SIAM J. SCI. COMPUT. © 2019 Society for Industrial and Applied Mathematics
Vol. 41, No. 2, pp. A1278–A1303

MULTILEVEL TECHNIQUES FOR COMPRESSION AND REDUCTION OF SCIENTIFIC DATA—THE MULTIVARIATE CASE[*]

MARK AINSWORTH[†,‡], OZAN TUGLUK[†], BEN WHITNEY[†], AND SCOTT KLASKY[‡]

Abstract. We develop a technique for multigrid adaptive reduction of data (`MGARD`). Special attention is given to the case of tensor product grids, where our approach permits the use of nonuniformly spaced grids in each direction, which can prove problematic for many types of data reduction methods. An important feature of our approach is the provision of guaranteed, computable bounds on the loss incurred by the reduction of the data. Many users are leery of lossy algorithms and will only consider using them provided that numerical bounds on the pointwise difference between the original and the reduced datasets are given. Accordingly, we develop techniques for bounding the loss measured in the $L^\infty(\Omega)$ norm, and we show that these bounds are realistic in the sense that they do not significantly overestimate the actual loss. The resulting loss indicators are used to guide the adaptive reduction of the data so that the reduced dataset meets a user-prescribed tolerance or memory constraint. Illustrative numerical examples, including the reduction of data arising from the simulation of a nonlinear reaction-diffusion problem, a turbulent channel flow, and a climate simulation, are provided.

Key words. data reduction, data compression, adaptive reduction

AMS subject classification. 65N30

DOI. 10.1137/18M1166651

1. Introduction.

President Obama's executive order mandating the creation of a National Strategic Computing Initiative[1] will further increase the already prohibitive volumes of data being produced by computational simulations on leadership computing facilities. Moreover, the types of subsystems likely to be used on exascale machines are improving at uneven rates, causing a widening gulf between floating point operations per second performance and memory speed (as measured by memory bandwidth, memory latency, and interconnect performance) [26]. These imbalances on the compute nodes are exacerbated by the desire to write simulation results to disc for postprocessing. For example, the exascale machines projected to come online in the next decade are expected to be able to write 10^{12} bytes to disc per second—an amount twice what is currently possible, but representing a compute speed-memory speed discrepancy 50 times greater than currently experienced [14]. The increasing computational power of modern machines therefore poses a great challenge if these machines are to be used to their full potential.

[*]Submitted to the journal's Methods and Algorithms for Scientific Computing section January 23, 2018; accepted for publication (in revised form) February 12, 2019; published electronically April 25, 2019.
http://www.siam.org/journals/sisc/41-2/M116665.html

Funding: This work was partially supported by the Exascale Computing Project (17-SC-20-SC) of the U.S. Department of Energy; the Advanced Scientific Research Office (ASCR) at the Department of Energy, under contract DE-AC02-06CH11357; the DOE Storage Systems and Input/Output for Extreme Scale Science project, announcement number LAB 15-1338; and DOE and UT–Battelle, LLC, contract number DE-AC05-00OR22725.

[†]Division of Applied Mathematics, Brown University, Providence, RI 02912 (Mark_Ainsworth@brown.edu, ozan_tugluk1@brown.edu, ben_whitney@brown.edu).

[‡]Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831 (klasky@ornl.gov).

[1]Executive Order – *Creating a National Strategic Computing Initiative*, E.O. 13702, July 29, 2015.

Enter compression and data reduction methods, which trade an increase in computational workload for a decrease in storage cost, resulting in improved data transfer and loading performance. Compression of scientific data is an active area of research, and a diverse range of methods have been developed, including methods based on polynomial interpolation and extrapolation [19, 13, 24, 1], Fourier transforms [32, 23], statistical methods [7, 10], tensor-based methods [4], reduced basis methods [18, 29], and hierarchical and wavelet-based methods [12, 16, 30, 25, 6, 9, 27]. The choice of a reduction method should be informed by the storage media available and the purposes for which the dataset (or a reduced representation thereof) will be needed. The storage hardware of a typical modern cluster consists of a heterogeneous collection of devices. Fastest but short-lived and least capacious is the main memory of the machine. At the opposite extreme, tape media can store enormous volumes of data for archival purposes but require weeks or months for data retrieval. In between we have nonvolatile RAM, solid state drives, hard disk drives, optical media, network storage, etc. Hierarchical reduction methods decompose the data into a sequence of components better suited to these types of heterogeneous storage media. Such methods are especially appropriate when some of the applications for which the data are to be used do not require the full resolution of the original dataset. For instance, the user may simply require that the reduced dataset meet a prescribed tolerance or, alternatively, that the reduced dataset may be stored in the available RAM. These two typical workflows are illustrated in Figure 1.

In previous work, we developed a hierarchical reduction method for univariate data on uniformly spaced grids [3]. The current effort extends these ideas to include multiple dimensions and adaptive reduction to a user-prescribed tolerance or memory constraint. We give special attention to the case of tensor product grids, which frequently occur in the solution of differential equations on regular domains and in timestepping applications. Our approach permits the use of nonuniformly spaced grids, which can prove problematic for many types of data reduction methods. Such grids arise, for example, in the simulation of turbulent flows, where Chebyshev nodes are often used.

An important feature of our approach is the provision of guaranteed, computable bounds on the loss incurred by the reduction of the data. Many users are leery of lossy algorithms and will only consider using them provided that numerical bounds on the pointwise difference between the original and the reduced datasets are given. Accordingly, we develop techniques for bounding the loss measured in the $L^\infty(\Omega)$ norm, and we show that these bounds are realistic in the sense that they do not significantly overestimate the actual loss. The resulting *loss indicators* are used to guide the adaptive reduction of the data so that the reduced dataset meets the storage or accuracy requirements of the user.

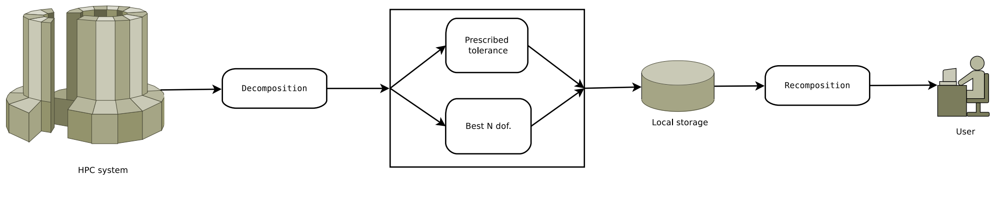


FIG. 1. *Illustration of the workflow for hierarchical reduction methods.*

The remainder of this paper is organized as follows. In section 2, we present the abstract framework of the reduction technique, which is applied to the tensor product case in section 3. Section 4 concerns the implementation of the core decomposition and recomposition procedures. In section 5, we present applications to datasets arising from the simulation of a nonlinear reaction-diffusion problem, a turbulent channel flow, and a climate simulation. Details regarding the derivation of the loss indicator are given in Appendix A.

2. Algorithms for data reduction based on orthogonal projection.

Consider a function u defined on some domain $\Omega \subset \mathbb{R}^d$ representing the solution to a problem of interest. Typically u (or an approximation to u) is obtained in digital form through a discretization consisting of the selection of a set of degrees of freedom and the determination of their values. These degrees of freedom may be, for example,

- values taken by u (or a derivative) at certain points within Ω,
- coefficients for u with respect to some basis of functions on Ω, or
- a collection of derived quantities uniquely determining u.

In order to output or store the digital form of u, it is often necessary to create reduced representations of u, which sacrifice some fidelity in the discretization in exchange for a reduction in filesize. To produce a reduced representation, one may choose a smaller set of degrees of freedom to represent an approximation to the original function. In the simplest case, this set may be a subset of the original collection. For example, a signal stored as a collection of frequencies may be filtered by discarding high-frequency components, or a timeseries may be decimated, with only a fraction of the original datapoints retained [1]. In general, though, the second set of degrees of freedom may bear little or no relation to the first. In this work, we present a hierarchical set of degrees of freedom which can be efficiently determined and naturally manipulated to produce reduced representations of an input function u with no increase in storage cost. In subsection 2.1 we establish the setting and define the degrees of freedom used. In subsection 2.2 we develop algorithms for generating reduced representations of arbitrary accuracy or size from these degrees of freedom.

2.1. Notation and degrees of freedom.

Let $\Omega \subset \mathbb{R}^d$ be a domain with a regular partition $\mathcal{P}_L$, and let $u \in V_L$ be a given function, where V_L is the space of continuous piecewise (multi)linear functions on $\mathcal{P}_L$. Suppose that $\mathcal{P}_L$ is obtained by repeatedly refining an initial partition $\mathcal{P}_0$, generating a sequence of partitions $\{\mathcal{P}_\ell : 0 \leq \ell \leq L\}$ of Ω. Let $\mathcal{N}_\ell$ denote the set of vertices of the elements in $\mathcal{P}_\ell$, and let V_ℓ denote the space of continuous piecewise (multi)linear functions with knots $\mathcal{N}_\ell$, so that $V_0 \subseteq \cdots \subseteq V_L$. Let $Q_\ell \colon V_L \to V_\ell$ denote the $L^2(\Omega)$ projection onto V_ℓ, which can be computed by inverting the Gramian (or mass) matrix. Let $\Pi_\ell \colon V_L \to V_\ell$ denote the piecewise linear interpolant onto V_ℓ; i.e., $\Pi_\ell v$ and v agree at the knots $\mathcal{N}_\ell$. We define Q_{-1} and Π_{-1} to be the zero operator.

As mentioned in the introduction, modern day storage systems are often hierarchical in nature, and data stored in such a system must be judiciously distributed across the entire hierarchy. To this end, we consider hierarchical decompositions of the data. There are many possible hierarchical decompositions of the data u relative to the partitioning $\{\mathcal{P}_\ell : 0 \leq \ell \leq L\}$, but not all are suitable for data reduction. In this work, we shall make use of a particular decomposition defined as follows. Let $\Delta_\ell \colon V_L \to V_\ell$ be given by $\Delta_\ell u = (I - \Pi_{\ell-1})Q_\ell u$. We refer to $\{\Delta_\ell u : 0 \leq \ell \leq L\}$ as the *multilevel components* of u. The multilevel components do not, strictly speaking, define a decomposition since they do not sum directly to u. Nevertheless, given the multilevel components we can reconstruct the original function u thanks to the following identity:

$$\begin{aligned}\sum_{\ell=0}^{L}(I-Q_{\ell-1})\Delta_\ell u &= \sum_{\ell=0}^{L}(I-\Pi_{\ell-1}-Q_{\ell-1}+Q_{\ell-1}\Pi_{\ell-1})Q_\ell u \\ &= \sum_{\ell=0}^{L}(I-Q_{\ell-1})Q_\ell u = \sum_{\ell=0}^{L}(Q_\ell - Q_{\ell-1})u = u,\end{aligned} \tag{1}$$

where we use the identities $Q_{\ell-1}\Pi_{\ell-1}=\Pi_{\ell-1}$ and $Q_{\ell-1}Q_\ell = Q_{\ell-1}$. In view of this identity, we can regard $\{\Delta_\ell u : 0 \le \ell \le L\}$ as a splitting:

$$V_L \ni u \longleftrightarrow (\Delta_0 u, \dots, \Delta_L u) \in V_0 \times \cdots \times V_L. \tag{2}$$

At first glance, it appears that it will be $L+1$ times as expensive to store the components in the *multilevel splitting* (2) as to store the original data u. Fortunately, this proves not to be the case. Consider how many degrees of freedom are required to store $\Delta_\ell u \in V_\ell$. Let $\{\phi_\ell(\,\cdot\,;x) : x \in \mathcal{N}_\ell\}$ be the Lagrange basis for V_ℓ, defined by $\phi_\ell(y;x) = \delta_{yx}$ for $x, y \in \mathcal{N}_\ell$, so that we may write

$$\Delta_\ell u(y) = \sum_{x\in\mathcal{N}_\ell} \Delta_\ell u(x)\phi_\ell(y;x) = \sum_{x\in\mathcal{N}_\ell\setminus\mathcal{N}_{\ell-1}} \Delta_\ell u(x)\phi_\ell(y;x), \tag{3}$$

where we have used that $\Delta_\ell u(x) = 0$ for $x \in \mathcal{N}_{\ell-1}$. (This follows from the fact that $Q_\ell u(x) = \Pi_{\ell-1}Q_\ell u(x)$ for $x \in \mathcal{N}_{\ell-1}$.) In other words, $\Delta_\ell u$ can be stored using only the values $\Delta_\ell u(x)$ at each of the "new" nodes on level ℓ, i.e., at each $x \in \mathcal{N}_\ell^* = \mathcal{N}_\ell \setminus \mathcal{N}_{\ell-1}$. Thus, the number of degrees of freedom required to store $\Delta_\ell u$ is not $\#\mathcal{N}_\ell$ but $\#\mathcal{N}_\ell^*$, the difference between $\#\mathcal{N}_\ell$ and $\#\mathcal{N}_{\ell-1}$. This means that the cost of storing all of the multilevel components $\{\Delta_\ell u : 0 \le \ell \le L\}$ is actually identical to the cost of storing the original function: $\#\mathcal{N}_L$ degrees of freedom.

The *hierarchical basis* for V_L, consisting of the Lagrange functions $\{\phi_\ell(\,\cdot\,;x) : x \in \mathcal{N}_\ell^*\}$ on each level ℓ, can also be used to represent the *hierarchical decomposition* [6]

$$V_L \ni u \longleftrightarrow ((\Pi_0 - \Pi_{-1})u, \dots, (\Pi_L - \Pi_{L-1})u) \in V_0 \times \cdots \times V_L. \tag{4}$$

The multilevel splitting is *not* the same as the hierarchical decomposition: as noted above, the sum of the multilevel components is not u. In fact, as (1) shows, the multilevel splitting is actually more akin to the *orthogonal decomposition* [9, 27]

$$V_L \ni u \longleftrightarrow ((Q_0 - Q_{-1})u, \dots, (Q_L - Q_{L-1})u) \in V_0 \times \cdots \times V_L, \tag{5}$$

which enjoys stability properties superior to those of the hierarchical decomposition. For instance, the orthogonal decomposition is well-defined for all $u \in L^2(\Omega)$, whereas the hierarchical decomposition requires continuous data (in order for the interpolation operator Π_ℓ to be bounded). Later, in Figures 14 and 15, we illustrate the comparative behavior of the hierarchical and orthogonal decompositions. In particular, Figure 15 shows the more uniform decay of the coefficients in the orthogonal decomposition. Storing the orthogonal decomposition directly, in the absence of bases for $\{V_\ell \cap V_{\ell-1}^\perp : 0 \le \ell \le L\}$, would require $\sum_{\ell=0}^{L} \#\mathcal{N}_\ell$ degrees of freedom. The multilevel splitting, which we use in the present work, enjoys the best of both worlds: minimal storage and superior stability.

2.2. Data reduction. The splitting defined by (2) will form the basis of our data reduction algorithms. As shown in (3), each multilevel component $\Delta_\ell u$ can be represented by the scalars $\{\Delta_\ell u(x) : x \in \mathcal{N}_\ell^*\}$. We refer to the collection over all levels $\ell \in \{0, \ldots, L\}$ of these scalars as the *multilevel coefficients* for u. In order to store the coefficients on a computer, we allocate storage `u_mc` and initialize it with values

$$\texttt{u\_mc}[x] = \Delta_\ell u(x) = (I - \Pi_{\ell-1}) Q_\ell u(x) \quad \text{for} \quad x \in \mathcal{N}_\ell^* \quad \text{and} \quad \ell \in \{0, \ldots, L\}. \tag{6}$$

In order to reduce the amount of storage required to represent u, we must discard some of these multilevel coefficients. Specifically, we retain only the values of `u_mc` on a subset $\mathcal{N} \subset \mathcal{N}_L$, thereby obtaining a set `ũ_mc` of reduced multilevel coefficients defined by

$$\tilde{\texttt{u}}\texttt{\_mc}[x] = \begin{cases} \texttt{u\_mc}[x] & x \in \mathcal{N}, \\ 0 & \text{otherwise.} \end{cases}$$

In order to define the corresponding reduced representation $\tilde{u}$, we first use (1) to express u in terms of `u_mc` as follows:

$$u = \sum_{\ell=0}^{L} (I - Q_{\ell-1}) \sum_{x \in \mathcal{N}_\ell^*} \texttt{u\_mc}[x]\, \phi_\ell(\,\cdot\,; x).$$

By analogy, the reduced coefficients `ũ_mc` define a function $\tilde{u} \in V_L$ by

$$\tilde{u} = \sum_{\ell=0}^{L} (I - Q_{\ell-1}) \sum_{x \in \mathcal{N}_\ell^*} \tilde{\texttt{u}}\texttt{\_mc}[x]\, \phi_\ell(\,\cdot\,; x).$$

Based on the above definition, Algorithm 4 in subsection 4.4 gives an efficient implementation for constructing the values of $\tilde{u}$ at the nodes of the original grid from a set of reduced multilevel coefficients `ũ_mc`. The resulting function $\tilde{u}$ is a lossy reduced representation of u which requires the storage of $\#\mathcal{N}$ coefficients as compared with the original $\#\mathcal{N}_L$. Ideally, we would choose the set $\mathcal{N}$ to have as few entries as possible while at the same time ensuring that the overall loss satisfies $\|u - \tilde{u}\|_{L^\infty} / \|u\|_{L^\infty} \leq \tau$ for some user-prescribed tolerance $\tau \geq 0$. Conversely, in some applications hardware resources may impose a strict limit on the number N of degrees of freedom that can be stored. In this case, we would seek to minimize the loss $\|u - \tilde{u}\|_{L^\infty}$ subject to the condition $\#\mathcal{N} \leq N$. Two questions naturally arise in these scenarios:

1. How should the index set $\mathcal{N}$ be chosen?
2. For a given index set $\mathcal{N}$, how can we estimate the loss incurred?

In the latter case, we seek a *loss indicator*: a function $\eta_\infty \colon V_L \to [0, \infty)$ satisfying

$$C_1 \|u - \tilde{u}\|_{L^\infty} \leq \eta_\infty(u - \tilde{u}) \leq C_2 \|u - \tilde{u}\|_{L^\infty} \quad \text{for all} \quad u - \tilde{u} \in V_L$$

for some constants $0 < C_1 \leq C_2$. The left-hand bound establishes that the indicator is *reliable*—that the true loss can always be guaranteed to meet a specified tolerance. The right-hand bound establishes that it is *efficient*—that it will not indicate a large loss where none exists. The notion of a loss indicator is analogous to the notion of an a posteriori error estimator for a numerical method [2].

In general, we will be able to express any linear indicator $\eta_\infty(u - \tilde{u})$ explicitly as a function of the difference between the multilevel coefficients `u_mc` and `ũ_mc`.

This enables us to assess the effect of discarding coefficients in u_mc and thereby to provide a criterion for choosing the index set $\mathcal{N}$. In the next section we will give a concrete example of a loss indicator and give algorithms for selecting $\mathcal{N}$ in the case of tensor product grids on Cartesian product domains.

3. Multilevel reduction on tensor product grids.

In this section we apply the general framework given in section 2 to the specific case of tensor product grids. Let Ω be the domain $[a^1, b^1] \times \cdots \times [a^d, b^d]$. For simplicity, we will choose the initial partition $\mathcal{P}_0$ to consist of the single element Ω and recursively generate $\mathcal{P}_{\ell+1}$ from $\mathcal{P}_\ell$ by bisecting each of the intervals, i.e., subdividing each element in $\mathcal{P}_\ell$ into 2^d new elements in $\mathcal{P}_{\ell+1}$. This results in an increasing sequence $V_0 \subset \cdots \subset V_L$ of tensor product function spaces.

3.1. Loss indicator.

Following the general approach described in the previous section, we shall generate reduced representations for an input function $u \in V_L$ by discarding a subset of the multilevel coefficients u_mc. To this end, we define a loss indicator $\eta_\infty \colon V_L \to [0, \infty)$ as follows:

$$\eta_\infty(v) = \sum_{\ell=0}^{L} \|\Delta_\ell v\|_{L^\infty} = \sum_{\ell=0}^{L} \|(I - \Pi_{\ell-1}) Q_\ell v\|_{L^\infty}. \tag{7}$$

The function η_∞ is easily computable in terms of the multilevel coefficients v_mc for v:

$$\eta_\infty(v) = \sum_{\ell=0}^{L} \|\Delta_\ell v\|_{L^\infty} = \sum_{\ell=0}^{L} \Big\| \sum_{x \in \mathcal{N}_\ell^*} \texttt{v\_mc}[x] \phi_\ell(\,\cdot\,; x) \Big\|_{L^\infty} = \sum_{\ell=0}^{L} \max_{x \in \mathcal{N}_\ell^*} |\texttt{v\_mc}[x]|, \tag{8}$$

where the first equality holds by (7), the second holds by (6), and the third is a property of the piecewise linear Lagrange basis functions. The following result establishes that η_∞ is an efficient and reliable loss indicator.

THEOREM 1. *Let V_L be the space of continuous piecewise multilinear functions on a tensor product grid in d dimensions. Then*

$$C_1 \eta_\infty(v) \le \|v\|_{L^\infty} \le C_2 \eta_\infty(v) \quad \textit{for all} \quad v \in V_L$$

with $C_1 \ge 3^{-d}/(2(L+1))$ and $C_2 \le 1 + 3^d$.

Proof. We begin with the left-hand bound. For any $v \in V_L$,

$$\begin{aligned} \eta_\infty(v) &= \sum_{\ell=0}^{L} \|(I - \Pi_{\ell-1}) Q_\ell v\|_{L^\infty} \\ &\le (L+1) \max_{\ell \in \{0,\dots,L\}} \|I - \Pi_{\ell-1}\|_{L^\infty} \max_{\ell \in \{0,\dots,L\}} \|Q_\ell\|_{L^\infty} \|v\|_{L^\infty}, \end{aligned}$$

where $\|I - \Pi_{\ell-1}\|_{L^\infty}$ and $\|Q_\ell\|_{L^\infty}$ denote the operator norms of $I - \Pi_{\ell-1} \colon L^\infty(V_\ell) \to L^\infty(V_\ell)$ and $Q_\ell \colon L^\infty(V_L) \to L^\infty(V_\ell)$, respectively. The operator norm of $\Pi_{\ell-1} \colon V_\ell \to V_\ell$ is 1 for $\ell \neq 0$ and 0 for $\ell = 0$, so the first maximum is at most 2 by the triangle inequality. We show below, following the proof of [11, Theorem 1], that $\|Q_\ell\|_{L^\infty} \le 3$ when $d = 1$. For general d, then, $\|Q_\ell\|_{L^\infty} \le 3^d$ by the definition of the tensor product. As a result, $\eta_\infty(v) \le C_1^{-1} \|v\|_{L^\infty}$ with $C_1^{-1} \le 2 \cdot 3^d \cdot (L+1)$.

We next show that $\|Q_\ell\|_{L^\infty} \le 3$ when $d = 1$. The domain in this case is simply an interval $[a^1, b^1]$. Fix a level ℓ in the hierarchy, and enumerate $\mathcal{N}_\ell$ as $a^1 = x^1 < \cdots < x^n = b^1$ with $n = \#\mathcal{N}_\ell$. Let M be the mass matrix on V_ℓ defined by $M_{i,j} = (\phi_\ell(\,\cdot\,; x^i), \phi_\ell(\,\cdot\,; x^j))$. Write $M = D(I+K)$ with I the identity matrix, D the diagonal matrix given by $D_{i,i} = M_{i,i}$, and K the bidiagonal matrix with entries $K_{i,j}$ equal to $M_{i,j}/M_{i,i}$ if $j \in \{i-1, i+1\}$ and 0 otherwise. The norm of K as an operator $\ell^\infty(\mathbb{R}^n) \to \ell^\infty(\mathbb{R}^n)$, denoted $\|K\|_{\ell^\infty}$, is calculated by taking the maximum of the row sums:

$$\|K\|_{\ell^\infty} = \max_{i\in\{1,\dots,n\}} \sum_{j=1}^{n} |K_{i,j}| = \max_{i\in\{1,\dots,n\}} \frac{M_{i,i-1} + M_{i,i+1}}{M_{i,i}}$$

with the convention that $M_{i,0} = M_{n,n+1} = 0$. Let h_i^- and h_i^+ denote the element widths $x^i - x^{i-1}$ and $x^{i+1} - x^i$, respectively, with the convention that $h_1^- = h_n^+ = 0$. Simple integrations show that $M_{i,i-1} = h_i^-/6$, $M_{i,i+1} = h_i^+/6$, and $M_{i,i} = (h_i^- + h_i^+)/3$, so that $\|K\|_{\ell^\infty} = 1/2$. Thus, $(I+K)^{-1} = \sum_{m=0}^{\infty} (-K)^m$, and $\|(I+K)^{-1}\|_{\ell^\infty} \le \sum_{m=0}^{\infty} \|K\|_{\ell^\infty}^m = 2$, with $\|\cdot\|_{\ell^\infty}$ again denoting the $\ell^\infty(\mathbb{R}^n) \to \ell^\infty(\mathbb{R}^n)$ operator norm.

Fix a function $v \in V_L$, and let $\vec{\alpha}$ be the vector of nodal values of $Q_\ell v$ defined by $\alpha_i = Q_\ell v(x^i)$. $\vec{\alpha}$ solves the system $M\vec{\alpha} = (u, \vec{\phi}_\ell)$, where $(u, \vec{\phi}_\ell)$ is the vector of inner products given by $(u, \vec{\phi}_\ell)_i = (u, \phi_\ell(\,\cdot\,; x^i))$. Thus, $\vec{\alpha} = (I+K)^{-1}D^{-1}(u, \vec{\phi}_\ell)$, and so, with $|\cdot|_{\ell^\infty}$ denoting the norm of a vector in $\ell^\infty(\mathbb{R}^n)$,

$$\begin{aligned} |\vec{\alpha}|_{\ell^\infty} &\le \|(I+K)^{-1}\|_{\ell^\infty} |D^{-1}(u, \vec{\phi}_\ell)|_{\ell^\infty} \\ &\le 2 \max_{i\in\{1,\dots,n\}} |D_{i,i}^{-1}(u, \phi_\ell(\,\cdot\,; x^i))|_{\ell^\infty} \\ &\le 2 \max_{i\in\{1,\dots,n\}} (M_{i,i})^{-1} \|u\|_{L^\infty} \|\phi_\ell(\,\cdot\,; x^i)\|_{L^1}. \end{aligned}$$

Another integration shows $\|\phi_\ell(\,\cdot\,; x^i)\|_{L^1} = (h_i^- + h_i^+)/2 = 3M_{i,i}/2$, so $|\vec{\alpha}|_{\ell^\infty} \le 3\|u\|_{L^\infty}$. As $|\vec{\alpha}|_{\ell^\infty} = \|Q_\ell v\|_{L^\infty}$, we conclude that $\|Q_\ell\|_{L^\infty} \le 3$. This completes the derivation of the left-hand bound.

Now we turn to the right-hand bound. Recall that $v = \sum_{\ell=0}^{L} (I - Q_{\ell-1})\Delta_\ell v$ for any $v \in V_L$. By the triangle inequality,

$$\|v\|_{L^\infty} \le \max_{\ell\in\{0,\dots,L\}} \|I - Q_{\ell-1}\|_{L^\infty} \sum_{\ell=0}^{L} \|\Delta_\ell v\|_{L^\infty} = \max_{\ell\in\{0,\dots,L\}} \|I - Q_{\ell-1}\|_{L^\infty} \eta_\infty(v).$$

Here $\|I - Q_{\ell-1}\|_{L^\infty}$ denotes the operator norm of $I - Q_{\ell-1}\colon L^\infty(\operatorname{im}(\Delta_\ell)) \to L^\infty(V_\ell)$. As $\operatorname{im}(\Delta_\ell) \subseteq V_\ell$, we can use the triangle inequality and the previously obtained bound on $\|Q_{\ell-1}\|_{L^\infty}$ to conclude that $\|v\|_{L^\infty} \le C_2\eta_\infty(v)$ with $C_2 \le 1 + 3^d$. ∎

Theorem 1 holds for arbitrary mesh spacings. As a consequence, the constants C_1 and C_2 can be overly pessimistic in certain cases. In particular, in the case of meshes with uniform spacing the right-hand bound can be improved as follows.

THEOREM 2. *Let V_L be the space of continuous piecewise multilinear functions on a uniform tensor product grid in d dimensions. Then*

$$C_1\eta_\infty(v) \le \|v\|_{L^\infty} \le C_2\eta_\infty(v) \quad \textit{for all} \quad v \in V_L$$

with $C_1 \ge 3^{-d}/(2(L+1))$ and $C_2 \le 1 + (\sqrt{3}/2)^d$.

This improved result follows from Theorem 1 and Appendix A, where it is shown that $\|I - Q_{\ell-1}\|_{L^\infty} \le 1 + (\sqrt{3}/2)^d$ in the case of uniform meshes. Alternative specializations can be obtained when a particular mesh is in hand. For example, in subsection 5.2 reductions are carried out on a tensor product grid made of uniform elements in the x and z dimensions and Chebyshev elements in the y dimension. Applying the general bound in the y dimension and the improved bound for uniform grids in the x and z dimensions, we obtain the following.

Corollary 3. *Let V_L be the space of continuous piecewise multilinear functions on a tensor product grid in three dimensions with uniform spacing in two dimensions and Chebyshev spacing in the third. Then*

$$C_1\eta_\infty(v) \le \|v\|_{L^\infty} \le C_2\eta_\infty(v) \quad \textit{for all} \quad v \in V_L$$

with $C_1 \ge 3^{-3}/(2(L+1))$ and $C_2 \le 1 + 3^2/2^2$.

3.2. Reduction algorithms. We can now develop algorithms for the use cases of constrained loss and constrained storage. Of course, one cannot simultaneously specify both a prescribed loss tolerance and a prescribed reduction ratio; each case must be dealt with separately. We begin with constrained loss. Given a function $u \in V_L$ and some tolerance $\tau \ge 0$, we seek a reduced representation $\tilde{u}$ represented by a minimal number of nonzero degrees of freedom such that $\|u-\tilde{u}\|_{L^\infty}/\|u\|_{L^\infty} \le \tau$. As in the previous section, we will generate $\tilde{u}$ by selectively discarding certain multilevel coefficients `u_mc[`x`]`, letting the retained coefficients be indexed by $\mathcal{N}$. The loss indicator $\eta_\infty(u-\tilde{u})$ can be easily determined given the multilevel coefficients for $u-\tilde{u}$, as in (8). The multilevel coefficients depend linearly on the function, so these are simply the differences between `u_mc`, the coefficients for u, and `ũ_mc`, the coefficients for $\tilde{u}$. As a result,

$$\eta_\infty(u-\tilde{u}) = \sum_{\ell=0}^{L} \max_{x\in\mathcal{N}_\ell^*} |\texttt{u\_mc[}x\texttt{]} - \tilde{\texttt{u}}\texttt{\_mc[}x\texttt{]}| = \sum_{\ell=0}^{L} \max_{\substack{x\in\mathcal{N}_\ell^*\\ x\notin\mathcal{N}}} |\texttt{u\_mc[}x\texttt{]}|.$$

Taking the maximum over all levels ℓ,

$$\eta_\infty(u-\tilde{u}) \le (L+1) \max_{\substack{x\in\mathcal{N}_L\\ x\notin\mathcal{N}}} |\texttt{u\_mc[}x\texttt{]}|.$$

Using Theorem 1, it suffices to find $\tilde{u}$ such that $\eta_\infty(u-\tilde{u}) \le \tau\|u\|_{L^\infty}/C_2$. So, if we retain `u_mc[`$x$`]` for all $x \in \mathcal{N}_L$ satisfying $|\texttt{u\_mc[}x\texttt{]}| > (\tau\|u\|_{L^\infty})/(C_2(L+1))$, then the resulting reduced representation $\tilde{u}$ will meet the prescribed bound on the relative loss. This gives rise to Algorithm 1.

The loss indicator may also be used to guide an algorithm for the use case of constrained storage. Given a budget of N degrees of freedom, which multilevel coefficients should we keep? The natural course is to minimize the loss indicator

$$\eta_\infty(u-\tilde{u}) = \sum_{\ell=0}^{L} \|\Delta_\ell(u-\tilde{u})\|_{L^\infty} = \sum_{\ell=0}^{L} \max_{\substack{x\in\mathcal{N}_\ell^*\\ x\notin\mathcal{N}}} |\texttt{u\_mc[}x\texttt{]}|.$$

This *knapsack problem* [8] can be solved to give the set $\mathcal{N}$ of N coefficients which should be retained. For simplicity we instead choose to minimize

$$(L+1) \max_{\substack{x\in\mathcal{N}_L\\ x\notin\mathcal{N}}} |\texttt{u\_mc[}x\texttt{]}| \ge \sum_{\ell=0}^{L} \max_{\substack{x\in\mathcal{N}_\ell^*\\ x\notin\mathcal{N}}} |\texttt{u\_mc[}x\texttt{]}| = \eta_\infty(u-\tilde{u}),$$

giving rise to Algorithm 2.

Algorithm 1. Reduction given a constraint on the maximum allowable loss in the reduced representation.

```
function AdaptiveReductionErrorConstraint(multilevel coefficients u_mc,
norm $\|u\|_\infty$, maximum level $L$, tolerance $\tau$)
    ũ_mc ← []
    $\mathcal{N} \leftarrow \{\}$
    for $x \in \mathcal{N}_L$ do
        if $|$u_mc[$x$]$| > (\tau\|u\|_\infty)/(C_2(L+1))$ then
            ũ_mc[$x$] ← u_mc[$x$]
            $\mathcal{N} \leftarrow \mathcal{N} \cup \{x\}$
        else
            ũ_mc[$x$] ← 0
        end if
    end for
    return ũ_mc, $\mathcal{N}$
end function
```

Algorithm 2. Approximation given a constraint on the degrees of freedom available for the reduced representation.

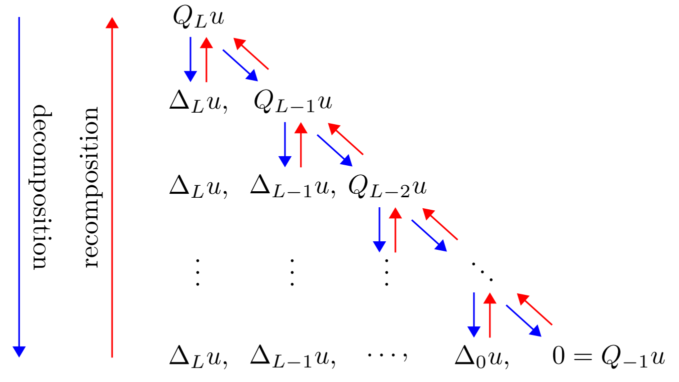
```
function AdaptiveReductionDimensionConstraint(multilevel
coefficients u_mc, number of degrees of freedom $N$)
    ũ_mc ← []
    $\mathcal{N} \leftarrow \{\}$
    for $x \in \mathcal{N}_L$ do
        if $|$u_mc[$x$]$|$ is among the $N$ greatest of u_mc then
            ũ_mc[$x$] ← u_mc[$x$]
            $\mathcal{N} \leftarrow \mathcal{N} \cup \{x\}$
        else
            ũ_mc[$x$] ← 0
        end if
    end for
    return ũ_mc, $\mathcal{N}$
end function
```

4. Implementation.

The transformation of u to and from the multilevel coefficients is straightforward mathematically, but care must be taken in practice so as not to increase the computational complexity or the memory requirements of the overall reduction process. In this section, which primarily concerns the tensor product case, we present recursive implementations for the decomposition and recomposition transformations requiring $O(\#\mathcal{N}_L)$ operations and a temporary storage buffer of size $O(\#\mathcal{N}_L)$. Figure 2 outlines the algorithms, which mirror one another. Decomposition requires a procedure to transform $Q_\ell u$ to $\Delta_\ell u$ and $Q_{\ell-1}u$, while recomposition requires a procedure to combine $\Delta_\ell u$ and $Q_{\ell-1}u$ into $Q_\ell u$.

4.1. Decomposition.

As illustrated in Figure 2, the decomposition of an input function u can be achieved by repeatedly splitting $Q_\ell u$ into $\Delta_\ell u$ and $Q_{\ell-1}u$. The first term is sent to storage, while the second is retained for further decomposition. We aim to represent $\Delta_\ell u$ and $Q_{\ell-1}u$ without requiring more than the $\#\mathcal{N}_\ell$ degrees



decomposition
recomposition
$Q_L u$
$\Delta_L u,\ Q_{L-1}u$
$\Delta_L u,\ \Delta_{L-1}u,\ Q_{L-2}u$
$\Delta_L u,\ \Delta_{L-1}u,\ \dots,\ \Delta_0 u,\ 0 = Q_{-1}u$

FIG. 2. *Illustration of the steps in the recursive decomposition and recomposition procedures.*

of freedom originally needed to represent $Q_\ell u$. The key idea is based on the following decomposition:

$$Q_\ell u = (I - \Pi_{\ell-1})Q_\ell u + Q_{\ell-1}u - \underbrace{(Q_{\ell-1}u - \Pi_{\ell-1}Q_\ell u)}_{z_{\ell-1}}. \tag{9}$$

Consider how each of the three right-hand terms can be represented. As discussed in subsection 2.1, the first term, $(I - \Pi_{\ell-1})Q_\ell u = \Delta_\ell u$, vanishes on $\mathcal{N}_{\ell-1}$ and can therefore be represented by the values it takes on $\mathcal{N}_\ell^*$. Conversely, the second term, $Q_{\ell-1}u$, is contained in $V_{\ell-1}$ and so can be represented by the values it takes on $\mathcal{N}_{\ell-1}$. Its values on $\mathcal{N}_\ell^*$ can be reconstructed through interpolation and as such do not need to be stored. The third term, $z_{\ell-1}$, need not be stored explicitly but can be reconstructed by projecting the first term onto $V_{\ell-1}$:

$$Q_{\ell-1}(I - \Pi_{\ell-1})Q_\ell u = Q_{\ell-1}IQ_\ell u - Q_{\ell-1}\Pi_{\ell-1}Q_\ell u = Q_{\ell-1}u - \Pi_{\ell-1}Q_\ell u = z_{\ell-1}.$$

Hence, $z_{\ell-1}$ is the solution to the variational problem

$$\text{find } z_{\ell-1} \in V_{\ell-1} \ :\ (z_{\ell-1}, v_{\ell-1}) = ((I - \Pi_{\ell-1})Q_\ell u, v_{\ell-1}) \text{ for all } v_{\ell-1} \in V_{\ell-1}, \tag{10}$$

and thus it can be generated from the data $(I - \Pi_{\ell-1})Q_\ell u$ (see subsection 4.3).

How can the decomposition in (9) be performed efficiently? The first term, $(I - \Pi_{\ell-1})Q_\ell u$, is constructed directly from $Q_\ell u$ by subtracting the interpolant $\Pi_{\ell-1}Q_\ell u$. Algorithmically, this can be accomplished by simply adjusting the values stored on $\mathcal{N}_\ell^*$. Following this step, $Q_\ell u$ has been decomposed as follows.

$$Q_\ell u = \underbrace{(I - \Pi_{\ell-1})Q_\ell u}_{\text{on } \mathcal{N}_\ell^*} + \underbrace{\Pi_{\ell-1}Q_\ell u}_{\text{on } \mathcal{N}_{\ell-1}}$$

The next step is to replace $\Pi_{\ell-1}Q_\ell u$ with $Q_{\ell-1}u$. To this end, observe that

$$Q_{\ell-1}u = \Pi_{\ell-1}Q_\ell u + (Q_{\ell-1} - \Pi_{\ell-1}Q_\ell)u = \Pi_{\ell-1}Q_\ell u + z_{\ell-1}.$$

Both $\Pi_{\ell-1}Q_\ell u$ and $z_{\ell-1}$ are contained in $V_{\ell-1}$ and hence are determined by their values on $\mathcal{N}_{\ell-1}$. Therefore, we can compute $Q_{\ell-1}u$ from $\Pi_{\ell-1}Q_\ell u$ and $z_{\ell-1}$ by adding their values on $\mathcal{N}_{\ell-1}$. In particular, the data stored on $\mathcal{N}_\ell^*$ are left untouched.

In summary, $Q_\ell u$ can be transformed to $(I-\Pi_{\ell-1})Q_\ell u$ and $Q_{\ell-1}u$ by the following procedure:

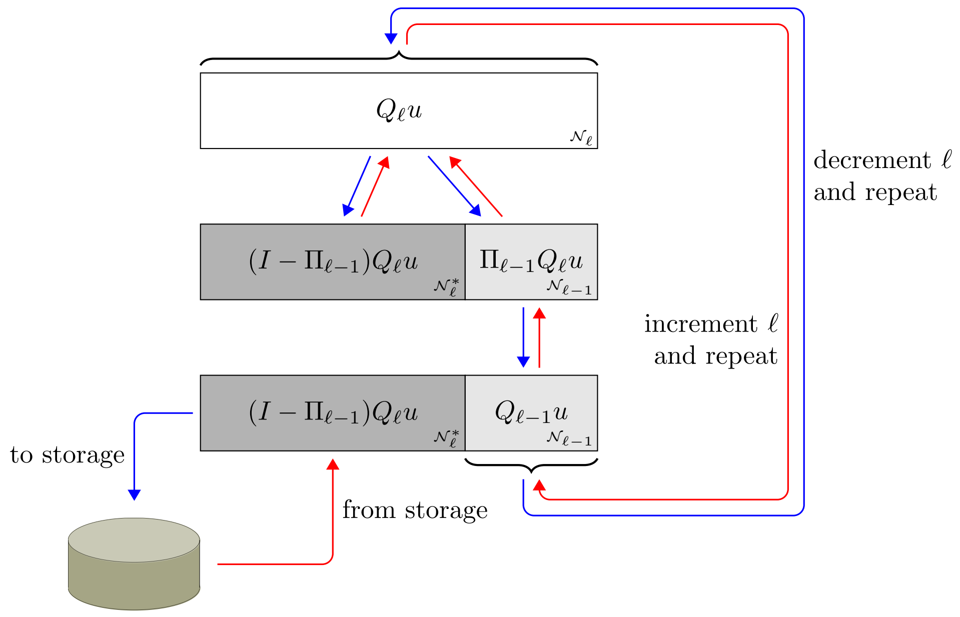


FIG. 3. *Illustration of recursive procedures for the decomposition into multilevel coefficients and the recomposition back to nodal coefficients.*

1. Modify the nodal values of $Q_\ell u$ on $\mathcal{N}_\ell^*$ to become those of $(I-\Pi_{\ell-1})Q_\ell u$.
2. Compute $z_{\ell-1}$ by solving (10).
3. Add $z_{\ell-1}$ to $\Pi_{\ell-1}Q_\ell u$ to obtain $Q_{\ell-1}u$ on $\mathcal{N}_{\ell-1}$.

After these steps, u has been transformed to

$$\underbrace{(I-\Pi_{L-1})Q_L u}_{\text{on } \mathcal{N}_L^*}, \dots, \underbrace{(I-\Pi_{\ell-1})Q_\ell u}_{\text{on } \mathcal{N}_\ell^*}, \underbrace{Q_{\ell-1}u}_{\text{on } \mathcal{N}_{\ell-1}}.$$

We can complete the decomposition by proceeding recursively, as illustrated in Figure 3.

4.2. Recomposition. The central step in the recomposition procedure is the compilation of $Q_\ell u$ from $(I-\Pi_{\ell-1})Q_\ell u$ and $Q_{\ell-1}u$. As shown in Figure 3, this can be accomplished by the reverse of the procedure used in the preceding subsection to split $Q_\ell u$ into $(I-\Pi_{\ell-1})Q_\ell u$ and $Q_{\ell-1}u$. We can recompute $Q_\ell u$ using (9):

$$Q_\ell u = (I-\Pi_{\ell-1})Q_\ell u + Q_{\ell-1}u - z_{\ell-1}. \tag{11}$$

Algorithmically, the sum in (11) involves the following steps:

1. Compute $z_{\ell-1}$ and then subtract from $Q_{\ell-1}u$. This leaves the nodal values of $(I-\Pi_{\ell-1})Q_\ell u$ on $\mathcal{N}_\ell^*$ and the nodal values of $Q_{\ell-1}u - z_{\ell-1} = \Pi_{\ell-1}Q_\ell u$ on $\mathcal{N}_{\ell-1}$.
2. Prolongate $\Pi_{\ell-1}Q_\ell u$ from $V_{\ell-1}$ to V_ℓ and add to $(I-\Pi_{\ell-1})Q_\ell u$ to obtain $Q_\ell u$.

These steps yield the partial recomposition

$$\underbrace{Q_\ell u}_{\text{on } \mathcal{N}_\ell}, \underbrace{(I-\Pi_\ell)Q_{\ell+1}}_{\text{on } \mathcal{N}_{\ell+1}^*}, \dots, \underbrace{(I-\Pi_{L-1})Q_L}_{\text{on } \mathcal{N}_L^*}.$$

Proceeding recursively, as illustrated in Figure 3, we can recompute the nodal values of u on $\mathcal{N}_L$ given the corrections $\{z_{\ell-1} : 0 < \ell \leq L\}$, which are computed as described in the following subsection.

4.3. Computation of the correction. The correction $z_{\ell-1}$ is obtained by solving (10). We can reformulate that variational problem as a system of linear equations by taking $v_{\ell-1}$ to be each of the coarse grid nodal basis functions $\{\phi_{\ell-1}(\,\cdot\,; x) : x \in \mathcal{N}_{\ell-1}\}$ in turn. The result is a matrix equation $M_{\ell-1}\vec{\mathtt{z}}_{\mathtt{l-1}} = \vec{\mathtt{f}}_{\mathtt{l-1}}$, where $M_{\ell-1}$ is the mass matrix on $V_{\ell-1}$ with respect to the nodal basis, $\vec{\mathtt{z}}_{\mathtt{l-1}}$ is the vector of coefficients for $z_{\ell-1}$ with respect to the same basis, and $\vec{\mathtt{f}}_{\mathtt{l-1}}$ is the load vector. Two issues need to be tackled: the inversion of the mass matrix and the computation of $\vec{\mathtt{f}}_{\mathtt{l-1}}$.

The mass matrix has a special structure in the tensor product case. By the construction of the function spaces, $V_{\ell-1}$ is given by the tensor product $V_{\ell-1}^1 \otimes \cdots \otimes V_{\ell-1}^d$, where $V_{\ell-1}^i$ is the space of continuous piecewise linear functions on $[a^i, b^i]$ with respect to the partition $\mathcal{P}_{\ell-1}^i$. Consequently, $M_{\ell-1} = M_{\ell-1}^1 \otimes \cdots \otimes M_{\ell-1}^d$, where $M_{\ell-1}^i$ is the mass matrix for $V_{\ell-1}^i$. The inverse of $M_{\ell-1}$ is thus given by $M_{\ell-1}^{-1} = (M_{\ell-1}^1)^{-1} \otimes \cdots \otimes (M_{\ell-1}^d)^{-1}$. Each $M_{\ell-1}^i$ is a tridiagonal symmetric positive definite matrix, so $\vec{\mathtt{z}}_{\mathtt{l-1}}$ can be determined using $O(\#\mathcal{N}_{\ell-1}^1) \times \cdots \times O(\#\mathcal{N}_{\ell-1}^d) = O(\#\mathcal{N}_{\ell-1})$ operations and a workspace of size $O(\max \#\mathcal{N}_{\ell-1}^i)$ [15, Algorithm 4.3.6]. This procedure overwrites $\vec{\mathtt{f}}_{\mathtt{l-1}}$ with $\vec{\mathtt{z}}_{\mathtt{l-1}}$, so no additional buffer is required for the storage of the solution.

The entries of the load vector are given by $\vec{\mathtt{f}}_{\mathtt{l-1}}\mathtt{[}x\mathtt{]} = ((I - \Pi_{\ell-1})Q_\ell u, \phi_{\ell-1}(\,\cdot\,; x))$ for $x \in \mathcal{N}_{\ell-1}$. That is, $\vec{\mathtt{f}}_{\mathtt{l-1}} = ((I - \Pi_{\ell-1})Q_\ell u, \vec{\phi}_{\ell-1})$. As $V_{\ell-1}$ is a subspace of V_ℓ, there exists a transfer matrix R_ℓ such that $\vec{\phi}_{\ell-1} = R_\ell \vec{\phi}_\ell$. Let $\vec{\alpha}$ be the vector of nodal values of $(I - \Pi_{\ell-1})Q_\ell u$. Then

$$\begin{aligned} \vec{\mathtt{f}}_{\mathtt{l-1}} &= ((I - \Pi_{\ell-1})Q_\ell u, \vec{\phi}_{\ell-1}) = ((I - \Pi_{\ell-1})Q_\ell u, R_\ell \vec{\phi}_\ell) \\ &= R_\ell((I - \Pi_{\ell-1})Q_\ell u, \vec{\phi}_\ell) = R_\ell M_\ell \vec{\alpha} \end{aligned}$$

because of the linearity of the inner product. Due to the tensor product structure of the function spaces, R_ℓ is given by the tensor product of the univariate transfer matrices $\{R_\ell^i : 1 \leq i \leq d\}$. As R_ℓ^i requires $O(\#\mathcal{N}_\ell^i)$ operations to apply, the matrix-vector multiplication of R_ℓ and $M_\ell \vec{\alpha}$ can be carried out in $O(\#\mathcal{N}_\ell)$ operations. A buffer of size $\#\mathcal{N}_\ell$ is required to store the intermediate vector $M_\ell \vec{\alpha}$.

4.4. Summary. The full decomposition and recomposition procedures are summarized in Algorithms 3 and 4. The algorithms are of optimal complexity and require little by way of additional storage. The only step requiring a significant buffer is the computation of the correction $\vec{\mathtt{z}}_{\mathtt{l-1}}$. The workspace needed can be allocated once and then used as a temporary buffer elsewhere in the implementation. In summary, we have shown the following.

THEOREM 4. *Let V_L be the space of continuous piecewise multilinear functions on a uniform tensor product grid in d dimensions. The decomposition of a function $u \in V_L$ into multilevel coefficients $\{\Delta_\ell u : 0 \leq \ell \leq L\}$ requires the storage of $\#\mathcal{N}_L$ degrees of freedom and can be computed in $O(\#\mathcal{N}_L)$ operations. The recomposition can be likewise computed in $O(\#\mathcal{N}_L)$ operations.*

Proof. The right-hand side $\vec{\mathtt{f}}_{\mathtt{l-1}}$ can be computed by performing matrix-vector multiplications with M_ℓ and R_ℓ at a cost of $O(\#\mathcal{N}_\ell)$ operations. The cost of solving the resulting system of equations is $O(\#\mathcal{N}_{\ell-1})$, and hence $z_{\ell-1}$ can be computed at a cost of $O(\#\mathcal{N}_\ell)$. The full decomposition requires the construction of $\{z_{\ell-1} : 0 < \ell \leq L\}$, costing $O(\#\mathcal{N}_1) + \cdots + O(\#\mathcal{N}_L)$. As $\#\mathcal{N}_\ell \simeq 2^{\ell d}$, the combined cost is $O(\#\mathcal{N}_L)$. Equally well, the reconstruction of u from the multilevel coefficients also requires the computation of $\{z_{\ell-1} : 0 < \ell \leq L\}$ at a cost of $O(\#\mathcal{N}_L)$ operations. □

Algorithm 3. Transformation from nodal to multilevel coefficients.

```
function DECOMPOSE(nodal coefficients &coeffs, maximum level $L$)
    for $\ell = L, \ldots, 0$ do                      //coeffs[$\mathcal{N}_\ell$] now represents $Q_\ell u$.
        for $x \in \mathcal{N}_\ell^*$ do
            compute $\Pi_{\ell-1} Q_\ell u(x)$
            coeffs[$x$] $\leftarrow$ coeffs[$x$] $- \Pi_{\ell-1} Q_\ell u(x)$
        end for                       //coeffs[$\mathcal{N}_\ell^*$] now represents $(I - \Pi_{\ell-1}) Q_\ell u$.
                                      //coeffs[$\mathcal{N}_{\ell-1}$] now represents $\Pi_{\ell-1} Q_\ell u$.
        $\vec{\texttt{f}}_{\texttt{l-1}} \leftarrow R_\ell M_\ell$coeffs[$\mathcal{N}_\ell^*$]
        $\vec{\texttt{z}}_{\texttt{l-1}} \leftarrow M_{\ell-1}^{-1} \vec{\texttt{f}}_{\texttt{l-1}}$                     //$\vec{\texttt{z}}_{\texttt{l-1}}$ now represents $z_{\ell-1}$.
        for $x \in \mathcal{N}_{\ell-1}$ do
            coeffs[$x$] $\leftarrow$ coeffs[$x$] $+ \vec{\texttt{z}}_{\texttt{l-1}}$[$x$]
        end for                           //coeffs[$\mathcal{N}_{\ell-1}$] now represents $Q_{\ell-1} u$.
    end for
    return coeffs
end function
```

Algorithm 4. Transformation from multilevel to nodal coefficients.

```
function RECOMPOSE(multilevel coefficients &coeffs, maximum level $L$)
    for $\ell = 0, \ldots, L$ do           //coeffs[$\mathcal{N}_{\ell-1}$] now represents $Q_{\ell-1} u$.
                                      //coeffs[$\mathcal{N}_\ell^*$] now represents $(I - \Pi_{\ell-1}) Q_\ell u$.
        $\vec{\texttt{f}}_{\texttt{l-1}} \leftarrow R_\ell M_\ell$coeffs[$\mathcal{N}_\ell^*$]
        $\vec{\texttt{z}}_{\texttt{l-1}} \leftarrow M_{\ell-1}^{-1} \vec{\texttt{f}}_{\texttt{l-1}}$                     //$\vec{\texttt{z}}_{\texttt{l-1}}$ now represents $z_{\ell-1}$.
        for $x \in \mathcal{N}_{\ell-1}$ do
            coeffs[$x$] $\leftarrow$ coeffs[$x$] $- \vec{\texttt{z}}_{\texttt{l-1}}$[$x$]
        end for                         //coeffs[$\mathcal{N}_{\ell-1}$] now represents $\Pi_{\ell-1} Q_\ell u$.
        for $x \in \mathcal{N}_\ell^*$ do
            compute $\Pi_{\ell-1} Q_\ell u(x)$
            coeffs[$x$] $\leftarrow$ coeffs[$x$] $+ \Pi_{\ell-1} Q_\ell u(x)$
        end for                              //coeffs[$\mathcal{N}_\ell$] now represents $Q_\ell u$.
    end for
    return coeffs
end function
```

5. Application examples. The pair of reduction algorithms developed in section 3 together constitute a MultiGrid Adaptive Reduction of Data algorithm, which we call MGARD. We now apply MGARD to three representative examples. In subsection 5.1, we apply adaptive reduction in both space and time to the solution of a coupled system of nonlinear PDEs on a uniform spatial mesh with uniform timestepping; in subsection 5.2, we demonstrate the flexibility of our approach by applying it to the reduction of data with nonuniform spatial grids; and in subsection 5.3, we compare the performance of MGARD with existing state-of-the-art reduction algorithms in the context of meteorological data.

5.1. Brusselator dataset. We now illustrate the performance of Algorithms 1 and 2 by applying them to the output of a Brusselator simulation. The Brusselator is a model of a chemical oscillator [31]. Two species U and V interact according to

$$\begin{aligned} U_t &= B + U^2V - (A+1)U + \alpha\nabla^2 U \\ V_t &= AU - U^2V + \alpha\nabla^2 V. \end{aligned}$$

A predictor-corrector scheme adapted from [17] is used to solve this system on a 65×65 grid for 1024 timesteps. The parameter values used are $A = 3$, $B = 1$, and $\alpha = 10^{-5}$, and the initial condition is random. For problems such as this one where there are a large number of solution snapshots available, one often employs a reduced basis approach [18, 29], which can, provided sufficiently many time snapshots are used to compute the reduced basis, achieve good compression. Here, we shall use this dataset to illustrate the applicability of our methodology to timeseries reduction. Figure 4 shows the concentration of U, the first reactant, at timestep 200 along with the reduced representations obtained by applying Algorithm 2 with $N \in \{50, 250, 1000\}$. Figure 5 illustrates the multilevel components used in carrying out the reductions.

Algorithm 2 is applied to every timestep in the solution with $N \in \{50, 250, 1000\}$. The resulting reduced representation of U uses N degrees of freedom per timestep. In this subsection, we give each compression ratio as the ratio of the number of degrees of freedom in the original dataset (65×65 per timestep) to the number of nonzero multilevel coefficients in the reduced representation (with Algorithm 2, N per

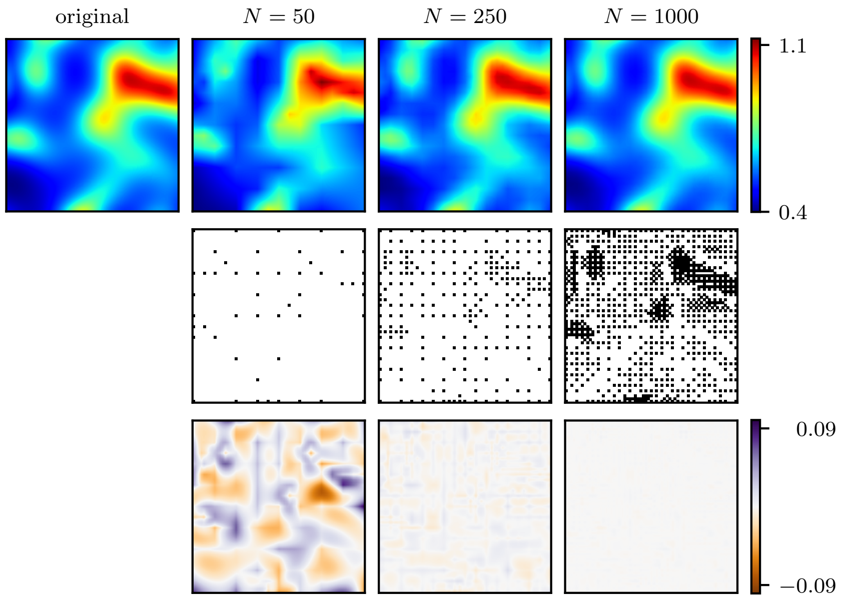


Fig. 4. *Illustration of the effect of applying Algorithm 2 to the concentration of U at timestep 200. The first row shows the original dataset u alongside the reduced representations $\tilde{u}$ obtained using Algorithm 2 with $N \in \{50, 250, 1000\}$. The third row shows the losses $u - \tilde{u}$ incurred in each case. The second row depicts the index sets $\mathcal{N}$ used to generate the reduced representations. Each node $x \in \mathcal{N}_L$ is colored black if the corresponding multilevel coefficient was retained ($x \in \mathcal{N}$) and white if it was discarded ($x \notin \mathcal{N}$). The adaptivity of the algorithm is on view: degrees of freedom are preferentially allocated to the regions of the domain where the solution is least regular.*

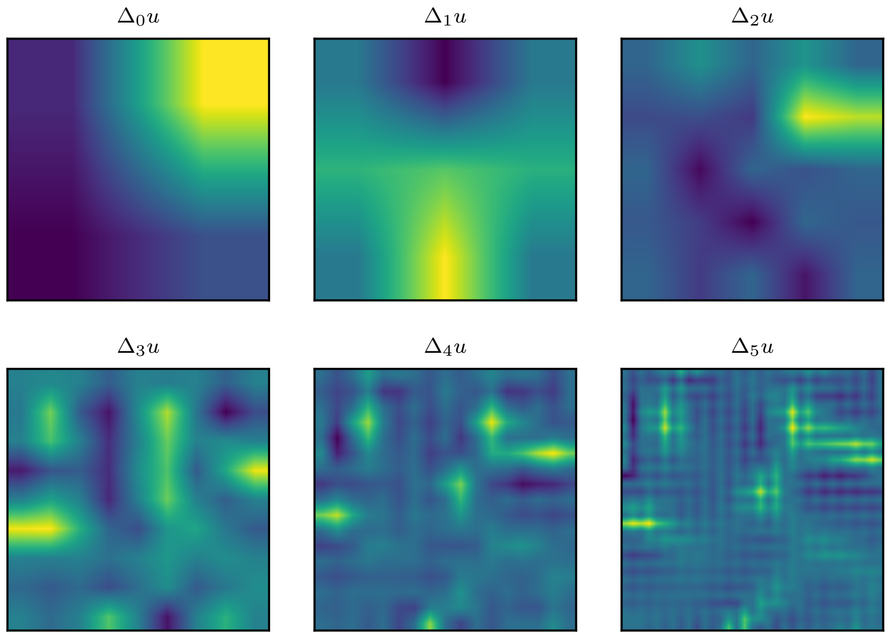


FIG. 5. *Illustration of the first six multilevel components of the concentration of U at timestep 200. Observe that the components contain progressively finer features as one moves from $\Delta_0 u$ to $\Delta_5 u$. The colormap is reset for each component. This is done for ease of visualization: the components on the finer scales have such small magnitude that they would be hardly discernible if plotted using the same colormap as $\Delta_0 u$.*

timestep). As a result, the compression ratio achieved is constant across timesteps for a given N (84.50 for $N = 50$, 16.90 for $N = 250$, and 4.23 for $N = 1000$). The loss, though, depends on the character of the solution at a given timestep. When the reaction is quiescent, little loss is incurred; when the concentrations are in a state of flux, a greater loss is experienced. This phenomenon is illustrated in Figure 6.

For some applications, this variable loss may be unacceptable. As mentioned above, many users will only use lossy algorithms if control of the loss can be guaranteed. For these use cases, Algorithm 1 is more appropriate. Figure 7 illustrates the loss caused by applying Algorithm 1 with tolerance $\tau = 0.1\%$ to each timestep of the solution.

Thus far, Algorithms 1 and 2 have been applied to the two-dimensional timeslabs of the solution one by one. Typically, though, a timestepping application will not send each timestep to disc separately. Instead, the output will be loaded into a cache or burst buffer, which, when full, is dumped to disc all at once. Given that multiple timesteps will be stored together in memory before writing, one might ask whether applying the reduction algorithms in time as well as space could improve their performance by taking advantage of possible redundancies between timeslabs. Figure 8, which compares the compression ratios achieved by Algorithm 1 with and without reduction in time, confirms that the algorithm can indeed take advantage of redundancies in the data in time, resulting in better compression ratios for a given tolerance. Figure 9, analogous to Figure 6, illustrates the compression ratios achieved when applying the reduction in space and time and varying the tolerance τ used.

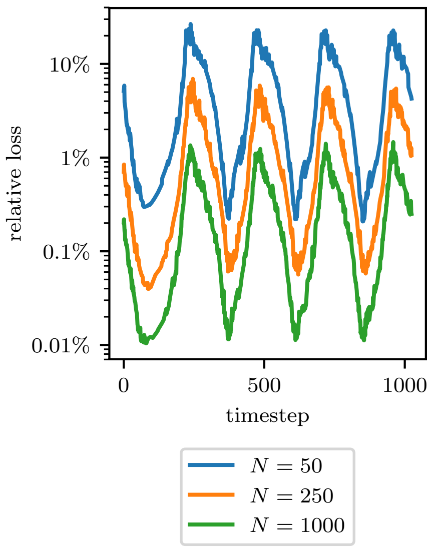


FIG. 6. *Illustration of the loss incurred by Algorithm 2 over time. The loss experienced varies according to the regularity of the solution over time, with the same pattern observed for different storage constraints N.*

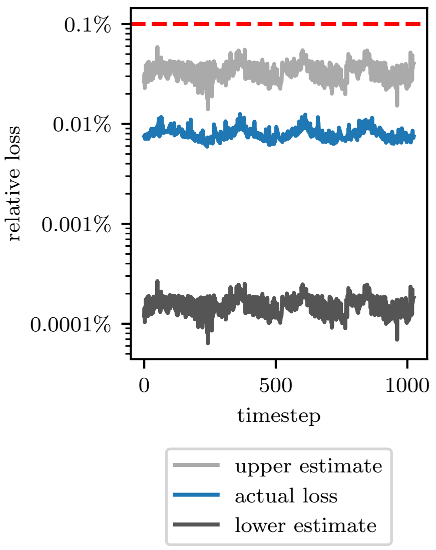


FIG. 7. *Illustration of the loss incurred by Algorithm 1 with $\tau = 0.1\%$ and the bounds of Theorem 1. The compression ratio achieved is 1.07. The actual loss $\|u - \tilde{u}\|_{L^\infty}$ is controlled by the loss indicator $\eta_\infty(u - \tilde{u})$, as desired.*

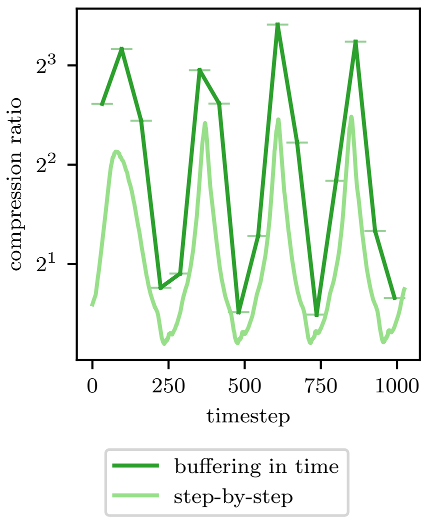


FIG. 8. *Illustration of the results obtained when applying Algorithm 1 (with $\tau = 1\%$ fixed) in time as well as space. The loss tolerance is always met, with better results achieved when the reduction is applied to more timesteps at once. Applying the reduction step-by-step yields a compression ratio of* 1.84. *When buffering in time, the ratio improves to* 2.91.

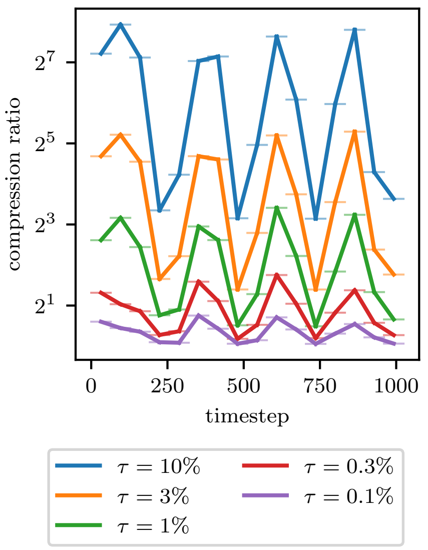


FIG. 9. *Illustration of the compression ratio achieved by Algorithm 1 over time. As with the loss in Figure 6, the compression ratio varies according to the regularity of the solution over time, with the same pattern observed for different tolerances τ. The compression ratios obtained are* 25.68 *with $\tau = 10\%$,* 6.78 *with $\tau = 3\%$,* 2.91 *with $\tau = 1\%$,* 1.66 *with $\tau = 0.3\%$, and* 1.22 *with $\tau = 0.1\%$.*

5.2. High Reynolds number channel flow. In this subsection we consider data [28, 22, 20] obtained by a direct numerical simulation [21] of channel flow at Reynolds number $\mathrm{Re}_\tau = 9.9935 \times 10^2$: one of the canonical examples of turbulent flow. The computational domain $L_x \times L_y \times L_z = 8\pi \times 2 \times 3\pi$ (dimensionless units) is discretized using a mesh of size $N_x \times N_y \times N_z = 2048 \times 512 \times 1536$ nodes. Each of the 4000 timeslices consists of the three velocity components at each of the 1.6 billion nodes, corresponding to a dataset of size 18.3 GB per timeslice.

We reduce the data on a single timeslice, treating each component of the velocity across the 1536 two-dimensional z-slices (xy planes) independently. Although the mesh is equispaced in the x and z directions, a Chebyshev grid is employed in the wall-normal direction (along the y axis). The nonuniform grid spacing can pose difficulties for many existing compression techniques, such as wavelets. However, `MGARD` is able to accommodate arbitrary spacings (see section 3). The overall reductions and achieved losses obtained by `MGARD`, `SZ` [13],[2] and `ZFP` [23][3] at a range of loss tolerances are reported in Table 1. In this subsection and in subsection 5.3, compression ratios are given as the ratio of the filesize of the original dataset to the filesize of the reduced representation. Figures 10 and 11 display the compression ratios obtained by `MGARD` for each velocity component on each z-slice when using loss tolerances of 1% and 10%, respectively. Figures 12 and 13 show the axial velocity fields of the reduced

Table 1
Compression ratios and achieved losses for reduction of the channel flow data [28, 22, 20] *by `MGARD`, `SZ`, and `ZFP` at various loss tolerances τ. The actual losses shown are the maxima over the three velocity components and are given to two significant figures.*

	Compression ratio			Actual loss (%)		
Target τ (%)	`MGARD`	`SZ`	`ZFP`	`MGARD`	`SZ`	`ZFP`
1	7.67	12.73	3.90	0.98	1.0	0.26
5	17.68	27.95	5.63	4.9	5.0	1.8
10	29.45	66.50	7.71	9.8	10	4.0
20	55.86	100.36	8.08	19	20	6.7

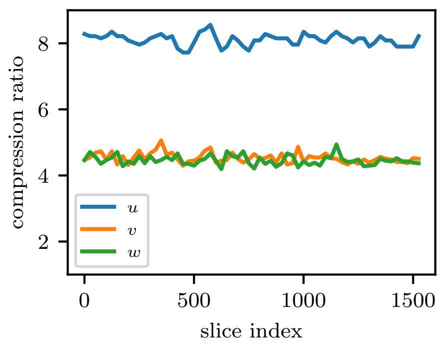


Fig. 10. *Compression ratio per z-slice for each velocity component with a loss tolerance of* 1%.

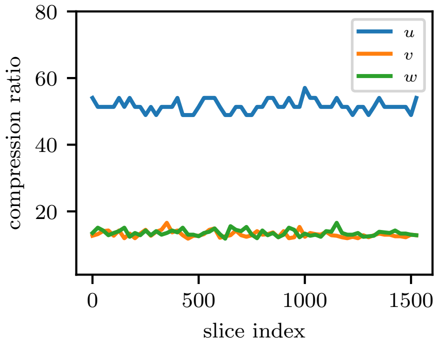


Fig. 11. *Compression ratio per z-slice for each velocity component with a loss tolerance of* 10%.

[2]The results in Tables 1 and 2 were obtained using `SZ` version 1.4.9.3 with parameters `SZ_DEFAULT_COMPRESSION`, `Gzip_DEFAULT_COMPRESSION`, and `max_quant_intervals = 65536`.
[3]The results in Tables 1 and 2 were obtained using `ZFP` version 0.5.1 with parameter `-a`.

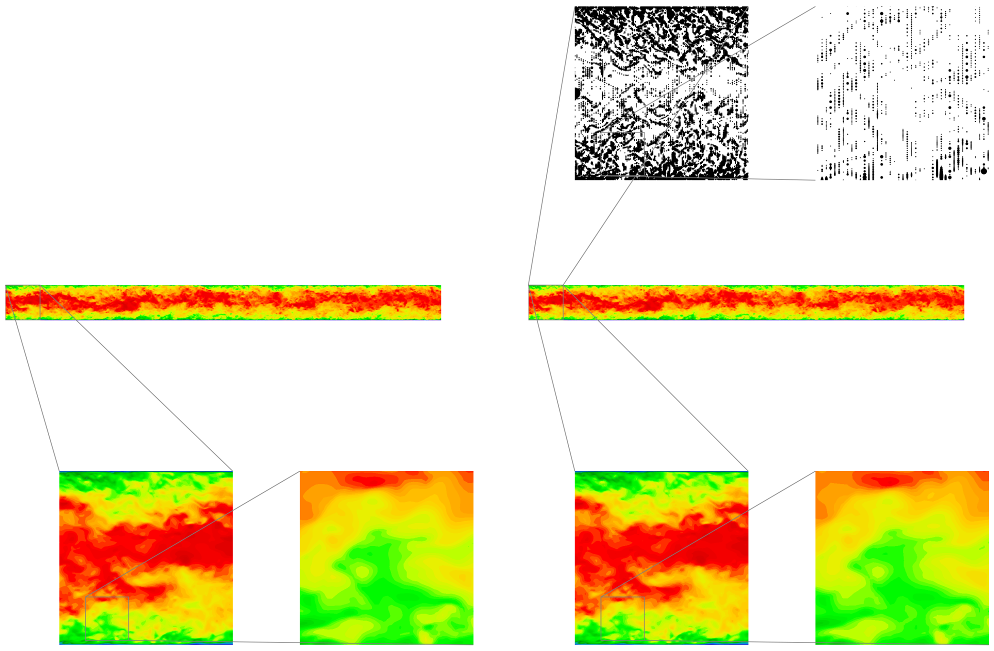


FIG. 12. *Contours of the axial velocity at the channel center. The middle row displays contours along the full length of the channel in the true (left) and reduced (right, using 1% tolerance) datasets. The bottom row displays fourfold zooms of selected square regions of the flows. The top row displays the locations of the retained coefficients.*

and original data. We repeatedly zoom a selected region of the domain fourfold to highlight the effects of reduction on progressively smaller scales of the flow. Scatter plots in the top right illustrate which of the multilevel coefficients are retained in the reduced representations.

We take this opportunity to illustrate the comparative behavior of the hierarchical and orthogonal decompositions. In Figure 14 we present partial recompositions of the axial velocity on the center z-slice obtained by truncating the orthogonal and hierarchical decompositions of the dataset. Figure 15 compares the magnitudes of the reduction coefficients obtained using the orthogonal and hierarchical decompositions. In particular, the more uniform decay of the coefficients in the orthogonal decomposition, observed in Figure 15, means that the orthogonal decomposition is more suitable for data reduction.

Finally, we consider the effect of reducing the data on the accuracy of some typical quantities of interest in turbulence simulations. To this end we decompose each component of the velocity field into the mean flow and fluctuations around the mean: $u_i = \overline{u}_i + u_i'$. Figures 16 to 18 show the mean axial velocity $\overline{u}_i$, the variance of the turbulent velocity fluctuations $\langle u_i' u_i' \rangle_{xz}$, and the Reynolds stress $\langle u_1' u_2' \rangle_{xz}$, respectively, versus the wall distance $y^+ = (1+y)\mathrm{Re}_\tau$. In all cases, we observe that reduction with a loss tolerance of 2% results in a negligible degradation in the quality of the quantity of interest. For example, the position and the maximum amplitude of the variances and the Reynolds stress is in good agreement with the location and values in the original.

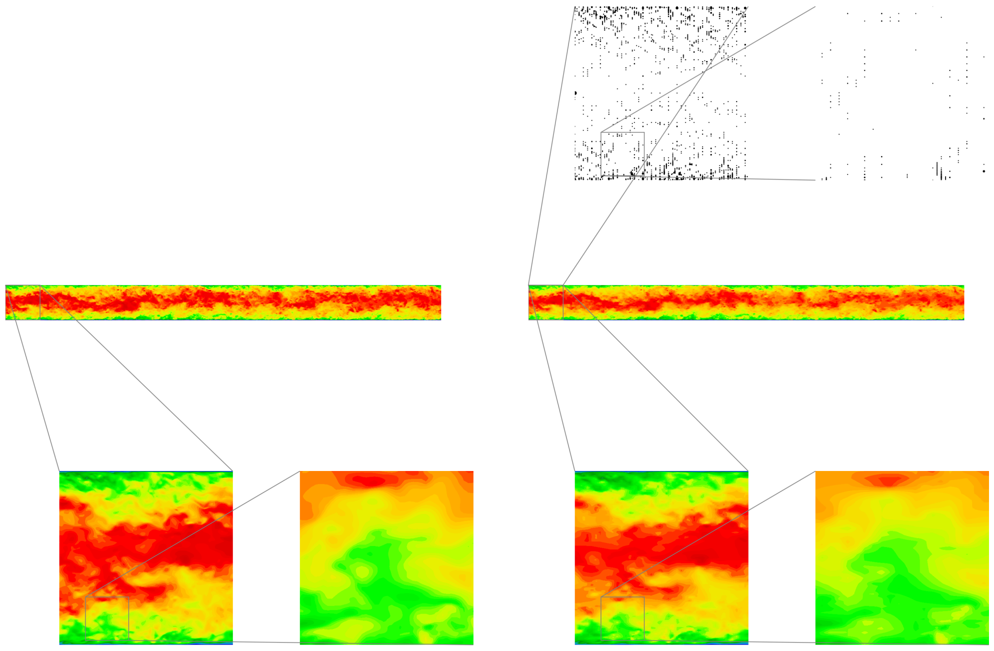


FIG. 13. *Contours of the axial velocity at the channel center. The middle row displays contours along the full length of the channel in the true (left) and reduced (right, using* 10% *tolerance) datasets. The bottom row displays fourfold zooms of selected square regions of the flows. The top row displays the locations of the retained coefficients.*

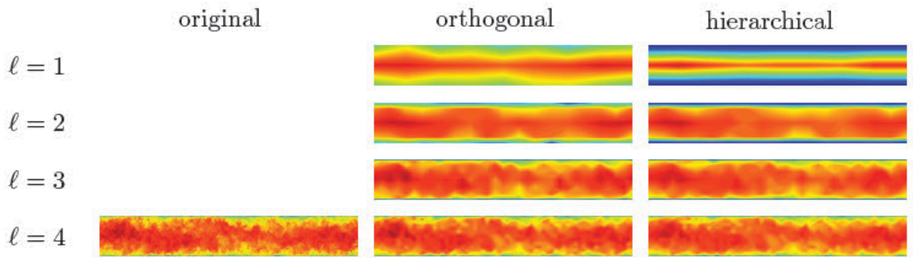


FIG. 14. *Illustration of recompositions obtained using the orthogonal decomposition* (5) *(shown in the center column) and the hierarchical decomposition* (4) *(shown in the right column) with contributions from the first* $\ell + 1$ *terms of each decomposition. Observe that the orthogonal decomposition performs better than the hierarchical decomposition, particularly at high compression. The original velocity field is shown in the bottom left. The errors are* 61%, 47%, 57%, *and* 38% *in the orthogonal case and* 81%, 71%, 66%, *and* 43% *in the hierarchical case.*

5.3. Community Earth System Model atmosphere data. We now apply MGARD to data obtained from the CESM atmosphere model [5]. The dataset, a field of low cloud coverage values called CLDLOW, comprises single-precision floating point data on a uniformly spaced 1800×3600 grid. Figure 19 shows the original data along with decompressed datasets obtained using various specified tolerances with MGARD. Table 2 shows the compression ratios and achieved losses obtained using MGARD, SZ, and ZFP. These preliminary results show that MGARD is competitive with existing state-of-the-art compressors.

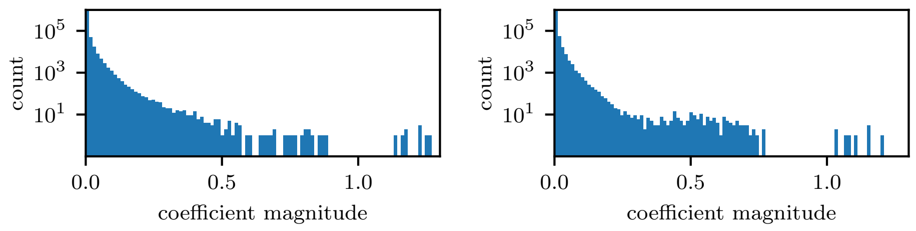


FIG. 15. *Histograms of absolute value of the reduction coefficients using the orthogonal decomposition* (5) *(left) and the hierarchical decomposition* (4) *(right). Observe the more uniform decay of the coefficients using the orthogonal decomposition compared with those using the hierarchical decomposition, corresponding to the superior stability properties of the orthogonal decomposition.*

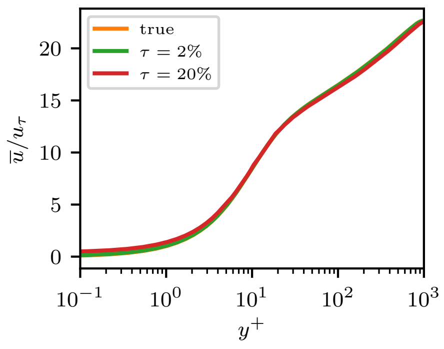


FIG. 16. *Mean axial velocity profile versus the wall coordinate y^+.*

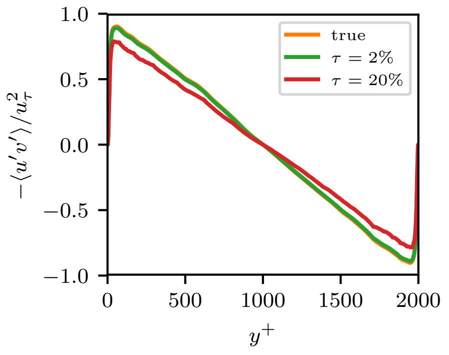


FIG. 17. *Reynolds stress $-\langle u'v' \rangle$ versus the wall coordinate y^+.*

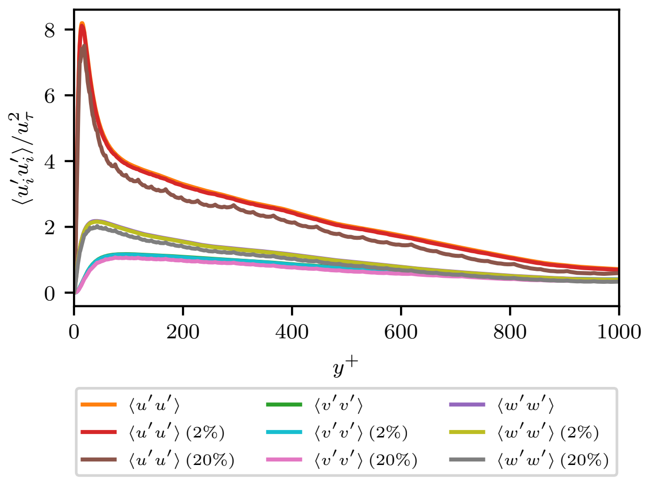


FIG. 18. *Variance of the turbulent velocity fluctuations versus the wall coordinate y^+.*

Appendix A. Estimation of $\|I - Q_{\ell-1}\|_{L^\infty}$.

The proof of Theorem 2 required a uniform bound on the operator norm of $I - Q_{\ell-1}\colon L^\infty(\mathrm{im}(\Pi_\ell - \Pi_{\ell-1})) \to L^\infty(\mathrm{im}(Q_\ell - Q_{\ell-1}))$. The objective of this section is to obtain that bound.

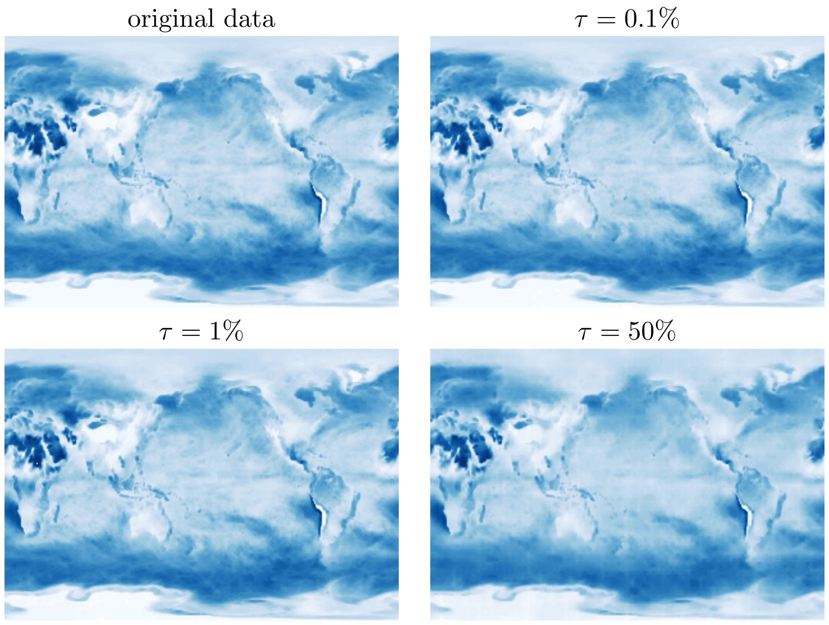


FIG. 19. *Illustration of the effect of compression using* MGARD *on the CLDLOW data* [5] *with indicated tolerances* τ.

Fix $\ell \in \{1, \ldots, L\}$. We begin with some notation. Let $N = \#\mathcal{N}_{\ell-1} - 1$. For $x \in \mathcal{N}_\ell$, let x^{L} and x^{R} denote $\#\{y \in \mathcal{N}_{\ell-1} : y < x\}$ and $\#\{y \in \mathcal{N}_{\ell-1} : y > x\}$, respectively. Set $\lambda_0 = -2 + \sqrt{3}$ and $\lambda_1 = -2 - \sqrt{3}$, and let Λ_0 and Λ_1 denote the absolute values of λ_0 and λ_1, respectively. Define $\cosh_\Lambda, \sinh_\Lambda \colon \mathbb{R} \to \mathbb{R}$ by $\cosh_\Lambda(n) = \frac{1}{2}(\Lambda_1^n + \Lambda_0^n)$ and $\sinh_\Lambda(n) = \frac{1}{2}(\Lambda_1^n - \Lambda_0^n)$. As $\Lambda_0\Lambda_1 = 1$, the addition and subtraction identities for cosh and sinh also hold for $\cosh_\Lambda$ and $\sinh_\Lambda$.

For all $x \in \mathcal{N}_\ell^*$ and $y \in \mathcal{N}_{\ell-1}$, $\phi_\ell(y; x) = 0$. As a result,

$$\max_{y \in \mathcal{N}_{\ell-1}} \sum_{x \in \mathcal{N}_\ell^*} |(I - Q_{\ell-1})\phi_\ell(\,\cdot\,; x)(y)| = \max_{y \in \mathcal{N}_{\ell-1}} \sum_{x \in \mathcal{N}_\ell^*} |Q_{\ell-1}\phi_\ell(\,\cdot\,; x)(y)|.$$

The first step in bounding this maximum is deriving an expression for the values taken by the projections $\{Q_{\ell-1}\phi_\ell(\,\cdot\,; x) : x \in \mathcal{N}_\ell^*\}$ on the nodes $\mathcal{N}_{\ell-1}$. The following result gives such an expression.

LEMMA 5. *For* $x \in \mathcal{N}_\ell^*$ *and* $y \in \mathcal{N}_{\ell-1}$,

$$Q_{\ell-1}\phi_\ell(\,\cdot\,; x)(y) = \frac{3(-1)^{N+1+k_0}}{2\sinh_\Lambda(N)\sinh_\Lambda(1)}(\cosh_\Lambda(k_1) - \cosh_\Lambda(k_1 - 1))\cosh_\Lambda(k_2)$$

$$\textit{where} \quad (k_0, k_1, k_2) = \begin{cases} (x^{\mathrm{R}} + y^{\mathrm{L}}, x^{\mathrm{R}}, y^{\mathrm{L}}) & y < x, \\ (x^{\mathrm{L}} + y^{\mathrm{R}}, x^{\mathrm{L}}, y^{\mathrm{R}}) & x < y. \end{cases}$$

Proof. Fix $x \in \mathcal{N}_\ell^*$. We will denote by $\{a_j : 0 \leq j < x^{\mathrm{L}}\}$ the nodal values $\{Q_{\ell-1}\phi_\ell(\,\cdot\,; x)(y) : y \in \mathcal{N}_{\ell-1}$ and $y < x\}$ with $Q_{\ell-1}\phi_\ell(\,\cdot\,; x)(y) = a_{y^{\mathrm{L}}}$. By definition, $(Q_{\ell-1}\phi_\ell(\,\cdot\,; x), \phi_{\ell-1}(\,\cdot\,; y)) = (\phi_\ell(\,\cdot\,; x), \phi_{\ell-1}(\,\cdot\,; y))$ for all $y \in \mathcal{N}_{\ell-1}$. Let y_- and y_+ denote the nodes of $\mathcal{N}_{\ell-1}$ to the immediate left and right, respectively, of x. Observe that $(\phi_\ell(\,\cdot\,; x), \phi_{\ell-1}(\,\cdot\,; y)) = 0$ if $y \notin \{y_-, y_+\}$. So, the nodal values $\{a_j : 0 \leq j < x^{\mathrm{L}}\}$ satisfy the linear recurrence relation

TABLE 2
Compression ratios and achieved losses for reduction of the CLDLOW data [5] by MGARD, SZ, and ZFP at various loss tolerances τ. The actual losses are given to two significant figures.

	Compression ratio			Actual loss (%)		
Target τ (%)	MGARD	SZ	ZFP	MGARD	SZ	ZFP
0.0001	2.03	2.32	1.76	0.00010	0.00010	0.000043
0.1	6.46	10.05	3.68	0.091	0.10	0.043
1	22.62	28.23	4.38	0.83	1.0	0.35
10	125.28	136.45	6.76	6.5	10	2.4
50	999.23	1030.01	9.75	30	50	7.8

$$\begin{cases} 0 = 2a_j + a_{j+1} & j = 0, \\ 0 = a_{j-1} + 4a_j + a_{j+1} & j \in \{1, \dots, x^{\mathrm{L}} - 2\}. \end{cases}$$

The characteristic polynomial $\lambda^2 + 4\lambda + 1$ has roots $\lambda_0 = -2 + \sqrt{3}$ and $\lambda_1 = -2 - \sqrt{3}$, so the nodal values are given, for some coefficients $\alpha_0, \alpha_1 \in \mathbb{R}$, by $a_j = \alpha_0 \lambda_0^j + \alpha_1 \lambda_1^j$. The boundary condition at $j = 0$ implies that $\alpha_0 = \alpha_1$:

$$\begin{aligned} 0 &= 2\alpha_0 + 2\alpha_1 + \lambda_0 \alpha_0 + \lambda_1 \alpha_1 \\ &= (2 + \lambda_0)\alpha_0 + (2 + \lambda_1)\alpha_1 \\ &= \sqrt{3}(\alpha_0 - \alpha_1). \end{aligned}$$

By the same token, the nodal values $\{Q_{\ell-1}\phi_\ell(\,\cdot\,; x)(y) : y \in \mathcal{N}_{\ell-1} \text{ and } x < y\}$, denoted $\{b_j : 0 \le j < x^{\mathrm{R}}\}$ with $Q_{\ell-1}\phi_\ell(\,\cdot\,; x)(y) = b_{y^{\mathrm{R}}}$, are given by $b_j = \beta_0 \lambda_0^j + \beta_0 \lambda_1^j$ for some coefficient $\beta_0 \in \mathbb{R}$. Define $\cosh_\lambda \colon \{0, \dots, N\} \to \mathbb{R}$ by $\cosh_\lambda(n) = \frac{1}{2}(\lambda_1^n + \lambda_0^n)$. Rescaling and denoting the unknown coefficients α and β, we conclude that $a_j = \alpha \cosh_\lambda(j)$ for $j \in \{0, \dots, x^{\mathrm{L}} - 1\}$ and $b_j = \beta \cosh_\lambda(j)$ for $j \in \{0, \dots, x^{\mathrm{R}} - 1\}$.

The final step is to determine α and β. Let h be the element size on $\mathcal{P}_{\ell-1}$. For $y \in \{y_-, y_+\}$, $(\phi_\ell(\,\cdot\,; x), \phi_{\ell-1}(\,\cdot\,; y)) = \frac{1}{2}(\phi_\ell(\,\cdot\,; x), \phi_{\ell-1}(\,\cdot\,; y_-) + \phi_{\ell-1}(\,\cdot\,; y_+)) = \frac{1}{4}h$, so that $\{a_j : 0 \le j < x^{\mathrm{L}}\}$ and $\{b_j : 0 \le j < x^{\mathrm{R}}\}$ are coupled by the conditions

$$\begin{aligned} \tfrac{1}{4}h &= \tfrac{1}{6}h a_{x^{\mathrm{L}}-2} + \tfrac{2}{3}h a_{x^{\mathrm{L}}-1} + \tfrac{1}{6}h b_{x^{\mathrm{R}}-1}, \\ \tfrac{1}{4}h &= \tfrac{1}{6}h a_{x^{\mathrm{L}}-1} + \tfrac{2}{3}h b_{x^{\mathrm{R}}-1} + \tfrac{1}{6}h b_{x^{\mathrm{R}}-2}. \end{aligned}$$

Substituting in the expressions for the nodal values, the first of these equations can be written $3/2 = \alpha \cosh_\lambda(x^{\mathrm{L}} - 2) + 4\alpha \cosh_\lambda(x^{\mathrm{L}} - 1) + \beta \cosh_\lambda(x^{\mathrm{R}} - 1)$. Observe that $0 = \alpha \cosh_\lambda(x^{\mathrm{L}} - 2) + 4\alpha \cosh_\lambda(x^{\mathrm{L}} - 1) + \alpha \cosh_\lambda(x^{\mathrm{L}})$. Subtracting, we find that $3/2 = \beta \cosh_\lambda(x^{\mathrm{R}} - 1) - \alpha \cosh_\lambda(x^{\mathrm{L}})$. Similarly, the second equation implies that $3/2 = \alpha \cosh_\lambda(x^{\mathrm{L}} - 1) - \beta \cosh_\lambda(x^{\mathrm{R}})$. Thus, α and β solve the linear system

$$\begin{bmatrix} -\cosh_\lambda(x^{\mathrm{L}}) & \cosh_\lambda(x^{\mathrm{R}} - 1) \\ \cosh_\lambda(x^{\mathrm{L}} - 1) & -\cosh_\lambda(x^{\mathrm{R}}) \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 3/2 \\ 3/2 \end{bmatrix}.$$

Denote this matrix M. The determinant of M is given by

$$\begin{aligned} \det(M) &= \cosh_\lambda(x^{\mathrm{L}}) \cosh_\lambda(x^{\mathrm{R}}) - \cosh_\lambda(x^{\mathrm{L}} - 1) \cosh_\lambda(x^{\mathrm{R}} - 1) \\ &= \tfrac{1}{2}\bigl(\cosh_\lambda(x^{\mathrm{L}} + x^{\mathrm{R}}) + \cosh_\lambda(x^{\mathrm{L}} - x^{\mathrm{R}}) \\ &\quad - \cosh_\lambda((x^{\mathrm{L}} - 1) + (x^{\mathrm{R}} - 1)) \\ &\quad - \cosh_\lambda((x^{\mathrm{L}} - 1) - (x^{\mathrm{R}} - 1))\bigr) \\ &= \tfrac{1}{2}\bigl(\cosh_\lambda(N + 1) - \cosh_\lambda(N - 1)\bigr) \end{aligned}$$

using the addition and subtraction identities for $\cosh_\lambda$ (which hold because $\lambda_0\lambda_1 = 1$). The solution of the system is then given by

$$\begin{bmatrix}\alpha\\ \beta\end{bmatrix} = \frac{1}{\det(M)}\begin{bmatrix}-\cosh_\lambda(x^{\mathrm{R}}) & -\cosh_\lambda(x^{\mathrm{R}}-1)\\ -\cosh_\lambda(x^{\mathrm{L}}-1) & -\cosh_\lambda(x^{\mathrm{L}})\end{bmatrix}\begin{bmatrix}3/2\\ 3/2\end{bmatrix}$$
$$= -\frac{3}{\cosh_\lambda(N+1)-\cosh_\lambda(N-1)}\begin{bmatrix}\cosh_\lambda(x^{\mathrm{R}})+\cosh_\lambda(x^{\mathrm{R}}-1)\\ \cosh_\lambda(x^{\mathrm{L}})+\cosh_\lambda(x^{\mathrm{L}}-1)\end{bmatrix}.$$

Thus,

$$Q_{\ell-1}\phi_\ell(\,\cdot\,;x)(y) = -\frac{3\big(\cosh_\lambda(x^{\mathrm{R}})+\cosh_\lambda(x^{\mathrm{R}}-1)\big)}{\cosh_\lambda(N+1)-\cosh_\lambda(N-1)}\cosh_\lambda(y^{\mathrm{L}})$$

if $y < x$, and

$$Q_{\ell-1}\phi_\ell(\,\cdot\,;x)(y) = -\frac{3\big(\cosh_\lambda(x^{\mathrm{L}})+\cosh_\lambda(x^{\mathrm{L}}-1)\big)}{\cosh_\lambda(N+1)-\cosh_\lambda(N-1)}\cosh_\lambda(y^{\mathrm{R}})$$

if $x < y$. We reach the desired expression using the identities

$$\cosh_\lambda(n) = (-1)^n\cosh_\Lambda(n)$$
$$\cosh_\Lambda(N+1)-\cosh_\Lambda(N-1) = 2\sinh_\Lambda(N)\sinh_\Lambda(1). \qquad \square$$

Next we take absolute values and sum over $x \in \mathcal{N}_\ell^*$.

Lemma 6. *For $y \in \mathcal{N}_{\ell-1}$,*

$$\sum_{x\in\mathcal{N}_\ell^*}|Q_{\ell-1}\phi_\ell(\,\cdot\,;x)(y)| = 3\frac{\cosh_\Lambda(N)+\cosh_\Lambda(y^{\mathrm{R}}-y^{\mathrm{L}}) - \cosh_\Lambda(y^{\mathrm{R}})-\cosh_\Lambda(y^{\mathrm{L}})}{2\sinh_\Lambda(N)\sinh_\Lambda(1)}.$$

Proof. Fix a node $y \in \mathcal{N}_{\ell-1}$ at which to evaluate the projections. $\mathcal{N}_\ell^*$ can be partitioned into the nodes to the left and to the right of y. Consider first those nodes to the left. We found in Lemma 5 that

$$|Q_{\ell-1}\phi_\ell(\,\cdot\,;x)(y)| = \frac{3\big(\cosh_\lambda(x^{\mathrm{L}})-\cosh_\lambda(x^{\mathrm{L}}-1)\big)}{2\sinh_\Lambda(N)\sinh_\Lambda(1)}\cosh_\Lambda(y^{\mathrm{R}})$$

if $x < y$. Summing over all such x,

$$\sum_{\substack{x\in\mathcal{N}_\ell^*\\ x<y}}|Q_{\ell-1}\phi_\ell(\,\cdot\,;x)(y)| = \frac{3\cosh_\Lambda(y^{\mathrm{R}})}{2\sinh_\Lambda(N)\sinh_\Lambda(1)}\sum_{x^{\mathrm{L}}=1}^{y^{\mathrm{L}}}\cosh_\Lambda(x^{\mathrm{L}}) - \cosh_\Lambda(x^{\mathrm{L}}-1)$$
$$= \frac{3\cosh_\Lambda(y^{\mathrm{R}})\big(\cosh_\Lambda(y^{\mathrm{L}})-\cosh_\Lambda(0)\big)}{2\sinh_\Lambda(N)\sinh_\Lambda(1)}.$$

Equally well, for the nodes to the right of y we have

$$\sum_{\substack{x\in\mathcal{N}_\ell^*\\ x>y}}|Q_{\ell-1}\phi_\ell(\,\cdot\,;x)(y)| = \frac{3\cosh_\Lambda(y^{\mathrm{L}})}{2\sinh_\Lambda(N)\sinh_\Lambda(1)}\sum_{x^{\mathrm{R}}=1}^{y^{\mathrm{R}}}\cosh_\Lambda(x^{\mathrm{R}}) - \cosh_\Lambda(x^{\mathrm{R}}-1)$$
$$= \frac{3\cosh_\Lambda(y^{\mathrm{L}})\big(\cosh_\Lambda(y^{\mathrm{R}})-\cosh_\Lambda(0)\big)}{2\sinh_\Lambda(N)\sinh_\Lambda(1)}.$$

Summing, we find that

$$\sum_{x\in\mathcal{N}_\ell^*} |Q_{\ell-1}\phi_\ell(\,\cdot\,;x)(y)| = 3\frac{\begin{array}{c}2\cosh_\Lambda(y^{\mathrm{R}})\cosh_\Lambda(y^{\mathrm{L}})\\ -\cosh_\Lambda(y^{\mathrm{R}})-\cosh_\Lambda(y^{\mathrm{L}})\end{array}}{2\sinh_\Lambda(N)\sinh_\Lambda(1)}.$$

The desired result now follows from the identity

$$2\cosh_\Lambda(y^{\mathrm{R}})\cosh_\Lambda(y^{\mathrm{L}}) = \cosh_\Lambda(N) + \cosh_\Lambda(y^{\mathrm{R}}-y^{\mathrm{L}}). \qquad \square$$

Maximizing the expression in Lemma 6 over $y \in \mathcal{N}_{\ell-1}$, we obtain the following bound.

Corollary 7.

$$\max_{y\in\mathcal{N}_{\ell-1}} \sum_{x\in\mathcal{N}_\ell^*} |Q_{\ell-1}\phi_\ell(\,\cdot\,;x)(y)| \le \frac{\sqrt{3}}{2}.$$

Proof. By Lemma 6,

$$\max_{y\in\mathcal{N}_{\ell-1}} \sum_{x\in\mathcal{N}_\ell^*} |Q_{\ell-1}\phi_\ell(\,\cdot\,;x)(y)| = \max_{y\in\mathcal{N}_{\ell-1}} 3\frac{\begin{array}{c}\cosh_\Lambda(N)+\cosh_\Lambda(y^{\mathrm{R}}-y^{\mathrm{L}})\\ -\cosh_\Lambda(y^{\mathrm{R}})-\cosh_\Lambda(y^{\mathrm{L}})\end{array}}{2\sinh_\Lambda(N)\sinh_\Lambda(1)}. \tag{12}$$

Since $y^{\mathrm{L}}, y^{\mathrm{R}} \in \{0,\dots,N\}$, $|\frac{1}{2}(y^{\mathrm{R}}-y^{\mathrm{L}})| \le N/2$. $\cosh_\Lambda(n)$ is increasing in $|n|$, so $\cosh_\Lambda(\frac{1}{2}(y^{\mathrm{R}}-y^{\mathrm{L}})) \le \cosh_\Lambda(N/2)$. As a result,

$$\begin{aligned}2\cosh_\Lambda(\tfrac{1}{2}(y^{\mathrm{R}}-y^{\mathrm{L}}))\cosh_\Lambda(\tfrac{1}{2}(y^{\mathrm{R}}-y^{\mathrm{L}})) &\le 2\cosh_\Lambda(\tfrac{N}{2})\cosh_\Lambda(\tfrac{1}{2}(y^{\mathrm{R}}-y^{\mathrm{L}})),\\ \cosh_\Lambda(y^{\mathrm{R}}-y^{\mathrm{L}})+\cosh_\Lambda(0) &\le \cosh_\Lambda(y^{\mathrm{R}})+\cosh_\Lambda(y^{\mathrm{L}}).\end{aligned}$$

In particular,

$$\cosh_\Lambda(N)+\cosh_\Lambda(y^{\mathrm{R}}-y^{\mathrm{L}})-\cosh_\Lambda(y^{\mathrm{R}})-\cosh_\Lambda(y^{\mathrm{L}}) \le \cosh_\Lambda(N)-\cosh_\Lambda(0).$$

Observe that this inequality becomes an equality when $y^{\mathrm{L}} \in \{0, N\}$. Thus,

$$\max_{y\in\mathcal{N}_{\ell-1}} \sum_{x\in\mathcal{N}_\ell^*} |Q_{\ell-1}\phi_\ell(\,\cdot\,;x)(y)| = 3\frac{\cosh_\Lambda(N)-\cosh_\Lambda(0)}{2\sinh_\Lambda(N)\sinh_\Lambda(1)}.$$

As $\sinh_\Lambda(N/2) \le \cosh_\Lambda(N/2)$,

$$\begin{aligned}2\sinh_\Lambda(N/2)\sinh_\Lambda(N/2) &\le 2\cosh_\Lambda(N/2)\sinh_\Lambda(N/2),\\ \cosh_\Lambda(N)-\cosh_\Lambda(0) &\le \sinh_\Lambda(N)-\sinh_\Lambda(0) = \sinh_\Lambda(N).\end{aligned}$$

We conclude that

$$\max_{y\in\mathcal{N}_{\ell-1}} \sum_{x\in\mathcal{N}_\ell^*} |Q_{\ell-1}\phi_\ell(\,\cdot\,;x)(y)| \le \frac{3}{2\sinh_\Lambda(1)} = \frac{\sqrt{3}}{2}. \qquad \square$$

We now arrive at the bound needed for Theorem 2.

Corollary 8. *Let V_L be the space of continuous piecewise multilinear functions on a uniform tensor product grid in d dimensions. Then $\|I-Q_{\ell-1}\|_{L^\infty} \le 1+(\sqrt{3}/2)^d$.*

Corollary 8 follows from Corollary 7, which gives a bound for $\|Q_{\ell-1}\|_{L^\infty}$ in the unidimensional case; the definition of the tensor product, allowing us to bound $\|Q_{\ell-1}\|_{L^\infty} \le (\sqrt{3}/2)^d$ in the multidimensional case; and the triangle inequality applied to $I - Q_{\ell-1}$.

REFERENCES

[1] M. Ainsworth, S. Klasky, and B. Whitney, *Compression using lossless decimation: Analysis and application*, SIAM J. Sci. Comput., 39 (2017), pp. B732–B757, https://doi.org/10.1137/16M1086248.

[2] M. Ainsworth and J. T. Oden, *A Posteriori Error Estimation in Finite Element Analysis*, Pure Appl. Math. (Hoboken) 37, John Wiley & Sons, 2011.

[3] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, *Multilevel techniques for compression and reduction of scientific data—the univariate case*, Comput. Vis. Sci., 19 (2018), pp. 65–76, https://doi.org/10.1007/s00791-018-00303-9.

[4] W. Austin, G. Ballard, and T. G. Kolda, *Parallel tensor compression for large-scale scientific data*, in Proceedings of the 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), May 2016, pp. 912–922, https://doi.org/10.1109/IPDPS.2016.67.

[5] A. H. Baker, H. Xu, J. M. Dennis, M. N. Levy, D. Nychka, S. A. Mickelson, J. Edwards, M. Vertenstein, and A. Wegener, *A methodology for evaluating the impact of data compression on climate simulation data*, in Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing, HPDC '14, New York, NY, 2014, ACM, pp. 203–214, https://doi.org/10.1145/2600212.2600217.

[6] R. E. Bank, T. F. Dupont, and H. Yserentant, *The hierarchical basis multigrid method*, Numer. Math., 52 (1988), pp. 427–458, https://doi.org/10.1007/BF01462238.

[7] L. A. Bautista Gomez and F. Cappello, *Improving floating point compression through binary masks*, in Proceedings of the 2013 IEEE International Conference on Big Data, Oct. 2013, pp. 326–331, https://doi.org/10.1109/BigData.2013.6691591.

[8] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*, Athena Scientific & Dynamic Ideas, Belmont, MA, 1997.

[9] F. Bornemann and H. Yserentant, *A basic norm equivalence for the theory of multilevel methods*, Numer. Math., 64 (1993), pp. 455–476, https://doi.org/10.1007/BF01388699.

[10] M. Burtscher, H. Mukka, A. Yang, and F. Hesaaraki, *Real-time synthesis of compression algorithms for scientific data*, in SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, Nov. 2016, pp. 264–275, https://doi.org/10.1109/SC.2016.22.

[11] M. Crouzeix and V. Thomée, *The stability in L_p and W_p^1 of the L_2-projection onto finite element function spaces*, Math. Comp., 48 (1987), pp. 521–532, https://doi.org/10.1090/S0025-5718-1987-0878688-2.

[12] I. Daubechies, *The wavelet transform, time-frequency localization and signal analysis*, IEEE Trans. Inform. Theory, 36 (1990), pp. 961–1005, https://doi.org/10.1109/18.57199.

[13] S. Di and F. Cappello, *Fast error-bounded lossy HPC data compression with SZ*, in Proceedings of the 2016 IEEE 30th International Parallel and Distributed Processing Symposium, Chicago, IL, USA, May 2016, IEEE, pp. 730–739, https://doi.org/10.1109/IPDPS.2016.11.

[14] I. Foster, M. Ainsworth, B. Allen, J. Bessac, F. Cappello, J. Y. Choi, E. Constantinescu, P. E. Davis, S. Di, W. Di, H. Guo, S. Klasky, K. K. Van Dam, T. Kurc, Q. Liu, A. Malik, K. Mehta, K. Mueller, T. Munson, G. Ostouchov, M. Parashar, T. Peterka, L. Pouchard, D. Tao, O. Tugluk, S. Wild, M. Wolf, J. M. Wozniak, W. Xu, and S. Yoo, *Computing just what you need: Online data analysis and reduction at extreme scales*, in Euro-Par 2017: Parallel Processing, F. F. Rivera, T. F. Pena, and J. C. Cabaleiro, eds., Springer International Publishing, Cham, 2017, pp. 3–19, https://doi.org/10.1007/978-3-319-64203-1_1.

[15] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1996.

[16] S. Grgic, K. Kers, and M. Grgic, *Image compression using wavelets*, in Proceedings of the IEEE International Symposium on Industrial Electronics, 1999, pp. 99–104, https://doi.org/10.1109/ISIE.1999.801765.

[17] A. Gumel, E. Twizell, M. Arigu, and F. Fakhir, *Numerical methods for a non-linear system arising in chemical kinetics*, Pertanika J. Sci. Tech., 5 (1997), pp. 191–200.

[18] J. S. Hesthaven, G. Rozza, and B. Stamm, *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*, Springer International Publishing, Cham, Switzerland, 2016, https://doi.org/10.1007/978-3-319-22470-1.

[19] L. Ibarria, P. Lindstrom, J. Rossignac, and A. Szymczak, *Out-of-core compression and decompression of large n-dimensional scalar fields*, Comput. Graph. Forum, 22 (2003), pp. 343–348, https://doi.org/10.1111/1467-8659.00681.

[20] Johns Hopkins Turbulence Database, Group, *Channel flow dataset*, Oct. 2017, https://doi.org/10.7281/T10K26QW, accessed Jan. 2, 2018.

[21] M. Lee and R. D. Moser, *Direct numerical simulation of turbulent channel flow up to $Re_\tau \approx$ 5200*, J. Fluid Mech., 774 (2015), 395–415, https://doi.org/10.1017/jfm.2015.268.

[22] Y. Li, E. Perlman, M. Wan, Y. Yang, C. Meneveau, R. Burns, S. Chen, A. Szalay, and G. Eyink, *A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence*, J. Turbul., 9 (2008), N31, https://doi.org/10.1080/14685240802376389.

[23] P. Lindstrom, *Fixed-rate compressed floating-point arrays*, IEEE Trans. Vis. Comput. Graph., 20 (2014), pp. 2674–2683, https://doi.org/10.1109/TVCG.2014.2346458.

[24] P. Lindstrom and M. Isenburg, *Fast and efficient compression of floating-point data*, IEEE Trans. Vis. Comput. Graph., 12 (2006), pp. 1245–1250, https://doi.org/10.1109/TVCG.2006.143.

[25] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek, *An overview of JPEG-2000*, in Proceedings of the Data Compression Conference, 2000, pp. 523–541, https://doi.org/10.1109/DCC.2000.838192.

[26] J. D. McCalpin, *Memory Bandwidth and System Balance in HPC Systems*, Nov. 2016, https://sites.utexas.edu/jdm4372/2016/11/22/sc16-invited-talk-memory-bandwidth-and-system-balance-in-hpc-systems/.

[27] P. Oswald, *Multilevel Finite Element Approximation*, Springer Vieweg Verlag, 1994.

[28] E. Perlman, R. Burns, Y. Li, and C. Meneveau, *Data exploration of turbulence simulations using a database cluster*, in Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, vol. 23, Reno, NV, Nov. 2007, ACM, https://doi.org/10.1145/1362622.1362654.

[29] A. Quarteroni, A. Manzoni, and F. Negri, *Reduced Basis Methods for Partial Differential Equations: An Introduction*, Springer, Cham, 2016, https://cds.cern.ch/record/2062564.

[30] M. Strengert, M. Magallón, D. Weiskopf, S. Guthe, and T. Ertl, *Hierarchical visualization and compression of large volume datasets using GPU clusters*, in the Proceedings of the Eurographics Workshop on Parallel Graphs and Visualization, 2004, pp. 41–48.

[31] E. H. Twizell, A. B. Gumel, and Q. Cao, *A second-order scheme for the "Brusselator" reaction–diffusion system*, J. Math. Chem., 26 (1999), pp. 297–316, https://doi.org/10.1023/A:1019158500612.

[32] G. K. Wallace, *The JPEG still picture compression standard*, IEEE Trans. Consumer Electron., 38 (1992), pp. xviii–xxxiv, https://doi.org/10.1109/30.125072.