Computing and Visualization in Science (2018) 19:65–76
https://doi.org/10.1007/s00791-018-00303-9

ORIGINAL ARTICLE

Multilevel techniques for compression and reduction of scientific data—the univariate case

Mark Ainsworth[1,2] · **Ozan Tugluk**[1,3] · **Ben Whitney**[1] · **Scott Klasky**[2]

Published online: 9 November 2018

Abstract
We present a multilevel technique for the compression and reduction of univariate data and give an optimal complexity algorithm for its implementation. A hierarchical scheme offers the flexibility to produce multiple levels of partial decompression of the data so that each user can work with a reduced representation that requires minimal storage whilst achieving the required level of tolerance. The algorithm is applied to the case of turbulence modelling in which the datasets are traditionally not only extremely large but inherently non-smooth and, as such, rather resistant to compression. We decompress the data for a range of relative errors, carry out the usual analysis procedures for turbulent data, and compare the results of the analysis on the reduced datasets to the results that would be obtained on the full dataset. The results obtained demonstrate the promise of multilevel compression techniques for the reduction of data arising from large scale simulations of complex phenomena such as turbulence modelling.

Keywords Data compression · Data reduction · Lossy compression · Multilevel compression · Error-controlled compression

1 Introduction

Computational simulation is now pervasive throughout science and engineering, providing new insights and unsurpassed levels of resolution for complex physical phenomena. However, even today's petascale leadership computing facilities are already producing data in such quantities that it is rapidly becoming infeasible, or even impossible, to store all of the output from a leading edge simulation. For instance, present day fusion simulations being executed on leadership computing facilities typically produce around 100 PB of data in a single run. The mismatch between compute speed and the speed with which data can be written to disc is set to become even more acute with the expectation that the machines anticipated to come online in 2024 will compute at 10^{18} operations per second, but will only be capable of writing 10^{12} bytes per second to disc—a situation 200 times worse than on current petascale systems.

The anticipated deluge of data places an emphasis on the development of techniques for the compression and reduction of scientific data being produced by very large scale simulations. However, currently available techniques for the compression of scientific data do not achieve sufficiently high compression rates to tackle the problems alluded to earlier. The introduction of compression into a scientific workflow can be beneficial in terms of reducing storage costs

This work is dedicated to Prof. Ulrich Langer on the occasion of his sixtieth birthday.

This research was supported in part 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.

✉ Mark Ainsworth
mark_ainsworth@brown.edu

Ozan Tugluk
ozan_tugluk1@brown.edu

Ben Whitney
ben_whitney@brown.edu

Scott Klasky
klasky@ornl.gov

[1] Division of Applied Mathematics, Brown University, 182 George St., Providence, RI 02912, USA

[2] Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

[3] Department of Astronautical Engineering, University of Turkish Aeronautical Association, 06790 Ankara, Turkey

and bandwidth, but it comes at a price: computer cycles must be redeployed for the compression and decompression steps. Furthermore, lossy compression introduces an additional source of error which many practitioners worry may compromise the very phenomena that they seek to uncover.

Data compression and reduction is a well-studied subject across a wide range of disciplines and data types [7,26] and many techniques are available for scientific data, including resampling and polynomial interpolation/extrapolation [1,10,16,22], Fourier transforms paired with coefficient quantization [21,31], wavelet decompositions [9,14,23,30], and more [2,4,6,19,27,29]. The selection of a particular compression technique (or the features to be included in a new algorithm) must take account of the usage of the data. For instance, streaming methods generally fail to attain optimal compression yet may be deemed appropriate for certain scenarios when the dataset is too large to be loaded into memory in its entirety. Some algorithms allow for random access of the compressed data, while others build in resilience against data corruption.

Generally speaking, a compression algorithm that is designed with a particular type of data in mind (e.g. text, images, or audio) is unlikely to perform well on another type of data. A case in point is scientific data, which is stored in high-precision formats and whose structure is quite different from the media data that most codecs target. The underlying need for high fidelity quantitative agreement means that compression of scientific data is generally regarded as quite challenging. Scientific computing requires specialized algorithms that take into account the provenance of the data as well as the objectives dictating the use of compression.

In the present work we present a multilevel technique for the compression and reduction of univariate data. Hierarchical compression techniques offer a high degree of flexibility which can be leveraged in a number of ways. For instance, different end users may be satisfied with quite different levels of accuracy when analyzing a dataset. A hierarchical scheme offers the flexibility to produce multiple levels of partial decompression of the data so that each user can work with a reduced representation that requires minimal storage whilst achieving the required level of tolerance. Alternatively, differing local computational resources may constrain the amount of data that a particular user can feasibly manage in their analysis. A hierarchical scheme is capable of providing a dataset that meets the storage constraint whilst minimising the lossiness.

In Sect. 2 we present algorithms that are designed with both of the above use cases in mind. The performance of the algorithms is quantified in terms of the underlying Sobolev regularity of the data and shown to be, in a sense to be made precise, quasioptimal. Importantly, the algorithms do not require any *a priori* knowledge of the regularity of the data. Section 3 describes the implementational details of the multilevel compression technique and, in particular, shows how the decomposition can be performed in optimal complexity without incurring any additional storage burden beyond that needed for the original dataset. Finally, in Sect. 4 the algorithm is applied to the case of turbulence modelling, where the datasets are traditionally not only extremely large but inherently non-smooth and, as such, rather resistant to compression. Note that wavelet techniques have been used successfully for modelling turbulence [28]. We apply the univariate algorithm to the array-valued data by regarding each snapshot in time as a datum, and then compressing the snapshots with time interpreted as the univariate index. We decompress the data for a range of relative errors, carry out the usual analysis procedures for turbulent data, and compare the results of the analysis on the reduced datasets to the results that would be obtained on the full dataset. The results obtained demonstrate the promise of multilevel compression techniques for the reduction of data arising from large scale simulations of complex phenomena such as turbulence modelling.

2 Algorithms for data reduction based on orthogonal projection

Let $[a, b]$ be an interval on the real line partitioned into non-overlapping intervals $\mathcal{P}_L$. Consider data representing a function u defined on $[a, b]$. Concretely, the data will consist of the values of certain degrees of freedom, e.g. the values taken by u at the knots $\mathcal{N}_L$ of the intervals in $\mathcal{P}_L$. To reduce the data, an alternative, smaller set of degrees of freedom must be chosen to represent the function (or an approximation thereof), and the values of these new degrees of freedom must be determined. We will assume that $\mathcal{P}_L$ is the result of repeated bisection of intervals in an initial partition $\mathcal{P}_0$, resulting in a hierarchy of increasingly fine partitions $\{\mathcal{P}_\ell : 0 \leq \ell \leq L\}$ and nodesets $\{\mathcal{N}_\ell : 0 \leq \ell \leq L\}$. Each successive partition has twice the number of intervals as the previous partition, so $\#\mathcal{P}_\ell = 2^\ell \#\mathcal{P}_0$ for $\ell \in \{0, \ldots, L\}$.

Let V_ℓ denote the space of continuous piecewise linear functions with respect to the partition $\mathcal{P}_\ell$ for $\ell \in \{0, \ldots, L\}$. Then the spaces $\{V_\ell : 0 \leq \ell \leq L\}$ are nested, in the sense that $V_0 \subset V_1 \subset \cdots \subset V_L \subset L^2([a, b])$, and a function in V_ℓ is determined by $\#\mathcal{N}_\ell = \#\mathcal{P}_\ell + 1 \simeq 2^\ell$ degrees of freedom. Let $Q_\ell \colon V_L \to V_\ell$ be the $L^2([a, b])$ projection onto V_ℓ for $\ell \in \{0, \ldots, L\}$. Given a function $u \in V_L$, we propose to select a reduced representation for u from among the projections $\{Q_\ell u : 0 \leq \ell \leq L\}$. $Q_L u = u$, clearly, so no loss of accuracy is entailed in using $Q_L u$ as a representation for u, although, of course, neither is there any reduction of degrees of freedom in this case. At the opposite extreme, using $Q_0 u$ as a reduced representation uses the fewest degrees of freedom but at the same time is the lossiest course. Between these two extremes one can choose a reduced representation $Q_\ell u \approx u$.

One use case that can be envisaged is a user interested in performing an analysis of a dataset but subject to memory constraints. In this scenario, the user wishes to have the best reconstruction that can fit in the available storage. Algorithm 1 gives a procedure that generates a reduced representation $Q_\ell u$ with ℓ the maximum possible such that the number n of degrees of freedom used is no more than N, a prescribed storage limit.

Algorithm 1 Reduction given a constraint on the degrees of freedom available for the reduced representation.

Require: $N \geq \#\mathcal{N}_0$.
  function APPROXIMATE(function u, number of DoFs N)
    for $\ell = 0, 1, 2, \ldots$ **do**
      if $\#\mathcal{N}_\ell \leq N$ and $\#\mathcal{N}_{\ell+1} > N$ **then**
        return $Q_\ell u$
      end if
    end for
  end function

A second use case might be a user seeking the reduced representation with the fewest degrees of freedom possible while meeting a prescribed tolerance on the relative error ϵ, i.e. satisfying for some $\tau \geq 0$

$$\epsilon = \frac{\|u - Q_\ell u\|}{\|u\|} \leq \tau$$

where $\|\cdot\|$ denotes the $L^2([a,b])$ norm. Algorithm 2 gives a method for achieving this objective.

Algorithm 2 Reduction given a constraint on the maximum allowable error in the reduced representation.

Require: $\tau \geq 0$.
  function APPROXIMATE(function u, tolerance τ)
    calculate $\|u\|$
    for $\ell = 0, 1, 2, \ldots$ **do**
      if $\|u - Q_\ell u\| \leq \tau \|u\|$ **then**
        return $Q_\ell u$
      end if
    end for
  end function

Observe that Algorithm 2 is guaranteed to terminate with $\ell \leq L$, since $Q_L u = u$. The calculation of the error $\|u - Q_\ell u\|$ used in the stopping criterion can be carried out efficiently using the Pythagorean identity $\|u - Q_\ell u\|^2 = \|u\|^2 - \|Q_\ell u\|^2$. Full details on how to compute $\|u\|$ and $\|Q_\ell u\|$, as well as a description of how to compute the projections $\{Q_\ell u : 0 \leq \ell \leq L\}$, are presented in Sect. 3.

The following result quantifies the performance of Algorithms 1 and 2 in terms of the Sobolev regularity r of the data u. We emphasize that in order to apply Algorithms 1 and 2 it is *not* necessary to know the regularity of the data, i.e., the largest value of r such that $u \in H^r([a,b])$.

Theorem 1 *Let $u \in H^r([a,b])$ for some $r \in (0, 3/2)$. Suppose $u \neq 0$.*

1. *Let $N \in \mathbb{N}$ be given. Then Algorithm 1 gives a reduced representation using $n \leq N$ degrees of freedom whilst achieving a relative error $\epsilon \lesssim N^{-r}\|u\|_r/\|u\|$.*
2. *Let $\tau \in (0, 1]$ be given. Then Algorithm 2 gives a reduced representation with relative error $\epsilon \leq \tau$ whilst requiring $n \lesssim (\tau\|u\|/\|u\|_r)^{-1/r}$ degrees of freedom.*

Moreover, the $L^2([a,b])$ projections used in Algorithms 1 and 2 are quasioptimal among linear reductions, in the sense that: any other method using N degrees of freedom cannot guarantee relative error below $O(N^{-r}\|u\|_r/\|u\|)$; or, any other method guaranteeing relative error $\tau\|u\|_r/\|u\|$ must use at least $O(\tau^{-1/r})$ degrees of freedom.

Proof We begin with Item 1. With $N \in \mathbb{N}$ given, take $M \in \{0, \ldots, L\}$ so that Algorithm 1 returns $Q_M u$. Observe that the projection error $u - Q_M u = (Q_L - Q_M)u$ can be written as the sum $\sum_{\ell=M+1}^{L} (Q_\ell - Q_{\ell-1})u$. We claim furthermore that

$$\|u - Q_M u\|^2 = \sum_{\ell=M+1}^{L} \|(Q_\ell - Q_{\ell-1})u\|^2. \tag{1}$$

It suffices to show that $\{(Q_\ell - Q_{\ell-1})u : M < \ell \leq L\}$ are pairwise orthogonal. Take $m, \ell \in \{M+1, \ldots, L\}$ differing. Without loss of generality, assume $m < \ell$. By the nestedness of the function spaces, $(Q_m - Q_{m-1})u \in V_m \subseteq V_{\ell-1}$. Consequently, with $(\cdot\,,\,\cdot)$ denoting the $L^2([a,b])$ inner product,

$$(Q_\ell u, (Q_m - Q_{m-1})u) = (Q_{\ell-1}u, (Q_m - Q_{m-1})u),$$

and so $(Q_\ell - Q_{\ell-1})u \perp (Q_m - Q_{m-1})u$. Thus (1) holds.

Next we wish to relate the right-hand side of (1) to the $H^r([a,b])$ norm of u. To this end we use the following norm equivalence, which holds for $r \in (0, 3/2)$ with constants independent of u [5,24]:

$$\|u\|_r^2 \simeq \sum_{\ell=0}^{L} 2^{2r\ell} \|(Q_\ell - Q_{\ell-1})u\|^2, \tag{2}$$

where Q_{-1} denotes the zero operator. The functions $\{(Q_\ell - Q_{\ell-1})u : M < \ell \leq L\}$ composing the projection error $u - Q_M u$ are exactly those that are weighted most heavily in (2). Multiplying each side of (1) by the least of these weights, $2^{2r(M+1)}$, we find that

$$\begin{aligned} 2^{2r(M+1)}\|u - Q_M u\|^2 &= \sum_{\ell=M+1}^{L} 2^{2r(M+1)} \|(Q_\ell - Q_{\ell-1})u\|^2 \\ &\leq \sum_{\ell=M+1}^{L} 2^{2r\ell} \|(Q_\ell - Q_{\ell-1})u\|^2 \lesssim \|u\|_r^2. \end{aligned}$$

Dividing through by $2^{2r(M+1)}$, we find that

$$\|u - Q_M u\|^2 \lesssim 2^{-2r(M+1)} \|u\|_r^2, \tag{3}$$

again with constants independent of u (and M). As Algorithm 1 returns $Q_M u$, N is bounded by $\#\mathcal{N}_M \leq N < \#\mathcal{N}_{M+1}$. In particular, $N < \#\mathcal{N}_{M+1} \simeq 2^{M+1}$. $2^{-2r(M+1)} \lesssim N^{-2r}$, so $\|u - Q_M u\|^2 \lesssim (N^{-2r}\|u\|_r^2/\|u\|^2)\|u\|^2$, as desired.

Next, consider Item 2. With $\tau \in (0, 1]$ given, take $M \in \{0, \ldots, L\}$ so that Algorithm 2 returns $Q_M u$, and let n be the number of degrees of freedom required to represent $Q_M u$. The case $M = 0$ is trivial. If $M > 0$,

$$\|u - Q_M u\|^2 \leq \tau^2 \|u\|^2 < \|u - Q_{M-1} u\|^2. \tag{4}$$

We remarked above that (3) holds with constants independent of u and M. That is, there exists some constant $\lambda > 0$ independent of u and M such that $\lambda \|u - Q_{M-1} u\|^2 \leq 2^{-2rM} \|u\|_r^2$. Combining with (4), we find that $\lambda^{-1} 2^{-2rM} \|u\|_r^2 \geq \tau^2 \|u\|^2$. Rearranging, $n \simeq 2^M \lesssim (\tau \|u\|/\|u\|_r)^{-1/r}$, as desired.

Finally, we consider the quasioptimality of Algorithms 1 and 2. Consider any linear reduction operator $T : H^r([a, b]) \to L^2([a, b])$. If N degrees of freedom are required to represent Tu (i.e., if $\operatorname{rank}(T) = N$), then $\|I - T\| \gtrsim N^{-r}$ (with constant independent of N) [12, p. 119]. In other words, there exists some $u \in H^r([a, b])$ such that $\|u - Tu\| \gtrsim (N^{-r}\|u\|_r/\|u\|)\|u\|$, so T can only guarantee accuracy of at best $O(N^{-r}\|u\|_r/\|u\|)$. On the other hand, if T can guarantee accuracy of $\tau\|u\|_r/\|u\|$ for all inputs u (i.e., if $\|u - Tu\| \leq \tau\|u\|_r$ for all nonzero $u \in H^r([a, b])$), we can conclude that T must require at least $O(\tau^{-1/r})$ degrees of freedom: $\|I - T\| \leq \tau$ and $\operatorname{rank}(T)^{-r} \lesssim \|I - T\|$, so $\operatorname{rank}(T) \gtrsim \tau^{-1/r}$. □

In order to verify the predictions of Theorem 1, we consider datasets corresponding to four functions of known Sobolev regularities $r \in \{0.2, 0.6, 1.0, 1.4\}$. Figure 1 shows the dataset corresponding to the case $r = 0.2$ along with a selection of its projections to intermediate levels. The performance of Algorithms 1 and 2 applied to each of the functions is illustrated in Figs. 2 and 3. Observe that the dependence of the number of degrees of freedom on the error tolerance (and vice versa) is consistent with the predictions of Theorem 1.

3 Implementation

As shown in the previous section, Algorithm 1 achieves quasioptimal errors when used to generate a reduced representation given a fixed number of degrees of freedom. Likewise, Algorithm 2 uses a quasioptimal number of degrees of freedom when used to generate a reduced representation achieving a fixed error bound. In practice, though,

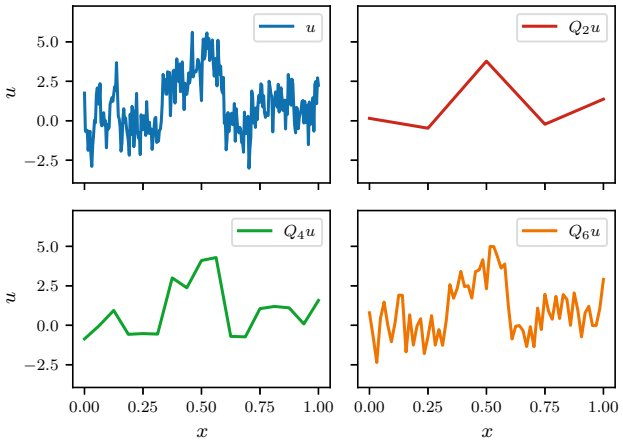

Fig. 1 Illustration of a function $u \in H^{0.2}([a, b])$ used in Figs. 2 and 3 along with three of its projections Q_2u, Q_4u, and Q_6u

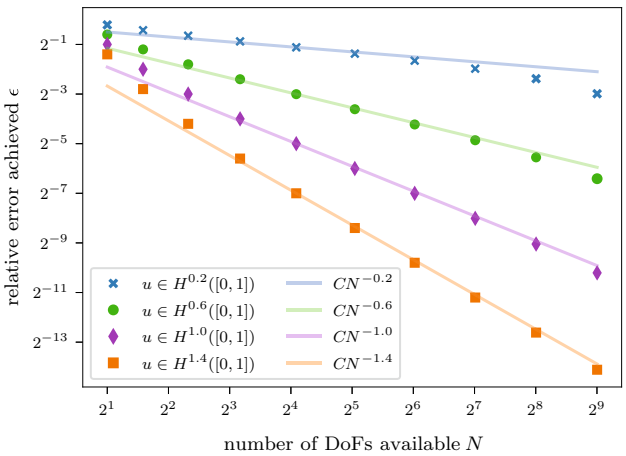

Fig. 2 Illustration of the error decay as Algorithm 1 is applied to data using increasing numbers of degrees of freedom. As expected, the error decreases as the number of degrees of freedom used increases, and the decay is fastest for the most regular data. The dependence of the error on the number of degrees of freedom supplied agrees with Theorem 1

scientists often use a dataset for multiple applications, each requiring different accuracies or allowing for different numbers of degrees of freedom. The projection onto a finer level may be needed to meet a prescribed tolerance. Conversely, if the data are to be transferred across a network or onto a machine with limited storage, then the projection onto a coarser level may be required.

In general, the accuracy needed or the number of degrees of freedom available may not be known in advance, and in such cases one would ideally like to calculate and store the entire hierarchy of projections $\{Q_\ell u : 0 \leq \ell \leq L\}$. Storing these projections naïvely, using the nodal values on $\mathcal{N}_\ell$ to represent $Q_\ell u$, will require $\sum_{\ell=0}^{L} \#\mathcal{N}_\ell \simeq 2\#\mathcal{N}_L$ degrees of freedom. As mentioned in the introduction, we are primarily concerned with data from extreme scale simulations which stress the available storage media. As such, we cannot

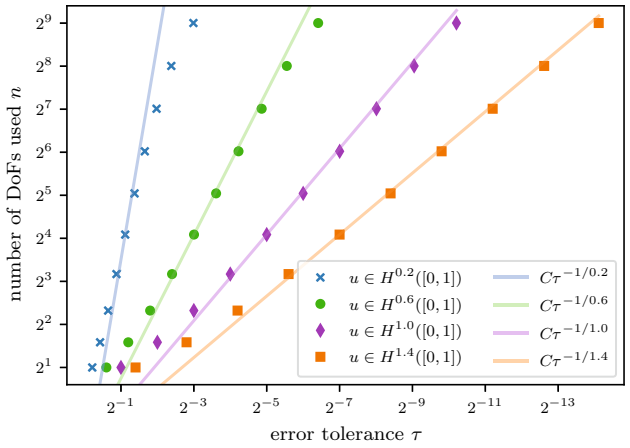

Fig. 3 Illustration of the growth of number of degrees of freedom needed as Algorithm 2 is applied to data with decreasing error tolerances. As expected, more degrees of freedom are required to generate a reduced representation with greater accuracy, and the growth is fastest for the least regular data. The dependence of the number of degrees of freedom needed on the error tolerance agrees with Theorem 1

afford to double the storage requirement in order to retain the intermediate projections. Alternatively, we could store only the original function u and recalculate the projections $\{Q_\ell u : 0 \le \ell \le L\}$ as required. The disadvantage of the latter approach is that computing any intermediate projection $Q_\ell u$ would require $O(\#\mathcal{N}_L)$ operations.

In this section we adopt an approach between these two extremes. Specifically, we present data structures and associated algorithms which enable us to reconstruct any of the projections $Q_\ell u$ in $O(\#\mathcal{N}_\ell)$ operations without increasing the overall level of storage beyond the $\#\mathcal{N}_L$ degrees of freedom required to store the original data u.

3.1 Decomposition

Let $\ell \in \{0, \ldots, L-1\}$ and suppose we have $Q_{\ell+1}u$ in hand. In particular, suppose we have an array of the nodal values of $Q_{\ell+1}u$ on $\mathcal{N}_{\ell+1}$. We aim to compute $Q_\ell u$, the next best reduced representation, and to store it, together with $Q_{\ell+1}u$, without requiring more than the $\#\mathcal{N}_{\ell+1}$ degrees of freedom originally needed to store $Q_{\ell+1}u$. The key idea is based on the following decomposition:

$$Q_{\ell+1}u = (I - \Pi_\ell)Q_{\ell+1}u + Q_\ell u - \underbrace{(Q_\ell u - \Pi_\ell Q_{\ell+1}u)}_{z_\ell}, \quad (5)$$

where $\Pi_\ell \colon V_L \to V_\ell$ is the piecewise linear interpolant. Consider how each of the three terms above can be represented. The first term, $(I - \Pi_\ell)Q_{\ell+1}u$, vanishes on $\mathcal{N}_\ell$ and can therefore be represented solely in terms of the values it takes on $\mathcal{N}_{\ell+1} \setminus \mathcal{N}_\ell$, the nodes in level $\mathcal{P}_{\ell+1}$ that are not present in $\mathcal{P}_\ell$. Conversely, the second term, $Q_\ell u \in V_\ell$, can be represented using the values at the nodes present in $\mathcal{P}_\ell$. The values taken by $Q_\ell u$ on $\mathcal{N}_{\ell+1}\setminus\mathcal{N}_\ell$ can be reconstructed through interpolation and as such do not need to be stored. The third term, z_ℓ, need not be stored explicitly but can be reconstructed from the first term thanks to the following observations:

- $z_\ell \in V_\ell$
- $z_\ell - (I - \Pi_\ell)Q_{\ell+1}u = -(Q_{\ell+1} - Q_\ell)u \in V_\ell^\perp$

In other words, z_ℓ is the orthogonal projection of the first term in (5) onto V_ℓ:

$$\begin{aligned} Q_\ell(I - \Pi_\ell)Q_{\ell+1}u &= Q_\ell I Q_{\ell+1}u - Q_\ell \Pi_\ell Q_{\ell+1}u \\ &= Q_\ell u - \Pi_\ell Q_{\ell+1}u = z_\ell. \end{aligned}$$

Hence, z_ℓ is the solution to the variational problem

$$\begin{aligned} &\text{find } z_\ell \in V_\ell \text{ such that } (z_\ell, v_\ell) \\ &= ((I - \Pi_\ell)Q_{\ell+1}u, v_\ell) \text{ for all } v_\ell \in V_\ell \end{aligned} \quad (6)$$

and thus it can be generated from the first term (see Sect. 3.3).

How can the decomposition in (5) be performed efficiently? The first term, $(I - \Pi_\ell)Q_{\ell+1}u$, is constructed directly from $Q_{\ell+1}u$ by subtracting the interpolant $\Pi_\ell Q_{\ell+1}u$. Algorithmically, this can be accomplished by simply adjusting the values stored on $\mathcal{N}_{\ell+1}\setminus\mathcal{N}_\ell$, as illustrated in Fig. 4. Following this step, the original data has been decomposed as follows.

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

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

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

Hence, $Q_\ell u$ can be easily obtained given z_ℓ. Both $\Pi_\ell Q_{\ell+1}u$ and z_ℓ are contained in V_ℓ and hence are specified by their values on $\mathcal{N}_\ell$. Therefore, we can compute $Q_\ell u$ from $\Pi_\ell Q_{\ell+1}u$ by adding the values of z_ℓ on $\mathcal{N}_\ell$. In particular, the values stored on $\mathcal{N}_{\ell+1}\setminus\mathcal{N}_\ell$ are left untouched.

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

1. Modify the nodal values of $Q_{\ell+1}u$ on $\mathcal{N}_{\ell+1}\setminus\mathcal{N}_\ell$ to become those of $(I - \Pi_\ell)Q_{\ell+1}u$, as shown in Fig. 4.
2. Compute z_ℓ by solving (6).
3. Add z_ℓ to $\Pi_\ell Q_{\ell+1}u$ to obtain $Q_\ell u$ on $\mathcal{N}_\ell$.

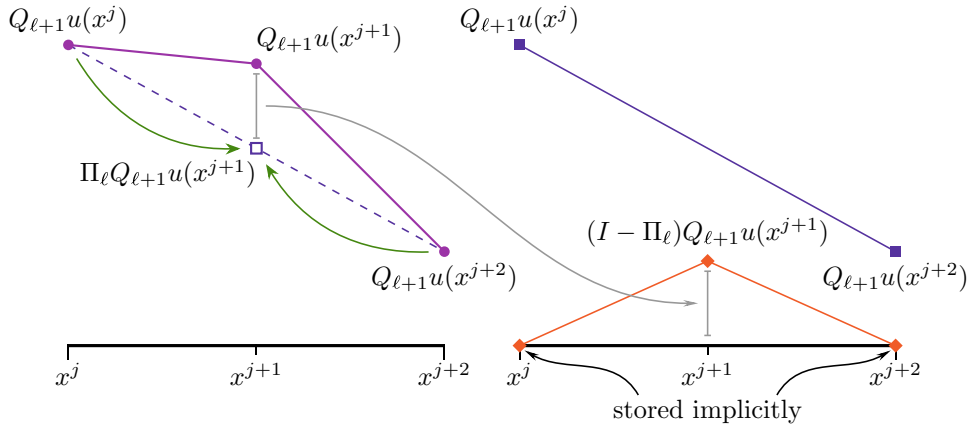

Fig. 4 Illustration of the calculation of $(I-\Pi_\ell)Q_{\ell+1}u$ and storage of $\Pi_\ell Q_{\ell+1}u$. Here the nodes x^j and x^{j+2} belong to $\mathcal{N}_\ell$ and the node x^{j+1} belongs to $\mathcal{N}_{\ell+1}\backslash\mathcal{N}_\ell$. The value stored at each node x^{j+1} in $\mathcal{N}_{\ell+1}\backslash\mathcal{N}_\ell$ is modified by subtracting the average of the values stored at the neighboring coarse level nodes, while the values at the coarse level nodes are left unchanged. This results in the values of $(I-\Pi_\ell)Q_{\ell+1}u$ being stored on $\mathcal{N}_{\ell+1}\backslash\mathcal{N}_\ell$ and the values of $\Pi_\ell Q_{\ell+1}u$ being stored on $\mathcal{N}_\ell$

By proceeding recursively, we can start with the nodal values of $u = Q_L u$ on $\mathcal{N}_L$ and successively compute

$$\underbrace{(I-\Pi_{L-1})Q_L}_{\text{on } \mathcal{N}_L\backslash\mathcal{N}_{L-1}}, \underbrace{(I-\Pi_{L-2})Q_{L-1}}_{\text{on } \mathcal{N}_{L-1}\backslash\mathcal{N}_{L-2}}, \ldots, \underbrace{(I-\Pi_0)Q_1}_{\text{on } \mathcal{N}_1\backslash\mathcal{N}_0}, \underbrace{Q_0 u}_{\text{on } \mathcal{N}_0}. \tag{7}$$

All of these functions can be represented without increasing the number of degrees of freedom that were required to store u. We can easily reconstruct any of the projections $\{Q_\ell u : 0 \le \ell \le L\}$ using the procedure described in the next subsection.

Algorithm 2 requires the calculation of the approximation error $\|u - Q_\ell u\|$ when determining whether to return a given projection $Q_\ell u$. These errors can be computed during the decomposition process. By the definition of the $L^2([a,b])$ projection, $\|u-Q_\ell u\|^2 = \|u\|^2 - \|Q_\ell u\|^2$. $\|u\|^2$ and $\|Q_\ell u\|^2$ can be calculated using Simpson's rule. With $\{\|Q_\ell u\|^2 : 0 \le \ell \le L\}$ calculated once and stored, the reduction errors $\{\|u - Q_\ell u\| : 0 \le \ell \le L\}$ can be determined in $O(1)$ operations using $O(L)$ additional storage.

3.2 Recomposition

Let $\ell \in \{0, \ldots, L-1\}$ and suppose we have $(I-\Pi_\ell)Q_{\ell+1}u$ and $Q_\ell u$ in hand. In particular, suppose we have one array of the nodal values of $(I-\Pi_\ell)Q_{\ell+1}u$ on $\mathcal{N}_{\ell+1}\backslash\mathcal{N}_\ell$ and another of the nodal values of $Q_\ell u$ on $\mathcal{N}_\ell$. We can recompute $Q_{\ell+1}u$ using (5):

$$Q_{\ell+1}u = (I-\Pi_\ell)Q_{\ell+1}u + Q_\ell u - z_\ell. \tag{8}$$

Algorithmically, (8) consists of the following steps:

1. Compute z_ℓ and then subtract it from $Q_\ell u$. This leaves the nodal values of $(I-\Pi_\ell)Q_{\ell+1}u$ stored on $\mathcal{N}_{\ell+1}\backslash\mathcal{N}_\ell$ and the nodal values of $Q_\ell u - z_\ell = \Pi_\ell Q_{\ell+1}u$ stored on $\mathcal{N}_\ell$.
2. Prolongate $\Pi_\ell Q_{\ell+1}u$ from V_ℓ to $V_{\ell+1}$ and add to $(I-\Pi_\ell)Q_{\ell+1}u$ to obtain $Q_{\ell+1}u$, as illustrated in Fig. 5.

By proceeding recursively, we can start with the decomposition given in (7) and recompute any of the projections $\{Q_\ell u : 0 \le \ell \le L\}$ in the hierarchy given the corrections $\{z_\ell : 0 \le \ell < L\}$, which we compute as described in the next subsection.

3.3 Computation of the correction

The correction z_ℓ is obtained by solving (6). We can reformulate (6) as a matrix equation by taking v_ℓ to be each of the coarse grid nodal basis functions $\{\phi_\ell^j : x^j \in \mathcal{N}_\ell\}$ in turn. This yields a matrix equation of the form $M_\ell \vec{\xi} = \vec{f}$ where M_ℓ is the mass matrix with respect to the nodal basis on V_ℓ, $\vec{\xi}$ is the vector of nodal values taken by z_ℓ, and $\vec{f}$ is the load vector. Two issues need to be tackled: the computation of $\vec{f}$ and the inversion of the mass matrix. The entries of $\vec{f}$ are given by

$$f^j = ((I-\Pi_\ell)Q_{\ell+1}u, \phi_\ell^j) = \tfrac{1}{4}h_\ell(\alpha_{\ell+1}^- + \alpha_{\ell+1}^+) \tag{9}$$

where the notation is given in Fig. 6. Finally, the system must be solved. M_ℓ is a tridiagonal symmetric positive definite matrix, so $\vec{\xi}$ can be determined in $O(\#\mathcal{N}_\ell)$ operations [13, Algorithm 4.3.6]. This method requires a workspace of size $O(\#\mathcal{N}_\ell)$, which can be allocated once and reused throughout the decomposition and recomposition procedures.

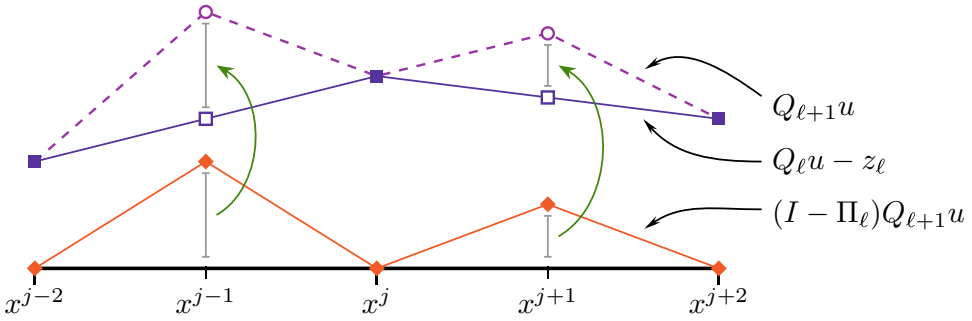

Fig. 5 Illustration of the recomposition of $Q_{\ell+1}u = (I-\Pi_\ell)Q_{\ell+1}u + (Q_\ell u - z_\ell)$. Here the nodes x^{j-2}, x^j, and x^{j+2} belong to $\mathcal{N}_\ell$ and the nodes x^{j-1} and x^{j+1} belong to $\mathcal{N}_{\ell+1}\backslash\mathcal{N}_\ell$. We are given the values of $Q_\ell u - z_\ell$ on the coarse nodes $\mathcal{N}_\ell$ and the values of $(I-\Pi_\ell)Q_{\ell+1}u$ on the nodes $\mathcal{N}_{\ell+1}\backslash\mathcal{N}_\ell$. Hence, the values of $Q_{\ell+1}u$ are obtained by adding the averages of the values stored on the neighboring coarse nodes to the values stored on $\mathcal{N}_{\ell+1}\backslash\mathcal{N}_\ell$

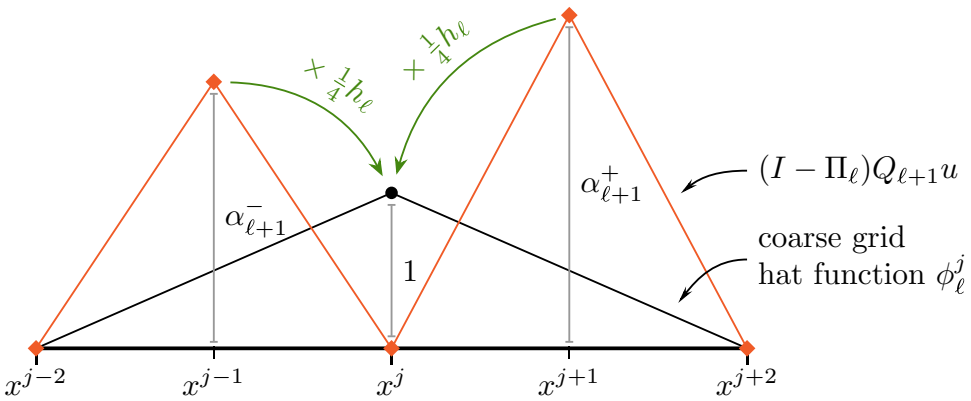

Fig. 6 Notation used in the evaluation of the right-hand side (9) used in the computation of the correction z_ℓ. Here x^{j-2}, x^j, and x^{j+2} belong to $\mathcal{N}_\ell$ and x^{j-1} and x^{j+1} belong to $\mathcal{N}_{\ell+1}\backslash\mathcal{N}_\ell$. The nodal values of $(I-\Pi_\ell)Q_{\ell+1}u$ on the neighboring nodes x^{j-1} and x^{j+1} are denoted by $\alpha^-_{\ell+1}$ and $\alpha^+_{\ell+1}$

3.4 Summary

The algorithms described above are all of optimal complexity and require little by way of additional storage. The only significant burden lies within the inversion of the mass matrix, which requires a local workspace of size $O(\#\mathcal{N}_\ell)$ on level ℓ. This necessary additional workspace can be allocated once and then used as a temporary workspace elsewhere in the implementation. In summary, we have shown:

Theorem 2 *The decomposition* (7) *requires the storage of* $\#\mathcal{N}_L$ *degrees of freedom and can be computed in* $O(\#\mathcal{N}_L)$ *operations. Moreover, the decomposition can be used to reconstruct* $Q_\ell u$ *where* $\ell \in \{0,\ldots,L\}$ *using* $O(\#\mathcal{N}_\ell)$ *operations.*

Proof The right-hand side $\vec{f}$ can be computed at a cost of $O(\#\mathcal{N}_\ell)$ operations using (9), while the cost of solving the resulting system of equations is also $O(\#\mathcal{N}_\ell)$. Hence, the cost of computing z_ℓ is $O(\#\mathcal{N}_\ell)$. The splitting (5) requires the construction of $\{z_\ell : 0 \le \ell < L\}$, which in total costs

$$O(\#\mathcal{N}_0) + O(\#\mathcal{N}_1) + \cdots + O(\#\mathcal{N}_{L-1}) = O(\#\mathcal{N}_L)$$

operations since $\#\mathcal{N}_\ell \simeq 2^\ell$. Equally well, the computation of $Q_M u$ from (7) also requires the computation of $\{z_\ell : 0 \le \ell < M\}$ at a cost of $O(\#\mathcal{N}_0)+\cdots+O(\#\mathcal{N}_{M-1}) = O(\#\mathcal{N}_M)$ operations. □

4 Application to reduction of turbulence data

In this section, we show how the hierarchical reduction scheme may be applied to a typical application where large datasets are generated. We consider the pseudo-spectral simulation of forced isotropic turbulence performed on an equispaced, periodic grid consisting of 1024^3 nodes over the box $[0, 2\pi]^3$ using a second-order Adams–Bashforth scheme and a Taylor-scale Reynolds number R_λ of approximately 433 [20,25]. The simulation was performed using a simulation timestep of size $\delta t = 2 \times 10^{-4}$ over 50,280 timesteps. Constraints on the amount of data that could be handled meant that the original simulation did not output the approximation at every timestep. Instead, decimation [1] was applied and the approximation was only output at every tenth timestep. The resulting dataset, comprising 5028 flow-field instances with a storage timestep of $\Delta t = 2 \times 10^{-3}$, is

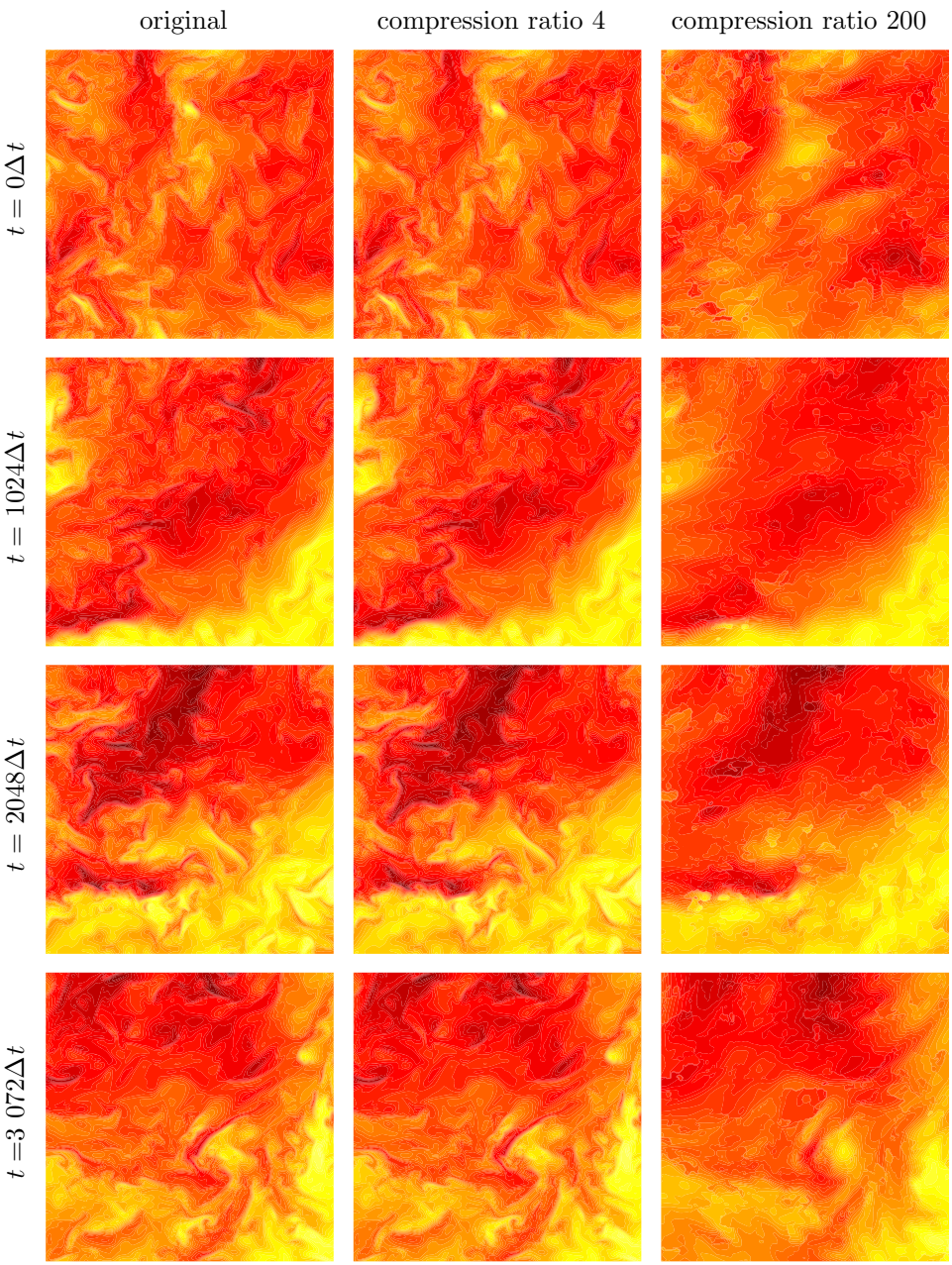

Fig. 7 Contours of the velocity component normal to the slice at various timesteps obtained using the original and reconstructed datasets with $\epsilon = 10^{-2}$ and $\epsilon = 2.5 \times 10^{-1}$. Compression ratios of 4 and 200 are obtained with $\epsilon = 10^{-2}$ and $\epsilon = 2.5 \times 10^{-1}$, respectively

in the public domain and available for download [17,20,25]. Even with decimation, the total amount of data stored from the run is 60 TB. We select a subset, the velocity field on a slice in the yz-plane consisting of 256^2 gridpoints at 4097 storage timesteps, approximately 3.3 GB in size for analysis.

Our objective is to apply the reduction algorithms described in Sect. 2 to this dataset. Using a variety of specified relative errors, we examine the relationship between error bound and compression ratio and the impact of the reduction on the results of post-processing and analysis typically applied to turbulence data. The usual quantities of interest when analyzing turbulence data are the fluctuations of the velocity around the mean, the time evolution of the kinetic energy, the energy spectrum, and two-point spatial and temporal correlations of velocity. It is also of interest to be able to make qualitative examinations of contour plots of the velocity.

The levels of accuracy required for the analysis of the quantities of interest are generally higher than the level needed to make a quick examination of the flow field at various timesteps. Moreover, the analyst would want a significant level of data reduction when performing visualisation of the velocity fields but would set a higher relative error threshold when carrying out quantitative analysis of the above quantities of interest. The hierarchical decomposition technique

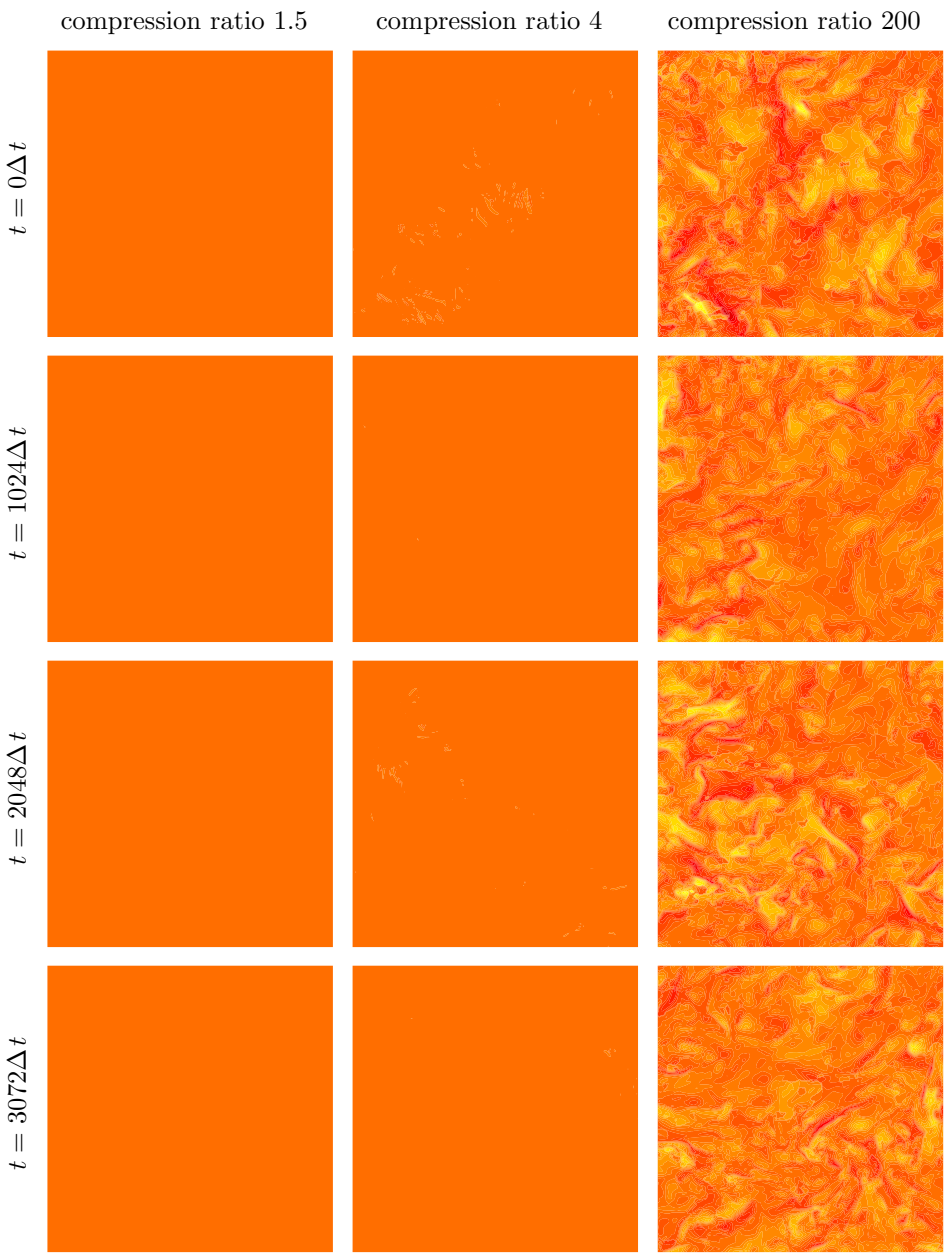

Fig. 8 Pointwise deviation of the velocity component normal to the slice at various timesteps obtained using reconstructed datasets with $\epsilon = 10^{-3}$, $\epsilon = 10^{-2}$, and $\epsilon = 2.5 \times 10^{-1}$. The compression ratios obtained are 1.5, 4, and 200, respectively

described in the previous sections provides the flexibility to accommodate these requirements.

We apply the univariate compression procedure by regarding the data as array-valued. That is, we take the dataset to be $\{\mathbf{u}(n\Delta t) : 0 \le n \le 4096\}$, where $\mathbf{u}(n\Delta t)$ is an array of the values of u on the spatial slice at timestep n. We generate a hierarchy of partitions $\{\mathcal{P}_\ell : 0 \le \ell \le 12\}$ of the time interval $[0, T]$ over which the simulation is sampled by repeatedly pruning half of the timesteps.

For a given tolerance $\epsilon > 0$, we wish to construct a reduced representation that has a relative error of at most ϵ measured in the norm defined by

$$\|\mathbf{u}\|^2 = \int_0^T |\mathbf{u}(t)|^2 \, \mathrm{d}t$$

where $|\mathbf{u}|$ denotes the Euclidean norm of the array-valued data (equivalent to the $L^2(\Omega)$ norm over the yz-spatial slice). Let M denote the number of nodes (here $M = 256^2$) over the spatial slice. Then we define an auxiliary tolerance $\delta = \epsilon/\sqrt{M}\|\mathbf{u}\| > 0$. For every node m in the yz-slice, we apply Algorithm 2 to the univariate data $\{u_m(n\Delta t) : 0 \le n \le 4096\}$ with the tolerance parameter δ to select a particular projection operator, which we denote by $Q^{(m)}$, for the reduced approximation at the point. This means that the pro-

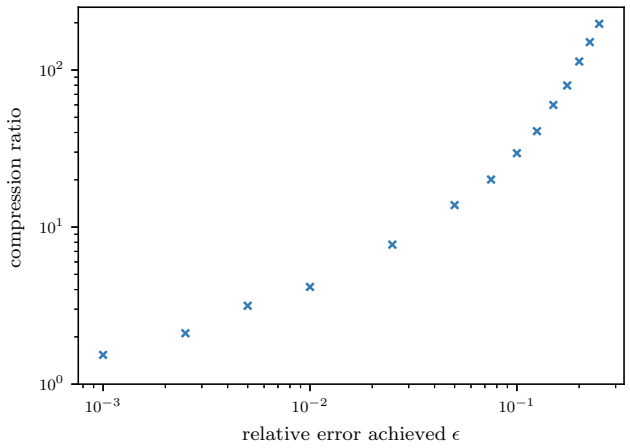

Fig. 9 Illustration of the compression ratios achieved using various levels of lossiness

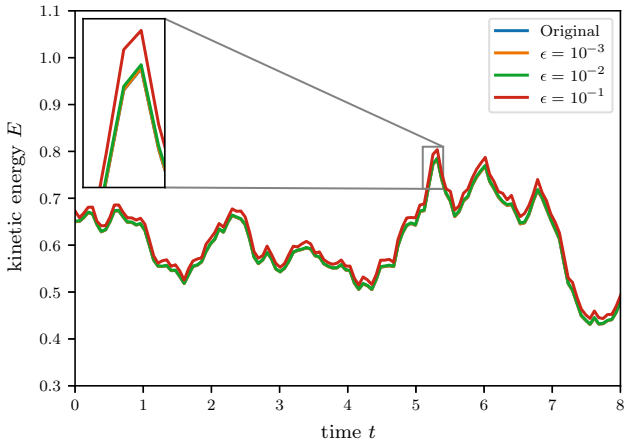

Fig. 10 Time histories of kinetic energy obtained using reduced representations at the error levels ϵ indicated

jection operator may be different at different nodes m, but in all cases we have

$$\|u_m - Q^{(m)}u_m\| \leq \delta.$$

It easily follows that

$$\|\mathbf{u} - \mathbf{Q}\mathbf{u}\|^2 \leq M\delta^2 = \epsilon^2\|\mathbf{u}\|^2$$

where $\mathbf{Q}$ is given by $Q^{(m)}$ at node m. In other words, the resulting reduced representation has relative error at most ϵ.

4.1 Visualizations and compression ratios

Firstly, we illustrate the performance of the reduction scheme when the primary interest is the visualisation of the velocity at a particular timestep. Figure 7 shows the contours of the component of the velocity normal to the spatial slice along with the contours obtained when the aforementioned reduction is performed with the choices $\epsilon = 10^{-2}$ and $\epsilon = 2.5 \times 10^{-1}$. The choice $\epsilon = 10^{-2}$ results in a 4-fold reduction in the data and the reduced and original flow fields are virtually indistinguishable to the naked eye, whilst the choice $\epsilon = 2.5 \times 10^{-1}$ results in a 200-fold reduction but still gives a faithful representation of the true flow pattern (Fig. 8).

Figure 9 presents the relationship between the error tolerance ϵ set and the number of degrees of freedom used. In particular, we give the values of the compression ratio, defined to be the ratio of the size of the original dataset to the size of the reduced representation. For instance, with a tolerance of $\epsilon = 10^{-1}$, the compressor is able to reduce the dataset size about 30-fold, requiring around 45s of compute time for the decomposition of the original 3.3 GB dataset on a single processor i7-3520m machine clocked at 2.9 GHz with 16 GB of RAM.

4.2 Kinetic energy

The time evolution of the kinetic energy is an important diagnostic for turbulence simulation since growth in kinetic energy is often the precursor to divergence of the numerical solver. The total kinetic energy is defined as

$$E_{\text{tot}} = \frac{1}{2}|\mathbf{u}|^2.$$

The average value of the kinetic energy is given [20] as $E_{\text{tot}} = 0.695$ for the present simulation. We wish to compare the time evolution of the kinetic energy computed using the reduced representation with the values obtained using the full dataset. Figure 10 shows that the time evolution of the kinetic energy is virtually identical to the true energy evolution when the value of the relative error is chosen to be $\epsilon = 10^{-2}$ and that the results obtained with $\epsilon = 10^{-1}$, while noticeably different from the true values, may still be acceptable for practical purposes.

4.3 Temporal power spectrum

The power spectrum is used to identify the distribution of the energy across different spatial and temporal scales of the flow. Energy accumulation in the smaller spatial scales is indicative of under-resolution in the numerical simulations. The temporal power spectrum is defined for $k \in \mathbb{N}$ by

$$E(k) = \langle \hat{u}^*(k)\hat{u}(k) \rangle,$$

where $\hat{u}$ is the Fourier transform of the velocity, $\hat{u}^*$ is its complex conjugate, and $\langle \cdot \rangle$ denotes the spatial average.

The temporal power spectrum of a velocity flow field measures the distribution of kinetic energy across different frequencies. For a purely random flow, the power spectrum would not vary with frequency, signalling a uniform distribu-

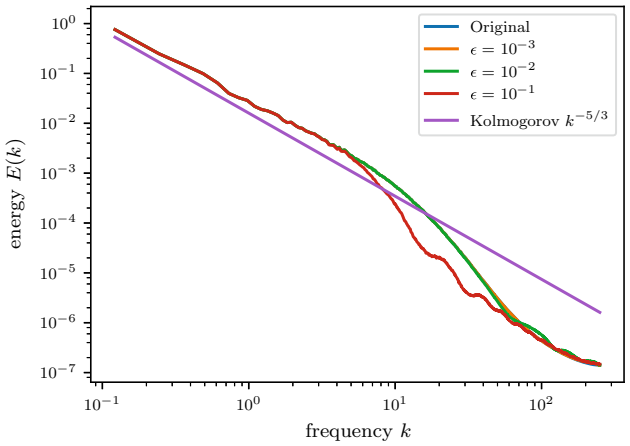

Fig. 11 Temporal power spectra obtained using reduced representations at the error levels ϵ indicated. For comparison, the line $k^{-5/3}$ is also plotted [18]

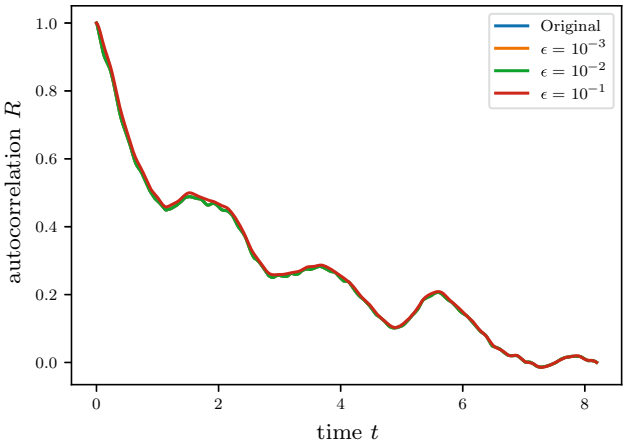

Fig. 12 Temporal autocorrelation of the component of velocity normal to the slice at a typical gridpoint obtained using reduced representations at the error levels ϵ indicated

tion of energy over different time scales. In fluid turbulence, by contrast, one expects to see a cascade of energies, with lower frequencies carrying most of the energy and higher frequency modes being less energetic.

Figure 11 presents the temporal power spectra for the original data and several reduced representations. At low levels of lossiness, the power spectrum obtained using the reduced representation is in good agreement with the true value. However, with $\epsilon = 10^{-1}$ the reduced spectrum differs noticeably from the original. The discrepancies occur chiefly in the mid-high frequencies; even with this large error, the reduced representation preserves the energy in the low and high frequency modes.

Overall, we see that using the reduced representation for the analyses does not introduce energy into the flow and does not interfere with the distribution of the low and high frequency ranges of the energy spectrum.

4.4 Temporal autocorrelation

The final quantity that we check is the autocorrelation of the velocities. This statistic allows us to ascertain whether any artificial correlation has been introduced into the velocity field by the reduction procedure. Temporal autocorrelation is defined to be

$$R(\tau) = \frac{\langle u(t)\, u(t+\tau)\rangle}{\langle u(t)^2\rangle}$$

where $\langle \cdot \rangle$ again denotes the spatial average. The autocorrelation function measures the dependence of the flow field at time $t + \tau$ on its state at t. In the case of turbulent flows, one expects that by increasing τ this function should tend to zero if the simulation is well-resolved. Figure 12 gives a sample plot of temporal correlation of axial velocity at a particular gridpoint. Even at high lossiness, the autocorrelation characteristics are preserved.

Overall, the results indicate that this univariate hierarchical reduction method can be effective even when applied to traditionally challenging multivariate applications such as turbulence.

5 Conclusion

The present work is the first step in developing a new approach to data reduction based on multilevel decomposition. The theory of multilevel decomposition is well-established —see for example [5,8,11,15,24] and the references therein—and it is widely used in the linear algebra community, underpinning the convergence analysis of multigrid methods and other related solvers. Here we are using that theory to develop data reduction algorithms instead. To the best of our knowledge, this is the first time that multilevel decomposition has been applied to data reduction.

The two most common multilevel decompositions are the hierarchical basis method [3] and the orthogonal decomposition. Here we select the orthogonal decomposition since the hierarchical basis is defined in terms of interpolants and so is not stable for Sobolev regularity below $d/2$ (where d is the spatial dimension of the domain over which the data are sampled). By contrast, the quasioptimality results given in Theorem 1 for the orthogonal decomposition hold for data in $H^r(\Omega)$ for any $r > 0$. The ability to work with data of low regularity is lost if a pure hierarchical basis approach is used.

The only problem with choosing the orthogonal basis decomposition is that it seems to require additional memory to store all of the projections $\{Q_\ell u : 0 \le \ell \le L\}$. In Sect. 3 we show how this can be avoided at the expense of computing certain $L^2([a, b])$ projections. We do not regard

these projections as cost prohibitive because, as shown in Theorem 2, they require only $O(\#\mathcal{N}_\ell)$ operations. Thus, the full hierarchy of projections may be efficiently computed and stored using no memory beyond that originally required for u.

Compression is achieved by truncating the representation at some level according to Algorithms 1 or 2. The resulting reduced representation is defined by its values at the nodes $\mathcal{N}_\ell$ of a grid $\mathcal{P}_\ell$ coarser than the original, full grid $\mathcal{P}_L$. It is a simple matter to generate the values at all of the nodes $\mathcal{N}_L$ by interpolating the values on $\mathcal{N}_\ell$ if required. That is, the grid resolution is not lost when using the multilevel technique presented in this work.

The final section, Sect. 4, is dedicated to an investigation of the performance of the reduction method on a turbulence dataset. We decompress the data with a range of error tolerances, carry out the usual analysis procedures on the reduced datasets, and compare the results to those obtained with the original data. The results show good agreement, demonstrating the promise of multilevel compression techniques for the reduction of data arising from large scale simulations of complex phenomena.

References

1. Ainsworth, M., Klasky, S., Whitney, B.: Compression using lossless decimation: analysis and application. SIAM J. Sci. Comput. **39**(4), B732–B757 (2017)
2. Austin, W., Ballard, G., Kolda, T. G.: Parallel tensor compression for large-scale scientific data. In: 2016 IEEE international parallel and distributed processing symposium (IPDPS), pp. 912–922, May 2016
3. Bank, R.E., Dupont, T.F., Yserentant, H.: The hierarchical basis multigrid method. Numer. Math. **52**(4), 427–458 (1988)
4. Bautista, G., Leonardo, A., Cappello, F.: Improving floating point compression through binary masks. In: 2013 IEEE international conference on big data, pp. 326–331, October 2013
5. Bornemann, F., Yserentant, H.: A basic norm equivalence for the theory of multilevel methods. Numer. Math. **64**(1), 455–476 (1993)
6. Burtscher, M., Hari, M., Annie, Y., Farbod, H.: 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, pp. 264–275, November 2016
7. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley Series in Telecommunications, 1st edn. Wiley, New York (1991)
8. Dahmen, W., Kunoth, A.: Multilevel preconditioning. Numer. Math. **63**(3), 315–344 (1992)
9. Daubechies, I.: The wavelet transform, time-frequency localization and signal analysis. IEEE Trans. Inf. Theory **36**(5), 961–1005 (1990)
10. Di, S., Cappello, F.: Fast error-bounded lossy HPC data compression with SZ. In: 2016 IEEE 30th international parallel and distributed processing symposium, IEEE, Chicago, IL, USA, pp. 730–739, May 2016
11. Donoho, D.L., Vetterli, M., DeVore, R.A., Daubechies, I.: Data compression and harmonic analysis. IEEE Trans. Inf. Theory **44**(6), 2435–2476 (1998)
12. Edmunds, D.E., Triebel, H.: Function Spaces, Entropy Numbers, Differential Operators, 1st edn. Cambridge University Press, Cambridge (1996)
13. Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd edn. The Johns Hopkins University Press, Baltimore (1996)
14. Grgic, S., Kers, K., Grgic, M.: Image compression using wavelets. In: Proceedings of the IEEE international symposium on industrial electronics, 1999. ISIE '99, vol. 1, pp. 99–104 (1999)
15. Griebel, M., Oswald, P.: Stable splittings of Hilbert spaces of functions of infinitely many variables. J. Complex. **41**, 126–151 (2017)
16. Ibarria, L., Lindstrom, P., Rossignac, J., Szymczak, A.: Out-of-core compression and decompression of large n-dimensional scalar fields. Comput. Graph. Forum **22**(3), 343–348 (2003)
17. Johns Hopkins Turbulence Databases. Forced isotropic turbulence dataset description, October 2017. Last update: 10/19/2017 5:55:14 PM. Accessed 01 Feb 2018
18. Kolmogorov, A.: The local structure of turbulence in incompressible viscous fluid for very large Reynolds' numbers. Akademiia Nauk SSSR Doklady **30**, 301–305 (1941)
19. Lakshminarasimhan, S., Shah, N., Ethier, S., Klasky, S., Latham, R., Ross, R., Samatova, N. F.: Compressing the incompressible with ISABELA: in-situ reduction of spatio-temporal data. In: Emmanuel J., Raymond N., Jean R. (eds) Euro-Par 2011: Parallel Processing Workshops, Lecture Notes in Computer Science, Bordeaux, France, Springer, Berlin, Heidelberg, vol. 6852, pp. 366–379, August 2011
20. Li, Y., Perlman, E., Wan, M., Yang, Y., Meneveau, C., Burns, R., Chen, S., Szalay, A., Eyink, G.: A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence. J. Turbul. **9**, N31 (2008)
21. Lindstrom, P.: Fixed-rate compressed floating-point arrays. IEEE Trans. Vis. Comput. Graph. **20**(12), 2674–2683 (2014)
22. Lindstrom, P., Isenburg, M.: Fast and efficient compression of floating-point data. IEEE Trans. Vis. Comput. Graph. **12**(5), 1245–1250 (2006)
23. Marcellin, M. W., Gormish, M. J., Bilgin, A., Boliek, M. P.: An overview of JPEG-2000. In: Proceedings DCC 2000. Data compression conference, pp. 523–541 (2000)
24. Oswald, P.: Multilevel Finite Element Approximation. Theory and Applications. Teubner Skripten zur Numerik. B. G. Teubner, Stuttgart (1994)
25. Perlman, E., Burns, R., Li, Y., Meneveau, C.: Data exploration of turbulence simulations using a database cluster. In: Proceedings of the 2007 ACM/IEEE conference on supercomputing, ACM, Reno, NV, USA, vol. 23, November 2007
26. Salomon, D.: Data Compression: The Complete Reference, 4th edn. Springer, London (2007)
27. Schendel, E. R., Jin, Y., Shah, N., Chen, J., Chang, C. S., Ku, S.-H., Ethier, S., Klasky, S., Latham, R., Ross, R., Samatova, N. F.: ISOBAR preconditioner for effective and high-throughput lossless data compression. In: 2012 IEEE 28th international conference on data engineering, pp. 138–149, April 2012
28. Schneider, K., Farge, M., Pellegrino, G., Rogers, M.M.: Coherent vertex simulation of three-dimensional turbulent mixing layers using orthogonal wavelets. J. Fluid Mech. **534**, 39–66 (2005)
29. Shah, N., Schendel, E. R., Lakshminarasimhan, S., Pendse, S. V., Rogers, T., Samatova, N. F.: Improving I/O throughput with PRIMACY: preconditioning ID-mapper for compressing incompressibility. In: 2012 IEEE international conference on cluster computing, pp. 209–219, September 2012
30. Strengert, M., Magallón, M., Weiskopf, D., Guthe, S., Ertl, T.: Hierarchical visualization and compression of large volume datasets using GPU clusters. EGPGV, pp. 41–48 (2004)
31. Wallace, G. K.: The JPEG still picture compression standard. IEEE Trans. Consum. Electron. **38**(1), xviii–xxxiv (1992)