^2+3x+1), \quad \phi_{0,1}(x) = -x(x-1)^3(3x+1),$$
$$\phi_{0,2}(x) = -\frac{1}{2}x^2(x-1)^3, \qquad \phi_{1,0}(x) = x^3(6x^2-15x+10),$$
$$\phi_{1,1}(x) = -x^3(x-1)(3x-4), \qquad \phi_{1,2}(x) = \frac{1}{2}x^3(x-1)^2$$

and

$$\psi_{0,0}(x)|_{I_l} = 16x^3(12x^2-15x+5), \quad \psi_{1,0}(x)|_{I_r} = -16(x-1)^3(12x^2-9x+2),$$
$$\psi_{0,1}(x)|_{I_l} = -8x^3(2x-1)(3x-2), \quad \psi_{1,1}(x)|_{I_r} = -8(x-1)^3(2x-1)(3x-1),$$
$$\psi_{0,2}(x)|_{I_l} = x^3(2x-1)^2, \qquad \psi_{1,2}(x)|_{I_r} = -(x-1)^3(2x-1)^2.$$

REFERENCES

[1] S. Adjerid, K. D. Devine, J. E. Flaherty, and L. Krivodonova, *A posteriori error estimation for discontinuous Galerkin solutions of hyperbolic problems*, Comput. Methods Appl. Mech. Engrg., 191 (2002), pp. 1097–1112.

[2] B. K. ALPERT, *A class of bases in L^2 for the sparse representation of integral operators*, SIAM J. Math. Anal., 24 (1993), pp. 246–262, https://doi.org/10.1137/0524016.

[3] M. A. ALVES, P. CRUZ, A. MENDES, F. D. MAGALHÃES, F. T. PINHO, AND P. J. OLIVEIRA, *Adaptive multiresolution approach for solution of hyperbolic PDEs*, Comput. Methods Appl. Mech. Engrg., 191 (2002), pp. 3909–3928.

[4] R. ARCHIBALD, G. FANN, AND W. SHELTON, *Adaptive discontinuous Galerkin methods in multiwavelets bases*, Appl. Numer. Math., 61 (2011), pp. 879–890.

[5] F. BASSI AND S. REBAY, *Accurate 2D Euler computations by means of a high order discontinuous finite element method*, in Proceedings of the Fourteenth International Conference on Numerical Methods in Fluid Dynamics, Lecture Notes in Phys. 453, Springer, Berlin, Heidelberg, 1995, pp. 234–240.

[6] K. S. BEY AND J. T. ODEN, *hp-version discontinuous Galerkin methods for hyperbolic conservation laws*, Comput. Methods Appl. Mech. Engrg., 133 (1996), pp. 259–286.

[7] B. BIHARI AND A. HARTEN, *Multiresolution schemes for the numerical solution of 2-D conservation laws I*, SIAM J. Sci. Comput., 18 (1997), pp. 315–354, https://doi.org/10.1137/S1064827594278848.

[8] R. BISWAS, K. D. DEVINE, AND J. E. FLAHERTY, *Parallel, adaptive finite element methods for conservation laws*, Appl. Numer. Math., 14 (1994), pp. 255–283.

[9] O. BOKANOWSKI, J. GARCKE, M. GRIEBEL, AND I. KLOMPMAKER, *An adaptive sparse grid semi-Lagrangian scheme for first order Hamilton-Jacobi Bellman equations*, J. Sci. Comput., 55 (2013), pp. 575–605.

[10] H.-J. BUNGARTZ AND M. GRIEBEL, *Sparse grids*, Acta Numer., 13 (2004), pp. 147–269.

[11] J. L. D. CALLE, P. R. B. DEVLOO, AND S. M. GOMES, *Wavelets and adaptive grids for the discontinuous Galerkin method*, Numer. Algorithms, 39 (2005), pp. 143–154, https://doi.org/10.1007/s11075-004-3626-9.

[12] G. CHIAVASSA, R. DONAT, AND S. MÜLLER, *Multiresolution-based adaptive schemes for hyperbolic conservation laws*, in Adaptive Mesh Refinement Theory and Applications, Lect. Notes Comput. Sci. Eng. 41, Springer, Berlin, 2005, pp. 137–159.

[13] B. COCKBURN, S. HOU, AND C.-W. SHU, *The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws, IV, The multidimensional case*, Math. Comp., 54 (1990), pp. 545–581.

[14] B. COCKBURN, G. E. KARNIADAKIS, AND C.-W. SHU, *The development of discontinuous Galerkin methods*, in Discontinuous Galerkin Methods, Lect. Notes Comput. Sci. Eng. 11, Springer, Berlin, 2000, pp. 3–50.

[15] B. COCKBURN AND C.-W. SHU, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework*, Math. Comp., 52 (1989), pp. 411–435.

[16] B. COCKBURN AND C.-W. SHU, *Runge–Kutta discontinuous Galerkin methods for convection-dominated problems*, J. Sci. Comput., 16 (2001), pp. 173–261.

[17] A. COHEN, *Numerical Analysis of Wavelet Methods*, Stud. Math. Appl. 32, North-Holland Publishing, Amsterdam, 2003.

[18] A. COHEN, S. KABER, S. MÜLLER, AND M. POSTEL, *Fully adaptive multiresolution finite volume schemes for conservation laws*, Math. Comp., 72 (2003), pp. 183–225, https://doi.org/10.1090/S0025-5718-01-01391-6.

[19] W. DAHMEN, B. GOTTSCHLICH–MÜLLER, AND S. MÜLLER, *Multiresolution schemes for conservation laws*, Numer. Math., 88 (2001), pp. 399–443, https://doi.org/10.1007/s211-001-8009-3.

[20] N. DISCACCIATI, *Controlling Oscillations in High-order Schemes Using Neural Networks*, master's thesis, Politecnico di Milano and Ecole Polytechnique Federale de Lausanne, Lausanne, France, 2018, http://infoscience.epfl.ch/record/263615.

[21] N. DISCACCIATI, J. S. HESTHAVEN, AND D. RAY, *Controlling Oscillations in High-order Discontinuous Galerkin Schemes Using Artificial Viscosity Tuned by Neural Networks*, Tech. report, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2019.

[22] N. GERHARD, *An Adaptive Multiresolution Discontinuous Galerkin Scheme for Conservation Laws*, Ph.D. thesis, RWTH Aachen University, Aaachen, Germany, 2017.

[23] N. GERHARD, D. CAVIEDES-VOULLIÈME, S. MÜLLER, AND G. KESSERWANI, *Multiwavelet-based grid adaptation with discontinuous Galerkin schemes for shallow water equations*, J. Comput. Phys., 301 (2015), pp. 265–288.

[24] N. GERHARD, F. IACONO, G. MAY, S. MÜLLER, AND R. SCHÄFER, *A high-order discontinuous Galerkin discretization with multiwavelet-based grid adaptation for compressible flows*, J. Sci. Comput., 62 (2015), pp. 25–52.

[25] N. GERHARD AND S. MÜLLER, *Adaptive multiresolution discontinuous Galerkin schemes for*

conservation laws: Multi-dimensional case, Comput. Appl. Math., 35 (2016), pp. 321–349.

[26] M. Griebel and G. Zumbusch, *Adaptive sparse grids for hyperbolic conservation laws*, in Hyperbolic Problems: Theory, Numerics, Applications, Internat. Ser. Numer. Math. 129, Birkhäuser, Basel, 1999, pp. 411–422.

[27] J.-L. Guermond, R. Pasquetti, and B. Popov, *Entropy viscosity method for nonlinear conservation laws*, J. Comput. Phys., 230 (2011), pp. 4248–4267.

[28] W. Guo and Y. Cheng, *A sparse grid discontinuous Galerkin method for high-dimensional transport equations and its application to kinetic simulations*, SIAM J. Sci. Comput., 38 (2016), pp. A3381–A3409, https://doi.org/10.1137/16M1060017.

[29] W. Guo and Y. Cheng, *An adaptive multiresolution discontinuous Galerkin method for time-dependent transport equations in multidimensions*, SIAM J. Sci. Comput., 39 (2017), pp. A2962–A2992, https://doi.org/10.1137/16M1083190.

[30] T. Hagstrom and D. Appelö, *Solving PDEs with Hermite interpolation*, in Spectral and High Order Methods for Partial Differential Equations—ICOSAHOM 2014, Lect. Notes Comput. Sci. Eng. 106, Springer, Cham, 2015, pp. 31–49.

[31] A. Harten, *Multiresolution algorithms for the numerical solution of hyperbolic conservation laws*, Comm. Pure Appl. Math., 48 (1995), pp. 1305–1342, https://doi.org/10.1002/cpa.3160481201.

[32] R. Hartmann and P. Houston, *Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations*, J. Comput. Phys., 183 (2002), pp. 508–532.

[33] R. Hartmann and P. Houston, *Adaptive discontinuous Galerkin finite element methods for nonlinear hyperbolic conservation laws*, SIAM J. Sci. Comput., 24 (2002), pp. 979–1004, https://doi.org/10.1137/S1064827501389084.

[34] P. Houston, B. Senior, and E. Süli, *hp-Discontinuous Galerkin finite element methods for hyperbolic problems: Error analysis and adaptivity*, Internat. J. Numer. Methods Fluids, 40 (2002), pp. 153–169.

[35] N. Hovhannisyan, S. Müller, and R. Schäfer, *Adaptive multiresolution discontinuous Galerkin schemes for conservation laws*, Math. Comp., 83 (2014), pp. 113–151.

[36] J. Huang and C.-W. Shu, *Error estimates to smooth solutions of semi-discrete discontinuous Galerkin methods with quadrature rules for scalar conservation laws*, Numer. Methods Partial Differential Equations, 33 (2017), pp. 467–488.

[37] F. Iacono, G. May, S. Müller, and R. Schäfer, *An adaptive multiwavelet-based DG discretization for compressible fluid flow*, in Computational Fluid Dynamics 2010, Springer, Berlin, Heidelberg, 2011, pp. 813–820.

[38] G. Kesserwani, D. Caviedes-Voullieme, N. Gerhard, and S. Müller, *Multiwavelet discontinuous Galerkin h-adaptive shallow water model*, Comput. Methods Appl. Mech. Engrg., 294 (2015), pp. 56–71.

[39] A. Klöckner, T. Warburton, and J. S. Hesthaven, *Viscous shock capturing in a time-explicit discontinuous Galerkin method*, Math. Model. Nat. Phenom., 6 (2011), pp. 57–83.

[40] A. Kornelus and D. Appelö, *On the scaling of entropy viscosity in high order methods*, in Spectral and High Order Methods for Partial Differential Equations—ICOSAHOM 2016, Lect. Notes Comput. Sci. Eng. 119, Springer, Cham, 2017, pp. 175–187.

[41] A. Kurganov, G. Petrova, and B. Popov, *Adaptive semidiscrete central-upwind schemes for nonconvex hyperbolic conservation laws*, SIAM J. Sci. Comput., 29 (2007), pp. 2381–2401, https://doi.org/10.1137/040614189.

[42] B. Lastdrager, B. Koren, and J. Verwer, *The sparse-grid combination technique applied to time-dependent advection problems*, Appl. Numer. Math., 38 (2001), pp. 377–401.

[43] Y. Liu, Y. Cheng, S. Chen, and Y.-T. Zhang, *Krylov implicit integration factor discontinuous Galerkin methods on sparse grids for high dimensional reaction-diffusion equations*, J. Comput. Phys., 388 (2019), pp. 90–102.

[44] L. Pareschi and G. Russo, *Implicit–explicit Runge–Kutta schemes and applications to hyperbolic systems with relaxation*, J. Sci. Comput., 25 (2005), pp. 129–155.

[45] P.-O. Persson and J. Peraire, *Sub-cell shock capturing for discontinuous Galerkin methods*, in Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit, 2006, 112.

[46] J. Qiu and C.-W. Shu, *Runge–Kutta discontinuous Galerkin method using WENO limiters*, SIAM J. Sci. Comput., 26 (2005), pp. 907–929, https://doi.org/10.1137/S1064827503425298.

[47] D. Ray and J. S. Hesthaven, *An artificial neural network as a troubled-cell indicator*, J. Comput. Phys., 367 (2018), pp. 166–191.

[48] J.-F. Remacle, J. E. Flaherty, and M. S. Shephard, *An adaptive discontinuous Galerkin technique with an orthogonal basis applied to compressible flow problems*, SIAM Rev., 45 (2003), pp. 53–72, https://doi.org/10.1137/S00361445023830.

[49] A. B. SHELTON, *A Multi-resolution Discontinuous Galerkin Method for Unsteady Compressible Flows*, Ph.D. thesis, Georgia Institute of Technology, Atlanta, GA, 2008.

[50] J. SHEN AND H. YU, *Efficient spectral sparse grid methods and applications to high-dimensional elliptic problems*, SIAM J. Sci. Comput., 32 (2010), pp. 3228–3250, https://doi.org/10.1137/100787842.

[51] C.-W. SHU AND S. OSHER, *Efficient implementation of essentially non-oscillatory shock-capturing schemes*, J. Comput. Phys., 77 (1988), pp. 439–471.

[52] Z. TAO, W. GUO, AND Y. CHENG, *Sparse Grid Discontinuous Galerkin Methods for the Vlasov-Maxwell System*, preprint, https://arxiv.org/abs/1805.10747, 2018.

[53] Z. TAO, Y. JIANG, AND Y. CHENG, *An Adaptive High-Order Piecewise Polynomial Based Sparse Grid Collocation Method with Applications*, preprint, https://arxiv.org/abs/1912.03982, 2019.

[54] M. VUIK AND J. RYAN, *Multiwavelet troubled-cell indicator for discontinuity detection of discontinuous Galerkin schemes*, J. Comput. Phys., 270 (2014), pp. 138–160.

[55] Z. WANG, Q. TANG, W. GUO, AND Y. CHENG, *Sparse grid discontinuous Galerkin methods for high-dimensional elliptic equations*, J. Comput. Phys., 314 (2016), pp. 244–263.

[56] A. ZEISER, *Fast matrix-vector multiplication in the sparse-grid Galerkin method*, J. Sci. Comput., 47 (2011), pp. 328–346.