A Study of the Trade-off Between Reducing Precision and Reducing Resolution for Data Analysis and Visualization

Duong Hoang, Pavol Klacansky, Harsh Bhatia, Peer-Timo Bremer, Peter Lindstrom, Valerio Pascucci

Fig. 1: Visualization of the diffusivity field at 0.2 bits per sample (bps) and its Laplacian field at 1.5 bps, using two of the bit streams studied in the paper. Compared to the by bit plane stream, the by wavelet norm stream produces a better reconstruction of the original function (left, compare white features), and a slightly worse, if not comparable, reconstruction of the Laplacian field (right).

Abstract—There currently exist two dominant strategies to reduce data sizes in analysis and visualization: reducing the precision of the data, e.g., through quantization, or reducing its resolution, e.g., by subsampling. Both have advantages and disadvantages and both face fundamental limits at which the reduced information ceases to be useful. The paper explores the additional gains that could be achieved by combining both strategies. In particular, we present a common framework that allows us to study the trade-off in reducing precision and/or resolution in a principled manner. We represent data reduction schemes as progressive streams of bits and study how various bit orderings such as by resolution, by precision, etc., impact the resulting approximation error across a variety of data sets as well as analysis tasks. Furthermore, we compute streams that are optimized for different tasks to serve as lower bounds on the achievable error. Scientific data management systems can use the results presented in this paper as guidance on how to store and stream data to make efficient use of the limited storage and bandwidth in practice.

Index Terms—data compression, bit ordering, multi-resolution, data analysis

1 INTRODUCTION

As the gap between the available compute power and the cost of data movement increases, data transfer, whether from cache, main memory, or disk, becomes a major bottleneck in many workflows. However, it has been shown that not every bit of data is always necessary to answer scientific questions with required accuracy. In particular, for techniques at the end of scientific workflows, such as visualization and data analysis, lower fidelity representations of the data often provide adequate approximations [24, 38, 70]; even during simulation, some loss in precision is often acceptable [9, 38]. As a result, several different techniques have been proposed to reduce the size of data.

Broadly, these techniques either (i) reduce the data resolution, e.g., the number of data points stored, or (ii) reduce the precision of each data point. Examples of the first kind of approach are subsampling [49], adaptive mesh refinement [5], octrees and other tree structures [55], and wavelets [70], and those of the second are various forms of compression [34, 36, 41, 43, 48, 56, 62]. Traditionally, multiresolution structures have been used to accelerate dynamic queries, e.g., in rendering [37], since discarding data points based on the viewpoint or data complexity can result in significant speed-up. Compression based on quantization, on the other hand, is more common when storing data, where in the absence of other information, treating each sample as equally important is the null hypothesis. However, in many situations, a combination of both resolution and precision reduction may be appropriate. For example, high spatial resolution may be needed to resolve the topology of an isosurface, yet the corresponding data samples may be usable at less than full precision to adequately approximate the geometry. Conversely, accumulating accurate statistics may require high-precision values, yet a lower resolution subset of data points may be sufficient for the task.

In general, different levels of adaptivity in combinations of resolution and precision may be suitable for different types of analysis and visualization tasks, and for many, these requirements might be data dependent. Consequently, a globally optimal data organization may not exist. Instead, we consider a progressive setting in which some initial data is loaded and processed, and new data is requested selectively based on the requirements of the current task as well as the characteristics of the data already loaded. The result is a stream of bits ordered such that the error is minimized, considering the task at hand. However, although intuitively considering both resolution and precision in the ordering certainly has advantages, it is unclear how much the error could be reduced for a given data budget or how little data could be used to achieve the same error. Furthermore, optimal data-dependent orderings may be especially impractical since they assume perfect knowledge of the data. We therefore need to understand which of these potential gains are realizable. This paper aims to address these problems about the suitable bit orderings through extensive, empirical experiments. In particular, this paper presents:

• A framework that allows systematic studies of the resolution-versus-precision trade-off for common data analysis and visualization tasks. The core idea is to represent various data reduction techniques as bit streams that progressively improve data quality in either resolution or precision (Section 3). We can thus compare these techniques fairly, by comparing the
corresponding bit streams.

- empirical evidence that jointly optimizing resolution and precision can provide significant improvements on the results of analysis tasks over adjusting either independently. We present a diverse collection of data sets and data analysis tasks and also show how different types of data analysis might require substantially different data streams for optimal results.

- a greedy approach that gives estimations for lower bounds of error for various analysis tasks. In addition, we also identify practical streams that closely approximate these bounds for each task (Section 4.1, Section 4.2, Section 4.3, and Section 4.4) using a novel concept called stream signature (Section 3.4), which is a small matrix that captures the essence of how a bit stream navigates the precision-versus-resolution space.

In the third contribution, we focus on common analysis tasks, with a notable omission of volume rendering, which is dependent on many parameters, such as transfer function, camera position, and resolution of the image, among others. Thus, for volume rendering, a standalone study is likely required to ensure scientifically sound experiments with generalizable results.

2 RELATED WORK

Tree-based multiresolution hierarchies. A very common scheme to generate a tree-like hierarchy is to construct low-resolution copies of the data from higher resolution ones through downsampling. Examples include Gaussian and Laplacian pyramids [8] and mipmap [37, 51]. The data is often stored in blocks on each resolution level. To save bandwidth, low-resolution versions of distant blocks can be streamed during rendering. However, these methods increase storage requirements, making them unsuitable for very large data. Recent multiresolution techniques save storage by adapting to the data in such a way that different regions are stored with different resolutions. A very popular approach is sparse voxel octrees [16, 25] and variations [6, 22, 31]. Smooth-varying regions are stored at coarser octree levels, which significantly reduces storage. During rendering, blocks of samples are streamed from an appropriate resolution, determined by how far the queried samples are from the eye/camera. A sparse, multiresolution hierarchy can also be built using other trees such as B+ tree [46] and kd-tree [21, 69], or space-filling curves [26, 49], which reorder data samples to form a hierarchy without any filtering steps or redundant samples. Low-resolution levels are constructed via subsampling, which, unfortunately, is prone to aliasing problems.

Transform-based multiresolution hierarchies. Other multiresolution approaches reduce data by transform-based compression. For example, COVRA [24] constructs an octree of bricks (consisting of blocks) and learns a sparse representation for the blocks in terms of basis blocks. Similarly, Fout et al. [23] transform each block using the KLT transform to produce several resolution levels, and compress each level using vector quantization [48]. Schneider et al. [56] also use vector quantization but with a simple Haar-like transform that separates each block of 23 voxels into one average and seven difference coefficients. Other examples of transform-based hierarchies include works that use the Tucker decomposition [2, 3, 59, 60], which decomposes the input data (stored as a tensor) into n matrices of basis vectors and one core tensor. Tensor decomposition works for higher dimensional data and can achieve high compression ratios, albeit at the price of a costly transform step.

Wavelets. Transforms that use fixed bases avoid such high computation cost at the expense of slightly less effective compression. Perhaps the most popular transform that uses a fixed basis is the (discrete) wavelet transform (DWT), which constructs a hierarchy of resolution levels via low and high bandpass filters. The transform is recursively applied to the lower resolution band, resulting in a hierarchy of “details” at varying resolution. The DWT is merely a change of basis that does not increase the data size. Furthermore, the wavelet basis functions are defined everywhere in space, requiring no special interpolation rules when given some arbitrary subset of wavelet coefficients. One disadvantage of the DWT is the random access cost, which is not constant time, although there has been work to develop acceleration structures to speed up local queries [67].

Besides offering a multiresolution decomposition, enabling data streaming with level-of-detail support, wavelet coefficients are especially amenable for compression, through thresholding or entropy compression. In storing and visualizing scientific data, wavelets (with compression) are used in a wide variety of systems [12, 13, 70] and applications, such as volume rendering with level-of-detail [29, 30, 33, 47, 68], turbulence visualization [65], and particle visualization [52].

Most wavelet-based techniques employ tiling of wavelet coefficients in individual subbands to facilitate random access and spatial adaptivity in resolution. For example, the VAPOR toolkit [13] incorporates a multiresolution file format based on wavelets to allow data analysis on commodity hardware by storing individual tiles in separate files to allow loading of the region of interest. However, like most multiresolution work, only the resolution control is leveraged. The precision axis, which can potentially further reduce data transfer, is left unexplored.

Wavelet coders. Most work that explores the precision axis can be found in wavelet coders. Wavelet coefficients in corresponding regions across subbands can be thought of as belonging to a “tree”. The embedded zero trees (EZW) coder exploits the observation that in such trees, “parent” coefficients are often larger in magnitude than their “children”. EZW therefore locates trees of wavelet coefficients that are insignificant with regard to a series of thresholds and encodes such a tree with one single symbol. The thresholds are typically set at the bit planes, starting from the most significant one. In this way, the data can be progressively refined in precision during decompression. The SPIHT coder [54] improves on EZW by locating more general types of zero trees [10]. SPECK [50] extends SPIHT to also exploit spatial correlations among nearby coefficients on the same subband.

Floating-point compression. The ZFP compression scheme [41] also encodes transform coefficients by bit plane, in order of decreasing significance. By partitioning the domain into 4 × 4 × 4 independent blocks, ZFP supports fixed-rate compression, random access to the data, localized decompression, and fast inline compression. Extensions of ZFP allow it to vary either the bit rate or precision spatially over the domain, albeit at fixed resolution [42]. Other notable compression schemes for scientific data include scalar quantization encoding (SQE) [34], ISABELA [36], SZ [62], and FFZIP [43], which generally employ prediction and compress the residuals. ISABELA and SZ perform residual scalar quantization, whereas FFZIP executes floats, which can be seen as nonuniform scalar quantization. Similarly, the precision-based level of details (PLOD) scheme proposed in MLOC [26] also truncates floats by dividing a double-precision number into several parts. MLOC includes a multiresolution scheme based on Hilbert curves, but this scheme (based on resolution) and the PLOD scheme (based on precision) are exclusive.

Mixing resolution and precision. Schemes that allow progressive data access in both resolution and precision include SBHP [11] and JPEG2000 [64]. Both partition each subband into blocks and code each block independently, in bit plane order. By interleaving compressed bits across blocks, one can construct a purely resolution- progressive or a purely precision- progressive stream, or anything in between. JPEG2000 has found use in the compression of scientific data, e.g., by Woodring et al. [70]. Since most JPEG2000 implementations are limited to integer data, the authors apply uniform scalar quantization to convert floating point data to integer form. Even though JPEG2000 supports varying both resolution and precision, most works do not explore this capability but focus only on setting bit rate. In general, efficiently leveraging both axes of data reduction has not been well studied.

Error quantification. Several works have aimed at quantifying the error incurred by data reduction, by compression or otherwise, for the results of analysis tasks. Baker et al. [1] evaluate several compressors, including FPZIP [43] and ISABELA [36] on ensembles of climate simulation data. Laney et al. [38] study the effects of lossy compression
on the Miranda hydrodynamic simulation code. Li et al. [39] measure the error incurred by wavelet compression on turbulent-flow data. Wang et al. [66] assess the quality of distorted data using a combination of statistical metrics in the wavelet transform domain. Etienne et al. have published a series of studies on verification of isosurfaces in geometrical [20] and topological [19] terms, and verification of volume-rendered images [18]. They focus on order-of-accuracy and convergence analysis, where errors are introduced mostly through discretization, not necessarily for the purpose of reducing data bandwidth. For volume rendering, Ljung et al. [44] compute an error estimate based on color variation between low- and high-resolution levels in each block, which guides the selection of an appropriate resolution level for every block. Finally, the Z-checker framework [63] consists of a wide range of independent metrics to evaluate data quality after lossy compression. In all these works, errors are not quantified in the context of the tradeoff between resolution and precision.

Finally, for surveys of data reduction techniques in general, we refer the readers to the work of Rodríguez et al. [4] and Li et al. [40].

3 Data Reduction Schemes as Packet Streams

In order to systematically study the resolution-versus-precision trade-offs among different data reduction schemes, it is important to perform fair and consistent comparisons. In this section, we develop such a consistent methodology by proposing to model different data reduction schemes as streams of uniformly sized data packets, where the original data contains all the packets, and any reduction step removes a set of packets (comparable amounts of data). These data streams are transmitted using a client-server model. At any point, the client is assumed to have received a subset of packets (in some predetermined sequential order or in some order requested by the client), which can be used to reconstruct an approximation to the original data. Therefore, to compare different streams, we reconstruct the original data using the same number of packets from each stream and perform desired tasks on each of the (approximate) reconstructions. A stream is considered better suited for a given task if it produces results that are closer to the reference results computed from the original data. Fig. 2 gives a schematic view of our data streaming model.

Although both the server and the client in our model can be on the same physical machine, only the server has full knowledge of the data. Thus, when the client receives a packet, it might not know where that packet should be deposited. A common solution is to have both the client and the server agree beforehand on a static ordering of packets, independent of the data. We use the term data-independent streams to refer to streams using such solutions. In contrast, for data-dependent streams, an additional mechanism is needed to inform the client about the subbands and bit planes of incoming packets. In this paper, we consider both types of streams, as well as specialized task-dependent streams optimized for given tasks (see Table 1).

![Fig. 2: Our data streaming model. The input is a regular grid of floating-point samples; the output is a stream of packets. A packet consists of bits from the same bit plane, from a block of negabinary wavelet coefficients. Different data reduction schemes generate different streams. The wavelet subbands are separated by blue lines in the second image, with the coarsest subband at the top left corner. Quantization and negabinary conversion happen immediately after wavelet transform.](image)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Data Dependent</th>
<th>Task Dependent</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{vl}$</td>
<td>by level</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$D_{bit}$</td>
<td>by bit plane</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$D_{wavelet}$</td>
<td>by wavelet norm</td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$S_{mag}$</td>
<td>by magnitude</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>$S_{[task] \cdot opt}$</td>
<td>task-optimized</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>$S_{[task] \cdot sig}$</td>
<td>by signature</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
</tr>
</tbody>
</table>

Table 1: We define various types of data streams, including data-independent, data-dependent, task-independent, and task-dependent streams. $S_{[task] \cdot sig}$ can be data dependent or task dependent, depending on the stream from which it is derived.

3.1 Decomposition of Data into Packets

Although one way to compare different data reduction strategies is to restrict the techniques to the same data size and compare data quality, it is difficult to enforce consistency. For example, the amount of change (in data) in one step of multiresolution simplification may be different from removing one bit in the quantization of every sample. To make all data reduction schemes comparable, in each scheme, a data set is redefined as a stream of equally sized packets. These packets are the smallest units of data transfer in our framework. A packet consists of a relatively small number ($= 2^3$) of bits and is associated with a resolution level and a precision level (i.e., bit plane). In this framework, different data-reduction schemes become different orderings of packets, called streams. Restricting two (or more) data streams to the same number of packets allows us to perform fair and consistent comparisons.

Resolution levels. Although there exist several ways to define the notion of resolution/scale/frequency, we choose the multilevel basis functions of the wavelet transform because they have compact support, and they avoid interpolation problems associated with other representations. Wavelet transform enables spatial adaptivity (i.e., finer resolution in regions that contain sharp features, at the expense of coarser resolution elsewhere). In particular, we choose the CDF5/3 multilinear wavelets [14] for their balance between simplicity and effectiveness at decorrelating the input signal in practice [64].

A multidimensional wavelet transform can be performed in multiple passes, which partitions the original domain into subbands, each of which can be thought of as a resolution level associated with one or more spatial direction. One transform pass (in 3D) creates eight subbands, of which the first is a low-pass, downsampled version of the original data, and the remaining add fine details in each subset of the dimensions (see Fig. 2 for a visualization of subbands in 2D). A subsequent transform pass recovers only on the first subband (of the previous level), creating the next (resolution) level of subbands. We use $l (0 \leq l < L)$ to index the subbands, with $l = 0$ referring to the coarsest subband and $L$ denoting the number of subbands. In 3D, the eight subbands created after one transform pass are indexed in the following order, from coarse to fine: LLL, LLH, LHL, LHH, HLL, HHL, HHL, HHH (L stands for “low” and H stands for “high”, referring to the low- and high-pass filter pair that perform the wavelet transform). The LHL subband, for example, contains coefficients that are low-pass transformed along $X$ and $Z$, and high-pass transformed along $Y$.

In our experiments, the number of subbands, $L$, is fixed at $1 + 3 + 7 = 22$, corresponding to three transform passes in 3D.

Precision levels. For creating packets corresponding to different precision levels, we quantize floating-point wavelet coefficients to $B$-bit signed integers. For most of the experiments in this paper, $B = 16$. This quantization eliminates the floating-point exponent bits, such that every bit (except the sign bit) can be associated with a bit plane $b (0 \leq b < B)$. In our convention, the higher indexed bit planes are less significant. We convert quantized coefficients to the negabinary representation, where integers are represented in base $-2$, i.e., $\sum_{b=0}^{B} c_{b}(-2)^{b}$ with $c_{b} \in \{0, 1\}$. Negabinary encoding is preferred over two’s complement encoding, because we start data reconstruction by zero-initializing all bits, and negabinary encoding has no single dedicated sign bit and ensures that small coefficients have many leading zero-bits. This transformation increases the number of bit planes by one, i.e., $0 \leq b \leq B$. 
Blocks and packets. Precisely, a packet consists of bits from the same bit plane, from a block of negabinary wavelet coefficients. A block is a \([g \times g \times g]\) grid of adjacent coefficients from the same subband. We let \(g\) be a constant \((g = 2\) in this paper\), so that finer resolution subbands contain more packets, which presents a trade-off between packets that provide wider (but coarser) coverage and packets that provide finer (but more local) details. Every packet of \((\text{size one byte in this paper})\) comes from a bit plane \(b\) and a subband \(I\). \(g\) is chosen to be larger than one for performance reasons, as in practice, most systems read bits in batches.

3.2 Data-Dependent and Data-Independent Streams

We define two streams: by level and by bit plane, which model two common reduction schemes in the literature. The by level stream, \(S_{byl}\), orders the packets strictly from coarsest to finest subbands. Within the same subband, packets follow the row-major order of blocks and then bit plane order (from 0 to \(B\)) within each coefficient. All bits for each coefficient are streamed together. The other common ordering, by bit plane, or \(S_{bit}\), proceeds strictly from higher ordered to lower ordered bit planes. Within the same bit plane, packets follow the subband order (from 0 to \(L - 1\) and then row-major order in each subband. \(S_{byl}\) and \(S_{bit}\) are designed to mimic the way data is accessed in traditional methods that work either in resolution \((S_{byl})\) or in precision \((S_{bit})\).

Additionally, we define a third stream that combines these two dimensions and refer to it as by wavelet norm, or \(S_{wav}\). This stream orders packets in descending order of weights \(w_{wav}(p) = 2^{B - b(p)} \cdot ||\psi(l, p)||\), where \(p\) denotes a packet and \(\psi(l, p)\) represents wavelet basis on subband \(I\). The \(||\cdot||\) notation refers to the \(L_2\) norm of the wavelet basis function. The first term captures the contribution of a bit on bit plane \(b(p)\), and the second term captures the contribution of a wavelet coefficient on subband \(l(p)\). In the wavelet representation, a function \(f\) is written as a linear sum of wavelet basis functions, i.e., \(f = \sum c_i \psi_i\), where \(c_i\) are the coefficients. Since our wavelet transforms are based on lifting, this norm is usually not one, but it increases with level. Basis functions in the same subband share the same norm, hence \(w_{wav}(p)\) is simply the contribution (in \(L_2\) norm) of a bit on bit plane \(b(p)\) and subband \(l(p)\), to the whole function \(f\). This ordering based on norms of wavelet basis functions was proposed previously by Weiss et al. [67]. For details of computing the norms of basis functions, see Appendix A.

Another common way to reduce data in the wavelet domain is to leave out the coefficients of the smallest magnitudes. Note that coefficient magnitudes are only weakly related to error, as the error also depends on the wavelet basis function norm [67]. We model this scheme with a stream called by magnitude, or \(S_{mag}\). Here, the weight function is \(w_{mag}(p) = \sum \epsilon_{block(p)} ||\psi||\) (the sum is over all coefficients in the block that contains packet \(p\)). If two packets have the same weight, they are ordered by subband index, then by bit plane index.

Unlike \(S_{byl}\), \(S_{bit}\), and \(S_{wav}\), the \(S_{mag}\) stream is data dependent because the coefficient magnitudes are not known without the data. In principle, data-dependent streams are better than data-independent streams because they can prioritize important packets based on the actual data. However, data-dependent streams are ill-suited for practical purposes, because the cost of sending position information likely outweighs any potential benefit. Nevertheless, we study them for various reasons. First, the by magnitude scheme is well known in the literature [13]. Second, the “best” data-dependent streams (which do not include position information for packets) can serve as a baseline to evaluate the performance of their data-independent counterparts. Finally, in addition to being data dependent, streams can also be task dependent (Section 3.3), which may provide insights into how data should be queried to perform certain analysis tasks.

3.3 Task-Optimized Streams

Each analysis task may require a fundamentally different stream for optimal results. Studying such “optimal” streams is important because they not only serve as a baseline but also can provide insights into other, more practical streams. Given the original data set \(f\) and its reconstructed approximation \(f'\) using a subset of packets, let \(q\) represent some quantity of interest, e.g., histogram, isosurface, etc., computed on \(f\) or \(f'\). For a given \(q\), a well-defined error metric \(e(q(f'), q(f))\), which returns a single scalar, is needed. Given \(f\), \(q\), and \(e\), our goal is to generate an optimal (and data-dependent) stream, \(S_{opt}\), for \(q\) with respect to \(e\). One possible definition for \(S_{opt}\) is a stream such that the area under the plotted curve of \(e\), with respect to the number of bits, is minimized for all packets to be streamed. However, this definition is limited in practice because a stream should be able to terminate at any point and still produce as small an error as possible.

Instead, we employ a greedy approach to define the optimal stream. We notice through experiments, however, that a straightforward greedy algorithm can pick unimportant packets too early. For example, starting with an all-zero reconstruction \(f' = 0\) and an empty stream, we can repeatedly append new packets to the stream, which when included in the current \(f'\), would minimize \(e\) at every step. At some point, the algorithm might pick a packet that introduces the lowest error yet contributes very little to improve the quality of \(f'\) (because more important packets would increase the error), leading to a nonoptimal stream. To avoid this problem, we make a modification to this greedy algorithm and build the stream backwards. We start with a “lossless” \(f'\) (i.e., \(f' = f\)), and at each step we remove the packet that has the least impact on the error \(e\) from \(f'\). This modification largely avoids the previous problem where less important packets were added to \(S_{opt}\) too early, because by starting with the full (instead of empty) set of packets, our error measurement better captures the importance of packets.

Unfortunately, such a greedy algorithm is still expensive in practice, as its complexity is at least \(O(n^2)\) (\(n\) is the number of packets), due to the 2-level nested loop. For a \(nx ny nL\) volume, a block size of \(bx by bz\), and \(B + 1\) bit planes, \(n\) is \(\frac{nx}{bx} \times \frac{ny}{by} \times \frac{nL}{bz} \times (B + 1)\). Thus, even a small volume, e.g., \(nx ny nL = 64\) and \(bx by bz = 2\), can result in a prohibitively high run time, as \(n^2 = (32768 \times 17)^2\). Therefore, we adopt a simplified version of this algorithm, where only one pass through \(n\) packets is needed. In iteration \(i (0 \leq i < n)\), we set a new packet \(p_i\) to zero, compute and record the incurred error \(e_i\) using the error metric \(e\), and then enable \(p_i\) again at the end of iteration \(i\). After \(n\) iterations, each packet has an associated weight \(w_i\). The stream \(S_{opt}\) is simply the sorted list of packets in decreasing order of the weights. This simplified algorithm (Algorithm 1) has significantly lower running time, while (by observation) retaining the same quality for \(S_{opt}\).

**Algorithm 1 Computing a task-optimized stream**

1: **Inputs:**
   - An original function \(f\)
   - An unordered set of \(n\) packets \(P = \{p_i\}\), produced from \(f\)
   - A quantity of interest \(q\), and an error function \(e\)

2: **Initialize:**
   - A set of \(n\) weights \(\{w_i\}\)
   - For each packet \(p_i\) \(do\)
     4: \(p_i := 0\)
     5: \(P \rightarrow \text{wavelet coefficients} = \{c_j\}\)
     6: \(\text{inverse quantization and inverse negabinary transform)}\)
     7: \(\{c_j\} \rightarrow f'\) (inverse wavelet transform)
     8: \(w_i := e(q(f'), q(f))\)
     9: Restore \(p_i\)
10: **end for**
11: Sort the \(p_i\)'s in descending order of \(w_i\)
12: **Output:**
The \(q\)-optimized implementation, the inverse wavelet transform on line 7 can be replaced by “splattling” coefficient \(p_i\) onto the domain, due to the transform being linear and the fact that \(p_i\) is the only coefficient changed in the current iteration. Since it is almost never advantageous to stream bits belonging to one coefficient out-of-order, we also enforce that in the final stream, packets belonging to the same group follow the bit plane order (from 0 to \(B\)).

3.4 Stream Signatures

Unlike data-independent streams, data-dependent streams do not impose a static ordering of packets. To concisely represent and characterize the dynamic ordering of data-dependent
We define two streams: bit planes. Within the same bit plane, packets follow the subband because the coefficient magnitudes are not known without the data. Reconstructed approximation they not only serve as a baseline but also can provide insights into finally, in addition to being data dependent, streams can also be task to evaluate the performance of their data-independent counterparts. Another common way to reduce data in the wavelet domain is Additionally, we define a third stream that combines these two streams, we introduce the notion of a stream signature. Any stream can be represented with respect to the two-dimensional space of resolution (subbands) and precision (bit planes), i.e., \( L_{L,B} = \{(l,b) | 0 \leq l < L, 0 \leq b < B\} \). Given a stream, we define its signature \( A \) as an \( L \times (1+B) \) matrix, where each \((l,b)\) element is associated with \( p_{lb} \), the set of packets belonging to subband \( l \) and bit plane \( b \). In particular, \( A(l,b) \), i.e., the \((l,b)\) element of \( A \), is an integer in the range \([0,(1+B) \times L]\), and indicates, on average, the position at which packets in \( p_{lb} \) appear in the given stream. For example, the signature \( A = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \) indicates that the stream begins with packets that lie on the first bit plane of the first subband, as \( A(0,0) = 0 \). Those are followed by packets on the second bit plane of the first subband \( A(0,1) = 1 \), and then the first bit plane of the second subband \( A(1,0) = 2 \), and finally, the third bit plane of the second subband \( A(1,2) = 5 \). Thus, a stream’s signature shows how the stream traverses the space \( L_{L,B} \) and highlights the different resolution-versus-precision trade-offs among streams, especially among \( S_{opt} \) streams optimized for different tasks. In Fig. 3 we visualize the signatures of \( S_{bit}, S_{hid} \), and \( S_{wav} \), defined in Section 3.1.

To compute a stream signature, we partition the whole domain (not individual subbands) into several regions, compute one signature per region, and average these local signatures. Partitioning is used since it is only when packets are relatively well localized that their relative ordering in the \( L_{L,B} \) space becomes meaningful. For example, a packet at one corner of the domain may be streamed before one at an opposite corner, but this fact contains no useful information. We define a region to be the spatial volume that is covered by a packet in the coarsest subband. Algorithm 2 lists the steps of our approach.

Finally, a signature can be used to construct a stream denoted generically as \( S_{sig} \). This construction is done by iterating through each element \((l,b)\) in ascending order and adding to the end of \( S_{sig} \) all the packets in \( p_{lb} \). An \( S_{sig} \) captures the behavior (in the \( L_{L,B} \) space) of the stream it derives from, but it is stripped from any spatial adaptivity. Hence, when \( S_{sig} \) is derived from an \( S_{opt} \), it can serve as a bridge when comparing the resolution-versus-precision trade-offs between data-independent and data-dependent streams.

Algorithm 2 Computing a stream signature

1: Inputs:
   A stream \( P = \{p_i\} \)
2: Initialize:
   Per-region signature matrix \( A_r := 0 \)
   Global signature matrix \( A := 0 \)
3: for each packet \( p_i \) in \( P \) do
4:   Let \( l, b, i \) be the region, bit plane, and subband that \( p_i \) belongs
5:   \( A_r(l,b) := A_r(l,b) + i \)
6: end for
7: for each region \( r \) do
8:   Sort the elements of \( A_r \)
9:   Assign each element of \( A_r \) its index after sorting
10: \( A := A + A_r \)
11: end for
12: Sort the elements of \( A \)
13: Assign each element of \( A \) its index after sorting
14: Output:
   The signature matrix \( A \)

### Table 2: Data sets used in our experiments; all volumes are \( 64^3 \).

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>boiler [58]</td>
<td>combustion simulation</td>
<td>float64</td>
</tr>
<tr>
<td>plasma [28]</td>
<td>magnetic reconnection simulation</td>
<td>float32</td>
</tr>
<tr>
<td>diffusivity [15]</td>
<td>hydrodynamics simulation</td>
<td>float64</td>
</tr>
<tr>
<td>pressure [15]</td>
<td>hydrodynamics simulation</td>
<td>float64</td>
</tr>
<tr>
<td>turbulence [17]</td>
<td>fluid dynamics simulation</td>
<td>float32</td>
</tr>
<tr>
<td>kingsnake [27]</td>
<td>CT scan</td>
<td>uint8</td>
</tr>
<tr>
<td>foam [45]</td>
<td>CT scan</td>
<td>uint16</td>
</tr>
</tbody>
</table>

### 4 Evaluation on Different Analysis Tasks

Thus far, we have presented several types of streams: data-independent \( (S_{hid}, S_{bit}, S_{wav}) \), data-dependent and task-independent \( (S_{mag}) \), and task-dependent \( (S_{opt}, S_{sig}) \). In this section, we consider a variety of common analysis and visualization tasks to evaluate the performance of these streams. For each task, we define an error metric, \( e \), for the evaluation and comparison of streams. Using Algorithm 1, we compute streams specifically optimized for each task, \( S_{task-opt} \), and use its signature to compute the corresponding \( S_{task-sig} \). For a variety of data sets, we compare these streams by evaluating the error as a function of bits per samples (or \( bps \)) defined as the total number of bits received divided by the total number of samples. To mimic the effects of entropy compression commonly used in practice, we remove from each stream all packets that consist only of leading-zero bits. The wavelet basis allows us to always reconstruct data at full resolution, which greatly simplifies computation of errors, as there exists no standard method to compute error over grids of different dimensions.

#### 4.1 Function Reconstruction

One of the most fundamental analysis tasks is that of reconstructing the original function itself. A commonly used error metric in this case is the root-mean-square error (RMSE). Fig. 4 shows a comparison of the different streams for a variety of data sets. It can be noted that, in general, \( S_{rms-se-opt} \) (the stream optimized to minimize the RMSE) performs better than \( S_{rms-se-sig} \) due to spatial adaptivity, whereas \( S_{rms-se-sig} \) slightly outperforms \( S_{wav} \), followed by \( S_{hid}, S_{mag} \), and \( S_{hid} \). In particular, \( S_{hid} \) outperforms \( S_{hid} \) (for \( kingsnake \) and \( boiler \), it does so after approximately 1 bps), which can be attributed to the removal of leading-zero packets. Empirically, wavelet coefficients on finer scale subbands are much smaller in magnitude [54]. Such coefficients contain a majority of the leading-zero bits, whose removal benefits \( S_{hid} \) the most. Diffusivity and plasma contain a significant amount of empty space, which translates to more leading-zero bits after the wavelet transform that \( S_{hid} \) can take advantage of, and thus, it outperforms \( S_{hid} \) immediately from the beginning. \( S_{mag} \) underperforms for the same reason that \( S_{hid} \) does, but to a lesser extent, since \( S_{mag} \) is adapted to the data. \( S_{wav} \) outperforms both \( S_{hid} \) and \( S_{hid} \) because it follows the optimal (data-independent) bit ordering in \( L_{L,B} \) in the \( L_2 \) norm, which is also the norm that the RMSE is based upon. Unsurprisingly, \( S_{rms-se-opt} \) outperforms all others, as it is the most data-adaptive (i.e., it can optimize packet ordering in the spatial domain in addition to the \( L_{L,B} \) domain). \( S_{rms-se} \) is the second best stream, as it follows the bit ordering of \( S_{rms-se-opt} \) in \( L_{L,B} \) but lacks any spatial adaptivity. In general, \( S_{wav} \) and \( S_{hid} \) have similar performances, but \( S_{hid} \) performs better when the data is less smooth or noisy, as is the case for \( boiler \) and \( kingsnake \). For such data, fine-level packets tend to have very few leading one bits among a majority of leading zero bits, which \( S_{wav} \) does not take into account (data-independent streams in effect assume every packet contains all one bits).

We explore the errors visually by rendering the plasma volume at 0.2 bps, for all streams except \( S_{rms-se-opt} \) (Fig. 5). Although \( S_{hid} \) has the precision to obtain an accurate background, it lacks resolution to resolve the fine details. \( S_{hid} \), instead, lacks the precision to reconstruct the (mostly smooth) background, but it has enough resolution to capture the fine details well. \( S_{wav} \) balances both precision and resolution, producing a more accurate picture as a whole. In this case, the \( S_{hid} \) stream produces the most accurate rendering overall.
we use 32 bits for quantization to ensure enough precision for finite 
stream, \( \nabla \) and its low-bit-rate approximation 
\( \partial f \) on grids of different resolutions. 

4.2 Derivative Computation

Computation of derivative-based quantities is important in data analysis. 
Examples include vorticity (curl) computation from velocity fields to 
identify vortical structures, gradient computation for accurate Morse 
segmentation and shading, and ridge extraction (e.g., for Lagrangian 
coherent structures). In this paper, derivatives are always computed 
using finite differences, which is common in practice. In this section, 
we use 32 bits for quantization to ensure enough precision for finite 
differences. We always compute finite differences on the finest 
resolution grid to avoid computing distances between quantities defined 
on grids of different resolutions.

4.2.1 Gradient Computation

Given a function \( f \) defined on a grid, its gradient at a grid point 
\( x = (x, y, z) \) is \( \nabla f(x) = \left( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right) \). For accuracy, we use a five-point 
stencil to compute the gradient, i.e., \( \frac{\partial f}{\partial x} \approx \frac{1}{12} f(x + 1, y, z) - \frac{1}{2} f(x, y, z) + \frac{1}{12} f(x - 1, y, z) + \frac{1}{4} f(x + 1, y, z) - \frac{1}{4} f(x - 1, y, z) \), but we note that the relative 
performances of the streams stay the same, using the more common two- 
and three-point formulas. The error between a gradient field \( \nabla f \), 
and its low-bit-rate approximation \( \nabla f' \), is defined as \( e(\nabla f', \nabla f) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \| \nabla f'(x_i) - \nabla f(x_i) \|^2} \). Using Algorithm 1, we compute a 
gradient-optimized stream, \( \nabla_{\text{grad-opt}} \), that minimizes the difference 
between the reconstructed and the original gradient fields.

Fig. 7 shows the gradient error incurred by different streams for four 
data sets. In general, we observe the ordering of performance (from best 
to worst) as: \( \nabla_{\text{grad-opt}} \), \( \nabla_{\text{grad-sig}} \), \( \nabla_{\text{bit}} \), \( \nabla_{\text{mag}} \), \( \nabla_{\text{ lvl}} \). This ordering 
can also be seen in Fig. 8, where the \( \times \)-component of the gradient

4.2.2 Laplacian Computation

The Laplace operator is a second-order differential operator defined 
as the divergence of the gradient field. The Laplacian of a 3D 
field is defined as \( \Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} \). Using a five-point 
finite difference, we approximate \( \frac{\partial^2 f}{\partial x^2} \approx \frac{1}{12} f(x + 1, y, z) + \frac{1}{4} f(x + 1, y, z) + \frac{1}{12} f(x - 1, y, z) - \frac{1}{2} f(x, y, z) + \frac{1}{4} f(x + 1, y, z) - \frac{1}{4} f(x - 1, y, z) \). We use 
the root-mean-square error to compare two Laplacian fields, i.e., 
\( e(\Delta f', \Delta f) = \text{RMSE}(\Delta f', \Delta f) \). We use Algorithm 1 to compute a 
Laplacian-optimized stream, \( \Delta_{\text{lap-opt}} \), which minimizes \( e \), and an \( \Delta_{\text{lap-sig}} \) 
stream from its signature. Fig. 9 plots the errors for all relevant 
streams. The plots here largely follow the ones in Fig. 7, in terms 
of relative performance among the streams, but with more discernible 
gaps between \( \Delta_{\text{bit}} \) and \( \Delta_{\text{lap-sig}} \), as well as between \( \Delta_{\text{lap-sig}} \) and \( \Delta_{\text{lvl}} \). The 
results suggest that similar to the gradient case, computation of the 
Laplacian favors resolution over precision, but to a higher degree.

4.3 Histogram Computation

A histogram succinctly summarizes the distribution of sample values, 
and thus it is useful as a cursory “look” into the data and in guiding 
overall further analysis. For example, it can be used to guide the selection 
of colors and opacities in a transfer function. To decide on an error
metric to compare histograms, we experimented with several popular metrics such as Kolmogorov-Smirnov [57], Kullback-Leibler [35], and Earth Mover’s Distance [53], among others [7,32]. We chose histogram intersection [61] as the metric of choice, because it is fast to compute and is reasonably insensitive to changes in precision, as well as the number of bins. The intersection distance between two histograms $H_1$ and $H_2$ is defined as $d(H_1, H_2) = 1 - \sum \min(H_1(i), H_2(i))$ (the sum is over all bins $i$). Every histogram is normalized by dividing the value in each bin by the total number of samples. We decided that the error metric should take into account both the histogram shapes and the range of values, and we clamped the range of values in reconstructed functions to that of the original function, so that corresponding histogram bins, i.e., $H_1(i)$ and $H_2(i)$, share the same range.

As before, for each data set, we use Algorithm 1 to compute an $S_{\text{hist-opt}}$ stream, optimized for histogram error, and then construct an $S_{\text{hist-sig}}$ from its signature. We plot the error curves for all relevant streams using the Intersection error metric (compare Fig. 10). We use 64 for the number of bins but note that there exist no meaningful differences across a wide range of number of bins (from 64 to 512) in our experiments. In all cases, the group consisting of $S_{\text{bit}}, S_{\text{wav}},$ and $S_{\text{mag}}$ underperforms the other group by a large margin. $S_{\text{mag}}$ performs poorly, because it ignores regions of smooth variations, which nevertheless count toward the distribution. $S_{\text{bit}}$ generally outperforms $S_{\text{bit}}$ at low bit rates, although there are several crossover points between the two curves. As can be seen in Fig. 12, $S_{\text{wav}}$ outperforms $S_{\text{bit}}$ when leading zero packets are present, because increasing resolution does not help as much as increasing precision. This is because the histogram is oblivious to spatial locations of samples (which require resolution to resolve) but is sensitive to sample values (which require precision). However, when leading zero packets are removed, $S_{\text{mag}}$ benefits significantly more than $S_{\text{bit}}$ (for the same reason explained in Section 4.1), resulting in the observed crossovers.

In the latter group, the performances of $S_{\text{wav}}$ and $S_{\text{hist-sig}}$ and (even $S_{\text{hist-opt}}$) differ by a negligible amount. This observation is confirmed in Fig. 11, where we plot various histograms, reconstructed at 0.3 bps, for the boiler data set. The histograms produced by $S_{\text{wav}}$ and $S_{\text{hist-sig}}$ have approximately the same shape and are the closest to the reference histogram. The next best histogram is produced by $S_{\text{bit}}$, followed by the one produced by $S_{\text{wav}}$, and then $S_{\text{mag}}$. These results suggest that histogram computation benefits from a bit ordering that combines both resolution and precision, with a strong bias toward precision.

### 4.4 Isosurface Extraction

Studying isosurfaces of a given function is an essential task in many visualization and analysis pipelines, as they can highlight features of interest. For measuring error between isosurfaces, we have found that the commonly used Hausdorff distance does not work well in our case, because two very different reconstructed surfaces may share the same Hausdorff distance to the reference. More sophisticated metrics exist, focusing on different characteristics such as geometric [20] and topological [19] properties, but they assume the surface has certain properties. Since isosurfaces partition the domain into “inside” and “outside” regions, we opt for a simpler error metric that assumes nothing about the shape of the isosurfaces, but simply counts misclassified voxels. This metric differs from comparing histograms with two bins in that we care about the spatial position and not just voxel counts.
We focus on the trade-off between two prominent dimensions of S:

- terms of error, S_{hist-opt} \approx S_{hist-sig} \approx S_{wav} < S_{lvl}, S_{mag}.

The erratic behavior at the beginning for kingsnake is likely due to the data being too noisy. The especially poor performances of S_{mag} for boiler and foam are due to the “shifting” effect explained in Section 4.2.1. Crossover points between S_{mag} and S_{lvl} are explained in Fig. 12.

Fig. 10: Comparison of histogram errors among streams. Plots are truncated to highlight differences without hiding important trends. In general, in terms of error, S_{hist-opt} \approx S_{hist-sig} \approx S_{wav} < S_{lvl}, S_{mag}.

Fig. 11: Histograms of the boiler data set, reconstructed at 0.3 bps. S_{lvl}, S_{wav}, and S_{hist-sig} produce histograms that share a shape similar to the reference histogram, with most of the peaks and valleys preserved. In contrast, S_{mag} produces a spurious peak not found in the reference. Finally, S_{mag}’s histogram has a widely skewed distribution where too many values fall into the first bin.

However, if the error caused by discarding a packet is of subvoxel resolution, such a metric fails to capture the importance of that packet, causing S_{iso-opt} to be ineffective. Therefore, we add the relative difference in surface areas (\|A_1 - A_2\|/\|A_1\|) to the error term. This additional term is often between [0, 1] and is meant to capture the subvoxel error when the number of misclassified voxels is zero.

With the error metric defined, we can compute a data-dependent stream optimized for this metric (S_{iso-opt}) and a stream based on its signature (S_{iso-sig}) using Algorithm 1 and Algorithm 2. Fig. 13 compares the performances of these two streams, along with S_{bit}, S_{lvl}, S_{wav}, and S_{mag}. As can be observed, S_{mag} performs poorly, indicating that isosurface extraction requires higher levels of resolution compared to histogram computation, as we need to resolve a surface in the spatial domain and not just the range domain of the function. S_{wav} and S_{iso-sig} typically outperform S_{bit}, especially at low bit rates. Fig. 14 renders the isosurfaces reconstructed at 0.6 bps for all streams. In terms of the quality of the reconstructed surfaces, S_{iso-sig} \approx S_{wav} > S_{bit} > S_{mag} > S_{lvl}, which agrees with the plots in Fig. 13. For isosurface extraction, S_{wav} appears to be the only stream — among the data-independent ones — that consistently works well in all cases.

### Analysis and Future Work

5 DISCUSSION AND FUTURE WORK

This work addresses one of the biggest contemporary challenges in visualization: management of the enormous amounts of data. We focus on the trade-off between two prominent dimensions of data reduction, resolution and precision, with respect to common analysis and visualization tasks. To keep the study tractable while not compromising the generalizability of results, we target fundamental analysis and visualization tasks, with an outlook that these can serve as building blocks for more complex and multiparameter tasks in the future. Although the paper focuses on a small set of core tasks, the framework is generic and applies to any well-defined metric, and one future direction is to consider a broader set of tasks.

We present the first empirical study to demonstrate that combining reduction in precision and resolution can lead to a significant improvement in data quality for the same data budget, and that different tasks might prefer different resolution-versus-precision trade-offs. For example, whereas computing histograms requires high precision, computing derivatives benefits more from higher resolution, and function reconstruction and isosurface extraction require a suitable mix of the two (see Fig. 15). We also show that common reduction techniques, e.g., those based on S_{lvl} and S_{mag}, do not perform well when leading zero bits are removed (to simulate entropy compression). For each task, the relative ordering of the rate-distortion curves stays largely the same regardless of data sets, although the gaps among them vary depending on the smoothness and noisiness of the data. Compared to data-independent streams, signature-based streams often perform better because they can adapt to the data. They are also amenable for implementation (unlike S_{opt}), since a signature is negligibly small and thus can be precomputed and stored during preprocessing. It is also interesting to consider per-block signatures instead of a global one.

An important question is whether task- and data-dependent streams provide sufficient advantages over purely data-independent streams. In practice, data would be used for multiple, and not necessarily predefined, tasks, and maintaining multiple streams will likely lead to additional overheads. Here, we consider S_{mag} to be the best possible stream that could be realized. Improvements on S_{mag} in the resolution-versus-precision space are likely possible, but they are unlikely to be significant. Given these assumptions and the fact that S_{big} in most cases provides very similar results to S_{bit} or S_{wav}, the additional effort (and potential overheads) for task-dependent bit orderings is unlikely to be beneficial. This leaves a significant gap between the best data-independent streams and the optimal stream S_{opt}. Our experiments suggest that the majority of these differences can be attributed to spatial adaptivity (see Fig. 15). The prototypical example is isosurface computation, where S_{opt} can skip all regions that do not affect any portion of the surface. It may be worthwhile in future work to investigate solutions to spatial adaptivity to significantly improve the
We focus on the trade-off between two prominent dimensions of S in visualization: management of the enormous amounts of data.

The isosurfaces reconstructed at 0.6 bps for all streams. In terms of the resolution, such a metric fails to capture the importance of that packet, iso-opt to be ineffective. Therefore, we add the relative subvoxel error when the number of misclassified voxels is zero. 

Fig. 13: Comparison of isosurface errors among streams. Plots are truncated to highlight differences without hiding important trends. The number of cells that each original surface occupies is reported. The trend in error is $S_{iso-opt} < S_{iso-sig} \approx S_{wav} < S_{mag} << S_{h}$.

(a) by level ($S_{h}$) (b) by bit plane ($S_{bit}$) (c) by wavelet norm ($S_{mag}$) (d) by magnitude ($S_{mag}$) (e) by signature ($S_{iso-sig}$) (f) reference

Finally, for fair comparisons, we always reconstruct the data at full resolution using wavelets. However, processing and memory costs are important, and it is likely that adaptive representations would be used in practice [16, 25, 46]. In these cases the error of a given approximation depends not only on the available information, but also on the data structure and algorithm being used. For example, trilinear interpolation on a coarse grid might produce different results than wavelet reconstruction on the original mesh. There exist solutions where both interpolations are equivalent [67], but such solutions have not yet been implemented in standard tools. An important future research direction will be to understand the implications of the results presented here for existing toolchains such as VTK.

The data is a 2D slice from diffusivity. Subbands are separated by red lines. The color of each pixel indicates the bit plane at which the corresponding coefficient currently is. Brighter greens correspond to more precision. Both the top and the bottom rows show that $S_{hist-sig}$ allocates more bits to the lower subbands, while $S_{lap-sig}$ prefers to stream bits from higher subbands. $S_{rms-sig}$ is somewhere in the middle. Both $S_{lap-sig}$ and $S_{lap-opt}$ share the same signature by definition but only $S_{lap-opt}$ provides spatial adaptivity, seen as nonuniform colors in each subband.

A L2 NORMS OF CDF5/3 WAVELET BASIS FUNCTIONS

The $S_{wav}$ stream requires the norms of CDF5/3 wavelet basis functions, which the following pseudocode computes (in 1D).

```python
sca_weights = [1/2, 1/4] wav_weights = [-1/8, -1/4, 3/4, -1/4, -1/8] scal_func = function adds a 0 between every two adjacent samples wav_func = wavelet reconstruction on the original mesh. There exist solutions (a) and Algorithm 2. Fig. 13 shows how the histograms vary depending on the smoothness and noisiness of the data. Compared to the resolution-versus-precision space are likely possible, but they affect any portion of the surface. It may be worthwhile in future work to additional overheads. Here, we consider predefined, tasks, and maintaining multiple streams will likely lead to data-independent streams, signature-based streams often perform better.

An important question is whether task- and data-dependent streams vary depending on the smoothness and noisiness of the data. Compared to the resolution-versus-precision space are likely possible, but they affect any portion of the surface. It may be worthwhile in future work to additional overheads. Here, we consider predefined, tasks, and maintaining multiple streams will likely lead to data-independent streams, signature-based streams often perform better.

This study can be considered only a first step toward a system of solutions that can optimize storage, network, and I/O bandwidth to suit specific tasks at hand. Ultimately, our results can guide development of new data layouts and file formats for scientific data. $S_{wav}$ appears to provide the best all around performance. However, for a file format additional constraints must be considered, such as disk block sizes, cache coherence, and compression. Furthermore, an ideal format should allow task-dependent data queries even though task-dependent formats will likely be restricted to very specific situations.

Finally, for fair comparisons, we always reconstruct the data at full resolution using wavelets. However, processing and memory costs are important, and it is likely that adaptive representations would be used in practice [16, 25, 46]. In these cases the error of a given approximation depends not only on the available information, but also on the data structure and algorithm being used. For example, trilinear interpolation on a coarse grid might produce different results than wavelet reconstruction on the original mesh. There exist solutions where both interpolations are equivalent [67], but such solutions have not yet been implemented in standard tools. An important future research direction will be to understand the implications of the results presented here for existing toolchains such as VTK.

ACKNOWLEDGMENTS

We thank the reviewers for their valuable comments. Some of the data sets are kindly shared with us by Benjamin Isaac (boiler) and Kristoffer Matheson and Ashley Spear (foam – collected through work supported by NSF grant DMREF-1629660). This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was supported by the LLNL-LDRD Program under Project No. 17-SI-004. This work is supported in part by NSF: CGV: Award:1314896, NSF: IIP Award:1602127 NSF:ACI:award 1649923, DOE/SciDAC DESC0007446, CCMSC DE-NA0002375, and PIPER: ER26142 DE-SC0010498. This material is based upon work supported by the Department of Energy, National Nuclear Security Administration, under Award Number(s) DE-NA0002375. This work was also supported by the Department of Energy Office of Science, Office of Advanced Scientific Computing Research under the guidance of Dr. Lucy Nowell.

REFERENCES


