SIAM J. Sci. Comput.
Vol. 39, No. 4, pp. B732–B757

COMPRESSION USING LOSSLESS DECIMATION: ANALYSIS AND APPLICATION[*]

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

Abstract. A crude but commonly used technique for compressing ordered scientific data consists of simply retaining every sth datum (with a value of $s = 10$ generally the default) and discarding the remainder. Should the value of a discarded datum be required afterwards, an approximation is generated by linear interpolation of the two nearest retained values. Despite the widespread use of this and similar techniques, there is little by way of theoretical analysis of their expected performance. First, we quantify the accuracy achieved by linear interpolation when approximating values discarded by decimation, obtaining both deterministic bounds in terms of appropriate smoothness measures of the data and probabilistic bounds in terms of statistics of the data. Second, we investigate the efficiency of the lossless compression scheme consisting of decimation coupled with encoding of the interpolation errors. In particular, we bound the expected compression ratio in terms of the appropriate measures of the data. Finally, we provide numerical illustrations of the practical performance of the algorithm on some real datasets.

Key words. lossy compression, lossless compression, decimation, predictive coding

AMS subject classifications. 95B65, 68P30, 94A24

DOI. 10.1137/16M1086248

1. Introduction.

The advent of exascale systems [43] is expected to enable scientific computation and simulation at an unprecedented scale at drastically increased levels of resolution. Effective utilization of exascale systems will pose new challenges in terms of resiliency and fault tolerance of algorithms and, perhaps more significantly, in terms of the vastly increased amounts of data being produced by the simulation. Already, the widening gap between compute speed and I/O rates means that it is no longer a viable proposition to simply store all of the data for offline analysis [3]. Meeting this challenge will require advances in hardware and software [31] including the development and analysis of numerical algorithms that are tailored to handling data produced by scientific simulation.

Consider data $\{u_j\}_{j=0}^{N}$ produced by a computational simulation of some system at a sequence of times $\{t_j\}_{j=0}^{N}$. 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 1. Should the value of a discarded datum u_j be required following decimation, it is tacitly

[*]Submitted to the journal's Computational Methods in Science and Engineering section July 25, 2016; accepted for publication (in revised form) April 4, 2017; published electronically August 30, 2017.

http://www.siam.org/journals/sisc/39-4/M108624.html

Funding: This work was partially supported by DOE Storage Systems and Input/Output for Extreme Scale Science project, announcement LAB 15-1338, and DOE and UT–Battelle, LLC, contract DE-AC05-00OR22725.

[†]Division of Applied Mathematics, Brown University, 182 George St., Providence, RI 02912 and Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831 (mark_ainsworth@brown.edu).

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

[§]Division of Applied Mathematics, Brown University, 182 George St., Providence, RI 02912 (ben_whitney@brown.edu).

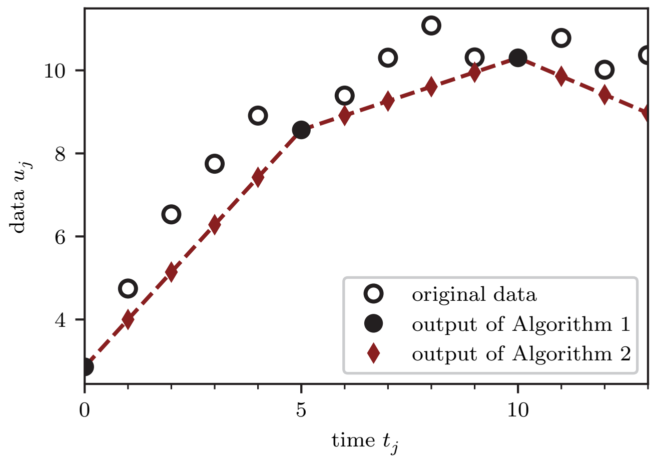

FIG. 1. *Illustration of the use of Algorithms* 1 *and* 2 *with stride* $s = 5$. *The original data* ○ *are reduced to the decimated data* ●, *giving a compression ratio of* s. *The discarded values are approximated by the interpolations* ♦.

assumed that an approximation will be generated by simple linear interpolation of the two nearest retained values, as in Algorithm 2. See Figure 1.

Algorithm 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.

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

Decimation has the obvious benefit of a guaranteed data reduction by a factor of 10 (s, in general), but it suffers from being inherently lossy. While judicious use of lossy procedures can have practical benefit, in the 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 often provide 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, claim that the surrogate data are acceptable? The key lies in distinguishing between *data* and *information*. The data $\{u_j\}_{j=0}^{N}$ embody information on the system being simulated. They could, for example, be the values of an underlying function u sampled at the time steps $\{t_j\}_{j=0}^{N}$. The data generally require an enormous amount of storage irrespective of the nature of the function u.

Algorithm 2 Approximation via linear interpolation. Each discarded value is approximated by a weighted average of the nearest retained values.

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

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 the function 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 produced by the stream.

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.

The heuristic explanation of how decimation can function in principle is compelling, 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*.

We view the lossless decimation technique considered in the present work as an archetype of the broad class of *predictive coding* data reduction techniques consisting of algorithms that use past data values to generate predictions for future data values and encode the resulting residuals. 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 [13]. Subsequently, DPCM was modified for use with nonuniformly sampled signals by replacing simple differences with prediction errors arising from extrapolating polynomials [16]. Predictive coding schemes based on polynomial extrapolation or interpolation have been applied in a range of areas, including mesh compression [44, 42, 26], volume compression [23, 17], audio compression [36, 11], image compression [45, 37, 27], and floating point compression [30, 14]. Predictive coding based on statistical techniques was used in [2], while hash functions were used in [35, 9, 10]. Alternative approaches attempt to preprocess datasets so as to make them more amenable to compression [28, 40, 38].

The main objective in the present work is to fill a gap in high performance computing 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 work will pave the way for further work in this direction.

2. Deterministic bounds on approximation errors.

Decimation in effect replaces the original dataset $\{u_j\}_{j=0}^N$ by a surrogate dataset $\{\tilde{u}_j\}_{j=0}^N$, 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 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* $\{\delta_j^1\}_{j=0}^{N-1}$ are defined by $\delta_j^1 = u_{j+1} - u_j$, and the *second order differences* $\{\delta_j^2\}_{j=1}^{N-1}$ are defined by $\delta_j^2 = \delta_j^1 - \delta_{j-1}^1$ [34]. We shall on occasion refer to $\{u_j\}_{j=0}^N$ 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 $\boldsymbol{u}\colon \{0, \ldots, N/s-1\} \to \mathbb{R}^{s+1}$, $\boldsymbol{\delta}^1\colon \{0, \ldots, N/s-1\} \to \mathbb{R}^s$, $\boldsymbol{\delta}^2\colon \{0, \ldots, N/s-1\} \to \mathbb{R}^{s-1}$, and $\boldsymbol{\Delta}\colon \{0, \ldots, N/s-1\} \to \mathbb{R}^{s-1}$ as follows:

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

These processes simply group scalars into vectors.

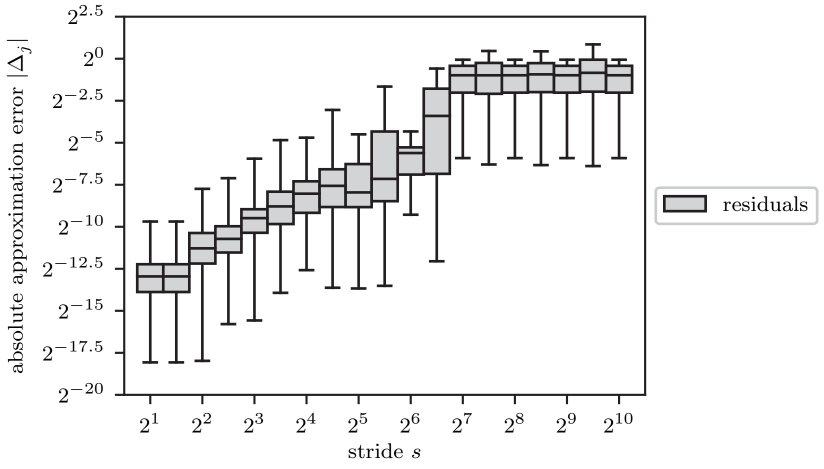

FIG. 2. *Illustration of the effect of stride size s on approximation errors. As one may expect, larger strides result in larger errors, but the relationship is nontrivial. There appear to be three regimes, with transitions between them occurring near $s = 2^4$ and $s = 2^7$. The data used for this plot 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 $\{\Delta_j\}_{j=0}^N$. 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* 3 *to* 6 *are generated in the same fashion.*

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

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

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

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

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

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

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

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

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

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

$$\begin{aligned}\Delta_{ms+i} &= u_{ms+i} - [(1-\tfrac{i}{s})u_{ms} + \tfrac{i}{s}u_{ms+s}] \\ &= (1-\tfrac{i}{s})(u_{ms+i} - u_{ms}) - \tfrac{i}{s}(u_{ms+s} - u_{ms+i}).\end{aligned} \tag{2}$$

$u_{ms+i} - u_{ms}$ 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), we find that

$$\Delta_{ms+i} = (1-\tfrac{i}{s}) \sum_{j=0}^{i-1} \delta^1_{ms+j} - \tfrac{i}{s} \sum_{j=i}^{s-1} \delta^1_{ms+j} = \sum_{j=0}^{s-1} \Psi^1_{i,j} \delta^1_{ms+j}. \tag{3}$$

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 (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 (3), we find that

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

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:*

$$|\Delta_{ms+i}| \le 2\mathfrak{M}^0, \qquad \textit{where} \quad \mathfrak{M}^0 = \max_{0 \le j \le N} |u_j|, \tag{4a}$$

$$|\Delta_{ms+i}| \le \tfrac{1}{2} s \mathfrak{M}^1, \qquad \textit{where} \quad \mathfrak{M}^1 = \max_{0 \le j \le N-1} |\delta^1_j|, \tag{4b}$$

$$|\Delta_{ms+i}| \le \tfrac{1}{8} s^2 \mathfrak{M}^2, \qquad \textit{where} \quad \mathfrak{M}^2 = \max_{1 \le j \le N-1} |\delta^2_j|. \tag{4c}$$

Moreover, the constant factors in these bounds are optimal.

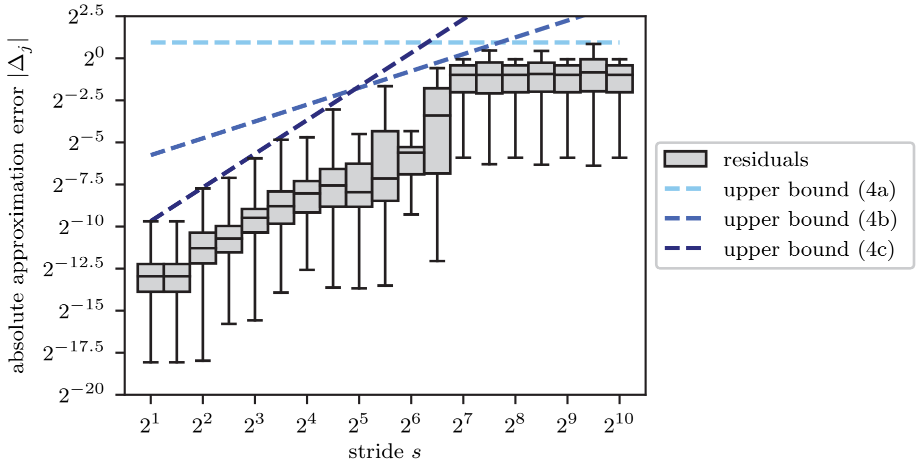

FIG. 3. *Illustration of the bounds in Theorem* 2.2. *The dataset used is the same as in Figure* 2.

Proof. $|\Delta_{ms+i}| \leq \|\boldsymbol{\Delta}_m\|_\infty$, 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 $\|\boldsymbol{u}_m\|_\infty \leq \mathfrak{M}^0$, so for (4a) it suffices to show that $\|\Psi^0\|_\infty \leq 2$. Similarly, for (4b) it suffices to show that $\|\Psi^1\|_\infty \leq s/2$, and for (4c) it suffices to show that $\|\Psi^2\|_\infty \leq s^2/8$. The matrix norm subordinate to the maximum vector norm is computed by finding the maximum of the absolute row sums [21]. Consequently,

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

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

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

The bounds (4a) to (4c) follow. For optimality, it suffices to find datasets for which each bound is achieved. For data given by $u_j = (-1)^j$, (4a) is tight; for data given by $u_j = |j - s/2|$, (4b) is tight; and for data given by $u_j = j^2$, (4c) is tight. ☐

Are the inequalities in Theorem 2.2 consistent with the behavior observed in Figure 2? In Figure 3 we plot these bounds along with the actual errors. It is observed that whilst no single one of the bounds (4a) to (4c) is tight for all strides used, their minimum is sharp over the full range.

Example 1. Theorem 2.2 is well suited to the case where $\{u_j\}_{j=0}^N$ are values taken by some deterministic function at uniformly spaced times $\{jh\}_{j=0}^N$. Suppose $u_j = f(jh)$ with $f \in C([0, Nh])$. $\mathfrak{M}^0 \leq \|f\|_\infty$, so (4a) becomes

$$|\Delta_{ms+i}| \leq 2\|f\|_\infty. \tag{5a}$$

We can make use of any additional smoothness of f by relating the differences of $\{u_j\}_{j=0}^N$ to the derivatives of f. If $f \in C^1([0, Nh])$, then $|\delta_j^1| = |\int_{jh}^{jh+h} f'(x)\,\mathrm{d}x| \leq h\|f'\|_\infty$, and (4b) becomes

$$|\Delta_{ms+i}| \leq \tfrac{1}{2}sh\|f'\|_\infty. \tag{5b}$$

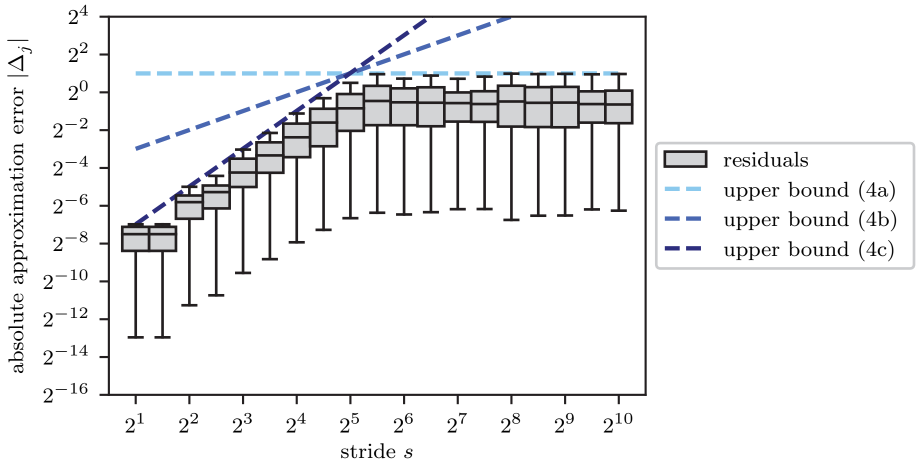

FIG. 4. *Illustration of the bounds of Theorem* 2.2 *when* $u_j = \sin(jh)$ *with* $h = 2^{-3}$ *and* $j \in \{0, \ldots, \lfloor 2^{12}\pi/h \rfloor\}$. *Upper bound* (4c) *is the tightest until sh is about half the period of* sin. *This matches the behavior predicted by the bounds* (5a) *to* (5c)*: with* $\|\sin\|_\infty = \|\sin'\|_\infty = \|\sin''\|_\infty$, (4c) *will be tightest whenever sh is less than* 2^2.

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

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

Taking absolute values and integrating, we find that $|\delta_j^2| \leq h^2\|f''\|_\infty$, and (4c) becomes

$$|\Delta_{ms+i}| \leq \tfrac{1}{8}(sh)^2\|f''\|_\infty, \tag{5c}$$

which is the well-known estimate for piecewise linear interpolation [34, pp. 54–55].

See Figure 4 for an illustration of the bounds in the case of sinusoidal data.

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. Let $\{u_j\}_{j=0}^N$ be the positions of a Brownian motion sample path at times $\{jh\}_{j=0}^N$ with $h = 2^{-13}$ and $N = 2^{20}$. In this case $u_{j+1} = u_j + X_j$, where $\{X_j\}_{j=0}^N$ are independent, identically distributed $\mathcal{N}(0, h)$ random variables. The actual approximation errors along with the bounds found in Theorem 2.2 are presented in Figure 5. All of the upper bounds (4a) to (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

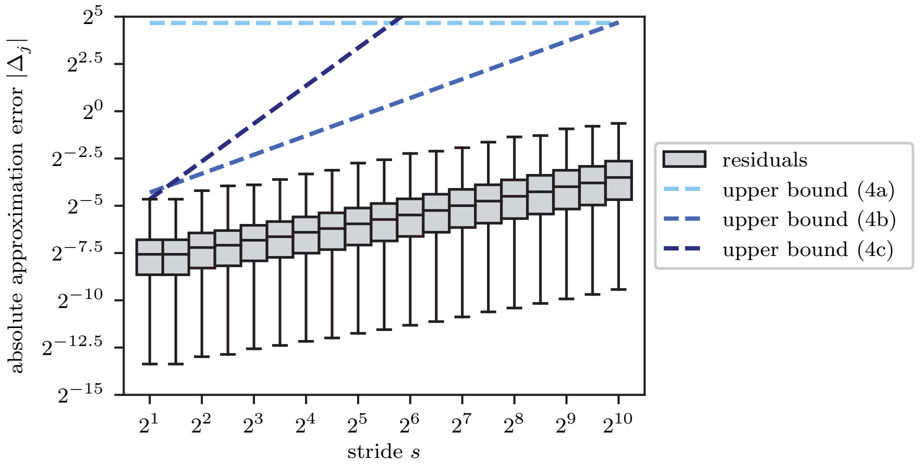

FIG. 5. *Illustration of the bounds of Theorem* 2.2 *with* $\{u_j\}_{j=0}^N$ *the positions of a Brownian motion sample path. The bounds fail to reflect the typical magnitudes of the approximation errors.*

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 $\{f_j\}_{j=0}^N$ is subject to measurement errors $\{X_j\}_{j=0}^N$, then observational data will take the form

$$u_j = f_j + X_j, \tag{6a}$$

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

$$u_{j+1} - u_j = f_j + X_j, \tag{6b}$$

where f_j is a deterministic drift and $\{X_j\}_{j=0}^N$ are again independent, identically distributed random variables as in (6a) but now representing the underlying volatility. Equation (6a), respectively, (6b), corresponds to the case that the zeroth, respectively, first, 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

$$u_{j+1} - 2u_j + u_{j-1} = f_j + X_j \tag{6c}$$

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

As Example 2 shows, Theorem 2.2 can sometimes give poor bounds when applied to data having one of the forms (6a) to (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 $\{X_j\}_{j=0}^N$ are known, though, they can be used to find information on the distribution of the approximation errors $\{\Delta_j\}_{j=0}^N$, as in the next result.

THEOREM 3.1. *Let $\{f_j\}_{j=0}^N$ be deterministic, and let $\{X_j\}_{j=0}^N$ be independent, identically distributed random variables with mean μ and variance σ^2. Define $\boldsymbol{f}\colon \{0,\dots,N/s-1\}\to\mathbb{R}^{s+1}$ 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 $\Psi^2\boldsymbol{f}_m+\mu\Psi^2\mathbf{1}$ and covariance matrix $\sigma^2\Psi^2\Psi^{2\intercal}$.*

Proof. Define $\boldsymbol{X}\colon\{0,\dots,N/s-1\}\to\mathbb{R}^{s+1}$ 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,

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

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

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

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

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

Next, suppose $\delta_j^1=f_j+X_j$. By the same argument as in the previous case, we find that $\mathbb{E}\,\boldsymbol{\Delta}_m=\Psi^1\boldsymbol{f}_m+\mu\Psi^1\mathbf{1}$. $\mathbf{1}\in\ker(\Psi^1)$, since $\sum_{j=0}^{s-1}\Psi_{i,j}^1=i(1-\frac{i}{s})-(s-i)\frac{i}{s}=0$, 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 $\operatorname{Cov}(\boldsymbol{\Delta}_m)=\sigma^2\Psi^1\Psi^{1\intercal}$.

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

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

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, $\mathbb{E}\,\boldsymbol{\Delta}_m=\Psi^2\boldsymbol{f}_m+\mu\Psi^2\mathbf{1}$. The covariance calculation proceeds exactly as before, and we find that $\operatorname{Cov}(\boldsymbol{\Delta}_m)=\sigma^2\Psi^2\Psi^{2\intercal}$. ∎

Let us now revisit the case of the Brownian motion data used in Example 2.

Example 2 (continued). Recall that $\{\delta_j^1\}_{j=0}^{N-1}$ are, as increments of a Brownian motion, independent $\mathcal{N}(0,h)$ random variables. As a linear transformation of $\boldsymbol{\delta}_m^1$, a Gaussian random vector $\boldsymbol{\Delta}_m$ is a Gaussian random vector with (by Theorem 3.1)

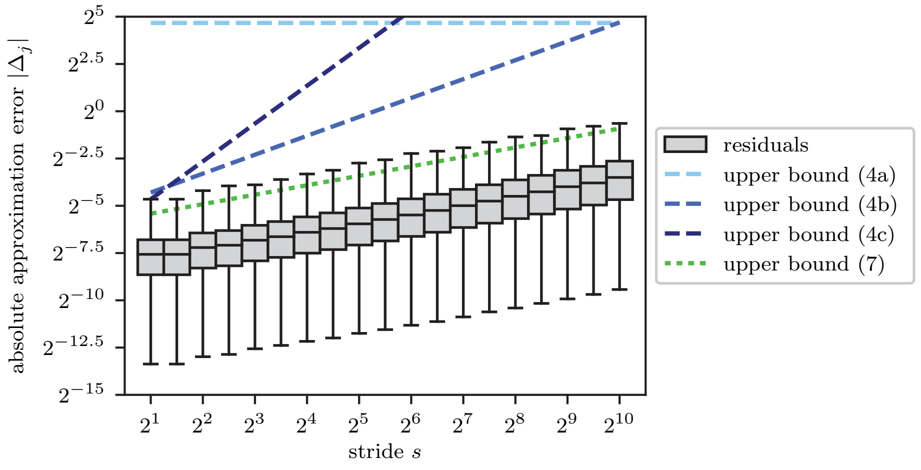

FIG. 6. *Illustration of upper bound* (7) *and the bounds of Theorem* 2.2 *with* $\{u_j\}_{j=0}^{N}$ *the positions of a Brownian motion sample path. Compare with Figure* 5*, which uses the same data. Upper bound* (7)*, derived using Theorem* 3.1*, tracks the growth of the approximation errors better 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.*

mean $\mathbf{0}$ and covariance matrix $h\Psi^1\Psi^{1\intercal}$. We show in Appendix A 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 $\sqrt{sh(1-\frac{i}{s})\frac{i}{s}}$. Maximizing over $i \in \{1, \dots, s-1\}$, we can use

$$\left(-\tfrac{3}{2}\sqrt{sh}, +\tfrac{3}{2}\sqrt{sh}\right) \tag{7}$$

as a 99.7% prediction interval for each Δ_{ms+i}. See Figure 6 for a comparison of (7) with upper bounds (4a) to (4c) for this data.

4. A lossless compression algorithm.

Algorithms 1 and 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 $\{\Delta_j\}_{j=0}^{N}$, 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 3 and 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 [41]. 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

Algorithm 3 Lossless decimation encoding: decimation (Algorithm 1), approximation (Algorithm 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: $s$ 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 4 Lossless decimation decoding: reconstruction of original data from decimated data and errors.

```
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
```

of X, denoted $H(X)$, is defined by

$$H(X) = -\sum_{x \in S} \mathbb{P}(X = x) \log_2 \mathbb{P}(X = x). \tag{8}$$

The compression ratio achieved by Algorithm 3, then, can be related to the Shannon entropy of the residuals $H(\boldsymbol{\Delta})$, as in the following result.

Theorem 4.1. *Let $\boldsymbol{\Delta}$ be an $\mathbb{R}^{s-1}$-valued process, and suppose that a dataset $\{u_j\}_{j=0}^{N}$ is generated in such a way that the approximation errors $\{\boldsymbol{\Delta}_m\}_{m=0}^{N/s-1}$ are sampled from $\boldsymbol{\Delta}$. If each datum requires b bits of storage, then the expected compression ratio $\mathfrak{R}$ achieved by Algorithm 3 when applied to $\{u_j\}_{j=0}^{N}$ is*

$$\mathfrak{R} = s\big[1 + \tfrac{1}{b} H(\boldsymbol{\Delta})\big]^{-1} \tag{9}$$

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 $\{\boldsymbol{\Delta}_m\}_{m=0}^{N/s-1}$, 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 [41]. 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 (9) in the limit $N \to \infty$. □

Theorem 4.1 provides a quantitative relationship between the entropy of the approximation errors and the compression ratio achieved by Algorithm 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 1; Algorithm 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 3.

In order to apply Theorem 4.1, 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 $\{\boldsymbol{\Delta}\cdot\boldsymbol{e}_i\}_{i=1}^{s-1}$, can be determined instead, then they may be used to bound $\mathfrak{R}$ according to the following result.

Corollary 4.2. *Let $\boldsymbol{\Delta}$ be an $\mathbb{R}^{s-1}$-valued process, and suppose that a dataset $\{u_j\}_{j=0}^{N}$ is generated in such a way that the approximation errors $\{\boldsymbol{\Delta}_m\}_{m=0}^{N/s-1}$ are sampled from $\boldsymbol{\Delta}$. If each datum requires b bits of storage, then the expected compression ratio $\mathfrak{R}$ achieved by Algorithm 3 when applied to $\{u_j\}_{j=0}^{N}$ satisfies*

$$\mathfrak{R} \geq s\left[1 + \frac{1}{b}\sum_{i=1}^{s-1} H(\boldsymbol{\Delta}\cdot\boldsymbol{e}_i)\right]^{-1} \tag{10}$$

in the limit $N \to \infty$.

Proof. $H(\boldsymbol{\Delta}) \leq \sum_{i=1}^{s-1} H(\boldsymbol{\Delta}\cdot\boldsymbol{e}_i)$ [41]. The result now follows from Theorem 4.1. □

Corollary 4.2 yields an inequality instead of an equality because it, unlike Theorem 4.1, 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 5 and 6.

Figure 7 illustrates the relationship between the value for $\mathfrak{R}$ obtained using (9), the bound on $\mathfrak{R}$ obtained using lower bound (10), and the actual compression ratio achieved by Algorithm 3. Figure 8 shows that the linear predictor is responsible for the bulk of the achieved compression. As written, Algorithm 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` [39], which uses the Burrows–Wheeler algorithm [6];
2. `gzip` [18], which uses the LZ77 algorithm [46];
3. `lzip` [15], which uses LZMA [32];

and four specialized floating point compressors:

1. `SPDP` [8], which is a lossless black box compressor;

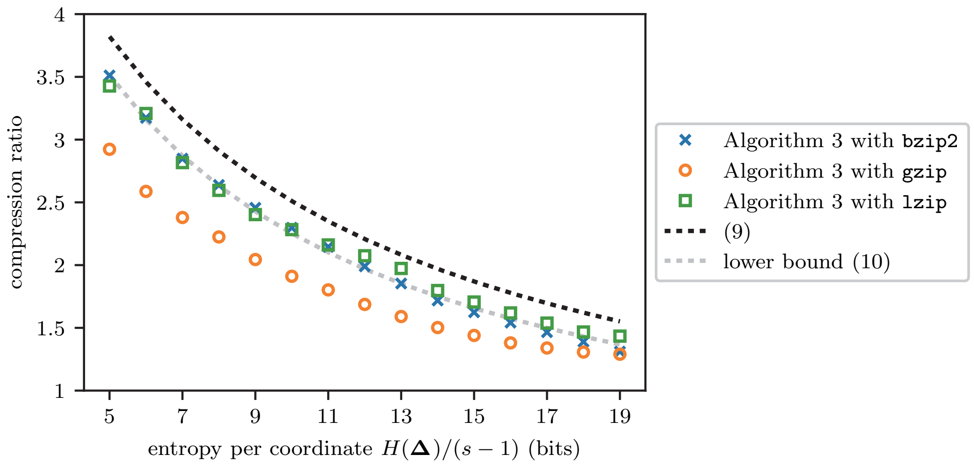

FIG. 7. *Illustration of the relationship between the Shannon entropy of the residuals and the compression ratio achieved by Algorithm* 3. *The data are constructed so that the* $\{\boldsymbol{\Delta}\cdot\boldsymbol{e}_i\}_{i=1}^{s-2}$ *are independent, identically distributed random variables of known entropy and* $\boldsymbol{\Delta}\cdot\boldsymbol{e}_{s-1}=\boldsymbol{\Delta}\cdot\boldsymbol{e}_{s-2}$. *Consequently,* $H(\boldsymbol{\Delta})\lneq\sum_{i=1}^{s-1}H(\boldsymbol{\Delta}\cdot\boldsymbol{e}_i)$*, and Theorem* 4.1 *and Corollary* 4.2 *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.*

2. `SZ` [14], which 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` [30], which can compress one dimensional (1D), two dimensional (2D), and three dimensional (3D) scalar fields in lossy and lossless configurations, and which we use in lossless 1D mode;
4. `fpc` [9], which is a lossless compressor for double-precision data. When pairing `fpc` with Algorithm 3, we will use only the encoder, not the predictor, of the former. We will refer to the combination as Algorithm 3 with `fpc_encode`.

We could obtain compression ratios closer to (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 [25, 2, 40, 9]. 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 [27, 22, 45].

5. Deterministic bounds on Shannon entropy of approximation errors.

In this section we use the results of section 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 existence of some $\mathbb{R}^{s-1}$-valued process modeling the approximation errors $\{\boldsymbol{\Delta}_m\}_{m=0}^{N/s-1}$, as in Theorem 4.1.

Suppose that the vectors of approximation errors $\boldsymbol{\Delta}_m=(\Delta_{ms+1},\dots,\Delta_{ms+s-1})$ 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 $\boldsymbol{\Delta}$. This turns out not to be the case, but it is possible to relate the volume of Ω to the *differential entropy* of $\boldsymbol{\Delta}$ instead. The differential entropy of an $\mathbb{R}^n$-valued random variable X with probability

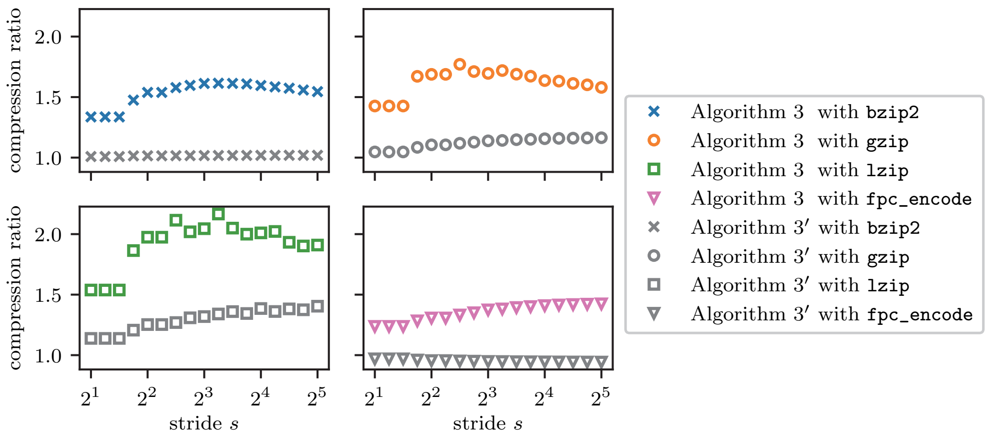

FIG. 8. *Illustration of the effect of replacing the linear predictor in Algorithm* 3 *with a constant predictor. Algorithm* 3′ *denotes the procedure obtained by replacing line* 3 *in Algorithm* 3 *with "*`surrogate` $\leftarrow$ `[0, ..., 0]`*." The results demonstrate that the linear predictor is responsible for the bulk of the achieved compression rates. The data used are given by* $u_j = \tan(jh)$ *with* $h = 2^{-15}$ *and* $j \in \{0, \ldots, \lfloor 2\pi/h \rfloor\}$.

density p is given by

$$\mathfrak{H}(X) = -\int_{\mathbb{R}^n} p(\boldsymbol{x}) \log_2 p(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}. \tag{11}$$

Differential entropy is related to Shannon entropy as follows [12, p. 229].[1]

LEMMA 5.1. *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*

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

as $w \to 0$.

Lemma 5.1 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 5.1 is of little use for most data. However, the precision of floating-point formats is fixed within each interval of the form $[\pm 2^n, \pm 2^{n+1})$ [24], 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 5.1, in what follows we will assume that $\{u_j\}_{j=0}^N$ is generated in such a way that we can model the approximation errors $\{\boldsymbol{\Delta}_m\}_{m=0}^{N/s-1}$ 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.

[1]The proof given in [12] is restricted to the $n = 1$ case, but it extends to the general case without significant modification.

Lemma 5.2. *Let X be a random variable taking values in $\Omega \subset \mathbb{R}^n$. Then $\mathfrak{H}(X) \leq \log_2 \operatorname{vol}_n(\Omega)$.*

Proof. Of all distributions with support contained in Ω, the differential entropy is maximized by $\operatorname{Unif}(\Omega)$, and the differential entropy of a uniform distribution is the logarithm of the volume of its support [41]. $\square$

Lemmas 5.1 and 5.2 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 $\{\boldsymbol{\Delta}_m\}_{m=0}^{N/s-1}$. 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 5.3. *The differential entropy of $\boldsymbol{\Delta}$ can be bounded as follows:*

$$\mathfrak{H}(\boldsymbol{\Delta}) \leq (s-1)(1+\log_2 \mathfrak{M}^0) + \log_2 \left[\tfrac{1}{6}(s+1)(s+2)\right], \tag{12a}$$

$$\mathfrak{H}(\boldsymbol{\Delta}) \leq (s-1)(1+\log_2 \mathfrak{M}^1), \tag{12b}$$

$$\mathfrak{H}(\boldsymbol{\Delta}) \leq (s-1)(1+\log_2 \mathfrak{M}^2) - \log_2 s. \tag{12c}$$

Proof. We start with (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 Appendix A that the volume of $\Psi^0[0,1]^{s+1}$ is $\frac{1}{6}(s+1)(s+2)$. Applying Lemma 5.2, we find that

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

The bounds (12b) and (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 Appendix A to be 1. So,

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

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 Appendix A to be $1/s$. So,

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

Now we can combine Lemma 5.3 with Theorem 4.1 to obtain bounds on the expected compression ratio achieved by Algorithm 3.

Theorem 5.4. *In the limit $N \to \infty$ and $w \to 0$, the expected compression ratio $\mathfrak{R}$ achieved by Algorithm 3 is bounded as follows:*

$$\mathfrak{R} \geq s\left[1+\tfrac{1}{b}\left((s-1)(1+\log_2 \mathfrak{M}^0 - \log_2 w) + \log_2\left[\tfrac{1}{6}(s+1)(s+2)\right]\right)\right]^{-1} \tag{13a}$$

$$\mathfrak{R} \geq s\left[1+\tfrac{1}{b}\left((s-1)(1+\log_2 \mathfrak{M}^1 - \log_2 w)\right)\right]^{-1} \tag{13b}$$

$$\mathfrak{R} \geq s\left[1+\tfrac{1}{b}\left((s-1)(1+\log_2 \mathfrak{M}^2 - \log_2 w) - \log_2 s\right)\right]^{-1}. \tag{13c}$$

Proof. Combine Lemmas 5.1 and 5.3 and Theorem 4.1. $\square$

Figure 9 illustrates the bounds in Theorem 5.4.

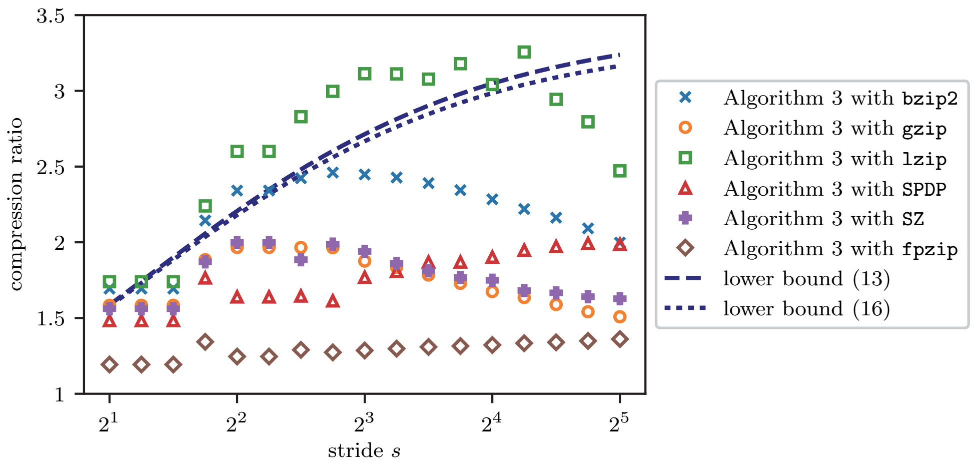

FIG. 9. *Illustration of the lower bounds on the expected compression ratio in Theorem* 5.4 *versus stride* s*. Lower bound* (13) *is the maximum of lower bounds* (13a) *to* (13c)*; lower bound* (16) *is the maximum of lower bounds* (16a) *to* (16c)*. Along with the bounds are plotted the actual compression ratios obtained using various standard compressors in lieu of an ideal entropy encoder. It is observed that using* `lzip` *comes close to achieving the theoretical ratio, 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* $j \in \{0, \ldots, \lfloor 2^6\pi/h \rfloor\}$.

6. Statistical bounds on Shannon entropy of approximation errors. Theorem 5.4 is based on the absolute bounds on $\{\Delta_j\}_{j=0}^N$ derived in Theorem 2.2. As we saw in Example 2, these bounds may be overly pessimistic when applied to random data. Consequently, we expect Theorem 5.4 to be similarly pessimistic in this case. We can obtain more realistic estimates by deriving an analogue to Theorem 5.4 based on Theorem 3.1, the analogue to Theorem 2.2 for random data. We begin with the following result relating variance to differential entropy [12, pp. 234–235]. It plays a role analogous to that of Lemma 5.2.

LEMMA 6.1. *Let* X *be an* $\mathbb{R}^n$*-valued continuous random variable with mean* $\boldsymbol{\mu} \in \mathbb{R}^n$ *and covariance matrix* $\Sigma \in \mathbb{R}^{n\times n}$*. Then*

$$\mathfrak{H}(X) \leq \tfrac{n}{2} \log_2 \left[2\pi \mathrm{e} |\det(\Sigma)|^{1/n}\right] \tag{14}$$

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

If $\{u_j\}_{j=0}^N$ is generated from a sequence of independent, identically distributed continuous random variables, then Lemma 6.1 can be used to bound the differential entropy of $\boldsymbol{\Delta}$ as follows.

LEMMA 6.2. *Let* $\{X_j\}_{j=0}^N$ *be independent, identically distributed random variables with mean* μ *and variance* σ^2.

(a) *If* $u_j = X_j$*, then*

$$\mathfrak{H}(\boldsymbol{\Delta}) \leq \tfrac{s-1}{2} \log_2 \left(2\pi \mathrm{e}\sigma^2\right) + \tfrac{1}{2} \log_2 \left[\tfrac{1}{12s}(s+1)^2(s+2)\right]. \tag{15a}$$

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

$$\mathfrak{H}(\boldsymbol{\Delta}) \leq \tfrac{s-1}{2} \log_2 \left(2\pi \mathrm{e}\sigma^2\right) - \tfrac{1}{2} \log_2(s). \tag{15b}$$

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

$$\mathfrak{H}(\boldsymbol{\Delta}) \leq \tfrac{s-1}{2}\log_2\left(2\pi e\sigma^2\right) - \log_2(s). \tag{15c}$$

Proof. Begin with (15a). We found in Theorem 3.1 that if $u_j = X_j$, then $\boldsymbol{\Delta}$ has mean $\mathbf{0}$ and covariance $\sigma^2\Psi^0\Psi^{0\intercal}$. Applying Lemma 6.1,

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

where we have used the expression for $\det(\Psi^0\Psi^{0\intercal})$ found in Appendix A. The bounds (15b) and (15c) are derived in exactly the same manner. ☐

Now we can combine Lemma 6.2 with Theorem 4.1 to obtain bounds on the expected compression ratio achieved by Algorithm 3.

THEOREM 6.3. *Let* $\{X_j\}_{j=0}^N$ *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* 3 *when applied to* $\{u_j\}_{j=0}^N$ *in the limit* $N \to \infty$ *and* $w \to 0$.

(a) *If* $u_j = X_j$, *then*

$$\mathfrak{R} \geq s\left[1 + \tfrac{1}{b}\left(\tfrac{s-1}{2}\log_2\left(2\pi e\sigma^2 w^2\right) + \tfrac{1}{2}\log_2\left[\tfrac{1}{12s}(s+1)^2(s+2)\right]\right)\right]^{-1}. \tag{16a}$$

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

$$\mathfrak{R} \geq s\left[1 + \tfrac{1}{b}\left(\tfrac{s-1}{2}\log_2\left(2\pi e\sigma^2 w^2\right) - \tfrac{1}{2}\log_2(s)\right)\right]^{-1}. \tag{16b}$$

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

$$\mathfrak{R} \geq s\left[1 + \tfrac{1}{b}\left(\tfrac{s-1}{2}\log_2\left(2\pi e\sigma^2 w^2\right) - \log_2(s)\right)\right]^{-1}. \tag{16c}$$

Proof. Combine Lemmas 5.1 and 6.2 and Theorem 4.1. ☐

Figure 10 illustrates the bounds in Theorem 6.3.

7. Applications.

In this section we evaluate the performance of Algorithm 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.

7.1. Black–Scholes simulation.

First, we consider data generated by a Black–Scholes solver [4] to simulate random fluctuations in the price of an asset over a time interval $[0, 1]$ with 2^{20} time steps. 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 11 shows that reduction of the data by up to 50% is achieved.

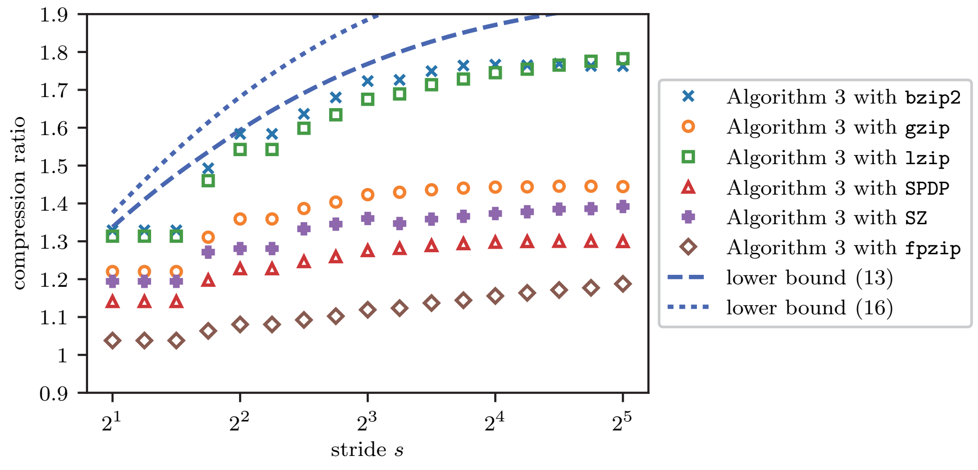

FIG. 10. *Illustration of the bounds of Theorem* 6.3 *when* $\{u_j\}_{j=0}^N$ *are the positions of a Brownian motion with time step* 2^{-10}*. The compression ratios shown are the ensemble averages across* 2^5 *trials with* 2^{20} *time steps each. After each trial, the zeroth, first, and second order differences are calculated, and the maxima and variances of the differences are used to generate bounds according to lower bounds* (13a) *to* (13c) *and lower bounds* (16a) *to* (16c)*. An ensemble average is calculated for each bound, and lower bounds* (13) *and* (16) *are the maxima of these averages. None of the compressors match the optimal compression ratios, but* `bzip2` *and* `lzip` *track* (13)*, achieving good performance.*

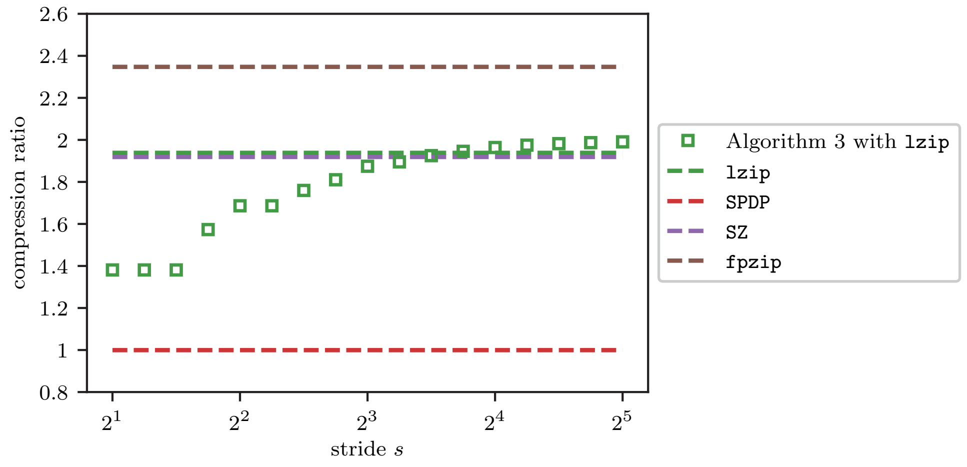

FIG. 11. *Illustration of the performance of Algorithm* 3 *on an asset path evolving according to the Black–Scholes equation.*

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 [5] to trace the trajectory of a particle with initial position $(8, 1, 1)$ over the interval $[0, 40]$ with $2 \cdot 10^5$ time steps. The parameters used are $\rho = 28$, $\sigma = 10$, and $\beta = 8/3$. The double-precision x coordinate is compressed using Algorithm 3. Figure 12 shows that a 35% reduction of the data is achieved.

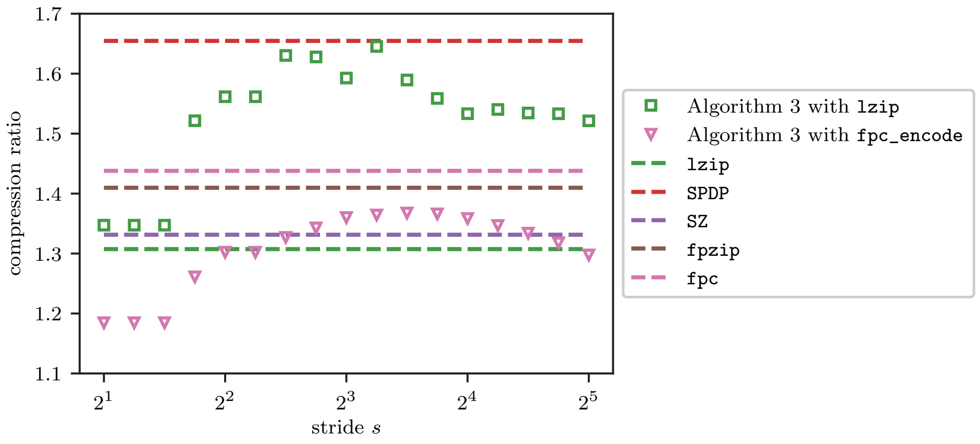

FIG. 12. *Illustration of the performance of Algorithm* 3 *on the x coordinate of a numerical solution of the Lorenz system.*

7.3. Forced isotropic turbulence. Third, we consider data from a large turbulence simulation [29, 33]. 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 time step (time step 2^{10}), traversed in lexicographic order. Figure 13 shows that the performance of Algorithm 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%.

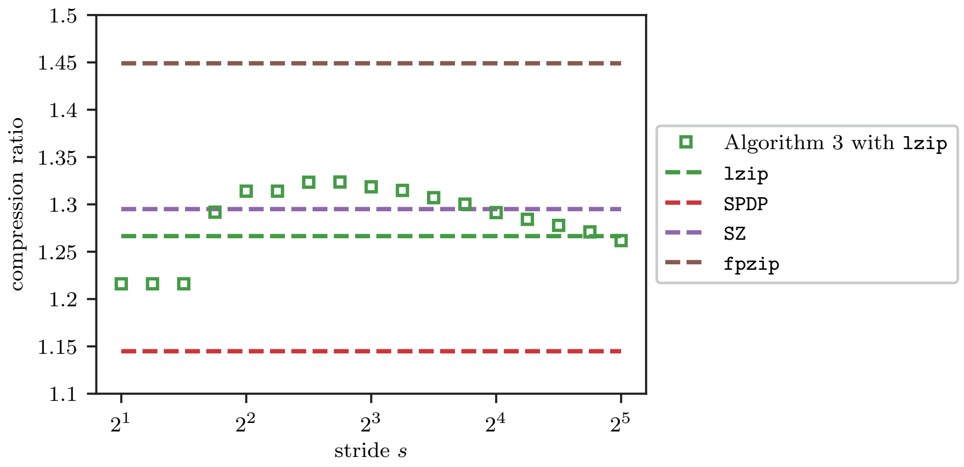

FIG. 13. *Illustration of the performance of Algorithm* 3 *on a single time slab of a* 3*D turbulence simulation.*

7.4. Head impact. Finally, we compare Algorithm 3 with the bitwise modal averaging algorithm described in [2] on a 135 MiB double-precision dataset produced by a head impact simulation [7]. Figure 14 shows that reductions of 8% are consistently achieved over the stride sizes $[2^1, 2^5]$. To facilitate a direct comparison, we

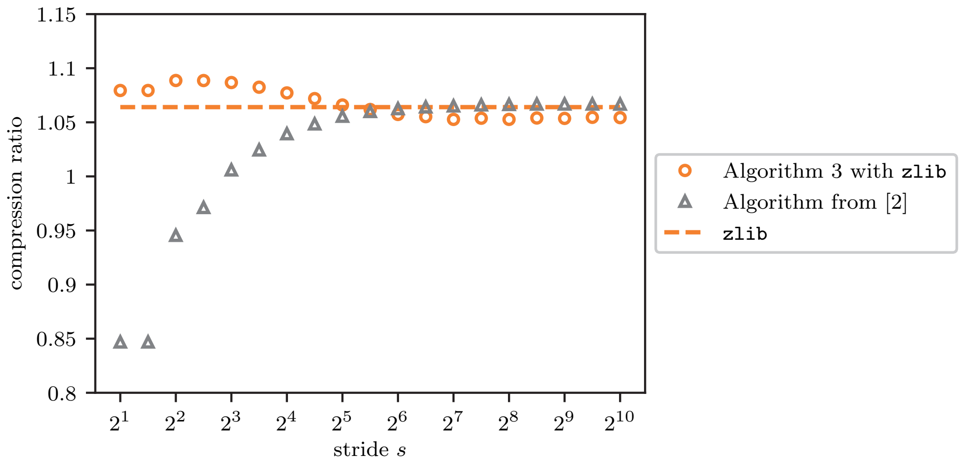

FIG. 14. *Illustration of the performance of Algorithm* 3 *on a head impact simulation. The compressor used by both methods is* `zlib` [19], *which uses the same algorithm as* `gzip`.

modify Algorithm 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 [2] gives a 7% reduction.

Note that the reductions of up to 11% reported in [2] were obtained by pairing the bitwise averaging procedure with a byte-level serialization technique [1].

Appendix A. 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 [20].

LEMMA A.1. *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 A.1 simplifies to the familiar statement that a linear mapping scales volumes by its determinant.

We next establish that Lemma A.1 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 A.2. *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 A.3. ∎

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 A.3. *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* $k \in \{1, \ldots, s-1\}$*, 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 $\operatorname{AbsDet}\colon (\mathbb{R}^{s-1})^{s-1} \to [0,\infty)$ 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 $k=s$, $|\det(M^0_{0,s})| = \operatorname{AbsDet}(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{s-1}) = 1$. If $j=0$ and $1 \le k \le s-1$,

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

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

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

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

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

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

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

These columns can be arranged into an upper triangular matrix with 1's on the diagonal. That matrix has determinant 1, so $|\det(M^1_j)| = 1/s$. The remaining absolute minor has the same value:

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

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

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

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

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

and if $j \ge i$,

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

Thus,

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

We can now use Lemma A.1 to prove the following result.

Theorem A.4.
(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 A.1 and A.3,

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

$$s\operatorname{vol}_{s-1}\left(\Psi^0[0,1]^{s+1}\right) = s + 2\sum_{k=1}^{s-1}k + \sum_{j=1}^{s-2}\sum_{k=j+1}^{s-1}(k-j) = \tfrac{1}{6}s(s+1)(s+2).$$

Next, consider Ψ^1. Combining Lemmas A.1 and A.3, $\operatorname{vol}_{s-1}(\Psi^1[0,1]^s) = \sum_{j=0}^{s-1} 1/s = 1$.

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 A.3 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 [20].

Lemma A.5. *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 A.2 that Ψ^0, Ψ^1, and Ψ^2 satisfy the hypotheses of Lemma A.5, so we may now apply the latter to prove the following result.

THEOREM A.6.
(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 A.3 and A.5,

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

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 $i \ge j$ follows by symmetry. Thus, $\det(\Psi^1\Psi^{1\intercal}) = (-1)^{s-1}\det(\Psi^2)$. In Lemma A.3 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 [20]. Alternatively, we can use Lemma A.5 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, $\det(\Psi^2\Psi^{2\intercal}) = \det(\Psi^2)\det(\Psi^{2\intercal}) = \det(\Psi^2)^2 = 1/s^2$. ◻

REFERENCES

[1] L. A. BAUTISTA GOMEZ, *Re: Reproducing Result from "Improving Floating Point Compression, through Binary Masks,"* manuscript.

[2] L. A. BAUTISTA GOMEZ AND F. CAPPELLO, *Improving floating point compression through binary masks*, in 2013 IEEE International Conference on Big Data, IEEE, Piscataway, NJ, 2013, pp. 326–331, https://doi.org/10.1109/BigData.2013.6691591.

[3] D. BRADBURY, *Hard drive capacity emphasized over performance for future enterprise hard drives*, Computer Weekly, http://www.computerweekly.com/feature/Hard-drive-capacity-emphasized-over-performancefor-future-enterprise-hard-drives. (2010).

[4] J. BURKHARDT, *BLACK_SCHOLES*, http://people.sc.fsu.edu/~jburkardt/cpp_src/black_scholes/black_scholes.html (2016).

[5] J. BURKHARDT, *LORENZ_ODE*, http://people.sc.fsu.edu/~jburkardt/cpp_src/lorenz_ode/lorenz_ode.html (2016).

[6] M. BURROWS AND D. J. WHEELER, *A block-sorting lossless data compression algorithm*, SRC Research report 124, Digital Systems Research Center, Palo Alto, CA, 1994.

[7] M. BURTSCHER, *Scientific IEEE* 754 64-*Bit Double-Precision Floating-Point Datasets*, http://cs.txstate.edu/~burtscher/research/datasets/FPdouble/ (2016).

[8] M. BURTSCHER AND S. CLAGGETT, *SPDP v*1.0, http://cs.txstate.edu/~burtscher/research/SPDPcompressor/ (2016).

[9] M. BURTSCHER AND P. RATANAWORABHAN, *High throughput compression of double-precision floating-point data*, in Proceedings of Data Compression Conference, DCC '07, IEEE Computer Society, Los Alamitos, CA, 2007, pp. 293–302, https://doi.org/10.1109/DCC.2007.44.

[10] M. BURTSCHER AND P. RATANAWORABHAN, *gFPC: A self-tuning compression algorithm*, in Proceedings of Data Compression Conference (DCC), IEEE Computer Society, Los Alamitos, CA, 2010, pp. 396–405, https://doi.org/10.1109/DCC.2010.42.

[11] J. Coalson, *FLAC*, https://xiph.org/flac/format.html (2014).

[12] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley Ser. Telecom., Wiley, New York, 1991.

[13] C. C. Cutler, *Differential Quantization of Communication Signals*, Patent US 2605361 A, 1952, http://www.google.com/patents/US2605361.

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

[15] A. Diaz Diaz, *Lzip*, http://lzip.nongnu.org/lzip.html (2016).

[16] V. Engelson, *Tools for Design, Interactive Simulation, and Visualization of Object-Oriented Models in Scientific Computing*, Ph.D. thesis, Linköping University, Linköping, Sweden, 2000, http://www.scansims.org/~vaden/thesis/.

[17] J. E. Fowler and R. Yagel, *Lossless compression of volume data*, in Proceedings of the 1994 Symposium on Volume Visualization, Washington, DC, 1994, ACM, New York, pp. 43–50, https://doi.org/10.1145/197938.197961.

[18] J.-l. Gailly and M. Adler, *The gzip home page*, http://www.gzip.org/ (2013).

[19] J.-l. Gailly and M. Adler, *zlib*, http://zlib.net/ (2013).

[20] E. Gover and N. Krikorian, *Determinants and the volumes of parallelotopes and zonotopes*, Linear Algebra Appl. 433, 2010, pp. 28–40, https://doi.org/10.1016/j.laa.2010.01.031.

[21] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, 1990.

[22] P. G. Howard and J. S. Vitter, *New methods for lossless image compression using arithmetic coding*, in Proceedings of Data Compression Conference, DCC '91, IEEE Computer Society, Los Alamitos, CA, 1991, pp. 257–266, https://doi.org/10.1109/DCC.1991.213355.

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

[24] 754-2008 *IEEE Standard for Floating-Point Arithmetic*, Institute of Electrical and Electronics Engineers, New York, 2008, http://ieeexplore.ieee.org/servlet/opac?punumber=4610933.

[25] M. Isenburg, P. Lindstrom, and J. Snoeyink, *Lossless compression of predicted floating-point geometry*, Comput.-Aided Des., 37 (2005), pp. 869–877, https://doi.org/10.1016/j.cad.2004.09.015.

[26] M. Isenburg, P. Lindstrom, and J. Snoeyink, *Streaming compression of triangle meshes*, in Proceedings of the Third Eurographics Symposium on Geometry Processing, Vienna, 2005, Eurographics Association, Aire-la-Ville, Switzerland, 2005, pp. 111–118, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1.3872&rep=rep1&type=pdf.

[27] 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, IEEE, Piscataway, NJ, pp. 283–286 vol.1, https://doi.org/10.1109/ISCAS.1993.393713.

[28] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova, *Compressing the Incompressible with ISABELA: In-situ reduction of spatio-temporal data*, in Euro-Par 2011: Parallel Processing Workshops, E. Jeannot, R. Namyst, and J. Roman, eds., Bordeaux, France, Lecture Notes in Comput. Sci. 6852, Springer, Berlin, 2011, pp. 366–379, https://doi.org/10.1007/978-3-642-23400-2_34.

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

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

[31] L. Nowell, *Science at extreme scale: Architectural challenges and opportunities*, DOE Computer Graphics Forum, 2014, http://www.mcs.anl.gov/~hereld/doecgf2014/slides/ScienceAtExtremeScale_DOECGF_Nowell_140424v2.pdf.

[32] I. Pavlov, *LZMA Specification (Draft)*, http://www.7-zip.org/a/lzma-specification.7z (2015).

[33] E. Perlman, R. Burns, Y. Li, and C. Meneveau, *Data exploration of turbulence simulations using a database cluster*, in Proceedings of the 2007 ACM/IEEE conference on Supercomputing, Vol. 23, Reno, NV, ACM, New York, 2007, https://doi.org/10.1145/1362622.1362654, http://dl.acm.org/citation.cfm?id=1362654.

[34] A. Ralston, *A First Course in Numerical Analysis*, 2nd ed., Dover, Mineola, NY, 2001.

[35] P. Ratanaworabhan, J. Ke, and M. Burtscher, *Fast lossless compression of scientific floating-point data*, in Proceedings of Data Compression Conference, DCC 2006, IEEE Computer Society, Los Alamitos, CA, 2006, pp. 133–142, https://doi.org/10.1109/DCC.2006.35.

[36] T. Robinson, *SHORTEN: Simple Lossless and Near-Lossless Waveform Compression*, Technical report CUED/F-INFENG/TR.156, Cambridge University Engineering Department, Cambridge, 1994.

[37] G. Roelofs, *PNG: The Definitive Guide*, 2nd ed., Greg Roelofs, San Jose, CA, 2003, http://www.libpng.org/pub/png/pngbook.html.

[38] E. R. Schendel, Y. Jin, N. Shah, J. Chen, C. S. Chang, S.-H. Ku, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova, *ISOBAR preconditioner for effective and high-throughput lossless data compression*, in Proceedings of IEEE 28th International Conference on Data Engineering, IEEE, Piscataway, NJ, 2012, pp. 138–149, https://doi.org/10.1109/ICDE.2012.114.

[39] J. Seward, *bzip*2, http://www.bzip.org (2010).

[40] N. Shah, E. R. Schendel, S. Lakshminarasimhan, S. V. Pendse, T. Rogers, and N. F. Samatova, *Improving I/O throughput with PRIMACY: Preconditioning ID-Mapper for compressing incompressibility*, in Proceedings of IEEE International Conference on Cluster Computing, 2012, pp. 209–219, https://doi.org/10.1109/CLUSTER.2012.16.

[41] C. E. Shannon, *A mathematical theory of communication*, ACM SIGMOBILE Mobile Comput. Commun. Rev., 5 (2001), pp. 3–55, https://doi.org/10.1145/584091.584093.

[42] G. Taubin and J. Rossignac, *Geometric compression through topological surgery*, ACM Trans. Graph., 17 (1998), pp. 84–115, https://doi.org/10.1145/274363.274365.

[43] P. Thibodeau, *Coming by* 2023*, an exascale supercomputer in the US*, Computerworld, www.computerworld.com/article/2849250 (2014).

[44] C. Touma and C. Gotsman, *Triangle Mesh Compression*, Patent US 6167159 A, 2000, http://www.google.com/patents/US6167159.

[45] M. J. Weinberger, G. Seroussi, and G. Sapiro, *The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS*, IEEE Trans. Image Process., 9 (2000), pp. 1309–1324, https://doi.org/10.1109/83.855427.

[46] J. Ziv and A. Lempel, *A universal algorithm for sequential data compression*, IEEE Trans. Inform. Theory, 23 (1977), pp. 337–343, https://doi.org/10.1109/TIT.1977.1055714.