Multilevel Techniques for Compression and Reduction of Scientific Data

Ben Whitney

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN THE DIVISION OF APPLIED MATHEMATICS AT BROWN UNIVERSITY

PROVIDENCE, RHODE ISLAND

2018-05-01

This dissertation by Ben Whitney is accepted in its present form by the Division of Applied Mathematics as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date: _____

_____
Mark Ainsworth
Advisor

Recommended to the Graduate Council

Date: _____

_____
Jérôme Darbon
Reader

Date: _____

_____
Scott Klasky
Reader

Approved by the Graduate Council

Date: _____

_____
Andrew G. Campbell
Dean of the Graduate School

Curriculum Vitae

Education

AB. Harvard University. Mathematics. 2009 to 2013.

ScM. Brown University. Applied Mathematics. 2013 to 2014.

Teaching

Community College of Rhode Island

Math 0500. Instructor. Summer 2014. John J. Moran Medium Security Facility.

Brown University

APMA 1650. Teaching assistant. Fall 2014.

APMA 0330. Teaching assistant. Spring 2015.

APMA 0330. Head teaching assistant. Fall 2015.

Publications

[AKW17] Mark Ainsworth, Scott Klasky, and Ben Whitney. Compression using lossless decimation: Analysis and application. *SIAM Journal on Scientific Computing*, 39(4):B732–B757, August 2017.

[ATWK17] Mark Ainsworth, Ozan Tugluk, Ben Whitney, and Scott Klasky. Multilevel techniques for compression and reduction of scientific data—the univariate case. *Computing and Visualization in Science*, June 2017. Accepted.

[ATWK18] Mark Ainsworth, Ozan Tugluk, Ben Whitney, and Scott Klasky. Multilevel techniques for compression and reduction of scientific data—the multivariate case. *Submitted*, January 2018.

[CCD+18] Jong Choi, Choong-Seock Chang, Julien Dominski, Scott Klasky, Gabriele Merlo, Eric Suchyta, Mark Ainsworth, Bryce Allen, Amitava Bhattacharjee, Franck Cappello, Michael Churchill, Philip Davis, Sheng Di, Greg Eisenhauer, Stephane Ethier, Ian Foster, Berk Geveci, Hanqi Guo, Kevin Huck, Frank Jenko, Mark Kim, James Kress, Seung-Hoe Ku, Qing Liu, Jeremy Logan, Allen Malony, Kshitij Mehta, Kenneth Moreland, Todd Munson, Manish Parashar, Tom Peterka, Norbert Podhorszki, Dave Pugmire, Ozan

Tugluk, Ruonan Wang, Ben Whitney, Matthew Wolf, and Chad Wood. Constructing multiphysics coupling for the exascale: Implementation and lessons learned. In *SC18: The International Conference for High Performance Computing, Networking, Storage, and Analysis*, Dallas, TX, 2018. Submitted.

[CHP+18] Jed Chou, Milena Hering, Sam Payne, Rebecca Tramel, and Ben Whitney. Diagonal splittings of toric varieties and unimodularity. *Proceedings of the American Mathematical Society*, 146(5):1911–1920, May 2018.

[KSA+17] Scott Klasky, Eric Suchyta, Mark Ainsworth, Qing Liu, Ben Whitney, Matthew Wolf, Jong Choi, Ian Foster, Mark Kim, Jeremy Logan, Kshitij Mehta, Todd Munson, George Ostrouchov, Manish Parashar, Norbert Podhorszki, David Pugmire, and Lipeng Wan. Exacution: Enhancing scientific data management for exascale. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 1927–1937, Atlanta, GA, USA, July 2017. IEEE.

Acknowledgments

I thank the people who made this work possible, especially my many teachers, chief among them Mark Ainsworth, and those who came before me. This work was partially supported by the DOE Storage Systems and Input/Output for Extreme Scale Science project, announcement number LAB 15-1338.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Computational simulation pervades science and engineering, enabling new insights about complex physical phenomena, and the advent of exascale systems [Thi14] is expected to drive the practice to an even greater scale. Effective utilization of exascale systems will pose new challenges, requiring algorithms with improved resiliency and fault tolerance and, perhaps more importantly, methods to cope with the vastly increased amounts of data generated by simulations. Today's petascale 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 example, present day fusion simulations typically produce around 100 PB per run [Cha17]. President Obama's executive order mandating the creation of a National Strategic Computing Initiative [Oba15] will only increase the amount of data being produced. Disk performance is an attractive target for optimizations, but nearterm improvements are expected to be too modest to alleviate the problem. The next decade's exascale machines are forecast to be able to write 10^{12} bytes to disk per second, a rate twice what is currently possible but corresponding to a compute speed–memory speed discrepancy 50 times greater than the present day's [FAA+17]. The various subsystems likely to be used on exascale machines are improving at uneven rates, causing a widening gulf between FLOPS performance and memory speed (as measured by memory bandwidth, memory latency, and interconnect performance) [McC16], as illustrated in Figure 1.1. The increasing computational power of modern machines therefore creates significant difficulties that must be addressed if these machines are to be used to their full potential. Meeting these challenges will require advances in hardware and software [Now14], including the development and analysis of numerical algorithms that are tailored to handling data produced by scientific simulation.

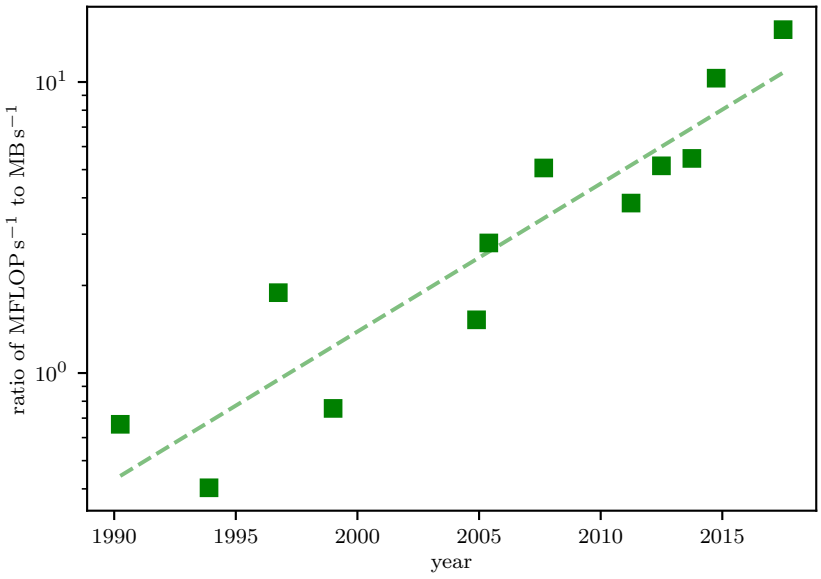

Figure 1.1: Illustration of the ratio of computational speed (measured in MFLOP s^{-1}) to memory bandwidth (measured in MB s^{-1}) for a selection of representative systems. The slope of the regression line corresponds to a 3.2× increase every decade. Memory bandwidth is measured using the STREAM benchmark [McC95]. The figure was reproduced from [McC16] with help from John McCalpin [McC18].

Enter data compression and reduction methods, which trade additional computational expense for memory savings. Compression and reduction are well-studied subjects across a wide range of disciplines and data types [CT91, Sal07]. Most familiar to the average person are probably the various codecs used to compress audiovisual data, among them JPEG [Wal92], MP3 [Bra99], and MP4 [Ric04]. Unfortunately, the success of these and other, general-purpose methods [Sew10, GA13a] on their respective domains does not automatically transfer to scientific data. 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. Scientific datasets, which are stored in high-precision formats and which exhibit structure closely tied to the phenomena they describe, require algorithms that account for these features. Many specialized techniques have been developed, including methods based on polynomial interpolation and extrapolation [ILRS03, DC16, LI06], Fourier transforms paired with coefficient quantization [Wal92, Lin14], statistical methods [BGC13, BMYH16], tensor-based methods [ABK16], hierarchical and wavelet-based methods [Dau90, GKG99, SMW$^+$04, MGBB00, BDY88, BY93, Osw94], and many more [ABK16, BGC13, BMYH16, LSE$^+$11, SJS$^+$12, SSL$^+$12]. Alternative approaches attempt to preprocess datasets so as to make them more amenable to compression [LSE$^+$11, SSL$^+$12, SJS$^+$12].

Another broad class of compression algorithms are *predictive coding* techniques, comprising algorithms that use past data values to generate predictions for future data values and encode the resulting residuals. Predictive coding has seen wide use in the data compression literature. For example, differential pulse-code modulation (DPCM), an early patented signal compression technique, consists of simply storing first (or higher) order differences in place of the original data [Cut52]. DPCM was subsequently modified for use with nonuniformly sampled signals by replacing simple differences with prediction errors arising from extrapolating polynomials [Eng00]. Predictive coding schemes based on polynomial extrapolation or interpolation have been applied in a range of areas, including mesh compression [TG00, TR98, ILS05b], volume compression [ILRS03, FY94], audio compression [Rob94, Coa14], image compression [WSS00, Roe03, JKHM93], and floating point compression [LI06, DC16]. Predictive coding based on statistical techniques was used in [BGC13], while hash functions were used in [RKB06, BR07, BR10]. Despite the wealth of predictive techniques that have been developed, there has been relatively little in the way of mathematical analysis of the algorithms' performance, which has in general been evaluated using *ad hoc* practical experiments instead. In Chapter 2 we conduct such an analysis on an archetypal predictive coding technique we call *lossless decimation*. Results on the structure of the residuals in the cases of deterministic and random data lead to bounds on the compression ratio achieved by the technique. We then compare these bounds with results obtained using a variety of standard compressors on a sample of representative datasets.

Lossless decimation splits its input into two components: the decimated data, consisting of every tenth (for example) value of the original, and the residuals, which encode the deviation of the data from piecewise linearity. Thus the technique can be viewed as a two-level method, with the first level (the decimated data) providing an initial approximation of the data and the second level (the residuals) correcting the errors in the first. Chapters 3 to 5 expand on this idea with the development of a suite of multilevel methods. Beyond their nice mathematical properties, these methods merit study because of a correspondence between the structure of the reduced output and the computational resources available to users of modern systems. The storage hardware of a typical cluster consists of a heterogeneous collection of devices. Fastest but shortlived 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 are NVRAM, solid state drives, hard disk drives, optical media, network storage, etc. Multilevel and hierarchical reduction methods decompose input data into a sequence of components tailored to these types of heterogeneous storage media. Such methods are especially appropriate when the applications for

which the data are to be used require varying levels of resolution. For instance, one user might require that the reduced dataset meet a prescribed error tolerance for the calculation of some derived quantity, while another might need it to fit in the available RAM for a responsive visualization. Chapter 3 presents univariate algorithms designed for these two use cases of constrained loss and constrained storage. Chapter 4 extends to the multivariate setting and introduces a loss estimator that permits the control of pointwise errors. Chapter 5 concerns reduction while limiting distortion in quantities of interest using a related loss estimator.

In Chapters 3 to 5 we will distinguish between *compression*, defined to be a decrease in filesize, and *reduction*, defined to be a reduction in the number of nonzero degrees of freedom required to represent a dataset. In the absence of a method of efficiently storing zero degrees of freedom, reduction does not guarantee any compression. Compression is achieved in Chapter 2 by encoding of the interpolation residuals and in the implementation of `MGARD` [ATWK17, ATWK18], the technique presented in Chapters 3 to 5, in ADIOS [LKS+08, LLT+14, PLL+16] by quantization of the multilevel coefficients. In Chapter 2, where the distinction is less relevant, we use the two terms interchangeably. Unless defined otherwise, $\|\cdot\|$ will denote the $L^2(\Omega)$ norm.

Chapter 2

Lossless Decimation

2.1 Introduction

Consider data $\{u_j : 0 \leq j \leq N\}$ produced by a computational simulation of some system at a sequence of times $\{t_j : 0 \leq j \leq N\}$. Each datum u_j may be, in the simplest case, a scalar, or, at the opposite extreme, an array containing every state variable at every point of a spatial grid. A crude but commonly used compression technique consists of simply retaining every sth datum (with a value of $s = 10$ generally the default) and discarding the remainder. This process, known as *decimation*, is described in Algorithm 2.1. Should the value of a discarded datum u_j be required following decimation, it is tacitly assumed that an approximation will be generated by simple linear interpolation of the two nearest retained values, as in Algorithm 2.2. See Figure 2.1.

Require: s divides N.
  function DECIMATE(uncompressed data `data`$[0, \ldots, N]$, stride s)
    `decimated` $\leftarrow$ `[]`
    for $m = 0, \ldots, N/s$ **do**
      `decimated`$[m]$ $\leftarrow$ `data`$[ms]$
    end for
    return `decimated`
  end function

Algorithm 2.1: Decimation. Every sth datum is retained, and the remainder are simply discarded. As the procedure's name suggests, s is commonly taken to be 10.

Decimation has the obvious benefit of guaranteed data reduction by a factor of 10 (s, in general), but it is inherently lossy. While judicious use of lossy procedures can have practical benefit, in the

This chapter was previously published as [AKW17].

function APPROXIMATE(decimated data `decimated`$[0, \ldots, N/s]$, stride s)
    `surrogate` $\leftarrow$ `[]`
    for $m = 0, \ldots, N/s$ **do**
        `surrogate`$[ms]$ $\leftarrow$ `decimated`$[m]$
    end for
    for $m = 0, \ldots, N/s - 1$ **do**
        for $i = 1, \ldots, s - 1$ **do**
            `surrogate`$[ms + i]$ $\leftarrow ((s - i) *$ `decimated`$[m] + i *$ `decimated`$[m + 1])/s$
        end for
    end for
    return `surrogate`
end function

Algorithm 2.2: Approximation via linear interpolation. Each discarded value is approximated by a weighted average of the nearest retained values.

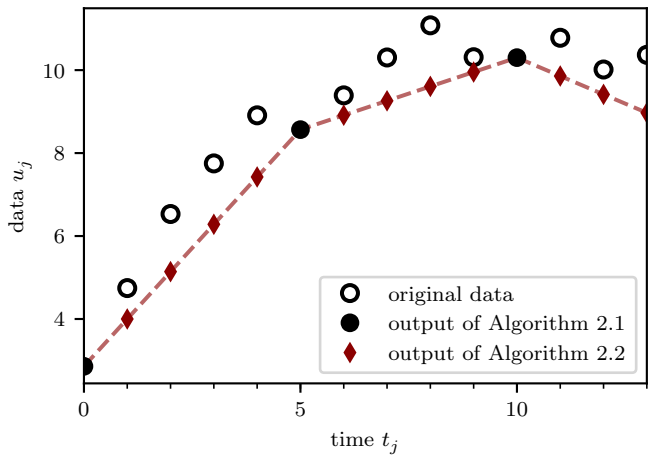

Figure 2.1: Illustration of the use of Algorithms 2.1 and 2.2 with stride $s = 5$. The original data ○ are reduced to the decimated data ●, giving a compression ratio of s. The discarded values are approximated by the interpolations ♦.

case of decimation data are simply discarded, with complete disregard of the errors arising from using linear interpolation to approximate the discarded values. While most practitioners are well aware of these shortcomings, the pressing need to compress the data means that this technique and related approaches are in widespread use nevertheless. If quizzed, the same practitioners might argue that the approximate values are often an acceptable surrogate for the discarded values.

How is it possible for a practitioner to, on the one hand, blindly discard 90% of the data while, on the other hand, claiming that the surrogate data are acceptable? The key lies in distinguishing between *data* and *information*. The data $\{u_j : 0 \leq j \leq N\}$ embody information on the system being simulated. They could, for example, be the values of an underlying function u sampled at timesteps $\{t_j : 0 \leq j \leq N\}$. The data generally require an enormous amount of storage irrespective of the nature of the function u. It may, however, be possible to store u itself in relatively little space. For instance, if u is smooth (or simple, or easily predictable, etc.), there will exist an alternative representation of u which can be communicated by, or stored with, only a small number of parameters. This is a classic case where the amount of data is large but the underlying information content is small. The key to effective data compression lies in extracting the information needed to represent the underlying function from the large volume of data containing it.

Suppose that decimation is applied to a data stream representing a smooth function u. The smoothness of u implies that linear interpolation can be expected to produce approximations that are in some sense close to the discarded values. In effect, the decimation procedure is seeking, albeit crudely, to extract a subset of the full dataset which captures the underlying information content. Viewing it in this way, one can begin to see how an *ad hoc* procedure such as decimation may be of practical value.

This heuristic explanation of how decimation can function in principle is instructive, but it invites more questions than it answers. For instance, in what circumstances does decimation used in conjunction with linear interpolation really result in little information loss? In these circumstances, the discrepancies between the actual values and those predicted via interpolation should, despite constituting a large dataset, carry little information. If so, is it possible to store these discrepancies, or *residuals*, with a modest amount of extra storage? One might then supplement the decimated data with the encoded residuals, allowing for lossless reproduction of the discarded values. We will refer to the resulting technique as *lossless decimation*.

The main objective in the present chapter is to fill a gap in HPC data compression by attempting to develop an approach to quantifying expected compression ratios. To this end, we first estimate the accuracy achieved by linear interpolation when approximating values discarded by decimation,

obtaining both

1. deterministic bounds in terms of appropriate smoothness measures of the data and

2. probabilistic bounds in terms of statistics of the data.

Second, we investigate the effectiveness of the lossless decimation scheme as a compressor. In particular, we bound the expected compression ratio in terms of the appropriate measures of the data. Finally, we provide numerical illustrations of the theoretical bounds as well as the practical performance of the algorithm on some datasets.

The arguments here are asymptotic in nature in that we assume that the dataset size N approaches infinity. Moreover, we do not attempt to use any *a priori* knowledge about the nature of the data beyond basic statistics. In practice, effective data compression algorithms often attempt to incorporate additional information on the provenance of the data. Our main focus is on deriving bounds on the compression ratio rather than proposing a particular predictor or method for encoding the approximation errors. Given the widespread use of predictive coding techniques, it is perhaps rather surprising that there is little by way of theoretical analysis of the expected performances of these methods, even in the simple cases we consider. We hope that the present chapter will pave the way for further work in this direction.

2.2 Deterministic Bounds on Approximation Errors

Decimation in effect replaces the original dataset $\{u_j : 0 \leq j \leq N\}$ by a surrogate dataset $\{\tilde{u}_j : 0 \leq j \leq N\}$ where

$$\tilde{u}_j = \begin{cases} u_{ms} & j = ms \\ (1 - \frac{i}{s})u_{ms} + \frac{i}{s}u_{ms+s} & j = ms + i. \end{cases}$$

This introduces an approximation error $\Delta_j = u_j - \tilde{u}_j$ for each $j \in \{0, \ldots, N\}$. The magnitude of the approximation errors is of considerable practical interest and will, in general, depend on both the stride size s and the nature of the data. Figure 2.2 illustrates the variation in the errors with the stride s for a particular dataset. It is observed that the relation between the errors and the stride size is rather nontrivial. In this section we seek an explanation for this behavior.

Our first result relates the approximation errors to derived quantities known as the nth order differences. The *first order differences* $\{\delta_j^1 : 0 \leq j \leq N-1\}$ are defined by $\delta_j^1 = u_{j+1} - u_j$, and the *second order differences* $\{\delta_j^2 : 1 \leq j \leq N-1\}$ are defined by $\delta_j^2 = \delta_j^1 - \delta_{j-1}^1$ [Ral01]. We

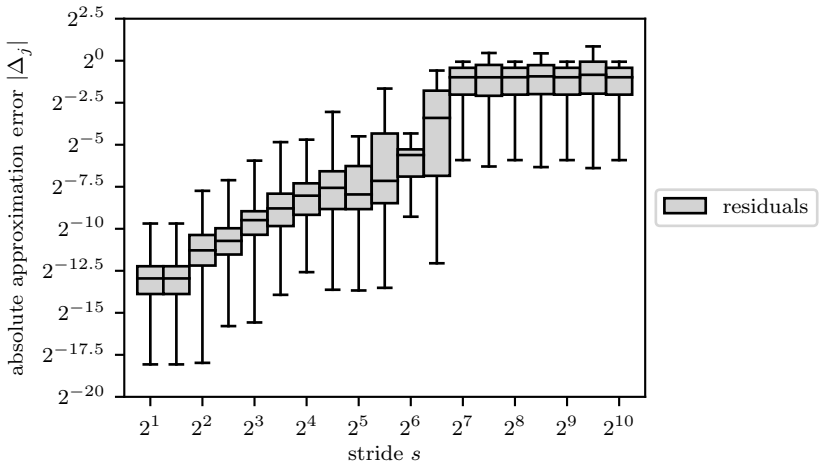

Figure 2.2: Illustration of the effect of stride size s on approximation errors. As one may expect, larger strides result in larger errors, but the relationship is nontrivial. There appear to be three regimes, with transitions between them occurring near $s = 2^4$ and $s = 2^7$. The data used for this plot are values taken by a truncated sawtooth wave on a uniform grid. For each stride s we decimate the data with factor s, interpolate, and calculate the residuals $\{\Delta_j : 0 \le j \le N\}$. The magnitudes of the errors are displayed in box plots with the top whiskers marking the maxima and the bottom whiskers marking the first percentiles. Outliers below the first percentiles are omitted. Figures 2.3 to 2.6 are generated in the same fashion.

shall on occasion refer to $\{u_j : 0 \le j \le N\}$ as the *zeroth order differences*. The relationships between the differences and the approximation errors are most conveniently stated in matrix–vector notation. We define processes $\boldsymbol{u}\colon \{0, \dots, N/s - 1\} \to \mathbb{R}^{s+1}$, $\boldsymbol{\delta}^1\colon \{0, \dots, N/s - 1\} \to \mathbb{R}^s$, $\boldsymbol{\delta}^2\colon \{0, \dots, N/s - 1\} \to \mathbb{R}^{s-1}$, and $\boldsymbol{\Delta}\colon \{0, \dots, N/s - 1\} \to \mathbb{R}^{s-1}$ as follows.

$$\boldsymbol{u}_m = (u_{ms}, \dots, u_{ms+s}) \qquad \boldsymbol{\delta}^1_m = (\delta^1_{ms}, \dots, \delta^1_{ms+s-1})$$
$$\boldsymbol{\Delta}_m = (\Delta_{ms+1}, \dots, \Delta_{ms+s-1}) \qquad \boldsymbol{\delta}^2_m = (\delta^2_{ms+1}, \dots, \delta^2_{ms+s-1})$$

These processes simply group scalars into vectors.

Lemma 2.1. *The approximation errors are related to the zeroth, first, and second order differences as follows:*

$$\boldsymbol{\Delta}_m = \Psi^0 \boldsymbol{u}_m = \Psi^1 \boldsymbol{\delta}^1_m = \Psi^2 \boldsymbol{\delta}^2_m$$

where Ψ^0 *is the* $(s-1)\times(s+1)$ *matrix given by*

$$\Psi^0_{i,j} = \begin{cases} -(1-\frac{i}{s}) & j=0 \\ -\frac{i}{s} & j=s \\ \delta_{ij} & j\notin\{0,s\} \end{cases} \tag{2.1a}$$

for $i\in\{1,\dots,s-1\}$ *and* $j\in\{0,\dots,s\}$, Ψ^1 *is the* $(s-1)\times s$ *matrix given by*

$$\Psi^1_{i,j} = \begin{cases} 1-\frac{i}{s} & j<i \\ -\frac{i}{s} & j\geq i \end{cases} \tag{2.1b}$$

for $i\in\{1,\dots,s-1\}$ *and* $j\in\{0,\dots,s-1\}$, *and* Ψ^2 *is the* $(s-1)\times(s-1)$ *matrix given by*

$$\Psi^2_{i,j} = \begin{cases} -s(1-\frac{i}{s})\frac{j}{s} & j\leq i \\ -s(1-\frac{j}{s})\frac{i}{s} & j\geq i \end{cases} \tag{2.1c}$$

for $i\in\{1,\dots,s-1\}$ *and* $j\in\{1,\dots,s-1\}$.

Proof. Let $m\in\{0,\dots,N/s-1\}$ and $i\in\{1,\dots,s-1\}$. Then

$$\begin{aligned}\Delta_{ms+i} &= u_{ms+i} - \tilde{u}_{ms+i} = u_{ms+i} - [(1-\tfrac{i}{s})u_{ms} + \tfrac{i}{s}u_{ms+s}] \\ &= -(1-\tfrac{i}{s})u_{ms} + u_{ms+i} - \tfrac{i}{s}u_{ms+s} = \sum_{j=0}^{s}\Psi^0_{i,j}u_{ms+j}.\end{aligned}$$

Thus, $\boldsymbol{\Delta}_m = \Psi^0\boldsymbol{u}_m$. Next, we seek to express $\boldsymbol{\Delta}_m$ in terms of $\boldsymbol{\delta}^1_m$. We can write each approximation error as a linear combination of differences of the data:

$$\begin{aligned}\Delta_{ms+i} &= u_{ms+i} - [(1-\tfrac{i}{s})u_{ms} + \tfrac{i}{s}u_{ms+s}] \\ &= (1-\tfrac{i}{s})(u_{ms+i}-u_{ms}) - \tfrac{i}{s}(u_{ms+s}-u_{ms+i}).\end{aligned} \tag{2.2}$$

$u_{ms+i}-u_{ms}$ may be written as the sum of first order differences $\sum_{j=0}^{i-1}\delta^1_{ms+j}$, and similarly $u_{ms+s}-u_{ms+i}$ is equal to $\sum_{j=i}^{s-1}\delta^1_{ms+j}$. Substituting into (2.2), we find that

$$\Delta_{ms+i} = (1-\tfrac{i}{s})\sum_{j=0}^{i-1}\delta^1_{ms+j} - \tfrac{i}{s}\sum_{j=i}^{s-1}\delta^1_{ms+j} = \sum_{j=0}^{s-1}\Psi^1_{i,j}\delta^1_{ms+j}. \tag{2.3}$$

Thus, $\boldsymbol{\Delta}_m = \Psi^1 \boldsymbol{\delta}^1_m$. Finally, we seek to express $\boldsymbol{\Delta}_m$ in terms of $\boldsymbol{\delta}^2_m$. This can be achieved by writing each δ^1_{ms+j} in (2.3) as the sum of δ^1_{ms+i} and second order differences. For $j < i$, $\delta^1_{ms+j} = \delta^1_{ms+i} - \sum_{k=j+1}^{i} \delta^2_{ms+k}$; for $j > i$, $\delta^1_{ms+j} = \delta^1_{ms+i} + \sum_{k=i+1}^{j} \delta^2_{ms+k}$. Substituting into (2.3), we find that

$$\begin{aligned}\Delta_{ms+i} &= (1-\tfrac{i}{s}) \sum_{j=0}^{i-1} \Big(\delta^1_{ms+i} - \sum_{k=j+1}^{i} \delta^2_{ms+k} \Big) - \tfrac{i}{s} \sum_{j=i}^{s-1} \Big(\delta^1_{ms+i} + \sum_{k=i+1}^{j} \delta^2_{ms+k} \Big) \\ &= (1-\tfrac{i}{s}) \Big(\sum_{j=0}^{i-1} \delta^1_{ms+i} - \sum_{k=1}^{i} \sum_{j=0}^{k-1} \delta^2_{ms+k} \Big) - \tfrac{i}{s} \Big(\sum_{j=i}^{s-1} \delta^1_{ms+i} + \sum_{k=i+1}^{s-1} \sum_{j=k}^{s-1} \delta^2_{ms+k} \Big).\end{aligned}$$

The δ^1_{ms+i} terms cancel, leaving

$$\Delta_{ms+i} = -(1-\tfrac{i}{s}) \sum_{k=1}^{i} k \delta^2_{ms+k} - \tfrac{i}{s} \sum_{k=i+1}^{s-1} (s-k) \delta^2_{ms+k} = \sum_{k=1}^{s-1} \Psi^2_{i,k} \delta^2_{ms+k}.$$

Thus, $\boldsymbol{\Delta}_m = \Psi^2 \boldsymbol{\delta}^2_m$. □

The results in Lemma 2.1 cannot be extended to third (or higher) order differences since if the data are quadratic the approximation errors are nonzero whereas the third (and higher) order differences vanish.

One consequence of Lemma 2.1 is that *a priori* information on the magnitudes of the differences can be translated into *a priori* bounds on the magnitudes of the approximation errors.

Theorem 2.2. *Let $m \in \{0, \ldots, N/s - 1\}$ and $i \in \{1, \ldots, s-1\}$. $|\Delta_{ms+i}|$ can be bounded as follows.*

$$|\Delta_{ms+i}| \le 2\mathfrak{M}^0 \qquad \textit{where} \quad \mathfrak{M}^0 = \max_{0 \le j \le N} |u_j| \tag{2.4a}$$

$$|\Delta_{ms+i}| \le \tfrac{1}{2} s \mathfrak{M}^1 \qquad \textit{where} \quad \mathfrak{M}^1 = \max_{0 \le j \le N-1} |\delta^1_j| \tag{2.4b}$$

$$|\Delta_{ms+i}| \le \tfrac{1}{8} s^2 \mathfrak{M}^2 \qquad \textit{where} \quad \mathfrak{M}^2 = \max_{1 \le j \le N-1} |\delta^2_j| \tag{2.4c}$$

Moreover, the constant factors in these bounds are optimal.

Proof. $|\Delta_{ms+i}| \le \|\boldsymbol{\Delta}_m\|_\infty$, so it suffices to show that $\|\boldsymbol{\Delta}_m\|_\infty$ is bounded above by $2\mathfrak{M}^0$, $\frac{1}{2} s \mathfrak{M}^1$, and $\frac{1}{8} s^2 \mathfrak{M}^2$. $\boldsymbol{\Delta}_m = \Psi^0 \boldsymbol{u}_m$ and by definition $\|\boldsymbol{u}_m\|_\infty \le \mathfrak{M}^0$, so for Upper Bound (2.4a) it suffices to show that $\|\Psi^0\|_\infty \le 2$. Similarly, for Upper Bound (2.4b) it suffices to show that $\|\Psi^1\|_\infty \le s/2$, and for Upper Bound (2.4c) it suffices to show that $\|\Psi^2\|_\infty \le s^2/8$. The matrix norm subordinate to the maximum vector norm is computed by finding the maximum of the absolute row sums [HJ90].

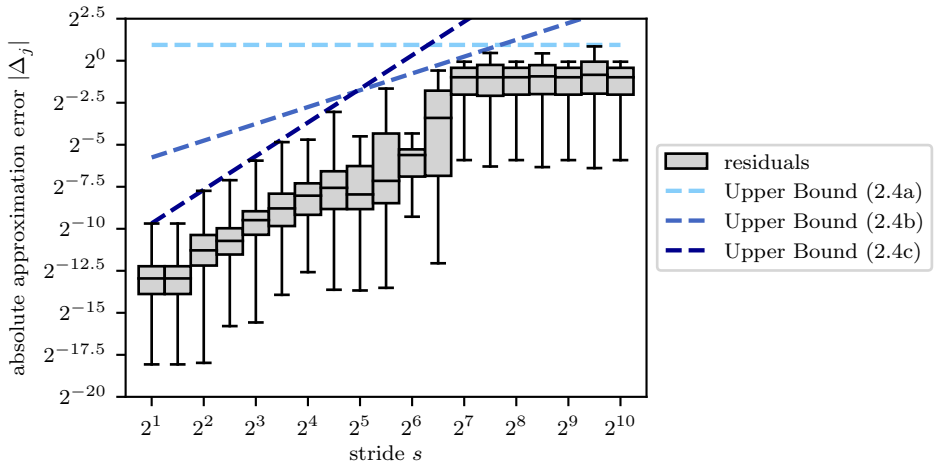

Figure 2.3: Illustration of the bounds of Theorem 2.2. The dataset used is the same as in Figure 2.2.

Consequently,

$$\|\Psi^0\|_\infty = \max_{1\le i\le s-1} (1-\tfrac{i}{s}) + 1 + \tfrac{i}{s} = 2,$$

$$\|\Psi^1\|_\infty = \max_{1\le i\le s-1} (1-\tfrac{i}{s})i + \tfrac{i}{s}(s-i) = \max_{1\le i\le s-1} 2s(1-\tfrac{i}{s})\tfrac{i}{s} \le \tfrac{s}{2}, \quad \text{and}$$

$$\|\Psi^2\|_\infty = \max_{1\le i\le s-1} \sum_{j=1}^{i} s(1-\tfrac{i}{s})\tfrac{j}{s} + \sum_{j=i+1}^{s-1} s(1-\tfrac{j}{s})\tfrac{i}{s} = \max_{1\le i\le s-1} \tfrac{s^2}{2}(1-\tfrac{i}{s})\tfrac{i}{s} \le \tfrac{s^2}{8}.$$

The bounds Upper Bounds (2.4a) to (2.4c) follow. For optimality, it suffices to find datasets for which each bound is achieved. For data given by $u_j = (-1)^j$, Upper Bound (2.4a) is tight; for data given by $u_j = |j - s/2|$, Upper Bound (2.4b) is tight; and for data given by $u_j = j^2$, Upper Bound (2.4c) is tight. □

Are the inequalities in Theorem 2.2 consistent with the behavior observed in Figure 2.2? In Figure 2.3 we plot these bounds along with the actual residuals. While no single one of the bounds Upper Bounds (2.4a) to (2.4c) is tight for all strides used, their minimum is sharp over the full range.

Example 2.1. Theorem 2.2 is well-suited to the case where $\{u_j : 0 \le j \le N\}$ are values taken by some deterministic function at uniformly spaced times $\{jh : 0 \le j \le N\}$. Suppose $u_j = f(jh)$, with $f \in C([0, Nh])$. $\mathfrak{M}^0 \le \|f\|_\infty$, so Upper Bound (2.4a) becomes

$$|\Delta_{ms+i}| \le 2\|f\|_\infty. \tag{2.5a}$$

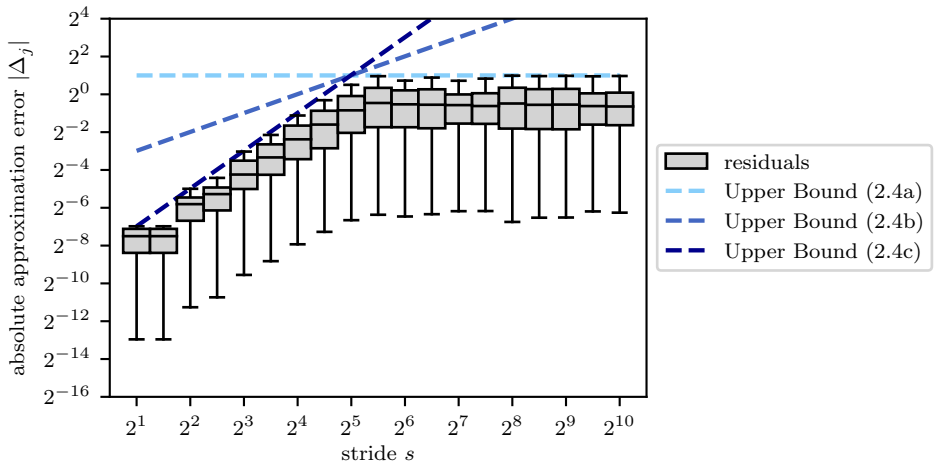

Figure 2.4: Illustration of the bounds of Theorem 2.2 when $u_j = \sin(jh)$ with $h = 2^{-3}$ and $j \in \{0, \dots, \lfloor 2^{12}\pi/h \rfloor\}$. Upper Bound (2.4c) is the tightest until sh is about half the period of sin. This matches the behavior predicted by the bounds Upper Bounds (2.5a) to (2.5c): with $\|\sin\|_\infty = \|\sin'\|_\infty = \|\sin''\|_\infty$, Upper Bound (2.4c) will be tightest whenever sh is less than 2^2.

We can make use of any additional smoothness of f by relating the differences of $\{u_j : 0 \le j \le N\}$ to the derivatives of f. If $f \in C^1([0, Nh])$, then $|\delta_j^1| = |\int_{jh}^{jh+h} f'(x)\,\mathrm{d}x| \le h\|f'\|_\infty$, and Upper Bound (2.4b) becomes

$$|\Delta_{ms+i}| \le \tfrac{1}{2} sh \|f'\|_\infty. \tag{2.5b}$$

If $f \in C^2([0, Nh])$, then

$$\begin{aligned}\delta_j^2 &= (u_{j+1} - u_j) - (u_j - u_{j-1}) = \int_{jh}^{jh+h} f'(x)\,\mathrm{d}x - \int_{jh-h}^{jh} f'(x)\,\mathrm{d}x \\ &= \int_0^h f'(jh+y) - f'(jh-y)\,\mathrm{d}y = \int_0^h \int_{-y}^{y} f''(jh+z)\,\mathrm{d}z\,\mathrm{d}y.\end{aligned}$$

Taking absolute values and integrating, we find that $|\delta_j^2| \le h^2\|f''\|_\infty$, and Upper Bound (2.4c) becomes

$$|\Delta_{ms+i}| \le \tfrac{1}{8}(sh)^2 \|f''\|_\infty, \tag{2.5c}$$

which is the well-known estimate for piecewise linear interpolation [Ral01, p. 54–55].

See Figure 2.4 for an illustration of the bounds in the case of sinusoidal data.

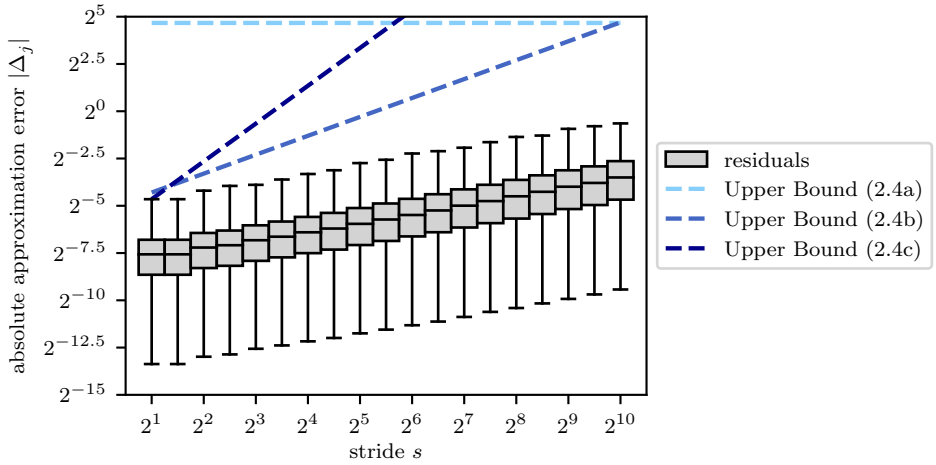

Figure 2.5: Illustration of the bounds of Theorem 2.2 with $\{u_j : 0 \le j \le N\}$ the positions of a Brownian motion sample path. The bounds fail to reflect the typical magnitudes of the approximation errors.

2.3 Statistical Bounds on Approximation Errors

In the previous section we presented bounds for approximation errors and applied them to deterministic, smooth data. How do these bounds fare when applied to nondeterministic data?

Example 2.2. Let $\{u_j : 0 \le j \le N\}$ be the positions of a Brownian motion sample path at times $\{jh : 0 \le j \le N\}$ with $h = 2^{-13}$ and $N = 2^{20}$. In this case $u_{j+1} = u_j + X_j$ where $\{X_j : 0 \le j \le N\}$ are independent, identically distributed $\mathrm{N}(0, h)$ random variables. The actual approximation errors along with the bounds found in Theorem 2.2 are presented in Figure 2.5. All of the upper bounds Upper Bounds (2.4a) to (2.4c) give guaranteed bounds, as expected, but none closely track the growth of the actual errors with s. Deterministic bounds of the type presented in Theorem 2.2 will be overly pessimistic when the most extreme differences of the data are likely to occur in isolation, since only clusters of large differences will cause large approximation errors. Such is the case with the Brownian motion sample path positions, since the increments of the data are independent of one another. With smooth data, by contrast, nearby differences are correlated, so the approximation errors are more closely related to the most extreme differences.

Datasets in which a random component is present are commonplace in both experiments and computational simulations. For instance, if a deterministic process $\{f_j : 0 \le j \le N\}$ is subject to measurement errors $\{X_j : 0 \le j \le N\}$, then observational data will take the form

$$u_j = f_j + X_j \tag{2.6a}$$

where $\{X_j : 0 \le j \le N\}$ are independent and identically distributed. Equally well, if $\{u_j : 0 \le j \le N\}$ are the prices of a stock over time, the change in value over a time interval can be modeled as

$$u_{j+1} - u_j = f_j + X_j \tag{2.6b}$$

where f_j is a deterministic drift and $\{X_j : 0 \le j \le N\}$ are again independent, identically distributed random variables as in (2.6a) but now representing the underlying volatility. Subequations (2.6a) and (2.6b) correspond to the cases that the zeroth and first, respectively, order differences have a random nature. If a numerical method is used to approximate the position of a particle under the influence of a force undergoing random fluctuations, then one may obtain a scheme of the form

$$u_{j+1} - 2u_j + u_{j-1} = f_j + X_j \tag{2.6c}$$

in which it is the second order differences that have a random nature.

As Example 2.2 shows, Theorem 2.2 can sometimes give poor bounds when applied to data having one of the forms (2.6a) to (2.6c). Rather than an absolute bound on the maximum possible error, a more appropriate measure of accuracy in these cases would be a statistical or probabilistic estimate of the *likely* error. Without additional information on the data, no such estimate can be found. If the mean and variance of the perturbations $\{X_j : 0 \le j \le N\}$ are known, though, they can be used to find information on the distribution of the approximation errors $\{\Delta_j : 0 \le j \le N\}$, as in the next result.

Theorem 2.3. *Let $\{f_j : 0 \le j \le N\}$ be deterministic, and let $\{X_j : 0 \le j \le N\}$ be independent, identically distributed random variables with mean μ and variance σ^2. Define $\boldsymbol{f} : \{0, \dots, N/s - 1\} \to \mathbb{R}^{s+1}$ by $\boldsymbol{f}_m = (f_{ms}, \dots, f_{ms+s})$.*

(a) If $u_j = f_j + X_j$, then $\boldsymbol{\Delta}_m$ has mean $\Psi^0 \boldsymbol{f}_m$ and covariance matrix $\sigma^2 \Psi^0 \Psi^{0\intercal}$.

(b) If $\delta_j^1 = f_j + X_j$, then $\boldsymbol{\Delta}_m$ has mean $\Psi^1 \boldsymbol{f}_m$ and covariance matrix $\sigma^2 \Psi^1 \Psi^{1\intercal}$.

(c) If $\delta_j^2 = f_j + X_j$, then $\boldsymbol{\Delta}_m$ has mean $\Psi^2 \boldsymbol{f}_m + \mu \Psi^2 \vec{1}$ and covariance matrix $\sigma^2 \Psi^2 \Psi^{2\intercal}$.

Proof. Define $\boldsymbol{X} : \{0, \dots, N/s - 1\} \to \mathbb{R}^{s+1}$ by $\boldsymbol{X}_m = (X_{ms}, \dots, X_{ms+s})$. We begin by calculating the mean in the case $u_j = f_j + X_j$. Recall from Lemma 2.1 that $\boldsymbol{\Delta}_m = \Psi^0 \boldsymbol{u}_m$. Using the linearity of expectation,

$$\mathbb{E}\, \boldsymbol{\Delta}_m = \mathbb{E}\, \Psi^0 \boldsymbol{u}_m = \Psi^0\, \mathbb{E}\, \boldsymbol{u}_m = \Psi^0 (\mathbb{E}\, \boldsymbol{f}_m + \boldsymbol{X}_m) = \Psi^0 \boldsymbol{f}_m + \mu \Psi^0 \vec{1}$$

where $\vec{1}$ is the vector all of whose entries are 1. Observe that each row of Ψ^0 sums to 0: $\sum_{j=0}^{s} \Psi^0_{i,j} = -(1-\frac{i}{s}) + 1 - \frac{i}{s} = 0$. That is, $\vec{1} \in \ker(\Psi^0)$, and so $\mathbb{E}\,\boldsymbol{\Delta}_m = \Psi^0 \boldsymbol{f}_m$.

Next, we calculate the covariance matrix. Using the bilinearity of covariance,

$$\mathrm{Cov}(\boldsymbol{\Delta}_m) = \mathrm{Cov}(\Psi^0 \boldsymbol{u}_m) = \Psi^0 \,\mathrm{Cov}(\boldsymbol{u}_m) \Psi^{0\intercal}.$$

Since $\{X_j : 0 \le j \le N\}$ are independent, the components of $\boldsymbol{u}_m$ are independent. Each has variance σ^2 (the deterministic components are constant), so $\mathrm{Cov}(\boldsymbol{u}_m) = \sigma^2 I$. Thus, $\mathrm{Cov}(\boldsymbol{\Delta}_m) = \sigma^2 \Psi^0 \Psi^{0\intercal}$.

Next, suppose $\delta^1_j = f_j + X_j$. By the same argument as in the previous case, we find that $\mathbb{E}\,\boldsymbol{\Delta}_m = \Psi^1 \boldsymbol{f}_m + \mu \Psi^1 \vec{1}$. $\vec{1} \in \ker(\Psi^1)$, since $\sum_{j=0}^{s-1} \Psi^1_{i,j} = i(1-\frac{i}{s}) - (s-i)\frac{i}{s} = 0$, so $\mathbb{E}\,\boldsymbol{\Delta}_m = \Psi^1 \boldsymbol{f}_m$. The calculation of the covariance matrix proceeds exactly as before, and we find that $\mathrm{Cov}(\boldsymbol{\Delta}_m) = \sigma^2 \Psi^1 \Psi^{1\intercal}$.

Finally, suppose $\delta^2_j = f_j + X_j$. As in the previous two cases, we find that $\mathbb{E}\,\boldsymbol{\Delta}_m = \Psi^1(\boldsymbol{f}_m + \mu\vec{1})$. But $\vec{1} \notin \ker(\Psi^2)$:

$$\begin{aligned}
\sum_{j=1}^{s-1} \Psi^2_{i,j} &= \sum_{j=1}^{i} -s(1-\tfrac{i}{s})\tfrac{j}{s} + \sum_{j=i+1}^{s-1} -s(1-\tfrac{j}{s})\tfrac{i}{s} = -(1-\tfrac{i}{s}) \sum_{j=1}^{i} j - \tfrac{i}{s} \sum_{j=i+1}^{s-1} s - j \\
&= -(1-\tfrac{i}{s}) \sum_{j=1}^{i} j - \tfrac{i}{s} \sum_{j=1}^{s-(i+1)} j = -(1-\tfrac{i}{s})\tfrac{i(i+1)}{2} - \tfrac{i}{s}\tfrac{(s-(i+1))(s-i)}{2} \\
&= -\tfrac{i}{2s}\big[(s-i)(i+1) - (s-i)(s-(i+1))\big] = -\tfrac{1}{2} s^2 (1-\tfrac{i}{s})\tfrac{i}{s}.
\end{aligned}$$

If $\mu \neq 0$, $\boldsymbol{X}_m$ contributes to $\boldsymbol{u}_m$ a quadratic component which will not be reproduced by the linear interpolant. So, $\mathbb{E}\,\boldsymbol{\Delta}_m = \Psi^2 \boldsymbol{f}_m + \mu \Psi^2 \vec{1}$. The covariance calculation proceeds exactly as before, and we find that $\mathrm{Cov}(\boldsymbol{\Delta}_m) = \sigma^2 \Psi^2 \Psi^{2\intercal}$. □

Let us now revisit the case of the Brownian motion data used in Example 2.2.

Example 2.2 (continued). Recall that $\{\delta^1_j : 0 \le j \le N-1\}$ are, as increments of a Brownian motion, independent $\mathrm{N}(0,h)$ random variables. As a linear transformation of $\boldsymbol{\delta}^1_m$, a Gaussian random vector, $\boldsymbol{\Delta}_m$ is a Gaussian random vector, with (by Theorem 2.3) mean $\vec{0}$ and covariance matrix $h\Psi^1\Psi^{1\intercal}$. We show in Section 2.8 that $\Psi^1\Psi^{1\intercal} = -\Psi^2$. Thus, each individual error Δ_{ms+i} is normally distributed with mean 0 and variance $-h\Psi^2_{i,i} = sh(1-\frac{i}{s})\frac{i}{s}$. The probability that a normal random variable falls within three standard deviations of its mean is approximately 99.7%, and the standard deviation of Δ_{ms+i} is $[sh(1-\frac{i}{s})\frac{i}{s}]^{1/2}$. Maximizing over $i \in \{1, \ldots, s-1\}$, we can use

$$\big(-\tfrac{3}{2}\sqrt{sh}, +\tfrac{3}{2}\sqrt{sh}\big) \tag{2.7}$$

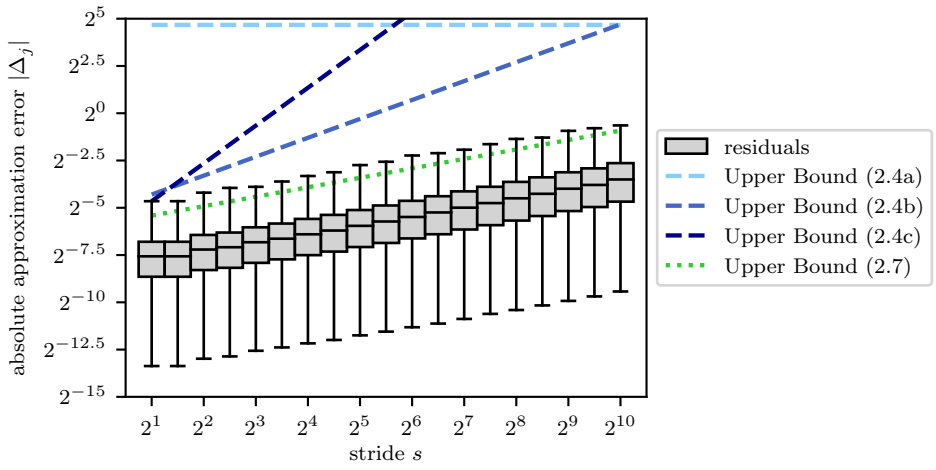

Figure 2.6: Illustration of Upper Bound (2.7) and the bounds of Theorem 2.2 with $\{u_j : 0 \le j \le N\}$ the positions of a Brownian motion sample path. Compare with Figure 2.5, which uses the same data. Upper Bound (2.7), derived using Theorem 2.3, tracks the growth of the approximation errors better than the bounds from Theorem 2.2. It is not, however, a strict bound: observe that the largest approximation errors, marked by the top whiskers, lie outside the prediction intervals.

as a 99.7% prediction interval for each Δ_{ms+i}. See Figure 2.6 for a comparison of Upper Bound (2.7) with Upper Bounds (2.4a) to (2.4c) for this data.

2.4 A Lossless Compression Algorithm

Algorithms 2.1 and 2.2 together define a lossy compression procedure, with perfect approximation achieved only for data that are piecewise linear between the values retained by decimation. The information lost is embodied by the approximation errors $\{\Delta_j : 0 \le j \le N\}$, meaning that the lossy algorithm can be made lossless by the simple expedient of saving these errors in the encoding step and using them to correct the approximations in the decoding step. Algorithms 2.3 and 2.4 give the resulting lossless compression and decompression procedures.

The lossless algorithm is only useful, of course, if its output is smaller than its input, the original data. The output consists of the decimated data, which will be smaller by a factor of s than the original data, and the residuals, which are compressed by an entropy encoder. A theoretical limit on the compression ratio achievable by an entropy encoder is given by Shannon's source coding theorem [Sha01]. Roughly speaking, the theorem states that a stream of symbols can be optimally encoded so that the average number of bits used per symbol is equal to the *Shannon entropy* of the stream. If X is a discrete random variable taking values in some set S, the Shannon entropy of X, denoted

Require: s divides N.
Require: EntropyEncoder is an ideal entropy encoder.
1: **function** CompressLossless(uncompressed data `data`$[0, \ldots, N]$, stride s)
2: `decimated` ←Decimated(`data`, s)
3: `surrogate` ←Approximate(`decimated`, s)
4: `errors` ← `[]`
5: **for** $i = 0, \ldots, N$ **do**
6: `errors`$[i]$ ← `data`$[i]$ − `surrogate`$[i]$
7: **end for**
8: **return** `decimated` and EntropyEncoder(`errors`)
9: **end function**

Algorithm 2.3: Lossless decimation encoding: decimation (Algorithm 2.1), approximation (Algorithm 2.2), and encoding of errors. By an *ideal entropy encoder* we mean an encoder than can compress its input with the maximum efficiency allowed by Shannon's source coding theorem.

Require: EntropyDecoder is the inverse of EntropyEncoder.
function DecompressLossless(decimated data `decimated`$[0, \ldots, N/s]$, stride s, compressed errors `errorsCompressed`)
`surrogate` ←Approximate(`decimated`, s)
`errors` ←EntropyDecoder(`errorsCompressed`)
`data` ← `[]`
for $i = 0, \ldots, N$ **do**
`data`$[i]$ ← `surrogate`$[i]$ + `errors`$[i]$
end for
return `data`
end function

Algorithm 2.4: Lossless decimation decoding: reconstruction of original data from decimated data and errors.

$H(X)$, is defined by

$$H(X) = -\sum_{x \in S} \mathbb{P}(X = x) \log_2 \mathbb{P}(X = x). \tag{2.8}$$

The compression ratio achieved by Algorithm 2.3, then, can be related to the Shannon entropy of the residuals $H(\boldsymbol{\Delta})$, as in the following result.

Theorem 2.4. *Let $\boldsymbol{\Delta}$ be an $\mathbb{R}^{s-1}$-valued process, and suppose that a dataset $\{u_j : 0 \leq j \leq N\}$ is generated in such a way that the approximation errors $\{\boldsymbol{\Delta}_m : 0 \leq m \leq N/s - 1\}$ are sampled from $\boldsymbol{\Delta}$. If each datum u_j requires b bits of storage, then the expected compression ratio $\mathfrak{R}$ achieved by Algorithm 2.3 when applied to $\{u_j : 0 \leq j \leq N\}$ is*

$$\mathfrak{R} = s\big[1 + \tfrac{1}{b} H(\boldsymbol{\Delta})\big]^{-1} \tag{2.9}$$

in the limit $N \to \infty$.

Proof. The original dataset requires $(1 + N)b$ bits of storage, and the decimated dataset requires $(1 + \frac{N}{s})b$ bits of storage. There are N/s realizations of $\boldsymbol{\Delta}$ in $\{\boldsymbol{\Delta}_m : 0 \leq m \leq N/s - 1\}$, so an ideal entropy encoder is expected (in the sense of probability) to use between $\frac{N}{s} H(\boldsymbol{\Delta})$ and $1 + \frac{N}{s} H(\boldsymbol{\Delta})$ bits to encode the approximation errors [Sha01]. So, the expected compression ratio is

$$\frac{(1 + N)b}{(1 + \frac{N}{s})b + 1 + \frac{N}{s} H(\boldsymbol{\Delta})},$$

which converges to (2.9) in the limit $N \to \infty$. □

Theorem 2.4 provides a quantitative relationship between the entropy of the approximation errors and the compression ratio achieved by Algorithm 2.3. If the entropy $H(\boldsymbol{\Delta})$ is negligible (meaning that the extra storage required for the approximation errors is small), then the limiting compression ratio of s is achieved. Note that this is the ratio achieved by Algorithm 2.1; Algorithm 2.3 being lossless, it is natural that its performance is limited by that of its lossy counterpart. If the approximation errors have little structure (in the sense that $H(\boldsymbol{\Delta})$ is large), then the compression ratio degenerates, reaching its minimum when $H(\boldsymbol{\Delta})$ is at its maximum. Each $\boldsymbol{\Delta}_m$ is a vector of length $s - 1$ with each entry requiring b bits of storage, so $H(\boldsymbol{\Delta})$ is at most $(s - 1)b$. In the worst case scenario, then, the expected compression ratio is 1, and there is no benefit to using Algorithm 2.3.

In order to apply Theorem 2.4, we must know the value of $H(\boldsymbol{\Delta})$, which may be difficult or impossible to compute in practice. If the Shannon entropies of the components of $\boldsymbol{\Delta}$, denoted $\{\boldsymbol{\Delta} \cdot \boldsymbol{e}_i : 1 \leq i \leq s - 1\}$, can be determined instead, then they may be used to bound $\mathfrak{R}$ according to

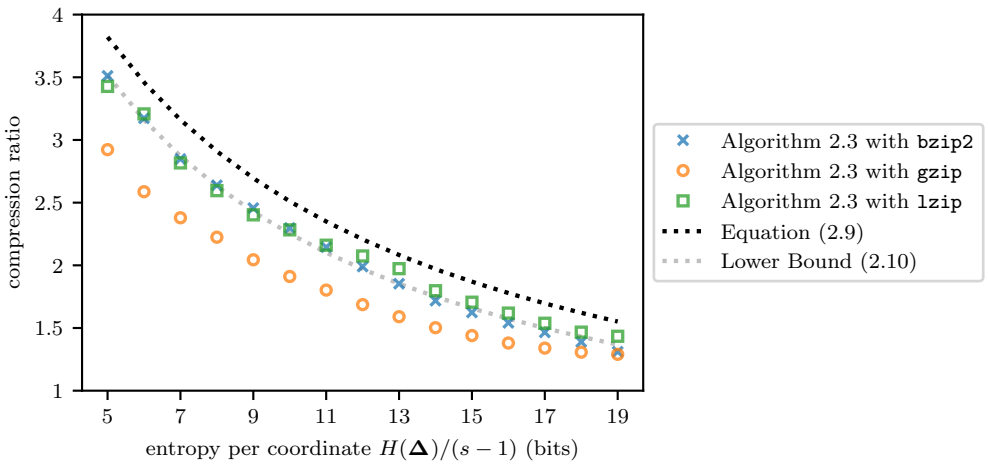

Figure 2.7: Illustration of the relationship between the Shannon entropy of the residuals and the compression ratio achieved by Algorithm 2.3. The data are constructed so that $\{\mathbf{\Delta} \cdot \boldsymbol{e}_i : 1 \leq i \leq s-2\}$ are independent, identically distributed random variables of known entropy and $\mathbf{\Delta} \cdot \boldsymbol{e}_{s-1} = \mathbf{\Delta} \cdot \boldsymbol{e}_{s-2}$. Consequently, $H(\mathbf{\Delta}) \leq \sum_{i=1}^{s-1} H(\mathbf{\Delta} \cdot \boldsymbol{e}_i)$, and Theorem 2.4 and Corollary 2.5 give different estimates. For each entropy value, 2^5 datasets of size 2^{20} are generated and the ensemble average of the compression ratios achieved is reported.

the following result.

Corollary 2.5. *Let $\mathbf{\Delta}$ be an $\mathbb{R}^{s-1}$-valued process, and suppose that a dataset $\{u_j : 0 \leq j \leq N\}$ is generated in such a way that the approximation errors $\{\mathbf{\Delta}_m : 0 \leq m \leq N/s - 1\}$ are sampled from $\mathbf{\Delta}$. If each datum u_j requires b bits of storage, then the expected compression ratio $\mathfrak{R}$ achieved by Algorithm 2.3 when applied to $\{u_j : 0 \leq j \leq N\}$ satisfies*

$$\mathfrak{R} \geq s\left[1 + \tfrac{1}{b}\sum_{i=1}^{s-1} H(\mathbf{\Delta} \cdot \boldsymbol{e}_i)\right]^{-1} \tag{2.10}$$

in the limit $N \to \infty$.

Proof. $H(\mathbf{\Delta}) \leq \sum_{i=1}^{s-1} H(\mathbf{\Delta} \cdot \boldsymbol{e}_i)$ [Sha01]. The result now follows from Theorem 2.4. □

Corollary 2.5 yields an inequality instead of an equality because it, unlike Theorem 2.4, ignores any correlation between neighboring residuals. Neither result can be applied without some means of computing or at least estimating the entropy of the residuals. We develop a few methods of doing this in Sections 2.5 and 2.6.

Figure 2.7 illustrates the relationship between the value for $\mathfrak{R}$ obtained using (2.9), the bound on $\mathfrak{R}$ obtained using Lower Bound (2.10), and the actual compression ratio achieved by Algorithm 2.3. Figure 2.8 shows that the linear predictor is responsible for the bulk of the achieved compression. As

written, Algorithm 2.3 requires an ideal entropy encoder able to compress the approximation errors to within the entropy of their generating process, whatever that process may be. Of course, no such encoder is available for general data. We instead use three general-purpose compressors:

1. `bzip2` [Sew10], which uses the Burrows–Wheeler algorithm [BW94].
2. `gzip` [GA13a], which uses the LZ77 algorithm [ZL77].
3. `lzip` [DD16], which uses LZMA [Pav15].

and four specialized floating point compressors:

1. `SPDP` [BC16]. `SPDP` is a lossless black box compressor.
2. `SZ` [DC16]. `SZ` is a lossy compressor with user-specifiable error bounds; we use it as a near-lossless compressor by requiring that the absolute error be at most 2^{-149} for single-precision and 2^{-1074} for double-precision data.
3. `fpzip` [LI06]. `fpzip` can compress 1D, 2D, and 3D scalar fields in lossy and lossless configurations. We use the lossless 1D mode.
4. `fpc` [BR07]. `fpc` is a lossless compressor for double-precision data. When pairing `fpc` with Algorithm 2.3, we will use only the encoder, not the predictor, of the former. We will refer to the combination as Algorithm 2.3 with `fpc_encode`.

We could obtain compression ratios closer to (2.9) by using a compressor tailored to the structure of our approximation errors. Similar optimizations have been taken in many previous algorithms. One approach is to use an encoder specifically designed for floating-point residuals [ILS05a, BGC13, SSL$^+$12, BR07]. Another is to seed an entropy encoder with the expected distribution of the prediction errors. This approach is often taken in image compression algorithms, since it has been empirically observed that prediction errors in that domain are often Laplace distributed [JKHM93, HV91, WSS00].

2.5 Deterministic Bounds on Shannon Entropy of Approximation Errors

In this section we use the results of Section 2.2 to obtain *a priori* bounds on the Shannon entropy of the approximation errors in terms of the differences of the data. In what follows we will always assume the

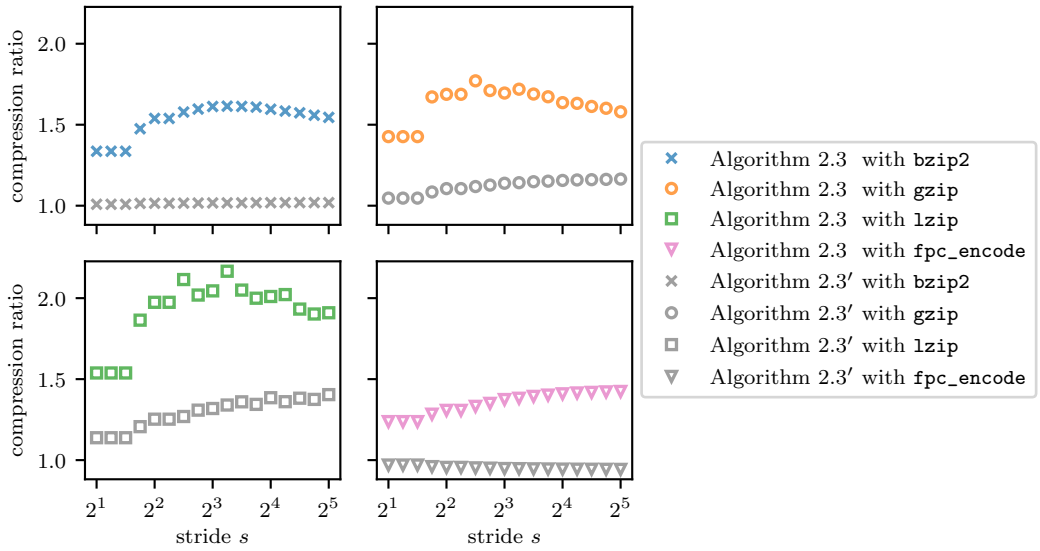

Figure 2.8: Illustration of the effect of replacing the linear predictor in Algorithm 2.3 with a constant predictor. Algorithm 2.3′ denotes the procedure obtained by replacing Line 3 in Algorithm 2.3 with '`surrogate` ← `[0,...,0]`.' The results demonstrate that the linear predictor is responsible for the bulk of the achieved compression rates. The data used are given by $u_j = \tan(jh)$ with $h = 2^{-15}$ and $j \in \{0, \dots, \lfloor 2\pi/h \rfloor\}$.

existence of some $\mathbb{R}^{s-1}$-valued process modelling the approximation errors $\{\mathbf{\Delta}_m : 0 \leq m \leq N/s - 1\}$, as in Theorem 2.4.

Suppose that the vectors of approximation errors $\mathbf{\Delta}_m = (\Delta_{ms+1}, \dots, \Delta_{ms+s-1})$ are contained in some bounding region $\Omega \subset \mathbb{R}^{s-1}$. Intuitively, one might expect the volume of Ω to be related to the Shannon entropy of $\mathbf{\Delta}$. This turns out not to be the case, but it is possible to relate the volume of Ω to the *differential entropy* of $\mathbf{\Delta}$ instead. The differential entropy of an $\mathbb{R}^n$-valued random variable X with probability density p is given by

$$\mathfrak{H}(X) = -\int_{\mathbb{R}^n} p(\vec{x}) \log_2 p(\vec{x}) \, \mathrm{d}\vec{x}. \tag{2.11}$$

Differential entropy is related to Shannon entropy as follows [CT91, p. 229].[1]

Lemma 2.6. *Let X be a continuous $\mathbb{R}^n$-valued random variable with Riemann integrable probability density. If X^Δ is the discretization of X with bin volume w^n, then*

$$H(X^\Delta) + n \log_2 w \to \mathfrak{H}(X)$$

[1] The proof given in [CT91] is restricted to the $n = 1$ case, but it extends to the general case without significant modification.

as $w \to 0$.

Lemma 2.6 is applicable when a continuous quantity is quantized on a uniform grid, as is the case with data stored in integer or fixed-point formats. In practice, scientific datasets are much more frequently represented using floating-point formats, whose hallmark is precision that varies with the magnitude of the number stored. It would appear, then, that Lemma 2.6 is of little use for most data. However, the precision of floating-point formats is fixed within each interval of the form $[\pm 2^n, \pm 2^{n+1})$ [IMII08], and the residuals only need to be discretized to the precision of the data, so this lemma will still be useful for data that do not change order of magnitude too quickly relative to the stride s. In order to make use of Lemma 2.6, in what follows we will assume that $\{u_j : 0 \leq j \leq N\}$ is generated in such a way that we can model the approximation errors $\{\boldsymbol{\Delta}_m : 0 \leq m \leq N/s - 1\}$ as discretizations with bin volume w^{s-1} of some stochastic process with sufficiently smooth density.

The next result quantifies the relationship between differential entropy and the volume of the bounding region Ω.

Lemma 2.7. *Let X be a random variable taking values in $\Omega \subset \mathbb{R}^n$. Then $\mathfrak{H}(X) \leq \log_2 \mathrm{vol}_n(\Omega)$.*

Proof. Of all distributions with support contained in Ω, the differential entropy is maximized by $\mathrm{Unif}(\Omega)$, and the differential entropy of a uniform distribution is the logarithm of the volume of its support [Sha01]. □

Lemmas 2.6 and 2.7 together mean that in order to bound the Shannon entropy of $\boldsymbol{\Delta}$ it suffices to estimate the volume of the region Ω enclosing the vectors $\{\boldsymbol{\Delta}_m : 0 \leq m \leq N/s - 1\}$. We can make use of the absolute bounds on the approximation errors derived in Theorem 2.2 to bound the volume of Ω, leading to the following bounds for the differential entropy of $\boldsymbol{\Delta}$.

Lemma 2.8. *The differential entropy of $\boldsymbol{\Delta}$ can be bounded as follows.*

$$\mathfrak{H}(\boldsymbol{\Delta}) \leq (s-1)(1 + \log_2 \mathfrak{M}^0) + \log_2 \left[\tfrac{1}{6}(s+1)(s+2)\right] \tag{2.12a}$$

$$\mathfrak{H}(\boldsymbol{\Delta}) \leq (s-1)(1 + \log_2 \mathfrak{M}^1) \tag{2.12b}$$

$$\mathfrak{H}(\boldsymbol{\Delta}) \leq (s-1)(1 + \log_2 \mathfrak{M}^2) - \log_2 s \tag{2.12c}$$

Proof. We start with Upper Bound (2.12a). By the definition of $\mathfrak{M}^0$, $\boldsymbol{u}$ is supported inside the $(s+1)$-cube $[-\mathfrak{M}^0, +\mathfrak{M}^0]^{s+1}$. According to Lemma 2.1, then, $\boldsymbol{\Delta}$ is supported inside the $(s-1)$-parallelepiped $\Psi^0[-\mathfrak{M}^0, +\mathfrak{M}^0]^{s+1}$. $\Psi^0[-\mathfrak{M}^0, +\mathfrak{M}^0]^{s+1}$ is a translation of $\Psi^0[0, 2\mathfrak{M}^0]^s$, which is itself a dilation by factor $2\mathfrak{M}^0$ of $\Psi^0[0,1]^{s+1}$. Thus, the volume of $\Psi^0[-\mathfrak{M}^0, +\mathfrak{M}^0]^{s+1}$ is $(2\mathfrak{M}^0)^{s-1}$ times

the volume of $\Psi^0[0,1]^{s+1}$. We show in Section 2.8 that the volume of $\Psi^0[0,1]^{s+1}$ is $\frac{1}{6}(s+1)(s+2)$. Applying Lemma 2.7, we find that

$$\begin{aligned}\mathfrak{H}(\boldsymbol{\Delta}) &\leq \log_2\big[(2\mathfrak{M}^0)^{s-1} \cdot \tfrac{1}{6}(s+1)(s+2)\big] \\ &= (s-1)(1+\log_2 \mathfrak{M}^0) + \log_2\big[\tfrac{1}{6}(s+1)(s+2)\big].\end{aligned}$$

The bounds Upper Bounds (2.12b) and (2.12c) are derived in a similar fashion. The volume of $\Psi^1[-\mathfrak{M}^1, +\mathfrak{M}^1]^s$ is $(2\mathfrak{M}^1)^{s-1}$ times the volume of $\Psi^1[0,1]^s$, which we calculate in Section 2.8 to be 1. So,

$$\mathfrak{H}(\boldsymbol{\Delta}) \leq \log_2\big[(2\mathfrak{M}^1)^{s-1} \cdot 1\big] = (s-1)(1+\log_2 \mathfrak{M}^1).$$

Similarly, the volume of $\Psi^2[-\mathfrak{M}^2, +\mathfrak{M}^2]^{s-1}$ is $(2\mathfrak{M}^2)^{s-1}$ times the volume of $\Psi^2[0,1]^{s-1}$, which we calculate in Section 2.8 to be $1/s$. So,

$$\mathfrak{H}(\boldsymbol{\Delta}) \leq \log_2\big[(2\mathfrak{M}^2)^{s-1} \cdot \tfrac{1}{s}\big] = (s-1)(1+\log_2 \mathfrak{M}^2) - \log_2 s. \qquad \square$$

Now we can combine Lemma 2.8 with Theorem 2.4 to obtain bounds on the expected compression ratio achieved by Algorithm 2.3.

Theorem 2.9. *In the limit $N \to \infty$ and $w \to 0$, the expected compression ratio $\mathfrak{R}$ achieved by Algorithm 2.3 is bounded as follows.*

$$\mathfrak{R} \geq s\big[1 + \tfrac{1}{b}\big((s-1)(1+\log_2 \mathfrak{M}^0 - \log_2 w) + \log_2[\tfrac{1}{6}(s+1)(s+2)]\big)\big]^{-1} \tag{2.13a}$$

$$\mathfrak{R} \geq s\big[1 + \tfrac{1}{b}\big((s-1)(1+\log_2 \mathfrak{M}^1 - \log_2 w)\big)\big]^{-1} \tag{2.13b}$$

$$\mathfrak{R} \geq s\big[1 + \tfrac{1}{b}\big((s-1)(1+\log_2 \mathfrak{M}^2 - \log_2 w) - \log_2 s\big)\big]^{-1} \tag{2.13c}$$

Proof. Combine Lemmas 2.6 and 2.8 and Theorem 2.4. $\square$

Figure 2.9 illustrates the bounds in Theorem 2.9.

2.6 Statistical Bounds on Shannon Entropy of Approximation Errors

Theorem 2.9 is based on the absolute bounds on $\{\Delta_j : 0 \leq j \leq N\}$ derived in Theorem 2.2. As we saw in Example 2.2, these bounds may be overly pessimistic when applied to random data. Consequently,

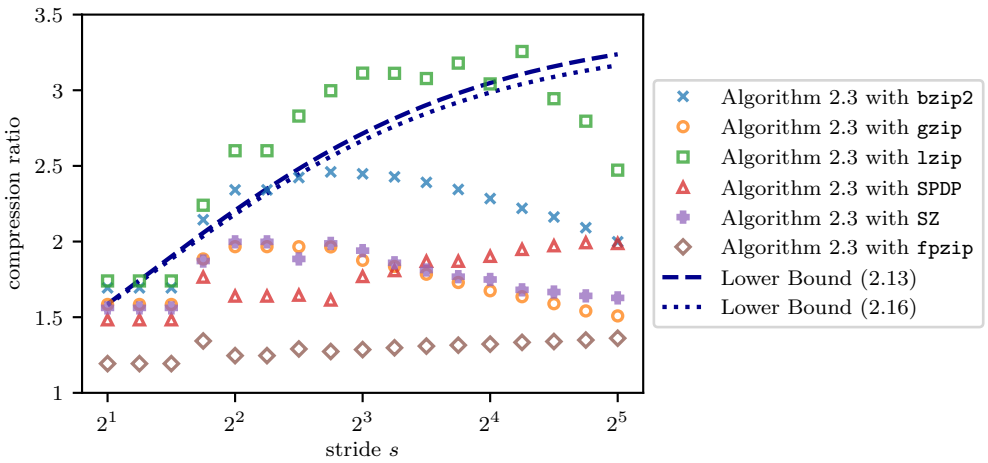

Figure 2.9: Illustration of the lower bounds on the expected compression ratio in Theorem 2.9 versus stride s. Lower Bound (2.13) is the maximum of Lower Bounds (2.13a) to (2.13c); Lower Bound (2.16) is the maximum of Lower Bounds (2.16a) to (2.16c). Along with the bounds are plotted the actual compression ratios obtained using various standard compressors in lieu of an ideal entropy encoder. For most strides tested `lzip` exceeds the theoretical lower bounds, whereas other compressors are less effective. The data are given by $u_j = 2^5(\frac{3}{2} + \frac{1}{2}\sin(jh))$ with $h = 2^{-7}$ and $j \in \{0, \dots, \lfloor 2^6\pi/h \rfloor\}$.

we expect Theorem 2.9 to be similarly pessimistic in this case. We can obtain more realistic estimates by deriving an analogue to Theorem 2.9 based on Theorem 2.3, the analogue to Theorem 2.2 for random data. We begin with the following result relating variance to differential entropy [CT91, p. 234–235]. It plays a role analogous to that of Lemma 2.7.

Lemma 2.10. *Let X be an $\mathbb{R}^n$-valued continuous random variable with mean $\vec{\mu} \in \mathbb{R}^n$ and covariance matrix $\Sigma \in \mathbb{R}^{n\times n}$. Then*

$$\mathfrak{H}(X) \leq \tfrac{n}{2}\log_2[2\pi\mathrm{e}|\det(\Sigma)|^{1/n}] \tag{2.14}$$

with equality holding iff $X \sim \mathrm{N}(\vec{\mu}, \Sigma)$.

If $\{u_j : 0 \leq j \leq N\}$ is generated from a sequence of independent, identically distributed continuous random variables, then Lemma 2.10 can be used to bound the differential entropy of $\boldsymbol{\Delta}$ as follows.

Lemma 2.11. *Let $\{X_j : 0 \leq j \leq N\}$ be independent, identically distributed random variables with mean μ and variance σ^2.*

(a) If $u_j = X_j$, then

$$\mathfrak{H}(\boldsymbol{\Delta}) \leq \tfrac{s-1}{2}\log_2(2\pi\mathrm{e}\sigma^2) + \tfrac{1}{2}\log_2\big[\tfrac{1}{12s}(s+1)^2(s+2)\big]. \tag{2.15a}$$

(b) If $\delta_j^1 = X_j$, then

$$\mathfrak{H}(\mathbf{\Delta}) \leq \tfrac{s-1}{2}\log_2(2\pi e\sigma^2) - \tfrac{1}{2}\log_2(s). \tag{2.15b}$$

(c) If $\delta_j^2 = X_j$, then

$$\mathfrak{H}(\mathbf{\Delta}) \leq \tfrac{s-1}{2}\log_2(2\pi e\sigma^2) - \log_2(s). \tag{2.15c}$$

Proof. Begin with Upper Bound (2.15a). We found in Theorem 2.3 that if $u_j = X_j$, then $\mathbf{\Delta}$ has mean $\vec{0}$ and covariance $\sigma^2\Psi^0\Psi^{0\intercal}$. Applying Lemma 2.10,

$$\begin{aligned}
\mathfrak{H}(\mathbf{\Delta}) &\leq \tfrac{s-1}{2}\log_2\big[2\pi e|\det(\sigma^2\Psi^0\Psi^{0\intercal})|^{1/(s-1)}\big] \\
&= \tfrac{s-1}{2}\log_2\big[2\pi e|(\sigma^2)^{s-1}\det(\Psi^0\Psi^{0\intercal})|^{1/(s-1)}\big] \\
&= \tfrac{s-1}{2}\log_2\big[2\pi e\sigma^2|\det(\Psi^0\Psi^{0\intercal})|^{1/(s-1)}\big] \\
&= \tfrac{s-1}{2}\log_2(2\pi e\sigma^2) + \tfrac{1}{2}\log_2\big[\tfrac{1}{12s}(s+1)^2(s+2)\big]
\end{aligned}$$

where we have used the expression for $\det(\Psi^0\Psi^{0\intercal})$ found in Section 2.8. The bounds Upper Bounds (2.15b) and (2.15c) are derived in exactly the same manner. □

Now we can combine Lemma 2.11 with Theorem 2.4 to obtain bounds on the expected compression ratio achieved by Algorithm 2.3.

Theorem 2.12. *Let $\{X_j : 0 \leq j \leq N\}$ be independent, identically distributed continuous random variables with mean μ and variance σ^2. Let $\mathfrak{R}$ be the expected compression ratio achieved by Algorithm 2.3 when applied to $\{u_j : 0 \leq j \leq N\}$ in the limit $N \to \infty$ and $w \to 0$.*

(a) If $u_j = X_j$, then

$$\mathfrak{R} \geq s\big[1 + \tfrac{1}{b}\big(\tfrac{s-1}{2}\log_2(2\pi e\sigma^2w^2) + \tfrac{1}{2}\log_2\big[\tfrac{1}{12s}(s+1)^2(s+2)\big]\big)\big]^{-1}. \tag{2.16a}$$

(b) If $\delta_j^1 = X_j$, then

$$\mathfrak{R} \geq s\big[1 + \tfrac{1}{b}\big(\tfrac{s-1}{2}\log_2(2\pi e\sigma^2w^2) - \tfrac{1}{2}\log_2(s)\big)\big]^{-1}. \tag{2.16b}$$

(c) If $\delta_j^2 = X_j$, then

$$\mathfrak{R} \geq s\big[1 + \tfrac{1}{b}\big(\tfrac{s-1}{2}\log_2(2\pi e\sigma^2w^2) - \log_2(s)\big)\big]^{-1}. \tag{2.16c}$$

Proof. Combine Lemmas 2.6 and 2.11 and Theorem 2.4. □

Figure 2.10 illustrates the bounds in Theorem 2.12.

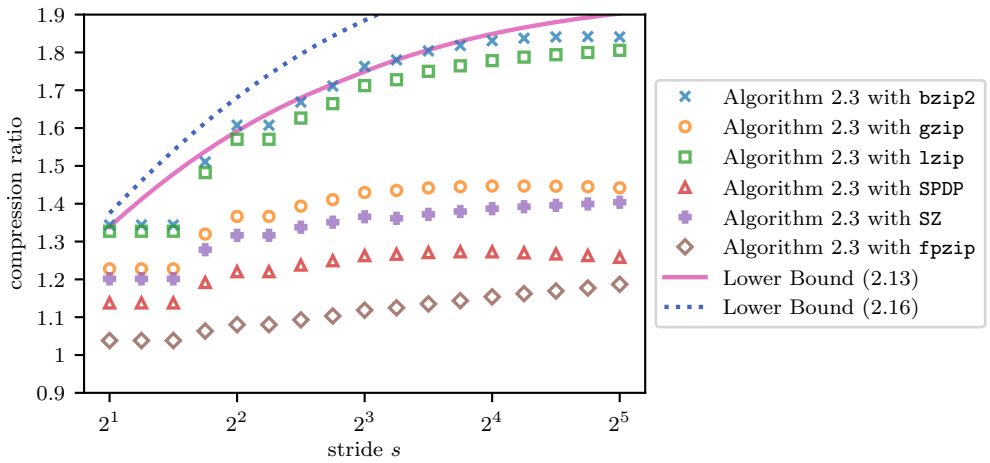

Figure 2.10: Illustration of the bounds of Theorem 2.12 when $\{u_j : 0 \le j \le N\}$ are the positions of a Brownian motion with timestep 2^{-10}. The compression ratios shown are the ensemble averages across 2^5 trials with 2^{20} timesteps each. After each trial, the zeroth, first, and second order differences are calculated, and the maxima and variances of the differences are used to generate bounds according to Lower Bounds (2.13a) to (2.13c) and (2.16a) to (2.16c). An ensemble average is calculated for each bound, and Lower Bounds (2.13) and (2.16) are the maxima of these averages. None of the compressors match the optimal compression ratios, but `bzip2` and `lzip` track Lower Bound (2.13), achieving good performance.

2.7 Applications

In this section we evaluate the performance of Algorithm 2.3 on a selection of representative datasets. In particular, we compare the compression ratios achieved with the ratios obtained using several state of the art floating point compressors applied directly to the original datasets.

2.7.1 Black–Scholes Simulation

First, we consider data generated by a Black–Scholes solver [Bur16a] to simulate random fluctuations in the price of an asset over a time interval $[0, 1]$ with 2^{20} timesteps. The starting price is 1, the expected growth rate is taken to be 1, and the volatility is set to 0.25. The output is an array of $2^{20}+1$ double-precision option values, which are cast down to single precision before being compressed. Figure 2.11 shows that reduction of the data by up to 50% is achieved.

2.7.2 Lorenz Attractor

Second, we consider data generated from the Lorenz equations, whose solutions follow deterministic (though not easily predictable) paths. We use an ODE solver [Bur16b] to trace the trajectory of a

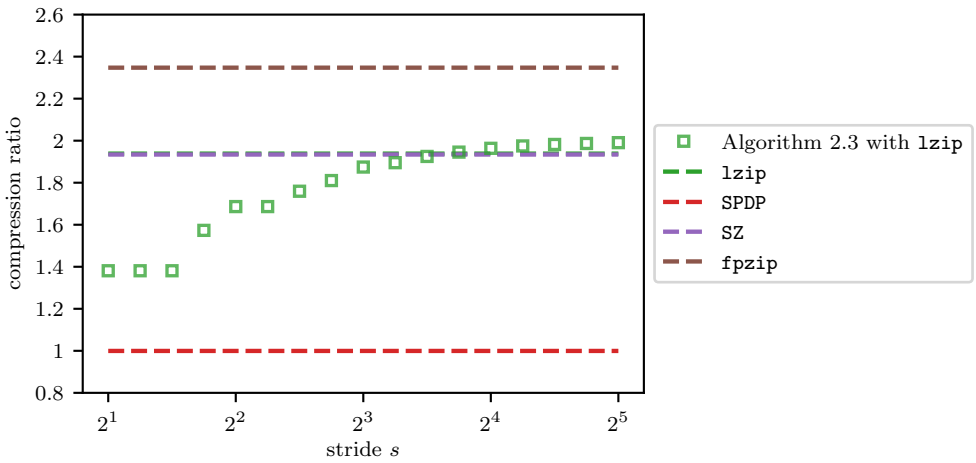

Figure 2.11: Illustration of the performance of Algorithm 2.3 on an asset path evolving according to the Black–Scholes equation.

particle with initial position $(8, 1, 1)$ over the interval $[0, 40]$ with 2×10^5 timesteps. The parameters used are $\rho = 28$, $\sigma = 10$, and $\beta = 8/3$. The double-precision x coordinate is compressed using Algorithm 2.3. Figure 2.12 shows that a 35% reduction of the data is achieved.

2.7.3 Forced Isotropic Turbulence

Third, we consider data from a large turbulence simulation [LPW+08, PBLM07]. The data are a $2^9 \times 2^9 \times 2^9$ cube of single-precision pressure values (512 MiB in total) at a single timestep (timestep 2^{10}), traversed in lexicographic order. Figure 2.13 shows that the performance of Algorithm 2.3 is comparable to that of `lzip`, `SPDP`, and `SZ`. In this example `fpzip` gives the clear best performance, achieving a reduction of approximately 30%.

2.7.4 Head Impact

Finally, we compare Algorithm 2.3 with the bitwise modal averaging algorithm described in [BGC13] on a 135 MiB double-precision dataset produced by a head impact simulation [Bur16c]. Figure 2.14 shows that reductions of 8% are consistently achieved over the stride sizes $[2^1, 2^5]$. To facilitate a direct comparison, we modify Algorithm 2.3 by encoding the decimated data in addition to the approximation errors, but the algorithm is otherwise applied directly as described earlier. Our implementation of the technique from [BGC13] gives a 7% reduction.

Note that the reductions of up to 11% reported in [BGC13] were obtained by pairing the bitwise averaging procedure with a byte-level serialization technique [BG16].

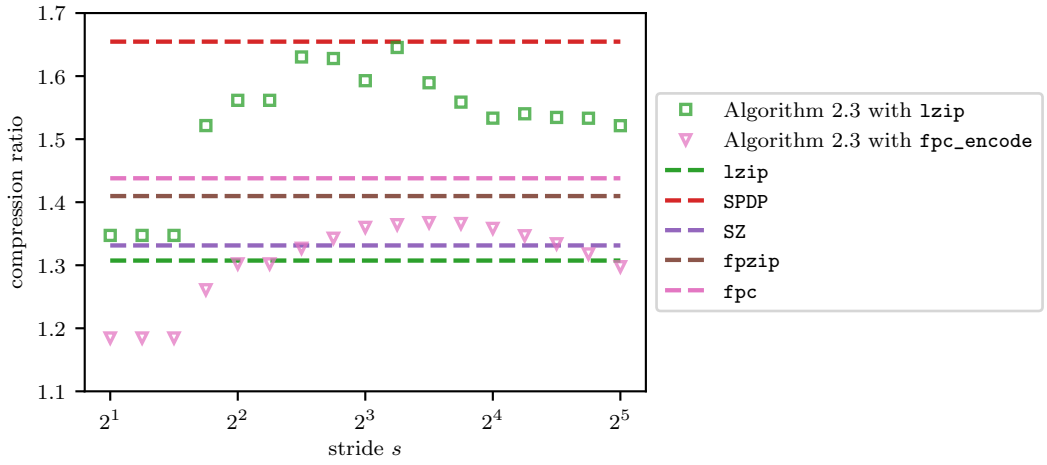

Figure 2.12: Illustration of the performance of Algorithm 2.3 on the x coordinate of a numerical solution of the Lorenz system.

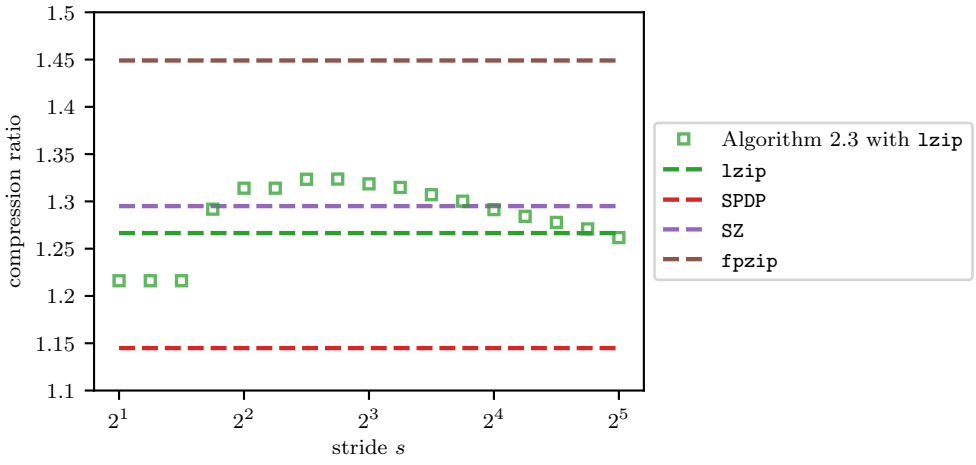

Figure 2.13: Illustration of the performance of Algorithm 2.3 on a single timeslab of a 3D turbulence simulation.

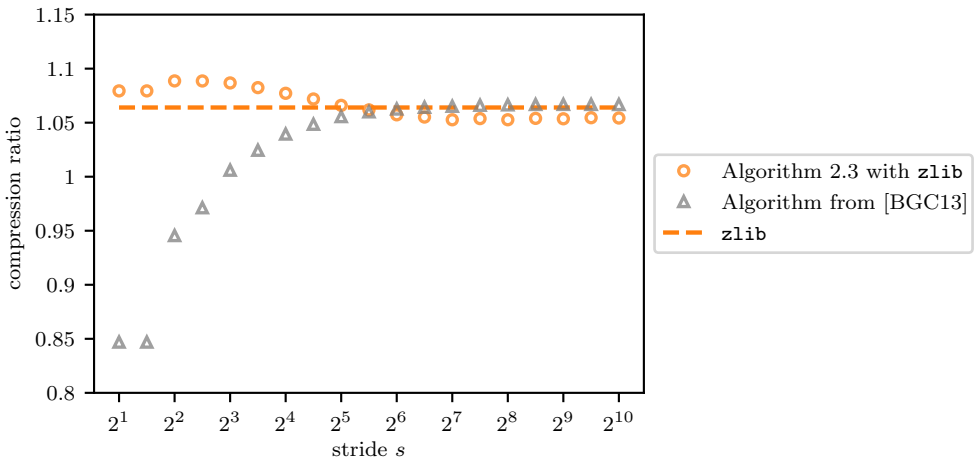

Figure 2.14: Illustration of the performance of Algorithm 2.3 on a head impact simulation. The compressor used by both methods is `zlib` [GA13b], which uses the same algorithm as `gzip`.

2.8 Properties of Ψ^0, Ψ^1, and Ψ^2

The aim of this section is twofold: first, to compute the volumes of $\Psi^0[0,1]^{s+1}$, $\Psi^1[0,1]^s$, and $\Psi^2[0,1]^{s-1}$, and second, to compute the determinants of $\Psi^0\Psi^{0\intercal}$, $\Psi^1\Psi^{1\intercal}$, and $\Psi^2\Psi^{2\intercal}$. We will compute the volumes using the following lemma, which relates the volume of the unit cube under a linear mapping to the mapping's minors [GK10].

Lemma 2.13. *Let $n \geq m$, and let $T\colon \mathbb{R}^n \to \mathbb{R}^m$ be a linear transformation with full rank. The m-volume of $T[0,1]^n$ is the sum of the absolute $m \times m$ minors of T.*

When $n = m$, Lemma 2.13 simplifies to the familiar statement that a linear mapping scales volumes by its determinant.

We next establish that Lemma 2.13 can be applied to Ψ^0, Ψ^1, and Ψ^2. Let $\boldsymbol{c}_0^0, \ldots, \boldsymbol{c}_s^0$, $\boldsymbol{c}_0^1, \ldots, \boldsymbol{c}_{s-1}^1$, and $\boldsymbol{c}_1^2, \ldots, \boldsymbol{c}_{s-1}^2$ denote the columns of Ψ^0, Ψ^1, and Ψ^2, respectively. Denote by $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{s-1}$ the standard basis of $\mathbb{R}^{s-1}$.

Lemma 2.14. *Ψ^0, Ψ^1, and Ψ^2 have full rank.*

Proof. $\boldsymbol{c}_j^0 = \boldsymbol{e}_j$ for $j \in \{1, \ldots, s-1\}$, so Ψ^0 has full rank. Observe that $\boldsymbol{c}_0^1 = \frac{1}{s}\sum_{j=1}^{s-1}\sum_{k=1}^{j}\boldsymbol{e}_k$ and $\boldsymbol{c}_j^1 = \boldsymbol{c}_0^1 - \sum_{k=1}^{j}\boldsymbol{e}_k$ for $j \in \{1, \ldots, s-1\}$, which implies that Ψ^1 has full rank. As Ψ^2 is a square matrix, it has full rank iff its determinant is nonzero. We establish that $\det(\Psi^2) \neq 0$ in Lemma 2.15. □

Denote by $M_{j,k}^0$ the submatrix produced by deleting $\boldsymbol{c}_j^0$ and $\boldsymbol{c}_k^0$ from Ψ^0. Denote by M_j^1 the

submatrix produced by deleting $\boldsymbol{c}_j^1$ from Ψ^1.

Lemma 2.15. *The absolute $(s-1)\times(s-1)$ minors of Ψ^0, Ψ^1, and Ψ^2 take on the following values:*

(a) for Ψ^0, 1 with multiplicity 1; k/s for $k \in \{1,\dots,s-1\}$, each with multiplicity 2; and $(k-j)/s$ for $1 \le j < k \le s-1$, each with multiplicity 1.

(b) for Ψ^1, $1/s$ with multiplicity s.

(c) for Ψ^2, $1/s$ with multiplicity 1.

Proof. Let $\mathrm{AbsDet}\colon (\mathbb{R}^{s-1})^{s-1} \to [0,\infty)$ be the function mapping an $(s-1)$-tuple of vectors in $\mathbb{R}^{s-1}$ to the absolute value of the determinant of the matrix whose columns are those vectors. We use the notation $(\,\cdot\,)_i$ for the ith component of a vector.

Begin with Ψ^0. We compute $|\det(M^0_{j,k})|$ case by case. If $j=0$ and $k=s$, $|\det(M^0_{0,s})| = \mathrm{AbsDet}(\boldsymbol{e}_1,\dots,\boldsymbol{e}_{s-1}) = 1$. If $j=0$ and $1 \le k \le s-1$,

$$|\det(M^0_{0,k})| = \mathrm{AbsDet}(\boldsymbol{e}_1,\dots,\boldsymbol{e}_{k-1},\boldsymbol{e}_{k+1},\dots,\boldsymbol{e}_{s-1},\boldsymbol{c}_s^0) = |(\boldsymbol{c}_s^0)_k| = |\Psi^0_{k,s}| = \tfrac{k}{s}.$$

Similarly, if $1 \le j \le s-1$ and $k=s$,

$$|\det(M^0_{j,s})| = \mathrm{AbsDet}(\boldsymbol{c}_0^0,\boldsymbol{e}_1,\dots,\boldsymbol{e}_{j-1},\boldsymbol{e}_{j+1},\dots,\boldsymbol{e}_{s-1}) = |(\boldsymbol{c}_0^0)_j| = |\Psi^0_{j,0}| = 1-\tfrac{j}{s}.$$

Finally, if $1 \le j < k \le s-1$,

$$\begin{aligned}|\det(M^0_{j,k})| &= \mathrm{AbsDet}(\boldsymbol{c}_0^0,\boldsymbol{e}_1,\dots,\boldsymbol{e}_{j-1},\boldsymbol{e}_{j+1},\dots,\boldsymbol{e}_{k-1},\boldsymbol{e}_{k+1},\dots,\boldsymbol{e}_{s-1},\boldsymbol{c}_s^0)\\ &= |\Psi^0_{j,0}\Psi^0_{k,s} - \Psi^0_{k,0}\Psi^0_{j,s}| = \tfrac{k-j}{s}.\end{aligned}$$

Next, consider Ψ^1. For $j \in \{1,\dots,s-1\}$,

$$\begin{aligned}|\det(M^1_j)| &= \mathrm{AbsDet}\Big(\boldsymbol{c}_0^1, \boldsymbol{c}_0^1 - \sum_{k=1}^{1}\boldsymbol{e}_k,\dots,\boldsymbol{c}_0^1 - \sum_{k=1}^{j-1}\boldsymbol{e}_k, \boldsymbol{c}_0^1 - \sum_{k=1}^{j+1}\boldsymbol{e}_k,\dots,\boldsymbol{c}_0^1 - \sum_{k=1}^{s-1}\boldsymbol{e}_k\Big)\\ &= \mathrm{AbsDet}\Big(\tfrac{1}{s}\sum_{j=1}^{s-1}\sum_{k=1}^{j}\boldsymbol{e}_k, \sum_{k=1}^{1}\boldsymbol{e}_k,\dots,\sum_{k=1}^{j-1}\boldsymbol{e}_k,\sum_{k=1}^{j+1}\boldsymbol{e}_k,\dots,\sum_{k=1}^{s-1}\boldsymbol{e}_k\Big)\\ &= \tfrac{1}{s}\,\mathrm{AbsDet}\Big(\sum_{k=1}^{j}\boldsymbol{e}_k, \sum_{k=1}^{1}\boldsymbol{e}_k,\dots,\sum_{k=1}^{j-1}\boldsymbol{e}_k,\sum_{k=1}^{j+1}\boldsymbol{e}_k,\dots,\sum_{k=1}^{s-1}\boldsymbol{e}_k\Big).\end{aligned}$$

These columns can be arranged into an upper triangular matrix with 1s on the diagonal. That matrix

has determinant 1, so $|\det(M_j^1)| = 1/s$. The remaining absolute minor has the same value:

$$\begin{aligned}
|\det(M_0^1)| &= \mathrm{AbsDet}\Big(\boldsymbol{c}_0^1 - \sum_{k=1}^{1} \boldsymbol{e}_k, \boldsymbol{c}_0^1 - \sum_{k=1}^{2} \boldsymbol{e}_k, \ldots, \boldsymbol{c}_0^1 - \sum_{k=1}^{s-1} \boldsymbol{e}_k\Big) \\
&= \mathrm{AbsDet}\Big(\boldsymbol{c}_0^1 - \sum_{k=1}^{1} \boldsymbol{e}_k, -\sum_{j=2}^{2} \boldsymbol{e}_k, \ldots, -\sum_{j=2}^{s-1} \boldsymbol{e}_k\Big) \\
&= \mathrm{AbsDet}(\boldsymbol{c}_0^1 - \boldsymbol{e}_1, -\boldsymbol{e}_2, \ldots, -\boldsymbol{e}_s) \\
&= |(\boldsymbol{c}_0^1 - \boldsymbol{e}_1)_1| = |(1 - \tfrac{1}{s}) - 1| = \tfrac{1}{s}.
\end{aligned}$$

Finally, consider Ψ^2. Ψ^2 is $(s-1) \times (s-1)$, so the only minor is $\det(\Psi^2)$. Observe that $\boldsymbol{c}_1^2 = -\frac{1}{s}\sum_{j=1}^{s-1}\sum_{k=1}^{j} \boldsymbol{e}_k$:

$$\Psi_{i,1}^2 = -s(1 - \tfrac{i}{s})\tfrac{1}{s} = -\tfrac{1}{s}(s-i) = -\tfrac{1}{s}\sum_{j=1}^{s-1}\sum_{k=1}^{j}\delta_{ik} = -\tfrac{1}{s}\Big(\sum_{j=1}^{s-1}\sum_{k=1}^{j}\boldsymbol{e}_k\Big)_i.$$

For $j \in \{2, \ldots, s-1\}$, we claim that $\boldsymbol{c}_j^2 = j\boldsymbol{c}_1^2 + \sum_{k=1}^{j-1}(j-k)\boldsymbol{e}_k$. Indeed, if $j \le i$,

$$\Psi_{i,j}^2 = -s(1 - \tfrac{i}{s})\tfrac{j}{s} = j\Psi_{i,1}^2 = \Big(j\boldsymbol{c}_1^2 + \sum_{k=1}^{j-1}(j-k)\boldsymbol{e}_k\Big)_i,$$

and if $j \ge i$,

$$\begin{aligned}
\Psi_{i,j}^2 &= -s(1 - \tfrac{j}{s})\tfrac{i}{s} = j\big[-s(1 - \tfrac{i}{s})\tfrac{1}{s}\big] - s\big[(1 - \tfrac{j}{s})\tfrac{i}{s} - (1 - \tfrac{i}{s})\tfrac{j}{s}\big] \\
&= j\Psi_{i,1}^2 + (j-i) = \Big(j\boldsymbol{c}_1^2 + \sum_{k=1}^{j-1}(j-k)\boldsymbol{e}_k\Big)_i.
\end{aligned}$$

Thus,

$$\begin{aligned}
|\det(\Psi^2)| &= \mathrm{AbsDet}\Big(\boldsymbol{c}_1^2, 2\boldsymbol{c}_1^2 + \sum_{k=1}^{1}(2-k)\boldsymbol{e}_k, \ldots, (s-1)\boldsymbol{c}_1^2 + \sum_{k=1}^{s-2}(s-1-k)\boldsymbol{e}_k\Big) \\
&= \mathrm{AbsDet}\Big(\boldsymbol{c}_1^2, \sum_{k=1}^{1}(2-k)\boldsymbol{e}_k, \ldots, \sum_{k=1}^{s-2}(s-1-k)\boldsymbol{e}_k\Big) \\
&= \mathrm{AbsDet}(\boldsymbol{c}_1^2, \boldsymbol{e}_1, \ldots, \boldsymbol{e}_{s-2}) = |\Psi_{s-1,1}^2| = \tfrac{1}{s}.
\end{aligned}$$

□

We can now use Lemma 2.13 to prove the following result.

Theorem 2.16.

(a) The volume of $\Psi^0[0,1]^{s+1}$ *is* $\frac{1}{6}(s+1)(s+2)$.

(b) The volume of $\Psi^1[0,1]^s$ *is* 1.

(c) The volume of $\Psi^2[0,1]^{s-1}$ *is* $1/s$.

Proof. Begin with Ψ^0. Combining Lemmas 2.13 and 2.15,

$$\operatorname{vol}_{s-1}(\Psi^0[0,1]^{s+1}) = \sum_{0\le j<k\le s} |\det(M^0_{j,k})| = 1 + 2\sum_{k=1}^{s-1} \tfrac{k}{s} + \sum_{j=1}^{s-2}\sum_{k=j+1}^{s-1} \tfrac{k-j}{s}$$
$$s\operatorname{vol}_{s-1}(\Psi^0[0,1]^{s+1}) = s + 2\sum_{k=1}^{s-1} k + \sum_{j=1}^{s-2}\sum_{k=j+1}^{s-1} (k-j) = \tfrac{1}{6}s(s+1)(s+2)$$

Next, consider Ψ^1. Combining Lemmas 2.13 and 2.15, $\operatorname{vol}_{s-1}(\Psi^1[0,1]^s) = \sum_{j=0}^{s-1} 1/s = 1$.

Finally, consider Ψ^2. Ψ^2 is a square matrix, so the volume of $\Psi^2[0,1]^{s-1}$ is given by the absolute value of the determinant of Ψ^2, which we computed in Lemma 2.15 to be $1/s$. □

Next, we will compute the determinants of $\Psi^0\Psi^{0\intercal}$, $\Psi^1\Psi^{1\intercal}$, and $\Psi^2\Psi^{2\intercal}$ using the following lemma, which relates the determinant of a linear mapping's Gramian matrix to the mapping's minors [GK10].

Lemma 2.17. *Let* $n \ge m$*, and let* $T\colon \mathbb{R}^n \to \mathbb{R}^m$ *be a linear transformation with full rank.* $\det(TT^\intercal)$ *is equal to the sum of the squared* $m \times m$ *minors of* T.

We showed in Lemma 2.14 that Ψ^0, Ψ^1, and Ψ^2 satisfy the hypotheses of Lemma 2.17, so we may now apply the latter to prove the following result.

Theorem 2.18.

(a) $\det(\Psi^0\Psi^{0\intercal}) = \frac{1}{12s}(s+1)^2(s+2)$.

(b) $\det(\Psi^1\Psi^{1\intercal}) = 1/s$.

(c) $\det(\Psi^2\Psi^{2\intercal}) = 1/s^2$.

Proof. Begin with Ψ^0. Combining Lemmas 2.15 and 2.17,

$$\begin{aligned}\det(\Psi^0\Psi^{0\intercal}) &= \sum_{0\le j<k\le s} \det(M^0_{j,k})^2 \\ &= 1^2 + 2\sum_{k=1}^{s-1} \left(\tfrac{k}{s}\right)^2 + \sum_{j=1}^{s-2}\sum_{k=j+1}^{s-1} \left(\tfrac{k-j}{s}\right)^2 = \tfrac{1}{12s}(s+1)^2(s+2).\end{aligned}$$

Next, consider Ψ^1. We claim that $\Psi^1\Psi^{1\intercal} = -\Psi^2$. Indeed, if $i \le j$,

$$(\Psi^1\Psi^{1\intercal})_{i,j} = \sum_{k=0}^{s-1} \Psi^1_{i,k}\Psi^{1\intercal}_{k,j} = s(1-\tfrac{j}{s})\tfrac{i}{s} = -\Psi^2_{i,j}.$$

The case $i \geq j$ follows by symmetry. Thus, $\det(\Psi^1 \Psi^{1\intercal}) = (-1)^{s-1} \det(\Psi^2)$. In Lemma 2.15 we found that $|\det(\Psi^2)| = 1/s$, so $\det(\Psi^1 \Psi^{1\intercal})$ is either $1/s$ or $-1/s$. It must be the former, since Gramian matrices have nonnegative determinants [GK10]. Alternatively, we can use Lemma 2.17 again: $\det(\Psi^1 \Psi^{1\intercal}) = \sum_{k=0}^{s-1} 1/s^2 = 1/s$.

Finally, consider Ψ^2. Using the basic properties of the determinant,

$$\det(\Psi^2 \Psi^{2\intercal}) = \det(\Psi^2) \det(\Psi^{2\intercal}) = \det(\Psi^2)^2 = 1/s^2. \qquad \square$$

Chapter 3

Univariate Multilevel Reduction

3.1 Introduction

In this chapter 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 while 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 while minimizing the lossiness.

In Section 3.2 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 3.3 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 Section 3.4 the algorithm is applied to the case of turbulence modelling, where the datasets are traditionally not only extremely large but

This chapter has been accepted to Computing and Visualization in Science with title *Multilevel Techniques for Compression and Reduction of Scientific Data—The Univariate Case* and coauthors Mark Ainsworth, Ozan Tugluk, and Scott Klasky.

inherently nonsmooth and, as such, rather resistant to compression. Note that wavelet techniques have been used successfully for modelling turbulence [SFPR05]. 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.

3.2 Algorithms for Data Reduction Based on Orthogonal Projection

Let $[a, b]$ be an interval on the real line partitioned into nonoverlapping 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 representation. 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 3.1 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.

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

Algorithm 3.1: Reduction given a constraint on the degrees of freedom available for the reduced representation.

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 3.2 gives a method for achieving this objective.

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

Algorithm 3.2: Reduction given a constraint on the maximum allowable error in the reduced representation.

Observe that Algorithm 3.2 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 Section 3.3.

The following result quantifies the performance of Algorithms 3.1 and 3.2 in terms of the Sobolev regularity r of the data u. We emphasize that in order to apply Algorithms 3.1 and 3.2 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 3.1. *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 3.1 gives a reduced representation using $n \leq N$ degrees of freedom while achieving a relative error $\epsilon \lesssim N^{-r}\|u\|_r/\|u\|$.*

2. *Let $\tau \in (0,1]$ be given. Then Algorithm 3.2 gives a reduced representation with relative error $\epsilon \leq \tau$ while requiring $n \lesssim (\tau\|u\|/\|u\|_r)^{-1/r}$ degrees of freedom.*

Moreover, the $L^2([a,b])$ projections used in Algorithms 3.1 and 3.2 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 $M \in \{0,\dots,L\}$ so that Algorithm 3.1 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

$$\|u - Q_M u\|^2 = \sum_{\ell=M+1}^{L} \|(Q_\ell - Q_{\ell-1})u\|^2. \tag{3.1}$$

It suffices to show that $\{(Q_\ell - Q_{\ell-1})u : M < \ell \leq L\}$ are pairwise orthogonal. Take $m, \ell \in \{M+1,\dots,L\}$ 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 (3.1) holds.

Next we wish to relate the righthand side of (3.1) 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 [BY93, Osw94]:

$$\|u\|_r^2 \simeq \sum_{\ell=0}^{L} 2^{2r\ell}\|(Q_\ell - Q_{\ell-1})u\|^2, \tag{3.2}$$

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 (3.2). Multiplying

each side of (3.1) 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

$$\|u - Q_M u\|^2 \lesssim 2^{-2r(M+1)}\|u\|_r^2, \tag{3.3}$$

again with constants independent of u (and M). As Algorithm 3.1 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 3.2 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 $M > 0$,

$$\|u - Q_M u\|^2 \leq \tau^2\|u\|^2 < \|u - Q_{M-1}u\|^2. \tag{3.4}$$

We remarked above that (3.3) 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 (3.4), 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 3.1 and 3.2. Consider any linear reduction operator $T\colon H^r([a, b]) \to L^2([a, b])$. 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) [ET96, 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 3.1, we consider datasets corresponding to four functions of known Sobolev regularities $r \in \{0.2, 0.6, 1.0, 1.4\}$. Figure 3.3 shows the dataset corresponding to the case $r = 0.2$ along with a selection of its projections to intermediate levels. The

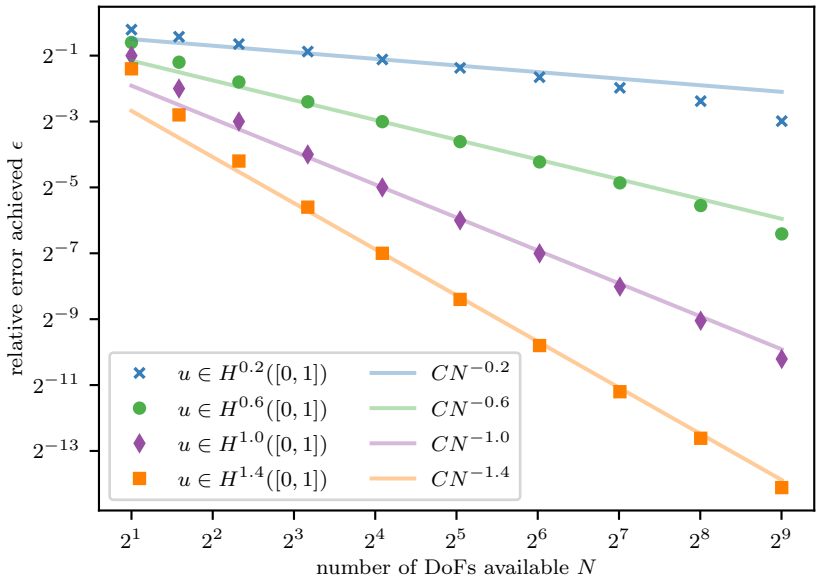

Figure 3.1: Illustration of the error decay as Algorithm 3.1 is applied to data using increasing numbers of degrees of freedom. As expected, the error decreases as the number of degrees of freedom used increases, and the decay is fastest for the most regular data. The dependence of the error on the number of degrees of freedom supplied agrees with Theorem 3.1.

performance of Algorithms 3.1 and 3.2 applied to each of the functions is illustrated in Figures 3.1 and 3.2. 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 3.1.

3.3 Implementation

As shown in the previous section, Algorithm 3.1 achieves quasioptimal errors when used to generate a reduced representation given a fixed number of degrees of freedom. Likewise, Algorithm 3.2 uses a quasioptimal number of degrees of freedom when used to generate a reduced representation achieving a fixed error bound. In practice, though, 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

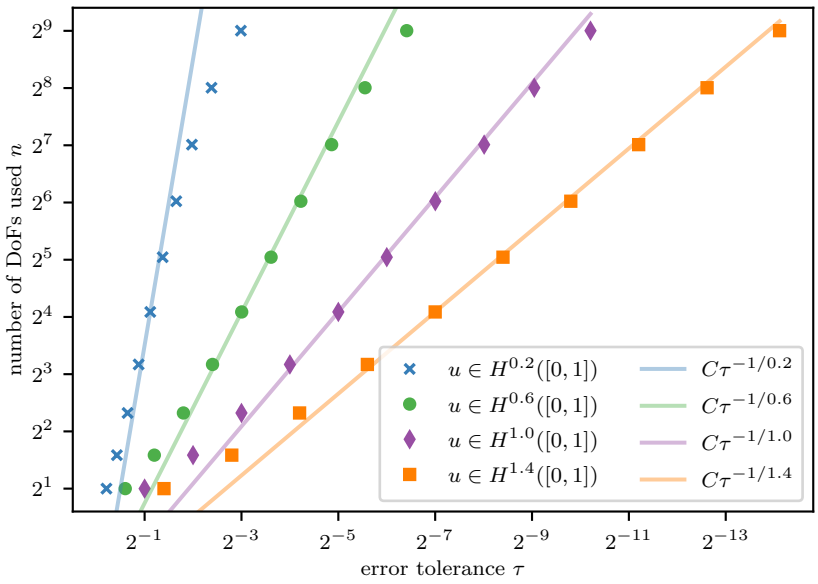

Figure 3.2: Illustration of the growth of number of degrees of freedom needed as Algorithm 3.2 is applied to data with decreasing error tolerances. As expected, more degrees of freedom are required to generate a reduced representation with greater accuracy, and the growth is fastest for the least regular data. The dependence of the number of degrees of freedom needed on the error tolerance agrees with Theorem 3.1.

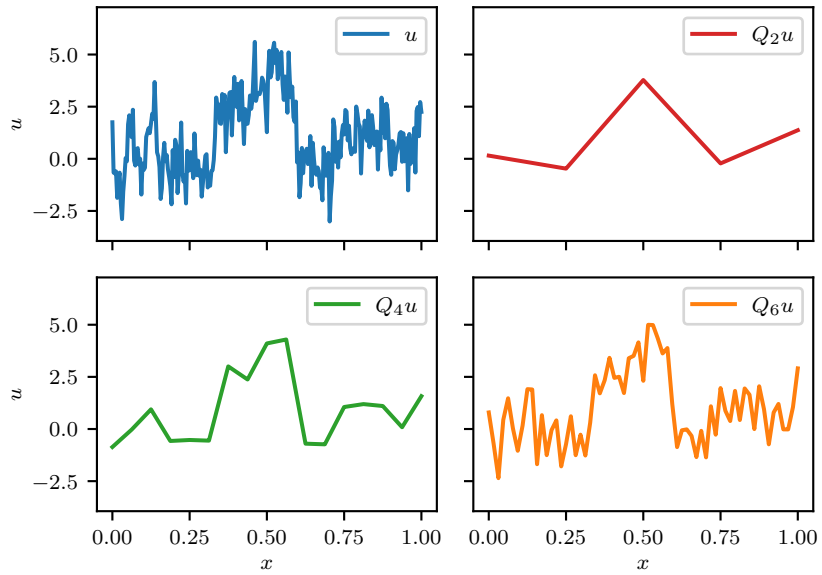

Figure 3.3: Illustration of a function $u \in H^{0.2}([a,b])$ used in Figures 3.1 and 3.2 along with three of its projections Q_2u, Q_4u, and Q_6u.

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 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 $\{Q_\ell u : 0 \leq \ell \leq L\}$ 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.

3.3.1 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:

$$Q_{\ell+1}u = (I - \Pi_\ell)Q_{\ell+1}u + Q_\ell u - \underbrace{(Q_\ell u - \Pi_\ell Q_{\ell+1}u)}_{z_\ell}, \tag{3.5}$$

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 (3.5) onto V_ℓ:

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

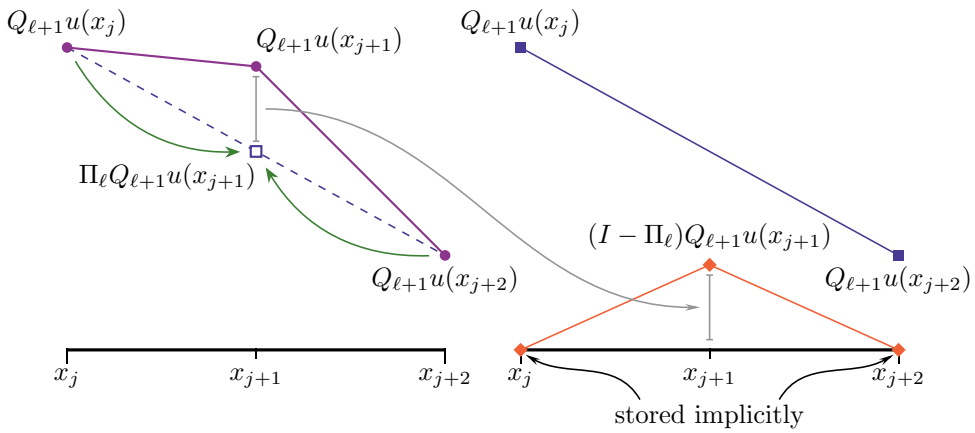

Figure 3.4: Illustration of the calculation of $(I - \Pi_\ell)Q_{\ell+1}u$ and storage of $\Pi_\ell Q_{\ell+1}u$. Here the nodes x_j and x_{j+2} belong to $\mathcal{N}_\ell$ and the node x_{j+1} belongs to $\mathcal{N}_{\ell+1} \setminus \mathcal{N}_\ell$. The value stored at each node x_{j+1} in $\mathcal{N}_{\ell+1} \setminus \mathcal{N}_\ell$ 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 $\mathcal{N}_{\ell+1} \setminus \mathcal{N}_\ell$ and the values of $\Pi_\ell Q_{\ell+1}u$ being stored on $\mathcal{N}_\ell$.

Hence, z_ℓ is the solution to the variational problem

$$\text{find } z_\ell \in V_\ell \text{ such that } (z_\ell, v_\ell) = ((I - \Pi_\ell)Q_{\ell+1}u, v_\ell) \text{ for all } v_\ell \in V_\ell \tag{3.6}$$

and thus it can be generated from the first term (see Subsection 3.3.3).

How can the decomposition in (3.5) 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 Figure 3.4. 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 Figure 3.4.

2. Compute z_ℓ by solving (3.6).

3. Add z_ℓ to $\Pi_\ell Q_{\ell+1}u$ to obtain $Q_\ell u$ 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

$$\underbrace{(I - \Pi_{L-1})Q_L}_{\text{on } \mathcal{N}_L \setminus \mathcal{N}_{L-1}}, \underbrace{(I - \Pi_{L-2})Q_{L-1}}_{\text{on } \mathcal{N}_{L-1} \setminus \mathcal{N}_{L-2}}, \ldots, \underbrace{(I - \Pi_0)Q_1}_{\text{on } \mathcal{N}_1 \setminus \mathcal{N}_0}, \underbrace{Q_0 u}_{\text{on } \mathcal{N}_0} . \tag{3.7}$$

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 3.2 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.

3.3.2 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 $\mathcal{N}_{\ell+1} \setminus \mathcal{N}_\ell$ and another of the nodal values of $Q_\ell u$ on $\mathcal{N}_\ell$. We can recompute $Q_{\ell+1}u$ using (3.5):

$$Q_{\ell+1}u = (I - \Pi_\ell)Q_{\ell+1}u + Q_\ell u - z_\ell. \tag{3.8}$$

Algorithmically, (3.8) 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 $\mathcal{N}_{\ell+1} \setminus \mathcal{N}_\ell$ 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 Figure 3.5.

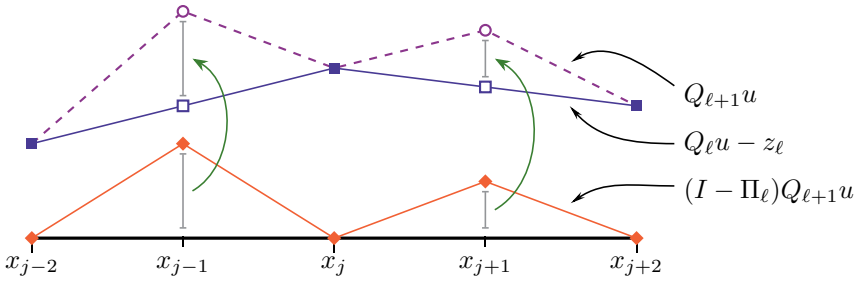

Figure 3.5: Illustration of the recomposition of $Q_{\ell+1}u = (I - \Pi_\ell)Q_{\ell+1}u + (Q_\ell u - z_\ell)$. Here the nodes x_{j-2}, x_j, and x_{j+2} belong to $\mathcal{N}_\ell$ and the nodes x_{j-1} and x_{j+1} belong to $\mathcal{N}_{\ell+1} \setminus \mathcal{N}_\ell$. We are given the values 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 $\mathcal{N}_{\ell+1} \setminus \mathcal{N}_\ell$. 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 $\mathcal{N}_{\ell+1} \setminus \mathcal{N}_\ell$.

By proceeding recursively, we can start with the decomposition given in (3.7) and recompute any of the projections $\{Q_\ell u : 0 \leq \ell \leq L\}$ in the hierarchy given the corrections $\{z_\ell : 0 \leq \ell < L\}$, which we compute as described in the next subsection.

3.3.3 Computation of the Correction

The correction z_ℓ is obtained by solving (3.6). We can reformulate (3.6) as a matrix equation by taking v_ℓ to be each of the coarse grid nodal basis functions $\{\phi_\ell(\,\cdot\,; x_j) : x_j \in \mathcal{N}_\ell\}$ 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

$$f^j = ((I - \Pi_\ell)Q_{\ell+1}u, \phi_\ell(\,\cdot\,; x_j)) = \tfrac{1}{4}h_\ell(\alpha^-_{\ell+1} + \alpha^+_{\ell+1}) \tag{3.9}$$

where the notation is given in Figure 3.6. 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 [GVL96, 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.

3.3.4 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

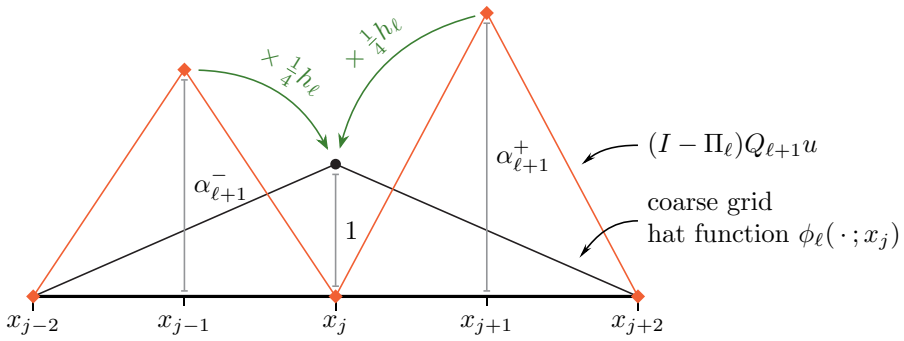

Figure 3.6: Notation used in the evaluation of the righthand side (3.9) used in the computation of the correction z_ℓ. Here x_{j-2}, x_j, and x_{j+2} belong to $\mathcal{N}_\ell$ and x_{j-1} and x_{j+1} belong to $\mathcal{N}_{\ell+1} \setminus \mathcal{N}_\ell$. The nodal values of $(I - \Pi_\ell) Q_{\ell+1} u$ on the neighboring nodes x_{j-1} and x_{j+1} are denoted by $\alpha^-_{\ell+1}$ and $\alpha^+_{\ell+1}$.

have shown:

Theorem 3.2. *The decomposition* (3.7) *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 righthand side $\vec{f}$ can be computed at a cost of $O(\#\mathcal{N}_\ell)$ operations using (3.9), 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 (3.5) requires the construction of $\{z_\ell : 0 \leq \ell < L\}$, 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 (3.7) also requires the computation of $\{z_\ell : 0 \leq \ell < M\}$ at a cost of $O(\#\mathcal{N}_0) + \cdots + O(\#\mathcal{N}_{M-1}) = O(\#\mathcal{N}_M)$ operations. □

3.4 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 [PBLM07, LPW+08]. 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 was applied and the approximation was only output at every tenth timestep. The resulting dataset, comprising 5028 flowfield instances with a storage timestep of $\Delta t = 2 \times 10^{-3}$, is in the public domain and available for download [PBLM07, LPW$^+$08, Joh17b]. 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 Section 3.2 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 postprocessing 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 visualization 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 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 3.2 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 projection 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.

3.4.1 Visualizations and Compression Ratios

First, we illustrate the performance of the reduction scheme when the primary interest is the visualization of the velocity at a particular timestep. Figure 3.7 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, while 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.

Figure 3.9 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 45 s 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.

3.4.2 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 [LPW+08] 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 3.10 shows that the time evolution of the

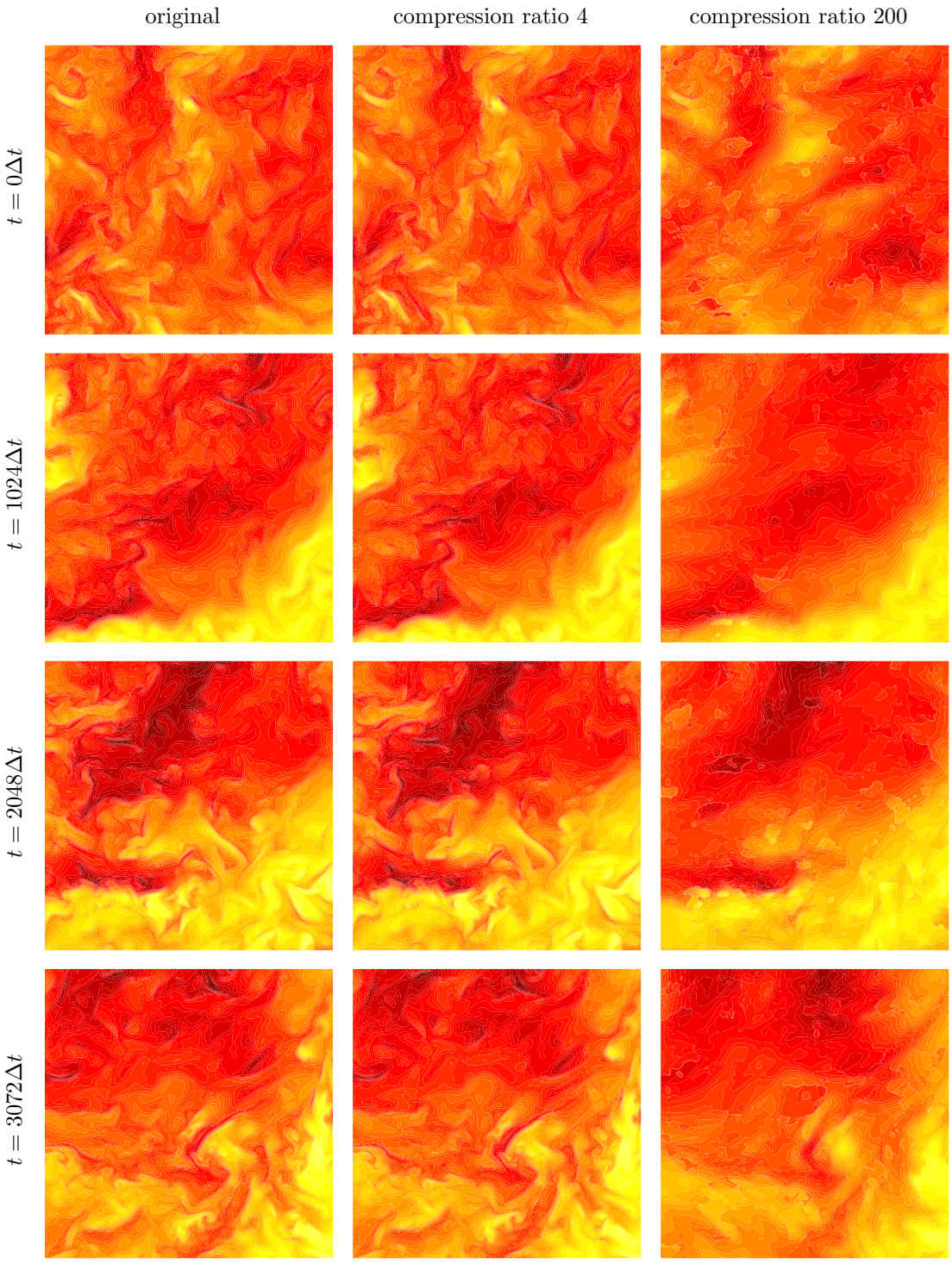

Figure 3.7: Contours of the velocity component normal to the slice at various timesteps obtained using the original and reconstructed datasets with $\epsilon = 10^{-2}$ and $\epsilon = 2.5 \times 10^{-1}$. Compression ratios of 4 and 200 are obtained with $\epsilon = 10^{-2}$ and $\epsilon = 2.5 \times 10^{-1}$, respectively.

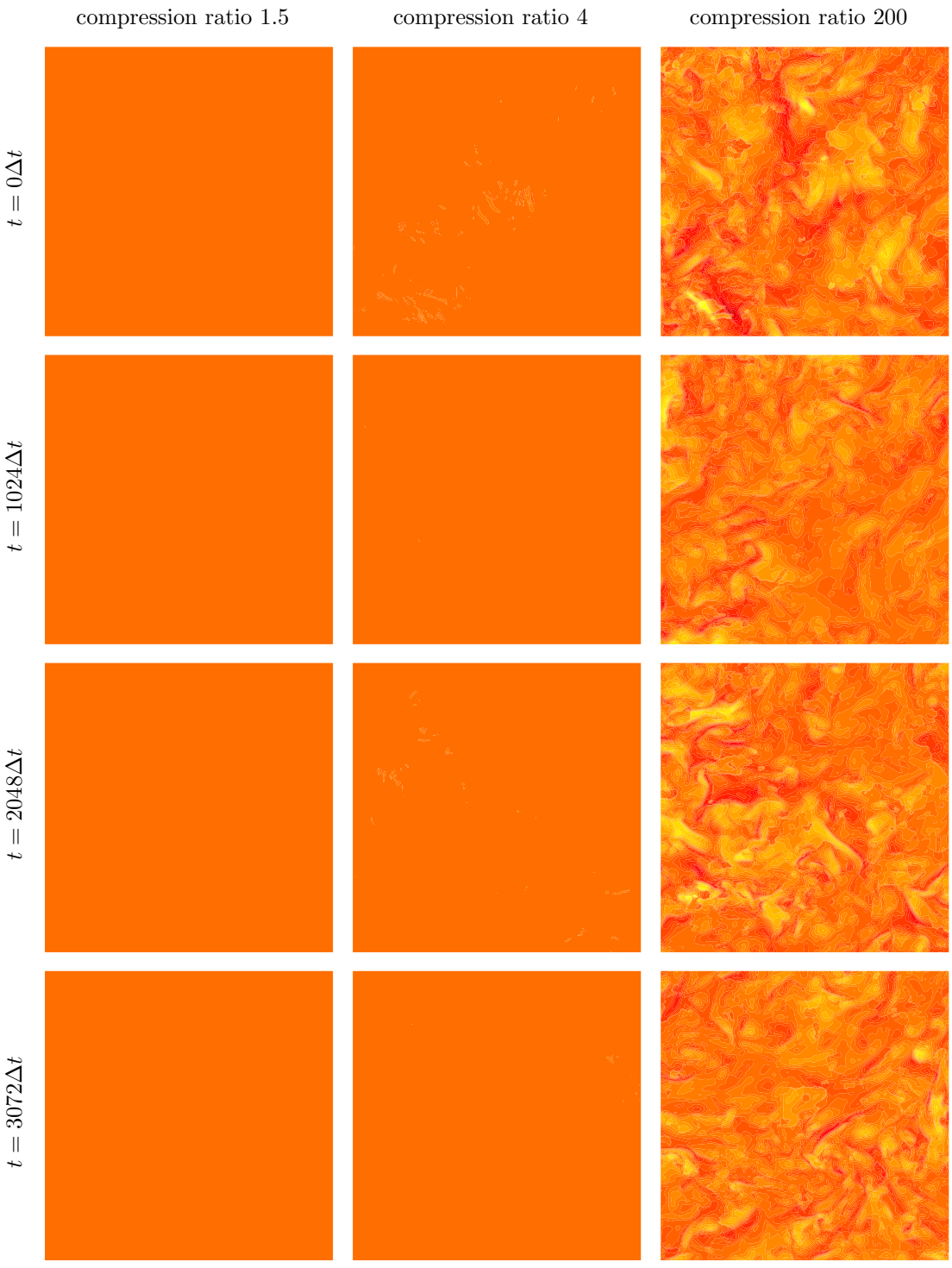

Figure 3.8: Pointwise deviation of the velocity component normal to the slice at various timesteps obtained using reconstructed datasets with $\epsilon = 10^{-3}$, $\epsilon = 10^{-2}$, and $\epsilon = 2.5 \times 10^{-1}$. The compression ratios obtained are 1.5, 4, and 200, respectively.

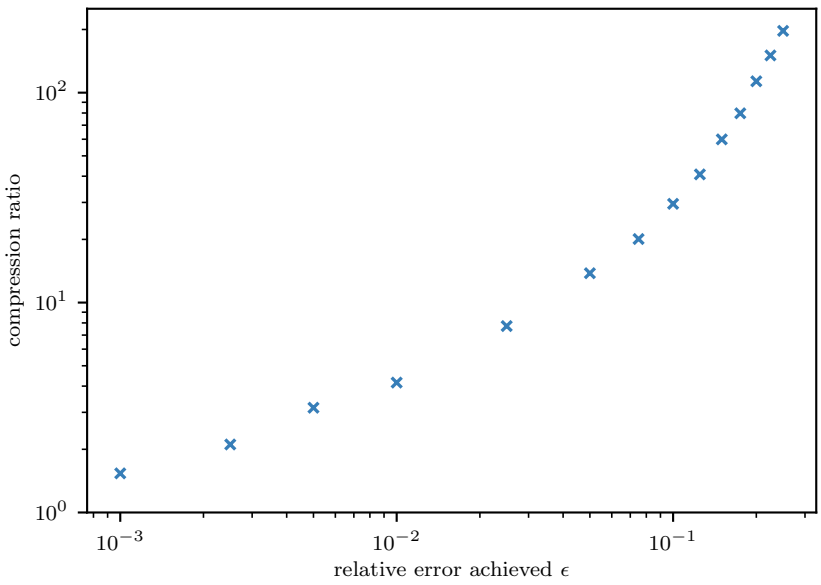

Figure 3.9: Illustration of the compression ratios achieved using various levels of lossiness.

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.

3.4.3 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 underresolution 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 distribution 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 3.11 presents the temporal power spectra for the original data and several reduced repre-

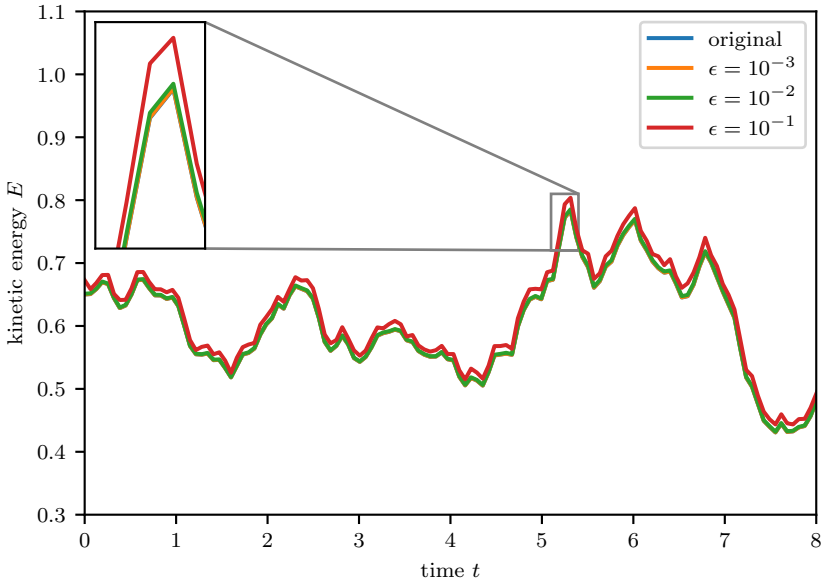

Figure 3.10: Time histories of kinetic energy obtained using reduced representations at the error levels ϵ indicated.

sentations. 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.

3.4.4 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 3.12 gives a

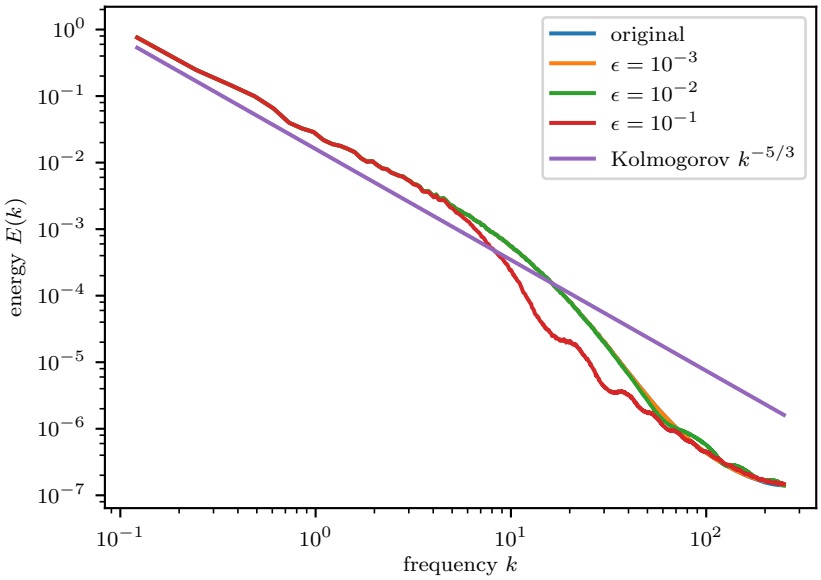

Figure 3.11: Temporal power spectra obtained using reduced representations at the error levels ϵ indicated. For comparison, the line $k^{-5/3}$ is also plotted [Kol41].

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.

3.5 Conclusion

The present chapter 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 [DK92, BY93, DVDD98, Osw94, GO17] 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 [BDY88] 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

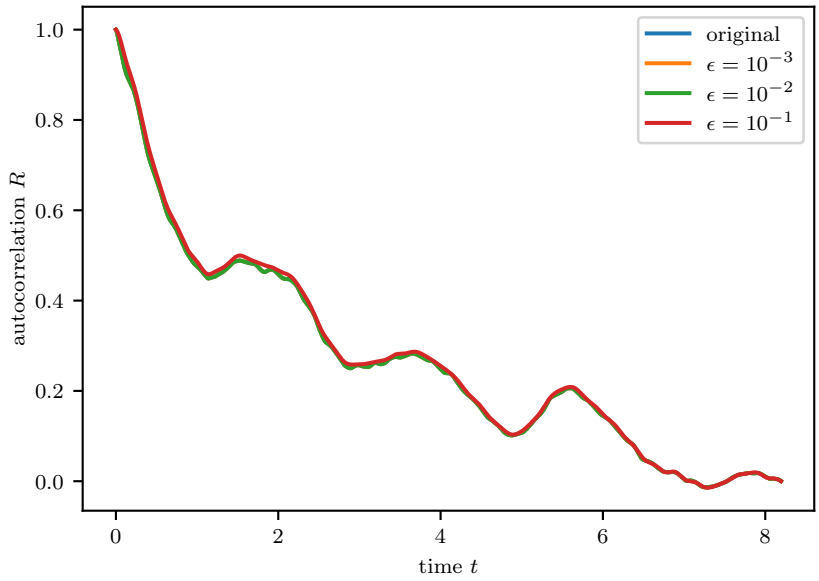

Figure 3.12: Temporal autocorrelation of the component of velocity normal to the slice at a typical gridpoint obtained using reduced representations at the error levels ϵ indicated.

quasioptimality results given in Theorem 3.1 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 $\{Q_\ell u : 0 \leq \ell \leq L\}$. In Section 3.3 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 3.2, 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 3.1 or 3.2. 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 chapter.

The final section, Section 3.4, 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 simulations of complex phenomena.

Chapter 4

Multivariate Multilevel Reduction with Pointwise Error Control

4.1 Introduction

In Chapter 3 we developed a hierarchical reduction method for univariate data on uniformly spaced grids. This chapter 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

This chapter has been submitted with title *Multilevel Techniques for Compression and Reduction of Scientific Data—The Multivariate Case* and coauthors Mark Ainsworth, Ozan Tugluk, and Scott Klasky. Ozan Tugluk performed the analyses and generated the figures for Subsections 4.4.2 and 4.4.3.

requirements of the user.

The remainder of this chapter is organized as follows. In Section 4.2 we present the abstract framework of the reduction technique, which is applied to the tensor product case in Section 4.3. In Section 4.4 we present applications to datasets arising from the simulation of a nonlinear reaction–diffusion problem, a turbulent channel flow, and a climate simulation. Section 4.5 concerns the implementation of the core decomposition and recomposition procedures. Details regarding the derivation of the loss indicator are given in Section 4.6.

4.2 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 time series may be decimated, with only a fraction of the original datapoints retained (see Chapter 2). In general, though, the second set of degrees of freedom may bear little or no relation to the first. In this chapter, 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 4.2.1 we establish the setting and define the degrees of freedom used. In Subsection 4.2.2 we develop algorithms for generating reduced representations of arbitrary accuracy or size from these degrees of freedom.

4.2.1 Notation and Degrees of Freedom

Let $\Omega \subset \mathbb{R}^d$ be a domain with a regular partition $\mathcal{P}_L$. 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, and let $\Pi_\ell \colon V_L \to V_\ell$ denote the piecewise linear interpolant onto V_ℓ.

As mentioned in Chapter 1, modern day storage systems are often hierarchical in nature, and data stored in 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 chapter, 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:

$$\begin{aligned} \sum_{\ell=0}^{L} (I - Q_{\ell-1})\Delta_\ell u &= \sum_{\ell=0}^{L} (I - \Pi_{\ell-1} - Q_{\ell-1} + Q_{\ell-1}\Pi_{\ell-1})Q_\ell u \\ &= \sum_{\ell=0}^{L} (I - Q_{\ell-1})Q_\ell u = \sum_{\ell=0}^{L} (Q_\ell - Q_{\ell-1})u = u \end{aligned} \tag{4.1}$$

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 $\{\Delta_\ell u : 0 \leq \ell \leq L\}$ as a splitting

$$V_L \ni u \longleftrightarrow (\Delta_0 u, \ldots, \Delta_L u) \in V_0 \times \cdots \times V_L. \tag{4.2}$$

At first glance, it appears that it will be $L + 1$ times as expensive to store the components in the *multilevel splitting* (4.2) 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

$$\Delta_\ell u(y) = \sum_{x \in \mathcal{N}_\ell} \Delta_\ell u(x)\phi_\ell(y; x) = \sum_{x \in \mathcal{N}_\ell \setminus \mathcal{N}_{\ell-1}} \Delta_\ell u(x)\phi_\ell(y; x), \tag{4.3}$$

where we have used the fact that $Q_\ell u(x) = \Pi_{\ell-1}Q_\ell u(x)$, and thus $\Delta_\ell u(x) = 0$, for all $x \in \mathcal{N}_{\ell-1}$. 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 basis decomposition* [BDY88]

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

The multilevel splitting is *not* the same as the hierarchical basis decomposition: as noted above, the sum of the multilevel components is not u. In fact, as (4.1) shows, the multilevel splitting is actually more akin to the *orthogonal decomposition* [BY93, Osw94]

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

which enjoys stability properties superior to those of the hierarchical basis 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 this chapter, enjoys the best of both worlds: compact storage and superior stability.

4.2.2 Data Reduction

The splitting defined by (4.2) will form the basis of our data reduction algorithms. As shown in (4.3), 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

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

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

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

In order to define the corresponding reduced representation $\tilde{u}$, we first use (4.1) 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

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

Based on the above definition, Algorithm 4.4 in Subsection 4.5.4 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 lefthand bound establishes that the indicator is *reliable* – that the true loss can always be guaranteed to meet a specified tolerance. The righthand bound establishes that 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 [AO11].

In general, we will be able to express any linear indicator $\eta_\infty(u - \tilde{u})$ explicitly as a function of

the difference between 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.

4.3 Multilevel Reduction on Tensor Product Grids

In this section we apply the general framework given in Section 4.2 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.

4.3.1 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:

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

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

$$\eta_\infty(v) = \sum_{\ell=0}^{L} \|\Delta_\ell v\|_{L^\infty} = \sum_{\ell=0}^{L} \Big\| \sum_{x \in \mathcal{N}_\ell^*} \texttt{v\_mc[}x\texttt{]}\, \phi_\ell(\,\cdot\,; x) \Big\|_{L^\infty} = \sum_{\ell=0}^{L} \max_{x \in \mathcal{N}_\ell^*} |\texttt{v\_mc[}x\texttt{]}|. \tag{4.4}$$

The following result establishes that η_∞ is an efficient and reliable loss indicator.

Theorem 4.1. *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 $C_1 \ge 3^{-d}/2$ and $C_2 \le 1 + 3^d$.

Proof. We begin with the lefthand bound. For any $v \in V_L$,

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

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 [CT87, Theorem 1], that $\|Q_\ell\|_{L^\infty} \leq 3$ when $d = 1$. For general d, then, $\|Q_\ell\|_{L^\infty} \leq 3^d$ by the definition of the tensor product. As a result, $\eta_\infty(v) \leq C_1^{-1} \|v\|_{L^\infty}$ with $C_1^{-1} \leq 2 \cdot 3^d$.

We next show that $\|Q_\ell\|_{L^\infty} \leq 3$ 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 $\|(I + K)^{-1}\|_{\ell^\infty} \leq \sum_{m=0}^{\infty} \|K\|_{\ell^\infty}^m = 2$, 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 $\ell^\infty(\mathbb{R}^n)$,

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

Another integration shows $\|\phi_\ell(\,\cdot\,; x_i)\|_{L^1} = (h_i^- + h_i^+)/2 = 3M_{i,i}/2$, so $|\vec{\alpha}|_{L^\infty} \leq 3\|u\|_{L^\infty}$. As

$|\vec{\alpha}|_{L^\infty} = \|Q_\ell v\|_{L^\infty}$, we conclude that $\|Q_\ell\|_{L^\infty} \leq 3$. This completes the derivation of the lefthand bound.

Now we turn to the righthand 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,

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

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

Theorem 4.1 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 lefthand bound can be improved as follows.

Theorem 4.2. *Let V_L be the space of continuous piecewise multilinear functions on a uniform tensor product grid in d dimensions. Then*

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

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

This improved result follows from Theorem 4.1 and Section 4.6, where it is shown that $\|I - Q_{\ell-1}\|_{L^\infty} \leq 1 + (\sqrt{3}/2)^d$ in the case of uniform meshes. Alternative specializations can be obtained when a particular mesh is in hand. For example, in Subsection 4.4.3 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 4.3. *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*

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

with $C_1 \geq 3^{-3}/2$ and $C_2 \leq 1 + 3^2/2^2$.

4.3.2 Reduction Algorithms

We can now develop algorithms for the use cases of constrained loss and constrained storage. We begin with constrained loss. Given a function $u \in V_L$ and some tolerance $\tau \geq 0$, we seek a reduced representation $\tilde{u}$ represented by a minimal number of nonzero degrees of freedom such that $\|u - \tilde{u}\|_{L^\infty} / \|u\|_{L^\infty} \leq \tau$. 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 (4.4). 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,

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

Taking the maximum over all levels ℓ,

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

Using Theorem 4.1, it suffices to find $\tilde{u}$ such that $\eta_\infty(u - \tilde{u}) \leq \tau \|u\|_{L^\infty} / C_2$. 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 4.1.

Require: $\tau \geq 0$.
  function REDUCE(multilevel coefficients `u_mc`, norm $\|u\|_{L^\infty}$, maximum level L, tolerance τ)
    `ũ_mc` $\leftarrow$ `[]`
    for $\ell = 0, \ldots, L$ **do**
      for $x \in \mathcal{N}_\ell^*$ **do**
        if $|$`u_mc[`x`]`$| > (\tau \|u\|_{L^\infty})/(C_2(L+1))$ **then**
          `ũ_mc[`x`]` $\leftarrow$ `u_mc[`x`]`
        else
          `ũ_mc[`x`]` $\leftarrow 0$
        end if
      end for
    end for
    return `ũ_mc`
  end function

Algorithm 4.1: Reduction given a constraint on the maximum allowable loss in the reduced representation.

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

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

This *knapsack problem* [BT97] can be solved to give the set $\mathcal{N}$ of N coefficients which should be retained. For simplicity we instead choose to minimize

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

giving rise to Algorithm 4.2.

Require: $N \geq 0$.
  function REDUCE(multilevel coefficients `u_mc`, number of degrees of freedom N)
    `ũ_mc` $\leftarrow$ `[]`
    for $\ell = 0, \ldots, L$ **do**
      for $x \in \mathcal{N}_\ell^*$ **do**
        if $|$`u_mc[`x`]`$|$ is among the N greatest of `u_mc` **then**
          `ũ_mc[`x`]` $\leftarrow$ `u_mc[`x`]`
        else
          `ũ_mc[`x`]` $\leftarrow 0$
        end if
      end for
    end for
    return `ũ_mc`
  end function

Algorithm 4.2: Reduction given a constraint on the degrees of freedom available for the reduced representation.

4.4 Application Examples

The pair of reduction algorithms developed in Section 4.3 together constitute a MultiGrid Adaptive Reduction of Data algorithm, which we call `MGARD`. We now apply `MGARD` to three representative examples. In Subsection 4.4.1, 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 4.4.2, we compare the performance of `MGARD` with existing state-of-the-art reduction algorithms in the context of meteorological data; and in Subsection 4.4.3, we demonstrate the flexibility of our approach by applying it to the reduction of data with nonuniform spatial grids.

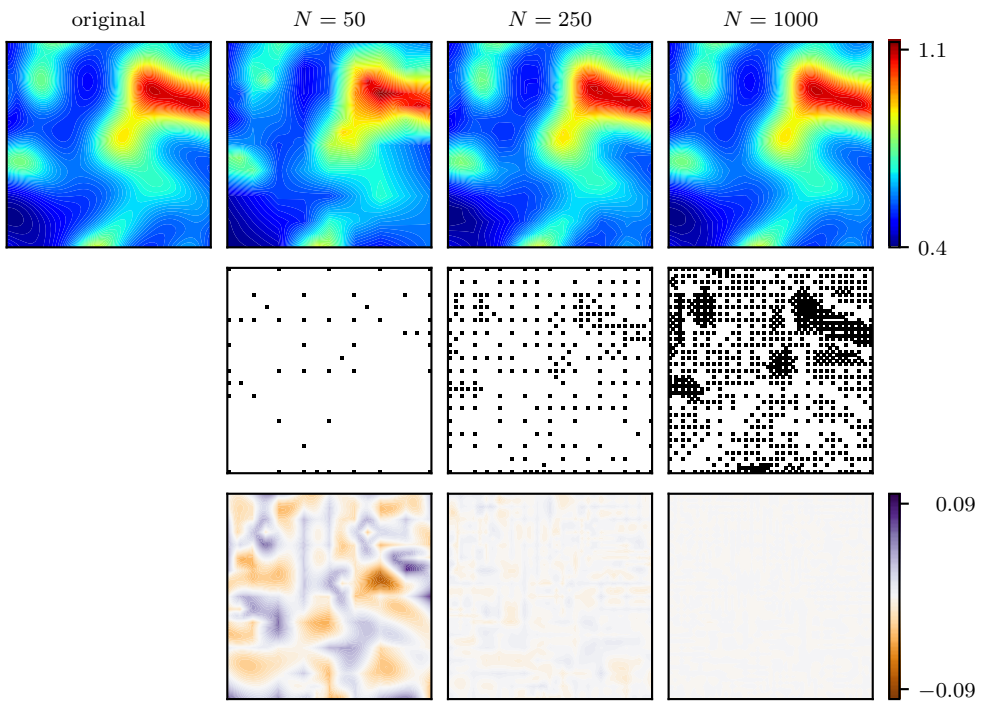

Figure 4.1: Illustration of the effect of applying Algorithm 4.2 to the concentration of U at timestep 200. The first row shows the original dataset u alongside the reduced representations $\tilde{u}$ obtained using Algorithm 4.2 with $N \in \{50, 250, 1000\}$. The third row shows the losses $u - \tilde{u}$ 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.

4.4.1 Brusselator Dataset

We now illustrate the performance of Algorithms 4.1 and 4.2 by applying them to the output of a Brusselator simulation. The Brusselator is a model of a chemical oscillator [TGC99]. Two species U and V interact according to

$$U_t = B + U^2V - (A + 1)U + \alpha\nabla^2 U$$
$$V_t = AU - U^2V + \alpha\nabla^2 V.$$

A predictor–corrector scheme adapted from [GTAF97] 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. Figure 4.1 shows the concentration of U, the first reactant, at timestep 200 along with the reduced representations obtained by applying Algorithm 4.2 with $N \in \{50, 250, 1000\}$.

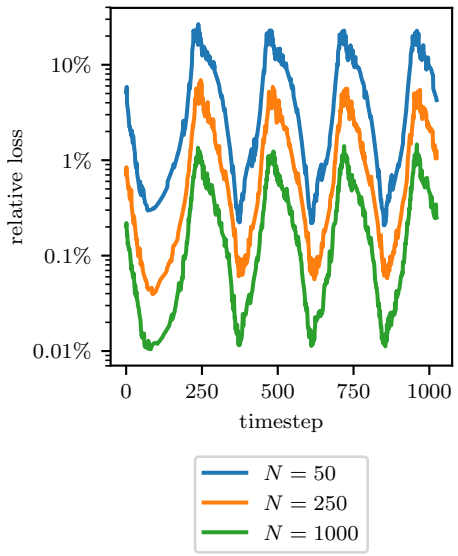

Figure 4.2: Illustration of the loss incurred by Algorithm 4.2 over time. The loss experienced varies according to the regularity of the solution over time, with the same pattern observed for different storage constraints N.

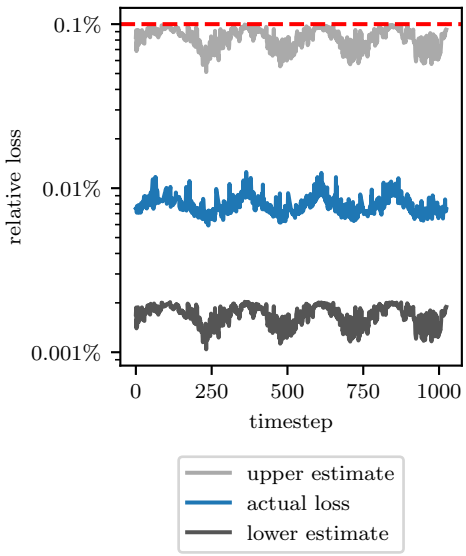

Figure 4.3: Illustration of the loss incurred by Algorithm 4.1 with $\tau = 0.1\%$ and the bounds of Theorem 4.1. The actual loss $\|u - \tilde{u}\|_{L^\infty}$ is controlled by the loss indicator $\eta_\infty(u - \tilde{u})$, as desired.

Algorithm 4.2 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. The compression ratio achieved is, therefore, constant across timesteps for a given N. The loss, though, depends on the character of the solution at a given timestep. When the reaction is quiescent, little loss in incurred; when the concentrations are in a state of flux, a greater loss is experienced. This phenomenon is illustrated in Figure 4.2.

For some applications, this variable loss may be unacceptable. As mentioned previously, many users will only use lossy algorithms if control of the loss can be guaranteed. For these use cases, Algorithm 4.1 is more appropriate. Figure 4.3 illustrates the loss caused by applying Algorithm 4.1 with tolerance $\tau = 0.1\%$ to each timestep of the solution.

Thus far, Algorithms 4.1 and 4.2 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.

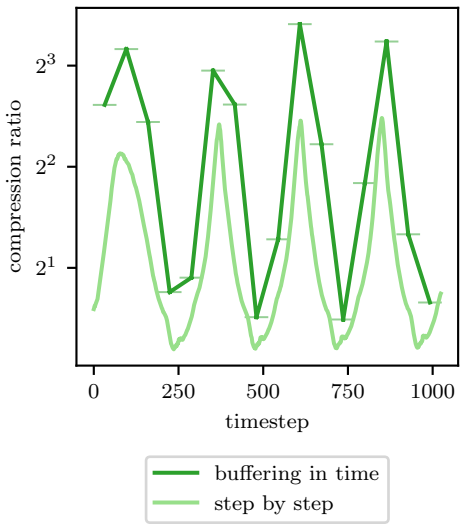

Figure 4.4: Illustration of the results obtained when applying Algorithm 4.1 (with $\tau = 1\%$ fixed) in time as well as space. The loss tolerance is always met, with better results achieved when the reduction is applied to more timesteps at once.

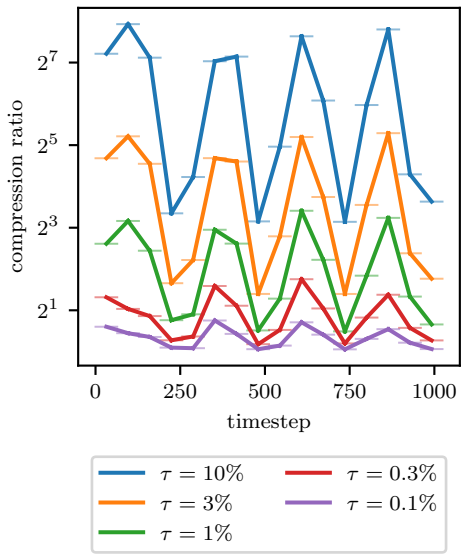

Figure 4.5: Illustration of the compression ratio achieved by Algorithm 4.2 over time. As with the loss in Figure 4.2, the compression ratio varies according to the regularity of the solution over time, with the same pattern observed for different tolerances τ.

Figure 4.4, which compares the compression ratios achieved by Algorithm 4.1 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 4.5, analogous to Figure 4.2, illustrates the compression ratios achieved when applying the reduction in space and time and varying the tolerance τ used.

4.4.2 CESM Atmosphere Data

We now apply `MGARD` to data obtained from the CESM Atmosphere model (CESM-ATM) [BXD+14]. The data comprises single-precision floating point data on a uniformly spaced 1800×3600 grid. Figure 4.6 shows the original data along with decompressed datasets obtained using various specified tolerances. Table 4.1 shows the compression ratios obtained using `MGARD`, `SZ` [DC16], and `zfp` [Lin14]. These preliminary results show that `MGARD` is competitive with existing state-of-the-art compressors.

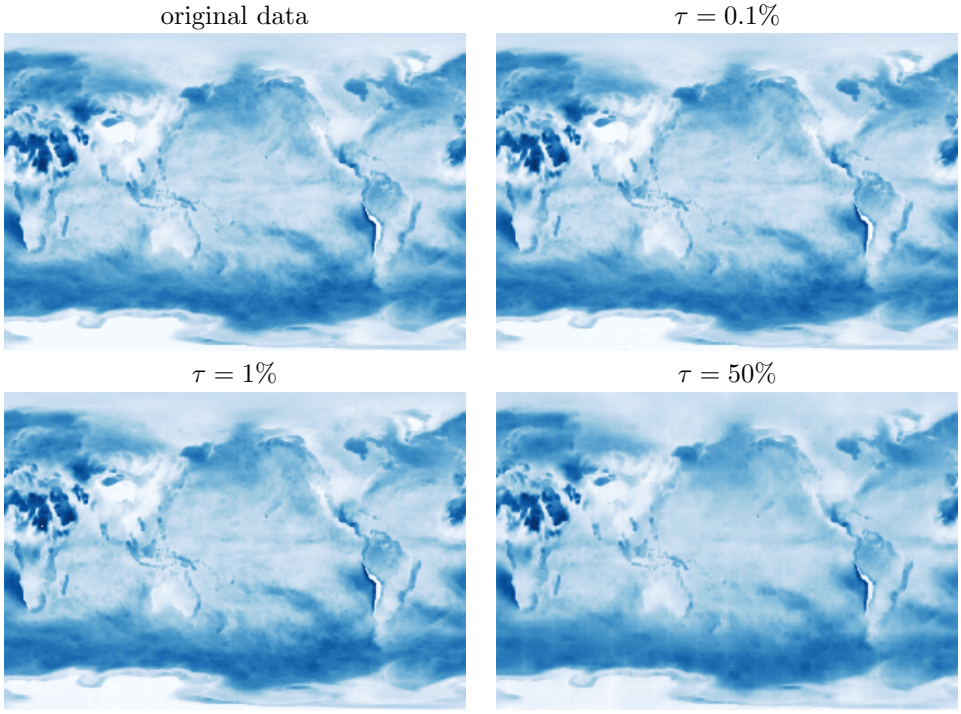

Figure 4.6: Illustration of the effect of compression using `MGARD` on the CLDLOW data [BXD+14] with indicated tolerances τ.

τ	`MGARD`	`SZ`	`zfp`
0.0001%	2.03	2.32	1.76
0.1%	6.46	10.05	3.68
1%	22.62	28.23	4.38
10%	125.28	136.45	6.76
50%	999.23	1030.01	9.75

Table 4.1: Compression ratios obtained for reduction of the CLDLOW data [BXD+14]. The results were obtained using `zfp` version 0.5.1 and `SZ` version 1.4.9.3.

loss	compression ratio
1%	5.71
5%	14.38
10%	26.14
15%	35.34

Table 4.2: Compression ratios and attained loss versus prescribed tolerance for channel flow data.

4.4.3 High Reynolds Number Channel Flow

In this subsection we consider data [PBLM07, LPW$^+$08, Joh17a] obtained by a direct numerical simulation [LM15] of channel flow (one of the canonical examples of turbulent flow) at Reynolds number $\mathrm{Re}_\tau = 9.9935 \times 10^2$. 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 apply `MGARD` to 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 4.3). The overall reduction obtained with a range of tolerances is reported in Subsection 4.4.3. Figures 4.7 and 4.8 display the compression ratios obtained for each velocity component on each z-slice when using loss tolerances of 1% and 10%. Figures 4.9 and 4.10 show the axial velocity fields of the reduced 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.

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 4.11 to 4.13 show the mean axial velocity, the variance of the turbulent velocity fluctuations $\langle u_i' u_i' \rangle_{xz}$ versus the wall distance $y^+ = (1 + y)\mathrm{Re}_\tau$, and the Reynolds stress $\langle u_1' u_2' \rangle_{xz}$ versus y^+. In all cases, we observe that reduction with a relative error of 1% 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.

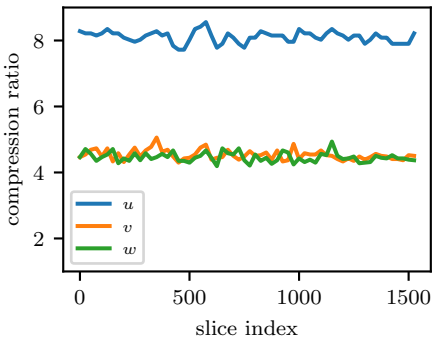

Figure 4.7: Compression ratio per z-slice for each velocity component with 1% loss.

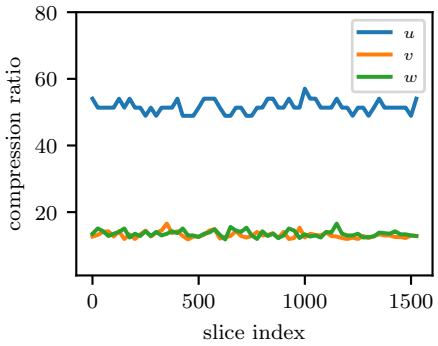

Figure 4.8: Compression ratio per z-slice for each velocity component with 10% loss.

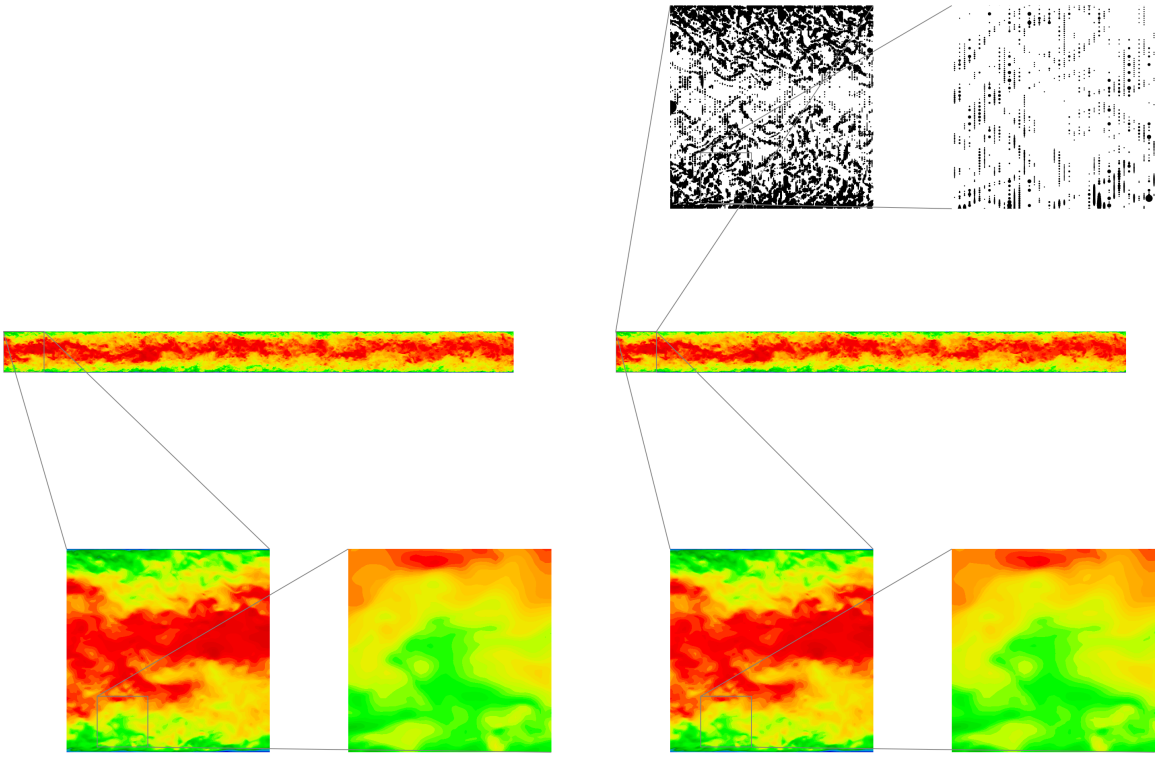

Figure 4.9: Contours of the axial velocity at the channel center. The middle row displays contours along the full length of the channel in the true (left) and reduced (right, using 1% tolerance) datasets. The bottom row displays 4 × zooms of selected square regions of the flows. The top row displays the locations of the retained coefficients.

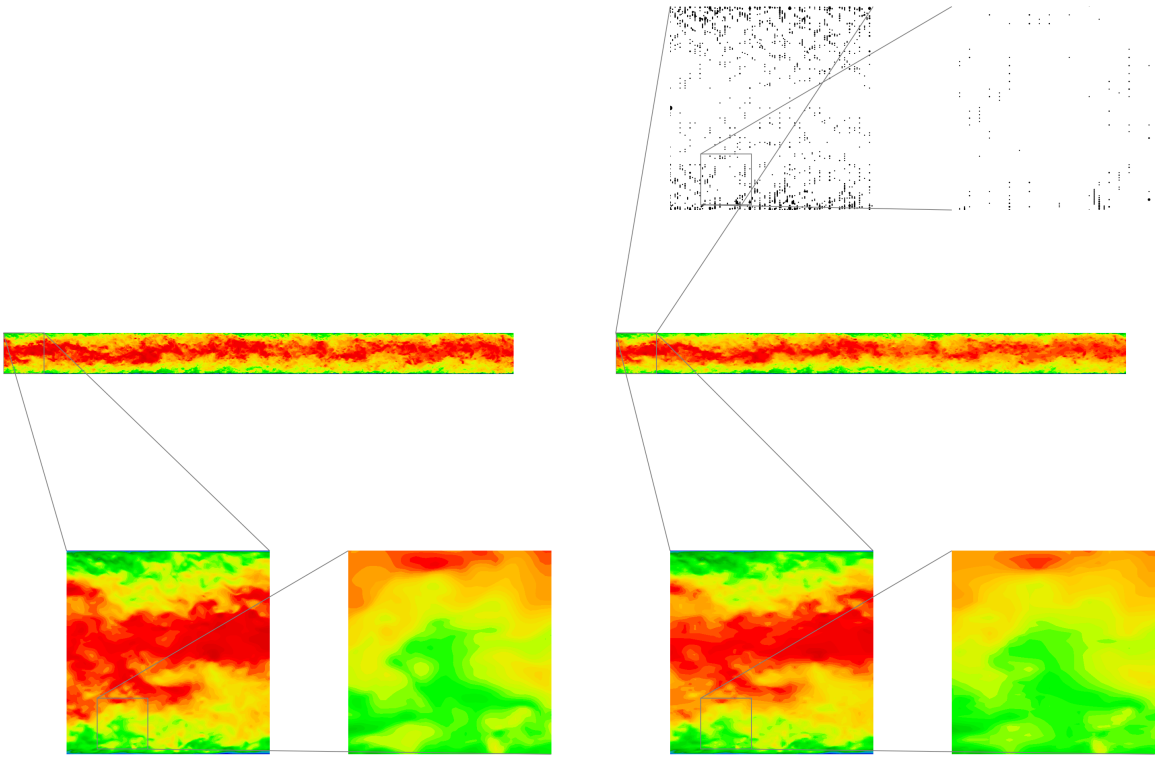

Figure 4.10: Contours of the axial velocity at the channel center. The middle row displays contours along the full length of the channel in the true (left) and reduced (right, using 10% tolerance) datasets. The bottom row displays 4 × zooms of selected square regions of the flows. The top row displays the locations of the retained coefficients.

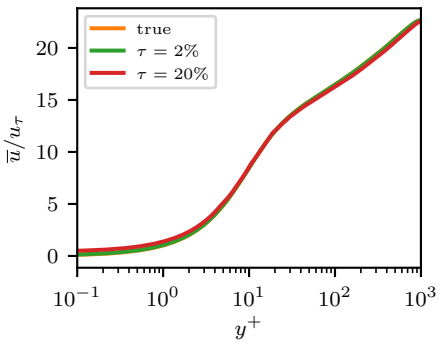

Figure 4.11: Mean axial velocity profile versus the wall coordinate y^+.

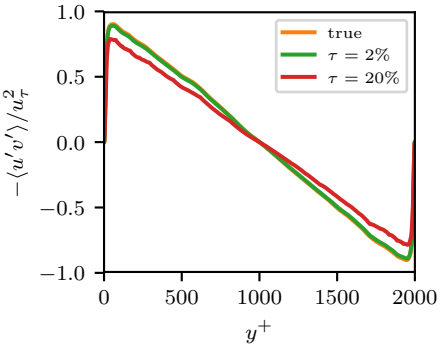

Figure 4.12: Reynolds stress $-\langle uv \rangle$ versus the wall coordinate y^+.

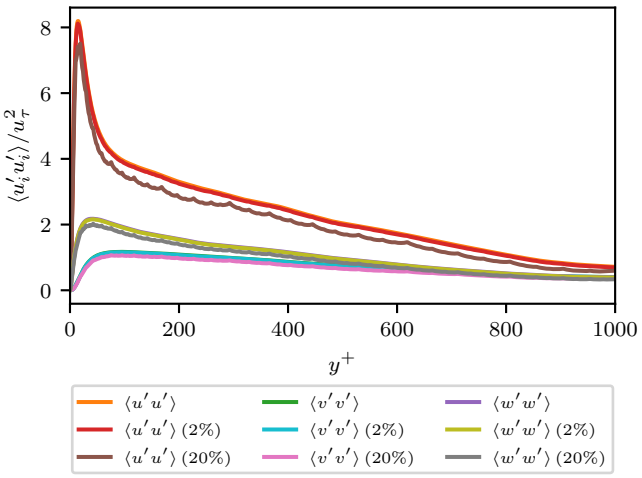

Figure 4.13: Variance of the turbulent velocity fluctuations versus the wall coordinate y^+.

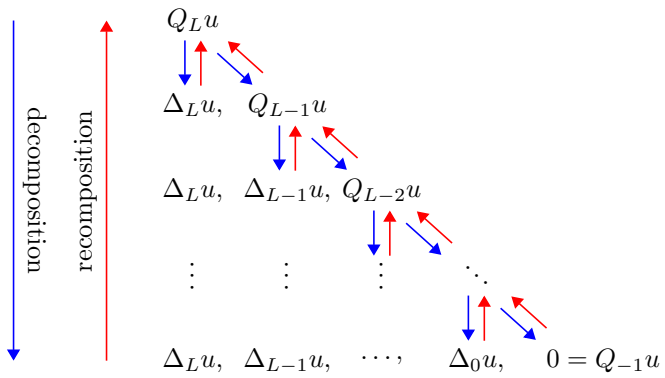

Figure 4.14: Illustration of the steps in the recursive decomposition and recomposition procedures.

4.5 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 4.14 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 $Q_\ell u$.

4.5.1 Decomposition

As illustrated in Figure 4.14, 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 of freedom originally needed to represent $Q_\ell u$. The key idea is based on the following decomposition:

$$Q_\ell u = (I - \Pi_{\ell-1}) Q_\ell u + Q_{\ell-1} u - \underbrace{(Q_{\ell-1} u - \Pi_{\ell-1} Q_\ell u)}_{z_{\ell-1}}. \tag{4.5}$$

Consider how each of the three righthand terms can be represented. As discussed in Subsection 4.2.1, 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

$$\text{find } z_{\ell-1}\in V_{\ell-1} \ : \ (z_{\ell-1},v_{\ell-1}) = ((I-\Pi_{\ell-1})Q_\ell u, v_{\ell-1}) \text{ for all } v_{\ell-1}\in V_{\ell-1} \tag{4.6}$$

and thus it can be generated from the data $(I-\Pi_{\ell-1})Q_\ell u$ (see Subsection 4.5.3).

How can the decomposition in (4.5) 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:

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 (4.6).
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

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

We can complete the decomposition by proceeding recursively, as illustrated in Figure 4.15.

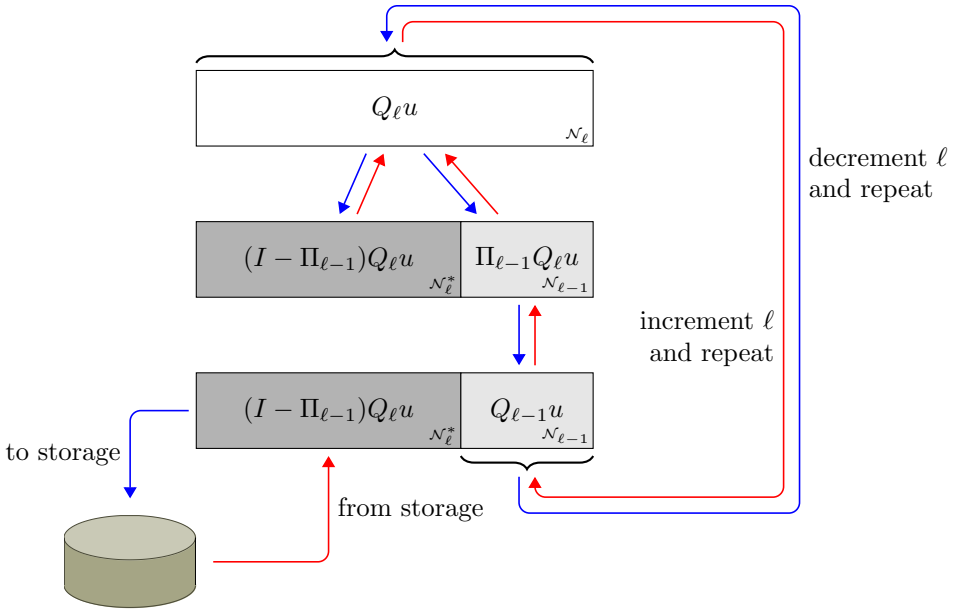

Figure 4.15: Illustration of recursive procedures for the decomposition into multilevel coefficients and the recomposition back to nodal coefficients.

4.5.2 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 4.15, 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 (4.5):

$$Q_\ell u = (I - \Pi_{\ell-1})Q_\ell u + Q_{\ell-1}u - z_{\ell-1}. \tag{4.7}$$

Algorithmically, the sum in (4.7) 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

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

Proceeding recursively, as illustrated in Figure 4.15, 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.

4.5.3 Computation of the Correction

The correction $z_{\ell-1}$ is obtained by solving (4.6). 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 $O(\max \#\mathcal{N}_{\ell-1}^i)$ [GVL96, 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 $\vec{\mathtt{f}}_{\mathtt{l-1}}[x] = ((I - \Pi_{\ell-1})Q_\ell u, \phi_{\ell-1}(\,\cdot\,;x))$ 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 $\{R_\ell^i : 1 \le i \le d\}$. 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 $M_\ell\vec{\alpha}$.

4.5.4 Summary

The full decomposition and recomposition procedures are summarized in Algorithms 4.3 and 4.4. 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 $\vec{\mathtt{z}}_{\mathtt{l\text{-}1}}$. 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 4.4. *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 $\{\Delta_\ell u : 0 \le \ell \le L\}$ 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 righthand side $\vec{\mathtt{f}}_{\mathtt{l\text{-}1}}$ 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 $\{z_{\ell-1} : 0 < \ell \le L\}$, 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 $\{z_{\ell-1} : 0 < \ell \le L\}$ at a cost of $O(\#\mathcal{N}_L)$ operations. □

---

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 $\Pi_{\ell-1} Q_\ell u$.
    $\vec{\mathtt{f}}_{\mathtt{l\text{-}1}} \leftarrow R_\ell M_\ell$`coeffs[`$\mathcal{N}_\ell^*$`]`
    $\vec{\mathtt{z}}_{\mathtt{l\text{-}1}} \leftarrow M_{\ell-1}^{-1} \vec{\mathtt{f}}_{\mathtt{l\text{-}1}}$ //$\vec{\mathtt{z}}_{\mathtt{l\text{-}1}}$ now represents $z_{\ell-1}$.
    for $x \in \mathcal{N}_{\ell-1}$ **do**
      `coeffs[`x`]` $\leftarrow$ `coeffs[`x`]` $+ \vec{\mathtt{z}}_{\mathtt{l\text{-}1}}$`[`$x$`]`
    end for //`coeffs[`$\mathcal{N}_{\ell-1}$`]` now represents $Q_{\ell-1} u$.
  end for
  return `coeffs`
end function

Algorithm 4.3: Transformation from nodal to multilevel coefficients.

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

The proof of Theorem 4.2 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.

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

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

Algorithm 4.4: Transformation from multilevel to nodal coefficients.

$\cosh_\Lambda, \sinh_\Lambda \colon \mathbb{R} \to \mathbb{R}$ 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 4.5. *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 $\{a_j : 0 \le j < x^{\mathrm{L}}\}$ the nodal values $\{Q_{\ell-1}\phi_\ell(\,\cdot\,; x)(y) : y \in \mathcal{N}_{\ell-1} \text{ and } y < x\}$, 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

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

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 $\alpha_0 = \alpha_1$:

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

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 $\{b_j : 0 \leq j < x^{\mathrm{R}}\}$ 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 $\{a_j : 0 \leq j < x^{\mathrm{L}}\}$ and $\{b_j : 0 \leq j < x^{\mathrm{R}}\}$ 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

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

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

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

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

$$\begin{aligned}\cosh_\lambda(n) &= (-1)^n\cosh_\Lambda(n)\\ \cosh_\Lambda(N+1)-\cosh_\Lambda(N-1) &= 2\sinh_\Lambda(N)\sinh_\Lambda(1).\end{aligned}$$

□

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

Lemma 4.6. *For $y \in \mathcal{N}_{\ell-1}$,*

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

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 4.5

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,

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

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

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

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

Corollary 4.7.

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

Proof. By Lemma 4.6,

$$\max_{y\in\mathcal{N}_{\ell-1}} \sum_{x\in\mathcal{N}_\ell^*} |Q_{\ell-1}\phi_\ell(\,\cdot\,;x)(y)| = \max_{y\in\mathcal{N}_{\ell-1}} 3\frac{\begin{array}{c}\cosh_\Lambda(N) + \cosh_\Lambda(y^{\mathrm{R}} - y^{\mathrm{L}})\\ -\cosh_\Lambda(y^{\mathrm{R}}) - \cosh_\Lambda(y^{\mathrm{L}})\end{array}}{2\sinh_\Lambda(N)\sinh_\Lambda(1)}. \tag{4.8}$$

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,

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

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

$$2\sinh_\Lambda(N/2)\sinh_\Lambda(N/2) \le 2\cosh_\Lambda(N/2)\sinh_\Lambda(N/2)$$
$$\cosh_\Lambda(N)-\cosh_\Lambda(0) \le \sinh_\Lambda(N)-\sinh_\Lambda(0) = \sinh_\Lambda(N).$$

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 4.2.

Corollary 4.8. *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 4.8 follows from Corollary 4.7, 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 $I - Q_{\ell-1}$.

Chapter 5

Multivariate Multilevel Reduction with Control of Quantities of Interest

5.1 Introduction

Compression methods for scientific data are a topic of interest because of the difficulty of transmitting, storing, analyzing, and understanding the mountains of simulation and experiment data produced each year. Though compression is typically separated from analysis as a distinct step of the scientific workflow, the two share a common goal: to extract from a mass of raw data the essential structure and key features of the phenomenon under study while ignoring or discarding the noise and simulation errors that contain no information of interest. Even so, compression is often viewed as a hindrance, begrudgingly tolerated out of practical necessity, that interferes with the project of understanding a dataset, and practitioners commonly restrict their use of compression to lossless methods, which make no change to the input data.

This blinkered approach to compression is ill-advised for several reasons. The first is practical: lossless algorithms have generally been unable to achieve the high compression ratios desired by users. More philosophically, there is noise in every measurement and error inherent to any numerical method, so to insist on bitwise reproduction of one's input data is to burn computational resources

treating meaningless errors with the same reverence afforded the scientifically relevant aspects of the data. Finally, there is the issue of how much information will realistically ever be extracted from a given dataset. A dataset of size 1 GiB is not, in all likelihood, being used to answer 2^{30} 'yes or no' questions. Instead, it was generated to answer a much smaller number of more complex questions: the width of a boundary layer, the stability of some parameter configuration, the correlation between two variables in a particular area of the domain, etc. Without knowing anything about the nature of the questions to be asked, one can do little but use a lossless method on the 1 GiB and hope for some space savings. If, though, these motivating questions are known, lossy methods can be of great utility.

Like their lossless counterparts, lossy compression algorithms may be evaluated using various performance metrics: execution time, memory usage, bandwidth achieved, scalability, and so on. Principally, though, a lossy algorithm is judged by the fidelity between original dataset it is given and the reduced dataset it returns. Fidelity and reduction are in direct competition, so it is worthwhile to discuss what exactly is required of a reduced dataset for it serve as a scientifically useful surrogate. Consider a typical numerical simulation outputting a representation of some field over a spatiotemporal domain. Generally, application scientists are not primarily concerned with the pointwise values taken by such a field – for instance, the pressure of a flow at location (x, y, z) and time t. Rather, the quantities of interest are derived statistics computed from these pointwise values. These quantities may be scalar-valued (for example, averages of some variable or its derivatives over subdomains of particular sizes) or vector-valued (for example, a gradient or a vorticity).

A wide variety of quantities of interest may be written in the form $\mathcal{Q}(u)$, where u is the function resulting from the numerical simulation and $\mathcal{Q}$ is a bounded linear functional defined on a function space containing u. Let $\tilde{u}$ be some reduced representation of u. Following reduction, the quantity of interest is calculated with $\tilde{u}$ instead of u, causing a loss $\mathcal{Q}(u) - \mathcal{Q}(\tilde{u})$. Given a particular reduced representation $\tilde{u}$, we would like to estimate this loss. Equally well, we would like to use a prescribed limit on the acceptable loss to inform the selection of $\tilde{u}$. In the previous chapter, we discussed a reduction procedure allowing for control of $\|u - \tilde{u}\|_{L^\infty}$, the $L^\infty(\Omega)$ loss in the original dataset. In that context, the loss in the quantity of interest may be bounded as follows:

$$|\mathcal{Q}(u) - \mathcal{Q}(\tilde{u})| = |\mathcal{Q}(u - \tilde{u})| \leq C_{\mathcal{Q},L^\infty} \|u - \tilde{u}\|_{L^\infty} \tag{5.1}$$

where $C_{\mathcal{Q},L^\infty}$ is a constant that depends on $\mathcal{Q}$ (in particular, the norm of $\mathcal{Q}$ as a linear functional on an appropriately defined function space). This approach is unsatisfying. The $L^\infty(\Omega)$ norm is

very exacting: a bound on $\|u - \tilde{u}\|_{L^\infty}$ imposes severe restrictions on the reduced representation $\tilde{u}$ at every point in the domain. In this chapter we will instead use a family $\{\|\cdot\|_s : |s| < 3/2\}$ of norms, generally more forgiving than $\|\cdot\|_{L^\infty}$, to measure the reduction loss $u - \tilde{u}$. Revisiting (5.1), we will then bound $|\mathcal{Q}(u) - \mathcal{Q}(\tilde{u})| \leq C_{\mathcal{Q},s}\|u - \tilde{u}\|_s$ where, as before, $C_{\mathcal{Q},s}$ is a constant depending on $\mathcal{Q}$. This approach allows for inexpensive reduction of u while limiting the resulting distortion in any quantities of interest.

The remainder of this chapter is organized as follows. Section 5.2 presents a pair of algorithms implementing a nonadaptive reduction technique, which is shown to be quasioptimal. In Section 5.3 these methods are generalized to adaptive reduction with control of quantities of interest. This step is accomplished with the help of a family of loss estimators which we show to be equivalent to the norms $\{\|\cdot\|_s : |s| < 3/2\}$. The performance of the adaptive reduction technique on a Brusselator dataset is investigated in Section 5.4. Intermediate results required for the preceding sections are collected in Section 5.5.

5.2 Nonadaptive Reduction

We begin by establishing the setting. Let $\Omega \subset \mathbb{R}^d$ be a Cartesian product of intervals. Let uniform initial meshes on each of these intervals be repeatedly uniformly refined, resulting in hierarchies of increasing meshes. Tensoring together yields a hierarchy $\{\mathcal{P}_\ell : 0 \leq \ell\}$ of uniform meshes on Ω. In practice, refinement is stopped at some maximum level L. Let V_ℓ denote the space of continuous functions piecewise multilinear with respect to $\mathcal{P}_\ell$, and let Q_ℓ and Π_ℓ denote the $L^2(\Omega)$ projection and piecewise multilinear interpolant, respectively, onto V_ℓ. Let $\mathcal{N}_\ell$ denote the set of nodes in $\mathcal{P}_\ell$, and let $\mathcal{N}_\ell^*$ denote $\mathcal{N}_\ell \setminus \mathcal{N}_{\ell-1}$.

Our data will be some function $u \in V_L$ defined on the finest mesh in the hierarchy. It will often be useful to think of u as the projection of a function in some Sobolev space $H^r(\Omega)$. Consider the family of norms $\{\|\cdot\|_s : |s| < 3/2\}$ on $\cup_{\ell=0}^\infty V_\ell$ defined by

$$\|u\|_s^2 = \sum_{\ell=0}^{\infty} 2^{2s\ell}\|(Q_\ell - Q_{\ell-1})u\|^2. \tag{5.2}$$

The norms thus defined are equivalent to the usual Sobolev norms $\{\|\cdot\|_{H^s} : |s| < 3/2\}$ [BY93, Osw94],[1] and so we will *define* the Sobolev norms by these sums. Consider the problem of finding accurate (as measured by some Sobolev norm $\|\cdot\|_s$ with $s \in [0, r)$) reduced representations of a given function

[1]The proof given in [BY93] assumes that the functions in question are piecewise linear, but the result may easily be generalized to multilinear functions.

$u \in V_L$. We will show that the simple approach of choosing $\tilde{u}$ to be one of the projections $Q_\ell u$ is quasioptimal among all linear methods.

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

Algorithm 5.1: Reduction given a constraint on the degrees of freedom available for the reduced representation.

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

Algorithm 5.2: Reduction given a constraint on the maximum allowable error in the reduced representation.

Algorithms 5.1 and 5.2 provide procedures for generating reduced representations given constraints on the number of degrees of freedom to be used and the maximum allowable error (defined by $\epsilon = \|u - \tilde{u}\|_s / \|u\|_s$), respectively. Compare with Algorithms 3.1 and 3.2 from Chapter 3. Given the constraint that the reduced representation for u come from $\{Q_\ell u : 0 \leq \ell \leq L\}$, the algorithms are perfectly straightforward: Algorithm 5.1 says we should pick the finest projection that fits in the available space, while Algorithm 5.2 says we should pick the smallest projection that satisfies the loss tolerance. What is notable about these procedures and the choice of $\{Q_\ell u : 0 \leq \ell \leq L\}$ as candidate reduced representations is that they yield quasioptimal reductions of the original data u. This claim is made precise in the following result.

Theorem 5.1. *Let $u \in H^r(\Omega)$ for some $r \in (0, 3/2)$, and take $s \in [0, r)$. Suppose $u \neq 0$.*

1. *Let $N \in \mathbb{N}$ be given. Then Algorithm 5.1 gives a reduced representation using $n \leq N$ degrees of freedom while achieving a relative error $\epsilon \lesssim N^{-(r-s)/d} \|u\|_r / \|u\|_s$.*

2. *Let $\tau \in (0, 1]$ be given. Then Algorithm 5.2 gives a reduced representation with relative error $\epsilon \leq \tau$ while requiring $n \lesssim (\tau \|u\|_s / \|u\|_r)^{-d/(r-s)}$ degrees of freedom.*

Moreover, the $L^2(\Omega)$ projections used in Algorithms 5.1 and 5.2 are quasioptimal among linear reductions, in the sense that: any other method using N degrees of freedom cannot guarantee relative error below $O(N^{-(r-s)/d}\|u\|_r/\|u\|_s)$; or, any other method guaranteeing relative error $\tau\|u\|_r/\|u\|_s$ must use at least $O(\tau^{-d/(r-s)})$ degrees of freedom.

The proof is nearly identical to the proof of Theorem 3.1.

Proof. We first derive a simple error bound for the $L^2(\Omega)$ projection. Let $M \in \{0, \dots, L\}$. Then

$$\begin{aligned}\|u - Q_M u\|_s^2 &= \sum_{\ell=0}^{L} 2^{2s\ell}\|(Q_\ell - Q_{\ell-1})(u - Q_M u)\|^2 = \sum_{\ell=M+1}^{L} 2^{2s\ell}\|(Q_\ell - Q_{\ell-1})u\|^2 \\ &= \sum_{\ell=M+1}^{L} 2^{2r\ell}2^{-2(r-s)\ell}\|(Q_\ell - Q_{\ell-1})u\|^2 \le 2^{-2(r-s)(M+1)}\|u\|_r^2. \end{aligned} \tag{5.3}$$

Now consider Item 1. With $N \in \mathbb{N}$ given, take $M \in \{0, \dots, L\}$ so that Algorithm 5.1 returns $Q_M u$. Clearly, $\#\mathcal{N}_M \le N < \#\mathcal{N}_{M+1}$. In particular, $N < \#\mathcal{N}_{M+1} \simeq 2^{(M+1)d}$. Substituting into (5.3),

$$\|u - Q_M u\|_s^2 \le 2^{-2(r-s)(M+1)}\|u\|_r^2 = \left(2^{(M+1)d}\right)^{-2(r-s)/d}\|u\|_r^2 \lesssim N^{-2(r-s)/d}\|u\|_r^2,$$

as desired.

Next, consider Item 2. With $\tau \in (0, 1]$ given, take $M \in \{0, \dots, L\}$ so that Algorithm 5.2 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 $M > 0$, $\|u - Q_M u\|_s^2 \le \tau^2\|u\|_s^2 < \|u - Q_{M-1}u\|_s^2$. Now apply (5.3).

$$\begin{aligned} 2^{-2(r-s)M}\|u\|_r^2 &\ge \|u - Q_{M-1}u\|_s^2 > \tau^2\|u\|_s^2 \\ N \simeq 2^{Md} &\le (\tau\|u\|_s/\|u\|_r)^{-d/(r-s)}. \end{aligned}$$

Finally, we consider the quasioptimality of Algorithms 5.1 and 5.2. Consider any linear reduction operator $T\colon H^r(\Omega) \to H^s(\Omega)$. If N degrees of freedom are required to represent Tu (i.e., if $\operatorname{rank}(T) = N$), then $\|I - T\| \gtrsim N^{-(r-s)/d}$ (with constant independent of N) [ET96]. In other words, there exists some $u \in H^r(\Omega)$ such that $\|u - Tu\|_s \gtrsim (N^{-(r-s)/d}\|u\|_r/\|u\|_s)\|u\|_s$, so T can only guarantee accuracy of at best $O(N^{-(r-s)/d}\|u\|_r/\|u\|_s)$. On the other hand, if T can guarantee accuracy of $\tau\|u\|_r/\|u\|_s$ for all inputs u (i.e., if $\|u - Tu\|_s \le \tau\|u\|_r$ for all nonzero $u \in H^r(\Omega)$), we can conclude that T must require at least $O(\tau^{-d/(r-s)})$ degrees of freedom: $\|I - T\| \le \tau$ and $\operatorname{rank}(T)^{-(r-s)/d} \lesssim \|I - T\|$, so $\operatorname{rank}(T) \gtrsim \tau^{-d/(r-s)}$. □

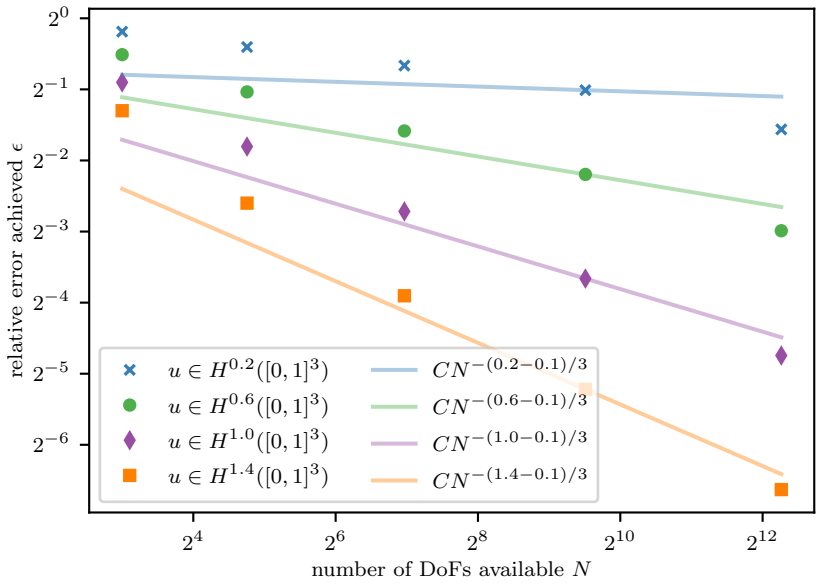

Figure 5.1: Illustration of the error decay as Algorithm 5.1 is applied to data using increasing numbers of degrees of freedom. As expected, the error decreases as the number of degrees of freedom used increases, and the decay is fastest for the most regular data. The decay rate given in Theorem 5.1 is somewhat pessimistic for the datasets used in this figure. The fit can be improved by, for example, replacing $\|u\|_r^2$ in (5.3) with $\|u - Q_M u\|_r^2$.

In order to verify the predictions of Theorem 5.1, we conduct two numerical experiments. In the first, four functions of known Sobolev regularities $r \in \{0.5, 0.75, 1, 1.25\}$ in three dimensions are reduced, with the errors measured in $H^s(\Omega)$ with $s = 0.25$. In the second, a single function of known Sobolev regularity $r = 1$ in two dimensions is reduced, with the errors measured in $H^s(\Omega)$ for $s \in \{0, 0.25, 0.5, 0.75\}$. The performance of Algorithms 5.1 and 5.2 applied to each of the functions is presented in Figures 5.1 and 5.2. 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 5.1.

5.3 Adaptive Reduction and *A Posteriori* Loss Estimator

Let a function $u \in V_L$ and a bounded linear functional $\mathcal{Q}\colon V_L \to \mathbb{R}$ be given. Consider the problem of finding a reduced representation $\tilde{u} \in V_L$ such that the reduction loss $|\mathcal{Q}(u) - \mathcal{Q}(\tilde{u})|$ is no more than some tolerance τ. As V_L has finite dimension, $\mathcal{Q}$ is bounded no matter the norm imposed on V_L. In particular, there exist constants $\{C_{\mathcal{Q},s} : |s| < 3/2\}$ such that $|\mathcal{Q}(u) - \mathcal{Q}(\tilde{u})| \leq C_{\mathcal{Q},s}\|u - \tilde{u}\|_s$. Here $C_{\mathcal{Q},s}$ is the norm of $\mathcal{Q}$ as a functional from V_L with the $\|\cdot\|_s$ norm to $\mathbb{R}$ with the standard norm. Thus, if $\|u - \tilde{u}\|_s \leq \tau / C_{\mathcal{Q},s}$ for any $|s| < 3/2$, then $|\mathcal{Q}(u) - \mathcal{Q}(\tilde{u})| \leq \tau$. Two tasks must be

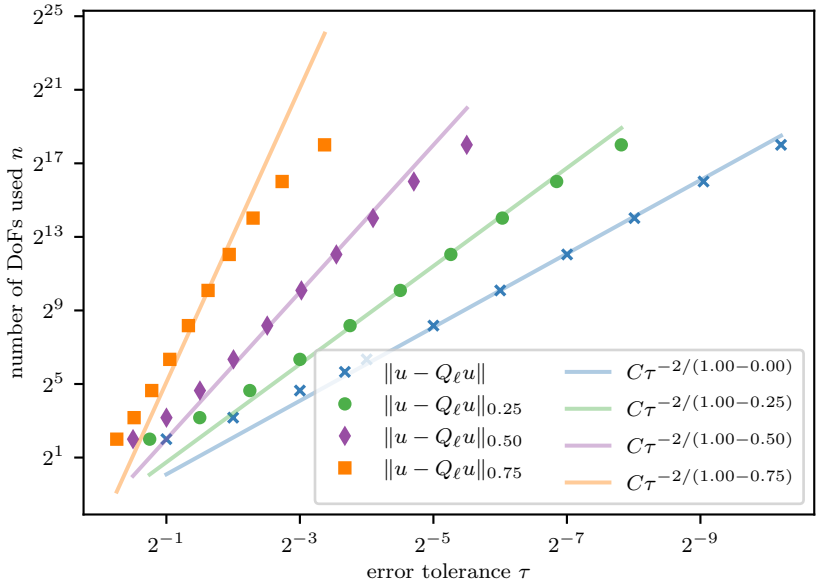

Figure 5.2: Illustration of the growth of number of degrees of freedom needed as Algorithm 5.2 is applied to data with decreasing error tolerances. As expected, more degrees of freedom are required to generate a reduced representation with greater accuracy, and the growth is fastest in the demanding case that s is close to r ($r - s$ is small).

tackled: first, the computation of $C_{\mathcal{Q},s}$, and second, the development of a procedure for generating $\tilde{u}$.

V_L is a Hilbert space with the usual $L^2(\Omega)$ inner product $(\,\cdot\,,\,\cdot\,)$, so by the Riesz representation theorem there exists some $\psi \in V_L$ such that $\mathcal{Q}(v) = (v, \psi)$ for all $v \in V_L$. ψ may be obtained by taking v to be each element of a basis for V_L (most naturally the Lagrange basis $\{\phi_L(\,\cdot\,; x) : x \in \mathcal{N}_L\}$), calculating $\mathcal{Q}(v)$ to populate a load vector, and then solving the resulting linear system. We show in Lemma 5.5 that $C_{\mathcal{Q},s}$ is related to ψ as follows:

$$C_{\mathcal{Q},s} = \sup_{\substack{v \in V_L \\ v \neq 0}} \frac{|\mathcal{Q}(v)|}{\|v\|_s} = \|\psi\|_{-s}.$$

Using (5.2), $\|\psi\|_{-s}$ can be computed by projecting ψ to each level $\{V_\ell : 0 \leq \ell \leq L\}$ in the function space hierarchy, computing norms, weighting, and summing.

It remains to develop a procedure for generating $\tilde{u}$. Our strategy is built around the *multilevel components* of u, a collection $\{\Delta_\ell u : 0 \leq \ell \leq L\}$ of functions defined by $\Delta_\ell u = (I - \Pi_{\ell-1})Q_\ell u$. The transformation $u \leftrightarrow \{\Delta_\ell u : 0 \leq \ell \leq L\}$ is invertible (see Subsection 4.2.1). So, if an alternative collection $\{\Delta_\ell \tilde{u} : 0 \leq \ell \leq L\}$ of components can be constructed, they will uniquely define a surrogate function $\tilde{u} \in V_L$. We will generate this collection $\{\Delta_\ell \tilde{u} : 0 \leq \ell \leq L\}$ by perturbing each $\Delta_\ell u$. In order to bound the resulting loss, we need a result relating the magnitude of these perturbations to the

error incurred in V_L. To this end we define a family of loss indicators $\{\eta_s \colon V_L \to [0,\infty) : |s| < 3/2\}$ by

$$\eta_s(v)^2 = \sum_{\ell=0}^{L} 2^{2s\ell} \|\Delta_\ell v\|^2.$$

The following result bounds $\|\cdot\|_s$ above and below by η_s.

Lemma 5.2. *Let V_L be the space of continuous piecewise multilinear functions on a uniform tensor product grid in d dimensions. Then*

$$C_1\eta_s(v) \le \|v\|_s \le C_2\eta_s(v) \quad \textit{for all} \quad v \in V_L \quad \textit{and all} \quad s \in (-\tfrac{3}{2}, \tfrac{3}{2})$$

with $C_1 \ge 2^{-d}$ and $C_2 \le 1$.

Proof. Observe that $(Q_\ell - Q_{\ell-1})\Delta_\ell v = (Q_\ell - Q_{\ell-1})v$ for all $v \in V_L$. As $Q_\ell - Q_{\ell-1}$ is an $L^2(\Omega)$ projection, $\|(Q_\ell - Q_{\ell-1})v\| \le \|\Delta_\ell v\|$. The righthand inequality follows:

$$\|v\|_s^2 = \sum_{\ell=0}^{L} 2^{2s\ell} \|(Q_\ell - Q_{\ell-1})v\|^2 \le \sum_{\ell=0}^{L} 2^{2s\ell} \|\Delta_\ell v\|^2 = \eta_s(v)^2.$$

For the lefthand inequality, observe that

$$\Delta_\ell v = (I - \Pi_{\ell-1})Q_\ell v = (I - \Pi_{\ell-1})(Q_\ell - Q_{\ell-1})v.$$

The result then follows from Corollary 5.8. □

Next we seek a method of generating $\Delta_\ell \tilde{u}$ close to $\Delta_\ell u$. Consider how $\Delta_\ell u$ might be represented as a set of coefficients. A natural basis is the subset $\{\phi_\ell(\,\cdot\,; x) : x \in \mathcal{N}_\ell^*\}$ of the Lagrange basis that is zero on $\mathcal{N}_{\ell-1}$. The coefficients for $\Delta_\ell u$ with respect to this basis are the nodal values $\{\Delta_\ell u(x) : x \in \mathcal{N}_\ell^*\}$. We gather these values into a single data structure `u_mc` defined by

$$\texttt{u\_mc}[x] = \Delta_\ell u(x) \quad \text{if} \quad x \in \mathcal{N}_\ell^*.$$

We call `u_mc` the *multilevel coefficients* of u. For more details on the multilevel coefficients, including efficient algorithms for computing them, see Chapter 4. The next result relates changes in the multilevel coefficients to the family of estimators. Let $\mathrm{vol}(\mathcal{P}_\ell)$ denote the (uniform) volume of the elements in $\mathcal{P}_\ell$.

Lemma 5.3. *Let $v \in V_L$, and let* `v_mc` *be the multilevel coefficients for v. Then*

$$C_1 \sum_{\ell=0}^{L} \operatorname{vol}(\mathcal{P}_\ell) \sum_{x \in \mathcal{N}_\ell^*} |\texttt{v\_mc}[x]|^2 \le \eta_s(v)^2 \le C_2 \sum_{\ell=0}^{L} \operatorname{vol}(\mathcal{P}_\ell) \sum_{x \in \mathcal{N}_\ell^*} |\texttt{v\_mc}[x]|^2$$

with $C_1 \ge 6^{-d}$ and $C_2 \le 1$.

Proof. It suffices to show that $\|\Delta_\ell u\|^2 \simeq \operatorname{vol}(\mathcal{P}_\ell)|\texttt{v\_mc}[\mathcal{N}_\ell^*]|^2$ with constants C_1 and C_2. Observe that, by the tensor product structure of the mesh, it suffices to show this for $d = 1$. Let $\vec{\alpha} = (\alpha_1, \ldots, \alpha_N)$ be the vector of nodal values of $\Delta_\ell v$ on $\mathcal{N}_\ell$, with α_1 and α_N the values on the endpoints. Using Simpson's rule,

$$\begin{aligned}\|\Delta_\ell v\|^2 &= \sum_{i=1}^{N-1} \tfrac{1}{6} \operatorname{vol}(\mathcal{P}_\ell)\big(\alpha_i^2 + 4(\tfrac{1}{2}\alpha_i + \tfrac{1}{2}\alpha_{i+1})^2 + \alpha_{i+1}^2\big) \\ &= \tfrac{1}{6} \operatorname{vol}(\mathcal{P}_\ell) \sum_{i=1}^{N-1} \begin{bmatrix} \alpha_i \\ \alpha_{i+1} \end{bmatrix}^\intercal \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} \alpha_i \\ \alpha_{i+1} \end{bmatrix}.\end{aligned}$$

The matrix has eigenvalues 1 and 3, so

$$\|\Delta_\ell v\|^2 \ge \tfrac{1}{6} \operatorname{vol}(\mathcal{P}_\ell) \sum_{i=1}^{N-1} 1 \left| \begin{bmatrix} \alpha_i \\ \alpha_{i+1} \end{bmatrix} \right|^2 = \tfrac{1}{6} \operatorname{vol}(\mathcal{P}_\ell) \sum_{i=1}^{N}{}'' 2|\alpha_i|^2 \ge \tfrac{1}{6} \operatorname{vol}(\mathcal{P}_\ell)|\vec{\alpha}|^2$$

$$\|\Delta_\ell v\|^2 \le \tfrac{1}{6} \operatorname{vol}(\mathcal{P}_\ell) \sum_{i=1}^{N-1} 3 \left| \begin{bmatrix} \alpha_i \\ \alpha_{i+1} \end{bmatrix} \right|^2 = \tfrac{1}{6} \operatorname{vol}(\mathcal{P}_\ell) \sum_{i=1}^{N}{}'' 6|\alpha_i|^2 \le \quad \operatorname{vol}(\mathcal{P}_\ell)|\vec{\alpha}|^2. \qquad \square$$

We can now relate changes in the multilevel coefficients to loss in quantities of interest.

Theorem 5.4. *Let $\mathcal{Q}$ be a bounded linear functional on V_L. Let ψ be the Riesz representative of $\mathcal{Q}$ on V_L. Let $u \in V_L$ with multilevel coefficients* `u_mc`*. Let $\tau \ge 0$. Let* `ũ_mc` *be a set of multilevel coefficients and let $\tilde{u} \in V_L$ be the corresponding function. If* `ũ_mc` *satisfies*

$$\sum_{\ell=0}^{L} 2^{2s\ell} \operatorname{vol}(\mathcal{P}_\ell) \sum_{x \in \mathcal{N}_\ell^*} |\texttt{u\_mc}[x] - \texttt{ũ\_mc}[x]|^2 \le \frac{\tau^2}{\|\psi\|_{-s}^2} \tag{5.4}$$

for any $s \in (-3/2, 3/2)$, then $|\mathcal{Q}(u) - \mathcal{Q}(\tilde{u})| \le \tau$.

Proof. Define a set `v_mc` of multilevel coefficients by $\texttt{v\_mc}[x] = \texttt{u\_mc}[x] - \texttt{ũ\_mc}[x]$ for $x \in \mathcal{N}_L$. By linearity, the corresponding function $v \in V_L$ is equal to $u - \tilde{u}$. Applying Lemma 5.3 to `v_mc`, we see that $\eta_s(u - \tilde{u})^2 \le \tau^2/\|\psi\|_{-s}^2$. The result then follows from Lemma 5.2. $\square$

Theorem 5.4 can be used to guide the generation of a reduced representation $\tilde{u}$ of u. Various strategies are feasible. For example, coefficients could be dropped by level ℓ, or all coefficients could be quantized. The simple approach taken by Algorithm 5.3 is to replace by zero as many coefficients, ordered according to the contribution to the righthand side of (5.4), as possible.

Require: $\tau \geq 0$.
  function REDUCE(linear functional $\mathcal{Q}$, multilevel coefficients `u_mc`, tolerance τ, smoothness parameter s)
    find Riesz representative ψ of $\mathcal{Q}$
    compute $\|\psi\|_{-s}$
    `ũ_mc` $\leftarrow$ `u_mc`
    `square_loss_estimate` $\leftarrow 0$
    for $x \in \mathcal{N}_L$ in order of increasing $2^{2s\ell}\,\mathrm{vol}(\mathcal{P}_\ell)|$`u_mc[`$x$`]`$|^2$ **do**
      find $\ell \in \{0, \ldots, L\}$ so that $x \in \mathcal{N}_\ell^*$
      `square_loss_estimate` $+= 2^{2s\ell}\,\mathrm{vol}(\mathcal{P}_\ell)|$`u_mc[`$x$`]`$|^2$
      if `square_loss_estimate` $> \tau^2$ **then**
        break
      else
        `ũ_mc[`x`]` $\leftarrow 0$
      end if
    end for
    return `ũ_mc`
  end function

Algorithm 5.3: Adaptive reduction to meet a loss tolerance.

5.4 Numerical Examples

In this section we apply Algorithm 5.3 while controlling, in Subsection 5.4.1, a measure of distortion, and, in Subsection 5.4.2, two quantities of interest. In all examples we use the Brusselator dataset from Subsection 4.4.1. That dataset comprises 1025 timesteps of the concentrations of two chemical species, U and V, on a spatial domain discretized with a 65×65 grid. We apply Algorithm 5.3 to the concentration of U only.

5.4.1 Peak Signal-to-Noise Ratio Control

We first consider the simple use case of reduction without any quantities of interest known *a priori*. In this scenario, a tolerance may be set on the loss induced by the reduction process so that the reduced representation $\tilde{u}$ can serve as a workable standin for u. One commonly used metric for this purpose is peak signal-to-noise ratio (PSNR), which can be defined for a general dataset u and

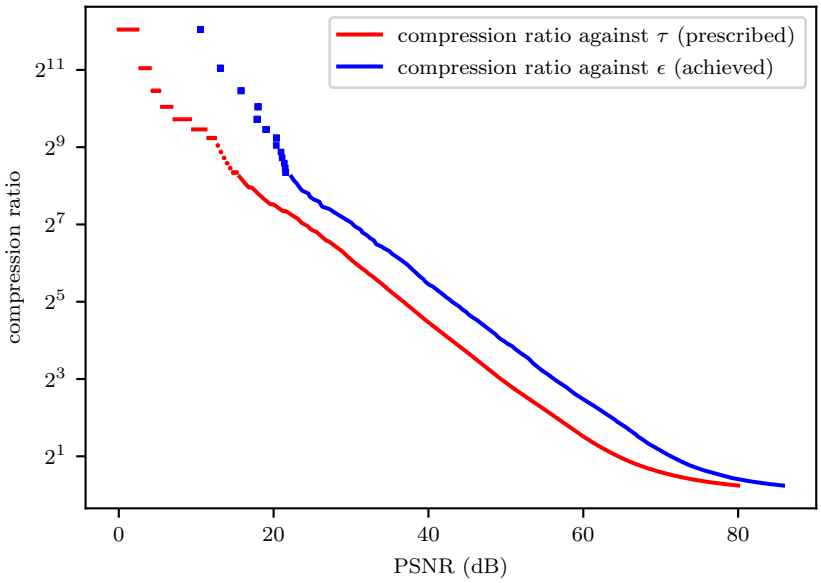

Figure 5.3: Illustration of results obtained when reducing timestep 300 of the Brusselator simulation to a range of PSNR tolerances.

reduced representation $\tilde{u}$ by

$$\mathrm{PSNR}(u, \tilde{u}) = 20 \log_{10} \Big[\frac{\max(u) - \min(u)}{|\Omega|^{-1/2} \|u - \tilde{u}\|} \Big].$$

We can use Lemma 5.2 to obtain a sufficient criterion for a reduced representation $\tilde{u}$ to meet a prescribed PSNR bound. Choosing $\tilde{u}$ to satisfy

$$|\Omega|^{-1/2} \|u - \tilde{u}\| \leq 10^{-\tau/20} \big[\max(u) - \min(u) \big]$$

will guarantee that the PSNR is at least τ, as can be seen by simply substituting into the definition of PSNR. Figure 5.3 shows the results of reducing one timestep of the Brusselator simulation so that the PSNR obtained meets a range of tolerances.

5.4.2 Average Control

We now turn to reduction while preserving quantities of interest. Perhaps the most common derived quantities are visualizations. Given a two-dimensional dataset u to be visualized, a strong condition we might put upon a reduced representation $\tilde{u}$ is that each pixel of the visualization produced with $\tilde{u}$ be within some tolerance τ of the corresponding pixel of the visualization produced with u. Assume

that each pixel of the visualization corresponds to some subset p of the domain Ω, and that the value of that pixel is given by the average of the function over p. The pixel fidelity condition is then written

$$\Big|\int_p u(x)\,\mathrm{d}x - \int_p \tilde{u}(x)\,\mathrm{d}x\Big| = \Big|\int_\Omega (u-\tilde{u})(x)\,\chi_p(x)\,\mathrm{d}x\Big| \leq \tau$$

where χ_p is the characteristic function of p. Let $\mathcal{P}$ denote the partition of Ω into subsets p, and suppose that each $p \in \mathcal{P}$ is a rectangle. Then $\chi_p \in H^s(\Omega)$ for $s \in [0, 1/2)$, and, applying Lemma 5.5,

$$\Big|\int_\Omega (u-\tilde{u})(x)\,\chi_p(x)\,\mathrm{d}x\Big| \leq \|u-\tilde{u}\|_{-s}\|\chi_p\|_s$$

for all $s \in [0, 1/2)$. So, if $\|u-\tilde{u}\|_{-s} \leq \tau / \max_{p\in\mathcal{P}} \|\chi_p\|_{-s}$, then

$$\max_{p\in\mathcal{P}} \Big|\int_\Omega (u-\tilde{u})(x)\,\chi_p(x)\,\mathrm{d}x\Big| \leq \|u-\tilde{u}\|_{-s} \max_{p\in\mathcal{P}} \|\chi_p\|_s \leq \tau$$

as desired. Figure 5.4 presents visualizations generated from a dataset reduced using this method and from the original dataset (timestep 480 of the Brusselator simulation).

A similar need for reduction while preserving averages occurs in the context of timestepping applications. In order to maintain stability, numerical methods for PDEs often require a simulation timestep significantly smaller than the timestep ultimately needed to resolve the phenomenon of interest. The output of such a simulation is a time series that may be unwieldy or impractical to process due to its size. A frequently used solution to this problem is decimation, i.e. replacing the original dataset with a reduced dataset consisting of every tenth (for example) timestep. As the intervening timesteps are simply discarded, this approach leaves the user vulnerable to data loss. One approach is to supplement the decimation process, rendering it lossless (see Chapter 2). Alternatively, the algorithm given in Section 5.3 can be used to produce a surrogate dataset that, though reduced, nonetheless preserves the pointwise time averages (for example) of the data. Suppose a simulation is carried out over a spatial domain Ω and temporal domain $[0, T]$. Let $[0, T]$ be subdivided into a collection $\mathcal{P}$ of time intervals. Let χ_p denote the characteristic function of a subinterval $p \in \mathcal{P}$. If

$$\|u(x,\,\cdot\,) - \tilde{u}(x,\,\cdot\,)\|_{H^{-s}([0,T])} \leq \frac{\tau}{\max_{p\in\mathcal{P}} \|\chi_p\|_{H^s([0,T])}},$$

for all $x \in \Omega$, then

$$\sup_{\substack{x\in\Omega \\ p\in\mathcal{P}}} \Big|\int_{[0,T]} u(x,t)\,\chi_p(t)\,\mathrm{d}t - \int_{[0,T]} \tilde{u}(x,t)\,\chi_p(t)\,\mathrm{d}t\Big| \leq \tau,$$

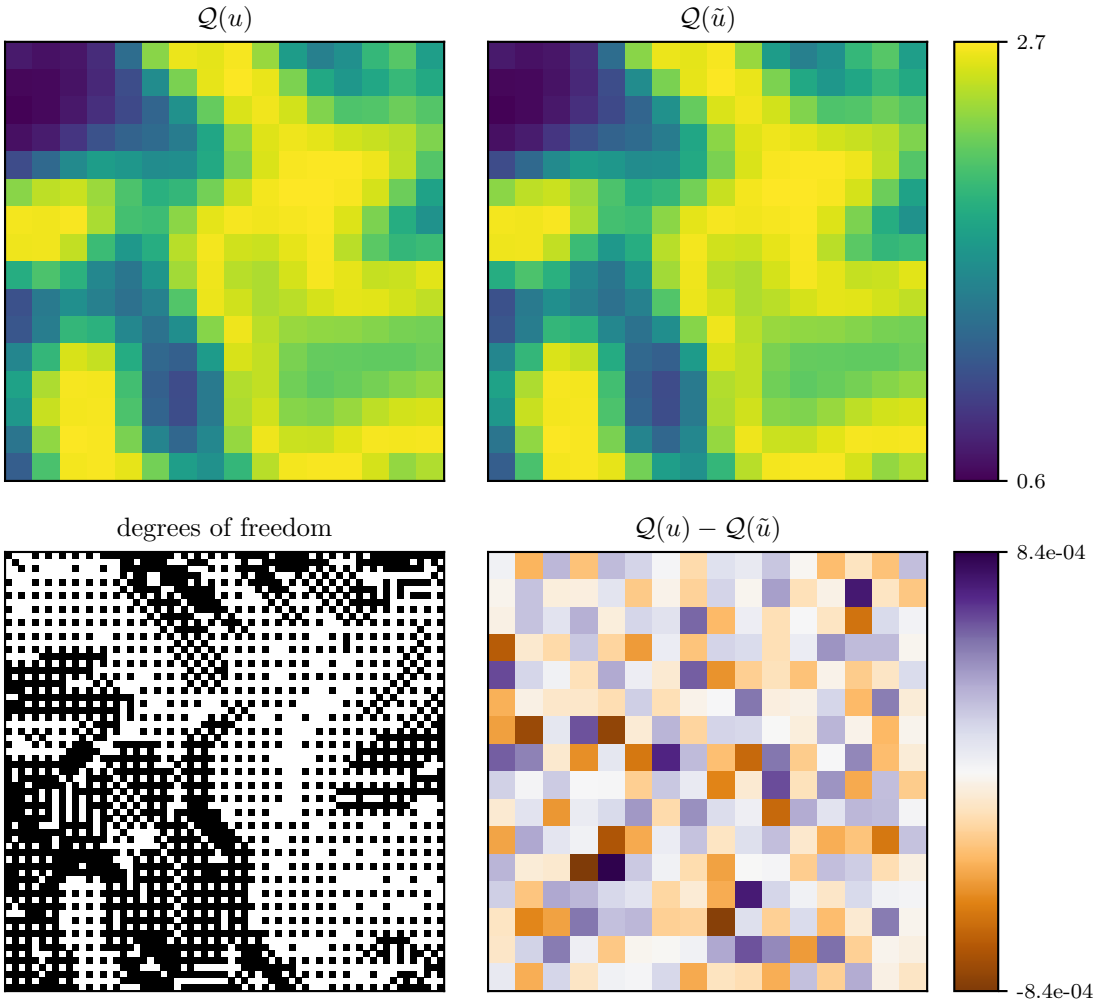

Figure 5.4: Illustration of results obtained when reducing timestep 480 of the Brusselator simulation using pixels of size 4×4 with a tolerance of $\tau = 5 \times 10^{-2}$.

	τ		
	10^{-2}	10^{-4}	10^{-6}
$\|\cdot\|$	4.467	1.190	1.001
$\|\cdot\|_{0.5}$	5.781	1.372	1.006
$\|\cdot\|_{L^\infty}$	2.493	1.026	1.000

Table 5.1: Compression ratios obtained using a decimation stride of 2^4 and a variety of tolerances and reduction norms. The compression ratios displayed are the averages over each of the 4225 time series in the Brusselator dataset.

following the same argument as above. As $\chi_p \in H^s([0,T])$ for any $s \in [0, 1/2)$, this inequality holds for any $s \in [-r, 1/2)$ if $u(x, \cdot) \in H^r([0,T])$. We may also use other norms to measure the reduction loss. For example, bounding

$$\Big| \int_{[0,T]} (u - \tilde{u})(x,t)\,\chi_p(t)\,\mathrm{d}t \Big| \leq \|u(x,\cdot) - \tilde{u}(x,\cdot)\|_{L^\infty([0,T])} \|\chi_p\|_{L^1([0,T])}$$

leads to the condition $\|u(x,\cdot) - \tilde{u}(x,\cdot)\|_{L^\infty} \leq \tau / \max_{p\in\mathcal{P}} \|\chi_p\|_{L^1}$ for all $x \in \Omega$. Table 5.1 compares the compression ratios achieved by this method when using $\|\cdot\|$, $\|\cdot\|_{0.5}$, and $\|\cdot\|_{L^\infty}$ and tolerances $\tau \in \{10^{-2}, 10^{-4}, 10^{-6}\}$.

5.5 Helpful Norm Results

We begin with a result needed to compute the bounding constants $C_{Q,s}$.

Lemma 5.5. *Let X be a Hilbert space with inner product $(\cdot,\cdot)$ and induced norm $\|\cdot\|$. Let $X_1 \oplus \cdots \oplus X_N$ be an orthogonal decomposition of X with associated projections $\Pi_1, \ldots, \Pi_N$. Let scalars $\alpha_1, \ldots, \alpha_N \in (0,\infty)$ be given and define a new inner product $(\cdot,\cdot)_H \colon X \to \mathbb{R}$ by $(u,v)_H = \sum_{i=1}^N \alpha_i^2 (\Pi_i u, \Pi_i v)$. Let H denote the Hilbert space with this inner product. For $\psi \in X$, let $\|\psi\|_{H^*}$ denote the operator norm of the functional $(\cdot,\psi)\colon H \to \mathbb{R}$. Let $|||\cdot||| \colon X \to [0,\infty)$ be the norm defined by $|||u|||^2 = \sum_{i=1}^N \alpha_i^{-2} \|\Pi_i u\|^2$. Then $|||\cdot||| = \|\cdot\|_{H^*}$.*

Proof. Fix $\psi \in X$. It suffices to bound $\|\psi\|_{H^*}$ above and below by $|||\psi|||$. We begin with the upper bound. By the continuous and discrete Cauchy–Schwarz inequalities,

$$(u,\psi) = \sum_{i=1}^N (\Pi_i u, \Pi_i \psi) \leq \sum_{i=1}^N \|\Pi_i u\| \|\Pi_i \psi\| \leq \Big(\sum_{i=1}^N \alpha_i^2 \|\Pi_i u\|^2 \Big)^{1/2} \Big(\sum_{i=1}^N \alpha_i^{-2} \|\Pi_i \psi\|^2 \Big)^{1/2} = \|u\|_H |||\psi|||$$

for all $u \in H$. As a result, $\|\psi\|_{H^*} \leq |||\psi|||$. For the lower bound, take u to be $\sum_{i=1}^N \alpha_i^{-2} \Pi_i \psi / |||\psi|||$.

Observe that

$$\|u\|_H^2 = \sum_{i=1}^N \alpha_i^2 \frac{\|\Pi_i\psi\|^2}{\alpha_i^4 |||\psi|||^2} = \frac{1}{|||\psi|||^2} \sum_{i=1}^N \frac{1}{\alpha_i^2} \|\Pi_i\psi\|^2 = 1$$

and that

$$(u,\psi) = \sum_{i=1}^N (\alpha_i^{-2}\Pi_i\psi/|||\psi|||, \Pi_i\psi) = \sum_{i=1}^N \frac{\|\Pi_i\psi\|^2}{\alpha_i^2|||\psi|||} = \frac{|||\psi|||^2}{|||\psi|||} = |||\psi|||.$$

As a result, $\|\psi\|_{H^*} \geq |||\psi|||$. □

The remaining results concern the norm of $I - \Pi_{\ell-1}$ as a operator $L^2(V_\ell) \to L^2(V_\ell)$. We begin with a general lemma about norms of projections.

Lemma 5.6. *Let X be an inner product space. Let $X_0, X_1 \subset X$ be nonempty subspaces satisfying $X = X_0 \oplus X_1$. Let Π_0 and Π_1 denote the projections to X_0 and X_1, respectively. Then*

$$\|\Pi_0\| = \sup_{\substack{u\in X \\ \|u\|=1}} \|\Pi_0 u\|_X = \sup_{\substack{u\in X \\ \|u\|=1}} \|\Pi_1 u\|_X = \|\Pi_1\|$$

where $\|\cdot\|_X$ is the norm given by the inner product $(\,\cdot\,,\,\cdot\,)_X$.

Proof. It suffices to show that for all $u \in X$ with $\|u\|_X = 1$ there exists $v \in X$ with $\|v\|_X = 1$ such that $\|\Pi_0 u\|_X = \|\Pi_1 v\|_X$. Write $u = \alpha a + \beta b$ with $a \in X_0$, $b \in X_1$, and $\|a\|_X = \|b\|_X = 1$. Take $v = \beta a + \alpha b$.[2] Then $\|v\|_X = 1$ and $\|\Pi_1 v\|_X = \|\Pi_0 u\|_X = \alpha$. □

In order to compute the operator norm of $I - \Pi_{\ell-1}$, we apply Lemma 5.6 with $X = V_\ell$, $\Pi_0 = \Pi_{\ell-1}$, and $\Pi_1 = I - \Pi_{\ell-1}$. We first calculate $\Pi_{\ell-1}$ when $d = 1$.

Lemma 5.7. *Suppose $d = 1$, and let ρ be the mesh ratio of $\mathcal{P}_\ell$. Then $\|\Pi_{\ell-1}\| \leq \sqrt{\rho} + \sqrt{1/\rho}$, with equality holding if $\rho = 1$.*

Proof. Observe that

$$\|\Pi_{\ell-1}\|^2 = \sup_{\substack{v\in V_\ell \\ v\neq 0}} \frac{\|\Pi_{\ell-1}v\|^2}{\|v\|^2} = \sup_{\substack{v\in V_\ell \\ v\neq 0}} \frac{\sum_{K\in\mathcal{P}_{\ell-1}} \|\Pi_{\ell-1}v\|_K^2}{\sum_{K\in\mathcal{P}_{\ell-1}} \|v\|_K^2}.$$

Consider an individual element $K \in \mathcal{P}_{\ell-1}$. Let h^- and h^+ be the widths of the elements into which K is subdivided in $\mathcal{P}_\ell$. We may then, without loss of generality, assume that $K = (-h^-, h^+)$. Let

[2] If $\beta \neq 0$, $v = [(\alpha^2 - 1)/\beta]a + \alpha b$ will also work.

$v \in V_\ell$ be arbitrary, and denote $v(-h^-)$, $v(0)$, and $v(h^+)$ by a, b, and c, respectively. Then

$$\begin{aligned}
\|\Pi_{\ell-1} v\|_K^2 &= \tfrac{1}{3}(h^- + h^+)(a^2 + ac + c^2) \\
&= \frac{1}{3}\frac{(h^- + h^+)^2}{h^- h^+}\left[h^- a^2 + h^- ab + (h^- + h^+)b^2 + h^+ bc + h^+ c^2\right] \\
&\quad - \frac{1}{3}\frac{h^- + h^+}{h^- h^+}\left[\tfrac{3}{4}(h^- a - h^+ c)^2 + \tfrac{1}{4}(h^- a + 2(h^- + h^+)b + h^+ c)^2\right] \\
&\le \frac{1}{3}\frac{(h^- + h^+)^2}{h^- h^+}\left[h^- a^2 + h^- ab + (h^- + h^+)b^2 + h^+ bc + h^+ c^2\right] \qquad (5.5) \\
&= \frac{(h^- + h^+)^2}{h^- h^+}\|v\|_K^2 = (\sqrt{h^-/h^+} + \sqrt{h^+/h^-})^2 \|v\|_K^2 \\
&\le (\sqrt{\rho} + \sqrt{1/\rho})^2 \|v\|_K^2. \qquad (5.6)
\end{aligned}$$

As a result,

$$\|\Pi_{\ell-1}\|^2 \le \sup_{\substack{v \in V_\ell \\ v \neq 0}} \frac{\sum_{K \in \mathcal{P}_{\ell-1}} (\sqrt{\rho} + \sqrt{1/\rho})^2 \|v\|_K^2}{\sum_{K \in \mathcal{P}_{\ell-1}} \|v\|_K^2} = (\sqrt{\rho} + \sqrt{1/\rho})^2 \qquad (5.7)$$

and so $\|\Pi_{\ell-1}\| \le \sqrt{\rho} + \sqrt{1/\rho}$.

Finally, consider the case of uniform grids. For each $K \in \mathcal{P}_{\ell-1}$, $h^- = h^+$. As a result, the inequality in (5.5) becomes an equality when $a = -2b = c$. Equality holds in (5.6) because $h^-/h^+ = h^+/h^- = \rho = 1$. The inequality in (5.7) becomes an equality because a single function $w \in V_\ell$ can maximize $\|\Pi_{\ell-1} w\|_K^2 / \|w\|_K^2$ on all elements $K \in \mathcal{P}_{\ell-1}$ simultaneously. Indeed, let $w \in V_\ell$ be defined by $w(x) = 2$ for $x \in \mathcal{N}_{\ell-1}$ and $w(x) = -1$ for $x \in \mathcal{N}_\ell^*$. Then

$$\|\Pi_{\ell-1}\|^2 = \sup_{\substack{v \in V_\ell \\ v \neq 0}} \frac{\|\Pi_{\ell-1} v\|^2}{\|v\|^2} \ge \frac{\sum_{K \in \mathcal{P}_{\ell-1}} (\sqrt{1} + \sqrt{1})^2 \|w\|_K^2}{\sum_{K \in \mathcal{P}_{\ell-1}} \|w\|_K^2} = 2^2. \qquad \square$$

Now we can extend to the case of general dimension to obtain the desired bound.

Corollary 5.8. *Let $\mathcal{P}_\ell$ be the tensor product of d one-dimensional meshes $\{\mathcal{P}_\ell^i : 1 \le i \le d\}$ with mesh ratios $\{\rho_i : 1 \le i \le d\}$. Then $\|I - \Pi_{\ell-1}\| \le \prod_{i=1}^d \sqrt{\rho_i} + \sqrt{1/\rho_i}$, with equality holding if $\rho_i = 1$ in each dimension.*

Proof. Because $\mathcal{P}_\ell$ is a tensor product mesh, the interpolation from $\mathcal{P}_\ell$ to $\mathcal{P}_{\ell-1}$ is given by the tensor product of the one-dimensional interpolations from $\mathcal{P}_\ell^i$ to $\mathcal{P}_{\ell-1}^i$: $\Pi_{\ell-1} = \Pi_{\ell-1}^1 \otimes \cdots \otimes \Pi_{\ell-1}^d$. By the definition of the tensor product, then, $\|\Pi_{\ell-1}\| = \|\Pi_{\ell-1}^1\| \cdots \|\Pi_{\ell-1}^d\|$, and the result follows by Lemma 5.7. $\square$

Bibliography

[ABK16] W. Austin, G. Ballard, and T. G. Kolda. Parallel tensor compression for large-scale scientific data. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 912–922, May 2016.

[AKW17] Mark Ainsworth, Scott Klasky, and Ben Whitney. Compression using lossless decimation: Analysis and application. *SIAM Journal on Scientific Computing*, 39(4):B732–B757, August 2017.

[AO11] Mark Ainsworth and J. Tinsley Oden. *A Posteriori Error Estimation in Finite Element Analysis*, volume 37 of *Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts*. John Wiley & Sons, Inc., 2011.

[ATWK17] Mark Ainsworth, Ozan Tugluk, Ben Whitney, and Scott Klasky. Multilevel techniques for compression and reduction of scientific data—the univariate case. *Computing and Visualization in Science*, June 2017. Accepted.

[ATWK18] Mark Ainsworth, Ozan Tugluk, Ben Whitney, and Scott Klasky. Multilevel techniques for compression and reduction of scientific data—the multivariate case. *Submitted*, January 2018.

[BC16] Martin Burtscher and Steven Claggett. SPDP v1.0. `http://cs.txstate.edu/~burtscher/research/SPDPcompressor/`, 2016.

[BDY88] Randolph E. Bank, Todd F. Dupont, and Harry Yserentant. The hierarchical basis multigrid method. *Numerische Mathematik*, 52(4):427–458, July 1988.

[BG16] Leonardo A. Bautista Gomez. Re: Reproducing result from "Improving Floating Point Compression,through Binary Masks", July 2016. The comma between 'Compression' and 'through' in the title was a typo in the original message.

[BGC13] Leonardo A. Bautista Gomez and Franck Cappello. Improving floating point compression through binary masks. In *2013 IEEE International Conference on Big Data*, pages 326–331, October 2013.

[BMYH16] Martin Burtscher, Hari Mukka, Annie Yang, and Farbod 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*, pages 264–275. IEEE, November 2016.

[BR07] Martin Burtscher and Paruj Ratanaworabhan. High throughput compression of double-precision floating-point data. In *Data Compression Conference, 2007. DCC '07*, pages 293–302, March 2007.

[BR10] Martin Burtscher and Paruj Ratanaworabhan. gFPC: A self-tuning compression algorithm. In *Data Compression Conference (DCC), 2010*, pages 396–405, March 2010.

[Bra99] Karlheinz Brandenburg. MP3 and AAC explained. In *Audio Engineering Society Conference: 17th International Conference: High-Quality Audio Coding*. Audio Engineering Society, September 1999.

[BT97] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific & Dynamic Ideas, Belmont, Massachusetts, 1st edition, 1997.

[Bur16a] John Burkhardt. BLACK_SCHOLES. `http://people.sc.fsu.edu/~jburkardt/cpp_src/black_scholes/black_scholes.html`, June 2016.

[Bur16b] John Burkhardt. LORENZ_ODE. `http://people.sc.fsu.edu/~jburkardt/cpp_src/lorenz_ode/lorenz_ode.html`, May 2016.

[Bur16c] Martin Burtscher. FPdouble. `http://cs.txstate.edu/~burtscher/research/datasets/FPdouble/`, February 2016.

[BW94] Michael Burrows and David J. Wheeler. A block-sorting lossless data compression algorithm. SRC Research Report 124, Digital Systems Research Center, Palo Alto, CA, USA, May 1994.

[BXD+14] Allison H. Baker, Haiying Xu, John M. Dennis, Michael N. Levy, Doug Nychka, Sheri A. Mickelson, Jim Edwards, Mariana Vertenstein, and Al 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, pages 203–214, New York, NY, USA, 2014. ACM.

[BY93] Folkmar Bornemann and Harry Yserentant. A basic norm equivalence for the theory of multilevel methods. *Numerische Mathematik*, 64(1):455–476, December 1993.

[Cha17] Choong-Seock Chang. Private correspondence, 2017.

[Coa14] Josh Coalson. FLAC format. `https://xiph.org/flac/`, 2014.

[CT87] M. Crouzeix and V. Thomée. The stability in L_p and W_p^1 of the L_2-projection onto finite element function spaces. *Mathematics of Computation*, 48(178):521–532, May 1987.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, Inc., New York, 1st edition, 1991.

[Cut52] Cassius C. Cutler. Differential quantization of communication signals, July 1952. U.S. Classification 375/247, 327/105, 340/870.19, 341/143, 341/200, 375/E07.265, 375/252, 327/75; International Classification H03M3/04, G06T9/00, H04N7/34; Cooperative Classification H04N19/00763, H03M3/04; European Classification H03M3/04, H04N7/34.

[Dau90] I. Daubechies. The wavelet transform, time-frequency localization and signal analysis. *IEEE Transactions on Information Theory*, 36(5):961–1005, September 1990.

[DC16] Sheng Di and Franck Cappello. Fast error-bounded lossy HPC data compression with SZ. In *2016 IEEE 30th International Parallel and Distributed Processing Symposium*, pages 730–739, Chicago, IL, USA, May 2016. IEEE.

[DD16] Antonio Diaz Diaz. Lzip. `http://lzip.nongnu.org/lzip.html`, June 2016.

[DK92] Wolfgang Dahmen and Angela Kunoth. Multilevel preconditioning. *Numerische Mathematik*, 63(3):315–344, December 1992.

[DVDD98] D. L. Donoho, M. Vetterli, R. A. DeVore, and I. Daubechies. Data compression and harmonic analysis. *IEEE Transactions on Information Theory*, 44(6):2435–2476, October 1998.

[Eng00] Vadim Engelson. *Tools for Design, Interactive Simulation, and Visualization of Object-Oriented Models in Scientific Computing*. PhD thesis, Linköping University, Linköping, Sweden, June 2000.

[ET96] D. E. Edmunds and H. Triebel. *Function Spaces, Entropy Numbers, Differential Operators*. Cambridge University Press, Cambridge New York, 1st edition, 1996.

[FAA+17] Ian Foster, Mark Ainsworth, Bryce Allen, Julie Bessac, Franck Cappello, Jong Youl Choi, Emil Constantinescu, Philip E. Davis, Sheng Di, Wendy Di, Hanqi Guo, Scott Klasky, Kerstin Kleese Van Dam, Tahsin Kurc, Qing Liu, Abid Malik, Kshitij Mehta, Klaus Mueller, Todd Munson, George Ostouchov, Manish Parashar, Tom Peterka, Line Pouchard, Dingwen Tao, Ozan Tugluk, Stefan Wild, Matthew Wolf, Justin M. Wozniak, Wei Xu, and Shinjae Yoo. Computing just what you need: Online data analysis and reduction at extreme scales. In Francisco F. Rivera, Tomás F. Pena, and José C. Cabaleiro, editors, *Euro-Par 2017: Parallel Processing*, volume 10417, pages 3–19. Springer International Publishing, Cham, 2017.

[FY94] James E. Fowler and Roni Yagel. Lossless compression of volume data. In *Proceedings of the 1994 Symposium on Volume Visualization*, pages 43–50, Washington, DC, USA, October 1994. ACM Press.

[GA13a] Jean-loup Gailly and Mark Adler. Gzip. `https://www.gnu.org/software/gzip/`, June 2013.

[GA13b] Jean-loup Gailly and Mark Adler. zlib. `https://zlib.net/`, April 2013.

[GK10] Eugene Gover and Nishan Krikorian. Determinants and the volumes of parallelotopes and zonotopes. *Linear Algebra and its Applications*, 433(1):28–40, July 2010.

[GKG99] S. Grgic, K. Kers, and M. Grgic. Image compression using wavelets. In *Proceedings of the IEEE International Symposium on Industrial Electronics, 1999. ISIE '99*, volume 1, pages 99–104, 1999.

[GO17] Michael Griebel and Peter Oswald. Stable splittings of Hilbert spaces of functions of infinitely many variables. *Journal of Complexity*, 41:126–151, August 2017.

[GTAF97] AB Gumel, EH Twizell, MA Arigu, and F Fakhir. Numerical methods for a non-linear system arising in chemical kinetics. *Pertanika Journal of Science & Technology*, 5(2):191–200, 1997.

[GVL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, Maryland, 3rd edition, 1996.

[HJ90] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, 1990. Corrected reprint of the 1985 original.

[HV91] Paul G. Howard and Jeffrey Scott Vitter. New methods for lossless image compression using arithmetic coding. In *Data Compression Conference, 1991. DCC '91.*, pages 257–266, April 1991.

[ILRS03] Lawrence Ibarria, Peter Lindstrom, Jarek Rossignac, and Andrzej Szymczak. Out-of-core compression and decompression of large n-dimensional scalar fields. *Computer Graphics Forum*, 22(3):343–348, September 2003.

[ILS05a] Martin Isenburg, Peter Lindstrom, and Jack Snoeyink. Lossless compression of predicted floating-point geometry. *Computer-Aided Design*, 37(8):869–877, July 2005.

[ILS05b] Martin Isenburg, Peter Lindstrom, and Jack Snoeyink. Streaming compression of triangle meshes. In *Proceedings of the Third Eurographics Symposium on Geometry Processing*, pages 111–118, Vienna, Austria, July 2005. Eurographics Association.

[IMII08] IEEE Computer Society, Microprocessor Standards Committee, Institute of Electrical and Electronics Engineers, and IEEE-SA Standards Board. *IEEE Standard for Floating-Point Arithmetic*. Institute of Electrical and Electronics Engineers, New York, NY, 2008.

[JKHM93] W. W. Jiang, S. Z. Kiang, N. Z. Hakim, and H. E. Meadows. Lossless compression for medical imaging systems using linear/nonlinear prediction and arithmetic coding. In *ISCAS '93, 1993 IEEE International Symposium on Circuits and Systems, 1993*, volume 1, pages 283–286, May 1993.

[Joh17a] Johns Hopkins Turbulence Databases. Channel flow dataset description. `http://turbulence.pha.jhu.edu/Channel_Flow.aspx`, October 2017.

[Joh17b] Johns Hopkins Turbulence Databases. Forced isotropic turbulence dataset description. `http://turbulence.pha.jhu.edu/Forced_isotropic_turbulence.aspx`, October 2017.

[Kol41] A. Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large Reynolds' numbers. *Akademiia Nauk SSSR Doklady*, 30:301–305, 1941.

[LI06] Peter Lindstrom and Martin Isenburg. Fast and efficient compression of floating-point data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1245–1250, September 2006.

[Lin14] Peter Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2674–2683, December 2014.

[LKS+08] Jay F. Lofstead, Scott Klasky, Karsten Schwan, Norbert Podhorszki, and Chen Jin. Flexible IO and integration for scientific codes through the adaptable IO system (ADIOS). In *Proceedings of the 6th International Workshop on Challenges of Large Applications in Distributed Environments*, CLADE '08, pages 15–24, New York, NY, USA, 2008. ACM.

[LLT+14] Qing Liu, Jeremy Logan, Yuan Tian, Hasan Abbasi, Norbert Podhorszki, Jong Youl Choi, Scott Klasky, Roselyne Tchoua, Jay Lofstead, Ron Oldfield, Manish Parashar, Nagiza Samatova, Karsten Schwan, Arie Shoshani, Matthew Wolf, Kesheng Wu, and Weikuan Yu. Hello ADIOS: The challenges and lessons of developing leadership class I/O frameworks. *Concurrency and Computation: Practice and Experience*, 26(7):1453–1473, May 2014.

[LM15] Myoungkyu Lee and Robert D. Moser. Direct numerical simulation of turbulent channel flow up to $Re_\tau \approx 5200$. *Journal of Fluid Mechanics*, 774:395–415, 2015.

[LPW+08] Yi Li, Eric Perlman, Minping Wan, Yunke Yang, Charles Meneveau, Randal Burns, Shiyi Chen, Alexander Szalay, and Gregory Eyink. A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence. *Journal of Turbulence*, 9:N31, July 2008.

[LSE+11] Sriram Lakshminarasimhan, Neil Shah, Stephane Ethier, Scott Klasky, Rob Latham, Rob Ross, and Nagiza F. Samatova. Compressing the incompressible with ISABELA: In-situ reduction of spatio-temporal data. In Emmanuel Jeannot, Raymond Namyst, and Jean Roman, editors, *Euro-Par 2011: Parallel Processing Workshops*, volume 6852 of *Lecture Notes in Computer Science*, pages 366–379, Bordeaux, France, August 2011. Springer Berlin Heidelberg.

[McC95] John D. McCalpin. Memory bandwidth and machine balance in current high performance computers. *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, pages 19–25, December 1995.

[McC16] John D. McCalpin. Memory bandwidth and system balance in HPC systems. `https://sites.utexas.edu/jdm4372/2016/11/22/sc16-invited-talk-memory-bandwidth-and-system-balance-in-hpc-systems/`, November 2016.

[McC18] John D. McCalpin. Re: Help reproducing FLOPS/memory figure, April 2018.

[MGBB00] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek. An overview of JPEG-2000. In *Proceedings DCC 2000. Data Compression Conference*, pages 523–541, 2000.

[Now14] Lucy Nowell. Science at extreme scale: Architectural challenges and opportunities, April 2014.

[Oba15] Barack Obama. Creating a national strategic computing initiative. Executive Order 13702, July 2015.

[Osw94] Peter Oswald. *Multilevel Finite Element Approximation*. Teubner Skripten zur Numerik. B. G. Teubner, Stuttgart, 1994. Theory and applications.

[Pav15] Igor Pavlov. LZMA specification (draft), June 2015.

[PBLM07] Eric Perlman, Randal Burns, Yi Li, and Charles Meneveau. Data exploration of turbulence simulations using a database cluster. In *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, volume 23, Reno, NV, USA, November 2007. ACM.

[PLL+16] Norbert Podhorszki, Qing Liu, Jeremy Logan, Jingqing Mu, Hasan Abbasi, Jong-Youl Choi, and Scott A. Klasky. ADIOS 1.10 user's manual. Technical report, U.S. Department of Energy Office of Science, July 2016.

[Ral01] Anthony Ralston. *A First Course in Numerical Analysis: Second Edition*. Dover Publications, Mineola, NY, 2nd edition, February 2001.

[Ric04] Iain E. G. Richardson. *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia*. John Wiley & Sons, Ltd., January 2004.

[RKB06] Paruj Ratanaworabhan, Jian Ke, and Martin Burtscher. Fast lossless compression of scientific floating-point data. In *Data Compression Conference, 2006. DCC 2006. Proceedings*, pages 133–142, March 2006.

[Rob94] Tony Robinson. SHORTEN: Simple lossless and near-lossless waveform compression. Technical report CUED/F-INFENG/TR.156, Citeseer, Cambridge University Engineering Department, 1994.

[Roe03] Greg Roelofs. *PNG: The Definitive Guide*. Greg Roelofs, San Jose, CA, USA, 2nd edition, July 2003.

[Sal07] David Salomon. *Data Compression: The Complete Reference*. Springer London, London, 4th edition, 2007.

[Sew10] Julian Seward. bzip2. `http://www.bzip.org/`, September 2010.

[SFPR05] Kai Schneider, Marie Farge, Giulio Pellegrino, and Michael M. Rogers. Coherent vertex simulation of three-dimensional turbulent mixing layers using orthogonal wavelets. *Journal of Fluid Mechanics*, 534:39–66, 2005.

[Sha01] Claude Elwood Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, January 2001.

[SJS+12] Eric R. Schendel, Ye Jin, Neil Shah, Jackie Chen, C. S. Chang, Seung-Hoe Ku, Stephane Ethier, Scott Klasky, Robert Latham, Robert Ross, and Nagiza F. Samatova. ISOBAR preconditioner for effective and high-throughput lossless data compression. In *2012 IEEE 28th International Conference on Data Engineering*, pages 138–149, April 2012.

[SMW+04] Magnus Strengert, Marcelo Magallón, Daniel Weiskopf, Stefan Guthe, and Thomas Ertl. Hierarchical visualization and compression of large volume datasets using GPU clusters. *EGPGV*, pages 41–48, 2004.

[SSL+12] Neil Shah, Eric R. Schendel, Sriram Lakshminarasimhan, Saurabh V. Pendse, Terry Rogers, and N. F. Samatova. Improving I/O throughput with PRIMACY: Preconditioning ID-mapper for compressing incompressibility. In *2012 IEEE International Conference on Cluster Computing*, pages 209–219, September 2012.

[TG00] Costa Touma and Craig Gotsman. Triangle mesh compression, December 2000. U.S. Classification 382/242, 382/243, 345/440, 345/419; International Classification G06T9/00, G06T17/20; Cooperative Classification G06T9/001, G06T17/20; European Classification G06T17/20, G06T9/00F.

[TGC99] E. H. Twizell, A. B. Gumel, and Q. Cao. A second-order scheme for the "Brusselator" reaction–diffusion system. *Journal of Mathematical Chemistry*, 26(4):297–316, November 1999.

[Thi14] Patrick Thibodeau. Coming by 2023, an exascale supercomputer in the U.S. *Computerworld*, October 2014.

[TR98] Gabriel Taubin and Jarek Rossignac. Geometric compression through topological surgery. *ACM Trans. Graph.*, 17(2):84–115, April 1998.

[Wal92] G. K. Wallace. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, February 1992.

[WSS00] M. J. Weinberger, G. Seroussi, and G. Sapiro. The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS. *IEEE Transactions on Image Processing*, 9(8):1309–1324, August 2000.

[ZL77] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, May 1977.