Preserving Topological Feature with Sign-of-Determinant Predicates in Lossy Compression: A Case Study of Vector Field Critical Points

Mingze Xia
University of Kentucky
Lexington, USA
mingze.xia@uky.edu

Sheng Di
Argonne National Laboratory
Lemont, USA
sdi1@anl.gov

Franck Cappello
Argonne National Laboratory
Lemont, USA
cappello@mcs.anl.gov

Pu Jiao
University of Kentucky
Lexington, USA
pujiao@uky.edu

Kai Zhao
Florida State University
Tallahassee, USA
kzhao@cs.fsu.edu

Jinyang Liu
University of California, Riverside
Riverside, USA
jliu447@ucr.edu

Xuan Wu
University of Kentucky
Lexington, USA
xuan.wu@uky.edu

Xin Liang*
University of Kentucky
Lexington, USA
xliang@uky.edu

Hanqi Guo
The Ohio State University
Columbus, USA
guo.2154@osu.edu

***Abstract*—Lossy compression has been employed to reduce the unprecedented amount of data produced by today's large-scale scientific simulations and high-resolution instruments. To avoid loss of critical information, state-of-the-art scientific lossy compressors provide error controls on relatively simple metrics such as absolute error bound. However, preserving these metrics does not translate to the preservation of topological features, such as critical points in vector fields. To address this problem, we investigate how to effectively preserve the sign of determinant in error-controlled lossy compression, as it is an important quantity of interest used for the robust detection of many topological features. Our contribution is three-fold. (1) We develop a generic theory to derive the allowable perturbation for one row of a matrix while preserving its sign of the determinant. As a practical use-case, we apply this theory to preserve critical points in vector fields because critical point detection can be reduced to the result of the point-in-simplex test that purely relies on the sign of determinants. (2) We optimize this algorithm with a speculative compression scheme to allow for high compression ratios and efficiently parallelize it in distributed environments. (3) We perform solid experiments with real-world datasets, demonstrating that our method achieves up to 440% improvements in compression ratios over state-of-the-art lossy compressors when all critical points need to be preserved. Using the parallelization strategies, our method delivers up to $1.25\times$ and $4.38\times$ performance speedup in data writing and reading compared with the vanilla approach without compression.**

***Index Terms*—High-performance computing, lossy compression, sign of determinant, critical points**

I. INTRODUCTION

Today's scientific applications are producing data at an unprecedented speed and amount. For a typical example, climate simulations with 1 km $\times$ 1 km resolution generate over 200 TB of data every 16 seconds [1]. Those data quickly occupy the storage capacity and/or network bandwidth, leading to severe problems in data storage and transmission.

To address such data challenges, error-controlled lossy compressors [2]–[6] are proposed and aggressively developed in the last decade to reduce the size of scientific data while maintaining accuracy. Featuring high reduction ratios compared with lossless compressors and better fidelity than classic lossy compressors, these compressors are widely used in scientific applications to solve diverse problems, including mitigating storage requirement [7], improving I/O performance [8], and accelerating computation [9].

In this work, we aim to preserve the sign of determinant during lossy compression and leverage this theory to preserve critical points in vector fields. In comparison, existing error-controlled lossy compressors preserve the quality of data via general metrics such as absolute errors and l^2 errors, but few of them provide the preservation of topological information. Sign of determinant is an important quantity of interest in computing geometry, which is used to express many predicates such as convex hulls [10]. It has also been used to detect topological features, including isosurface and critical points [11]. Critical points are defined as the locations where the vector field vanishes, and they are the key constituents of vector field topology that is essential for flow visualizations [12]–[15]. While our main goal is to preserve critical points that are essential for visual analysis, the impact extends beyond that as critical points usually represent important physical phenomena such as eddies in ocean [16], cyclones in climate

* Corresponding author: Xin Liang, Department of Computer Science, University of Kentucky, Lexington, KY 40506.

applications [17], and vortices in fluid dynamics [18].

In the visualization community, various topological information, such as locations and types of critical points, should be preserved during the lossy compression to ensure accurate analytics, while these properties are often overlooked by existing lossy compressors, leading to altered topology in the reconstructed data. Although attempts have been made to preserve topology in vector field compression, the existing works are either impractical or non-general. For example, iterative methods such as Delaunay simplification [19] lead to unbounded running time; the clustering methods [20] work only for 2D vector fields. A variation of error-controlled lossy compression was also proposed to preserve the locations and types of critical points [21]. This approach derives sufficient error bounds for each vertex based on how critical points are extracted and leverages those derived error bounds to guide compression. However, it usually over-preserves data as the derived error bounds are sufficient but unnecessary, leading to suboptimal compression ratios. In addition, its error bound derivation is based on critical point extraction via numerical methods, which may have ambiguity issues due to the inexact floating-point arithmetics.

In this work, we propose a novel method to preserve the sign of determinant during lossy compression and apply it to preserve topological features such as critical points. Our compression framework is based on the coupled scheme in [21] but heavily extends it for high robustness, quality, and scalability. To address the issue of loose sufficient error bounds in [21], we propose a novel concept, namely *speculative compression*, to trade off computation performance for high compression ratios. In addition, we propose two parallelization strategies to preserve topological features – critical points in border cells constituted by vertices from multiple processors. To the best of our knowledge, this is the first attempt to leverage error-controlled lossy compressors for topology preservation in distributed environments. Specifically, our contributions are summarized as follows.

- We develop a general theory to preserve the sign of determinant during lossy compression, which can be used to preserve topological features extensively. As a use case, we apply it to preserve critical points in vector field compression.
- We implement a compression framework using the derived theory based on the coupled compression scheme proposed in [21]. We also propose a speculative compression scheme that features a highly flexible balance between compression ratios and speed.
- We carefully parallelize our algorithm to achieve critical point preserving lossy compression in distributed environments. In particular, we propose both a simple method with no communication cost and an optimized method that provides better compression ratios than the simple one with minimal communication overhead.
- We evaluate our method using four real-world datasets from climate and computational fluid dynamics (CFD) simulations. Experiments demonstrate that the proposed method faithfully preserves the outcome of critical point extraction while providing a compression ratio up to $4.4\times$ higher than the state of the arts. This leads to $4.38\times$ speedup on data reading performance when evaluated with 768 GB data on a cluster using $4,096$ cores.

The remaining sections are organized as follows. Section II discusses the background and related works. Section III formulates the research problem and provides an overview. In Section IV, we propose the error bound derivation theory for robust critical point preservation. The detailed implementation, along with the speculative compression scheme, is presented in Section V. In Section VI, we demonstrate our parallelization strategies. In section VII, we present and analyze the evaluation results. Finally, we conclude our work with a vision for future work in Section VIII.

II. Background and Related Works

In this section, we review the background for critical point extraction in vector fields and the literature about lossy compression. Without loss of generality, we assume piecewise linear interpolation in each cell, which is widely used in the community.

A. Sign of determinant and robust critical point extraction

The sign of determinant is an important quantity of data that can be used as algorithmic solutions to a wide range of geometric problems. Typical applications that rely on the sign of determinant include point-in-polygon test [22], hyperplane in Euclidean space [23], nonvertical hyperlanes [24], and construction of convex hulls [10]. In the following, we only introduce the algorithm that leverages the sign of determinant for robust critical point extraction in vector fields because of limited space, and refer readers to [25] for a more detailed treatment.

A critical point is defined as the location where the vector field vanishes. Critical points can be extracted by numerical methods, which can be reduced to finding zero points in each cell where the vector field is linearly interpolated according to data values on the vertices of the cell. In a 2D case, this can be formulated as solving the barycentric coordinates (μ_0, μ_1, μ_2) in the equation below:

$$\begin{bmatrix} u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \end{bmatrix} \begin{bmatrix} \mu_0 \\ \mu_1 \\ \mu_2 \end{bmatrix} = \mathbf{0} \text{ and } \mu_0 + \mu_1 + \mu_2 = 1, \quad (1)$$

where (u_i, v_i) are the vectors in each vertex. A critical point is detected in the cell if $0 \leq \mu_k \leq 1$ holds for any $k \in \{0, 1, 2\}$; otherwise, there would be no critical points in the cell. While this method provides a means for critical point detection and extraction, it may lead to ambiguities due to the inexact, and thus unstable, nature of floating-point arithmetic: results of the detection could change with the order of the vertices. For instance, one critical point on the edge of two cells may be extracted as two separate critical points in the two cells or no critical points in both cells (see [11] for a concrete example). This will cause inconsistent results, such

as broken or branched traces in critical point tracing, which is undesired in most cases.

To address this issue, prior research [11], [26] has suggested the use of Simulation of Simplicity (SoS) [25] for robust critical point detection and tracking. This is based on an important lemma in [26]: a simplex contains a critical point if and only if the origin $\mathbf{0}$ lies in the convex hull of the vectors at the simplex's vertices. As such, the detection of the critical point in a cell reduces to a point-in-simplex test with $\mathbf{x} = \mathbf{0}$ as the target point, which is a well-defined problem that can be solved by computing the orientation. As shown in Algorithm 1, we can compute the orientation of the simplex and compare it with the orientation computed after changing one of the vertices to the target point $\mathbf{0}$. If all of the orientations have the same sign, $\mathbf{0}$ is regarded as inside the simplex, so a critical point is detected; otherwise, it would be outside of the simplex, and no critical points will be reported.

If a critical point is detected in a cell, it can be extracted using the numerical methods mentioned above. In addition, critical points are categorized into different types, each representing a specific topology, based on the signs and existence of imaginary parts in the eigenvalues of their Jacobian matrix. More details about the types can be found in [27], [28].

Algorithm 1 Point-in-simplex test [25]

Input: Point $\mathbf{x}$ and simplex $\mathcal{S} = \{\mathbf{x_0}, \mathbf{x_1}, \ldots, \mathbf{x_n}\}$.
Output: True if $\mathbf{x}$ is in $\mathcal{S}$ and false otherwise.

$s \leftarrow$ `Orientation`$(\mathbf{x_0}, \mathbf{x_1}, \ldots, \mathbf{x_n})$ ▷ compute the orientation for $\mathcal{S}$
for $i \leftarrow 0$ to n **do**
  $(\mathbf{x'_0}, \mathbf{x'_1}, \ldots, \mathbf{x'_n}) = (\mathbf{x_0}, \mathbf{x_1}, \ldots, \mathbf{x_n})$
  $\mathbf{x'_i} \leftarrow \mathbf{x}$ ▷ replace the i-th data with $\mathbf{x}$
  $s_i \leftarrow$ `Orientation`$(\mathbf{x'_0}, \mathbf{x'_1}, \ldots, \mathbf{x'_n})$ ▷ compute the new orientation
  if $s \neq s_i$ **then** ▷ $\mathbf{x}$ is not in $\mathcal{S}$ if the orientations do not align
    return *false*
  end if
end for
return *true* ▷ $\mathbf{x}$ is in $\mathcal{S}$ if all orientations align

B. Lossy compression for scientific data

Since the lossless compressors [29]–[31] fail to provide satisfactory compression ratios for scientific data, and the classic lossy compressors [32], [33] may lead to unbounded errors, error-controlled lossy compressors are considered as a viable option for scientific data compression. Error-controlled lossy compressors can be generally classified into two categories, namely prediction-based ones and transform-based ones, depending on how data are decorrelated. Prediction-based compressors such as SZ [2] and FPZIP [3] decorrelate data via certain prediction methods, while transform-based compressors such as ZFP [4] and MGARD [5] do that leveraging specific transforms. The decorrelated data will then be quantized to a small set of discrete values for a better compressibility while controlling the data distortion. After that, the quantized data will be fed into lossless encoders such as Huffman [34] and ZSTD [35] to achieve de-facto shrinking of data size. Despite multiple error controls (including those for certain quantities-of-interest [36], [37]) provided by these compressors, most of them are topology-agnostic thus failing to preserve topological information such as critical points.

Vector field compression has also been studied for years to reduce data size while preserving critical topology. Most of these approaches, however, are subject to iterative operations, suffering from long execution time by nature. This includes the iterative clustering method, which was used in [20] for compressing 2D vector fields, the iterative collapsion of edges [38], and an iterative segmentation-based approach [39]. Delaunay simplification was used in [19] to compress vector fields based on edge collapsing, but it does not explicitly preserve the topology.

The most relevant research to the proposed work is [21], which uses a variation of error-bounded lossy compression to reduce the size of vector fields while preserving critical points. This approach derives sufficient vertex-wise error bounds based on how critical points are extracted, ensuring the preservation of critical points when the derived error bounds are enforced at every vertex. Such enforcement is ensured by performing linear-scaling quantization [7], which in turn guarantees the preservation. Note that the derived error bounds need to be aggressively quantized for high compression ratios and stored for usage during decompression. Although this approach provides an elegant way for critical point preserving lossy compression, it has several limitations. First, it can only preserve critical points extracted by numerical methods, which may have ambiguous issues themselves due to the inexact floating-point arithmetics [11], [26]. Second, it leads to suboptimal compression ratios in general because the derivation only provides sufficient yet not necessary error bounds. Third, it is designed for compression with a single processor, which may not directly generalize to distributed cases. In this paper, we carefully address all these three limitations by proposing a new error bound derivation theory, a novel speculative compression scheme, and two effective parallelization strategies detailed in the later sections.

III. Problem Formulation and Design Overview

In this section, we formulate our research problem, followed by an overview of the proposed compression framework.

We follow [21] to define the false cases in critical point preservation. Specifically, we have three false cases, namely false positive (FP), false negative (FN), and false type (FT). FP occurs if a critical point is detected in a cell with decompressed data but does not exist with the original data. FN indicates that a critical point is identified in a cell with the original data but absent with the decompressed data. At last, FT means that while the critical point is present in a cell with both the original data and decompressed data, it has two different types. For instance, FT occurs when an attracting node in the original data becomes a repelling one in the decompressed data. If

there is a matched pair of critical points in the original and decompressed data, we call it true positive (TP).

With these evaluation metrics for critical points, we define the research objective as follows. Basically, we want to achieve as high compression ratios as possible while enforcing user-specified absolute error bound τ and preserving all the critical points. Mathematically, these two constraints can be formulated as $||\mathbf{d} - \mathbf{d'}||_{L^\infty} \leq \tau$ and $\mathbf{card}(FP) = \mathbf{card}(FN) = \mathbf{card}(FT) = 0$, where $\mathbf{d}$ and $\mathbf{d}'$ are the original and decompressed data, respectively, and $\mathbf{card}$ is the cardinality that indicates the number of false critical points. Note that we also care about the compression and decompression throughput, but we prioritize compression ratio because it is more important in multiple use cases, such as mitigation of storage pressure and transmission across wide area networks.

We first review the coupled compression scheme proposed in [21], which is the pioneering work to leverage error-controlled compression for critical point preservation. This prior compression scheme inherits the traditional prediction-based compression pipeline [40], but adds an error bound derivation module to compute a sufficient error bound, which guarantees the preservation of critical points extracted by numerical methods. This error bound is then fed into a linear-scaling quantizer [7], to ensure the error of the target data point is less than the bound. As its error bound derivation module is specifically tied to numerical methods, it does not generalize to other critical point detection methods directly. In other words, it cannot provide guaranteed preservation toward critical points detected by the point-in-simplex test, which is more widely used than numerical methods due to its robustness. Furthermore, by adopting the sufficient but not necessary error bounds to guide the compression, it only provides limited compression ratios (less than $10\times$ in most cases) that are usually insufficient for practical use. Last but not least, it offers no parallel support, and a direct parallelization with domain decomposition will lead to failures in preserving critical points in border cells constituted by vertices from different processors.

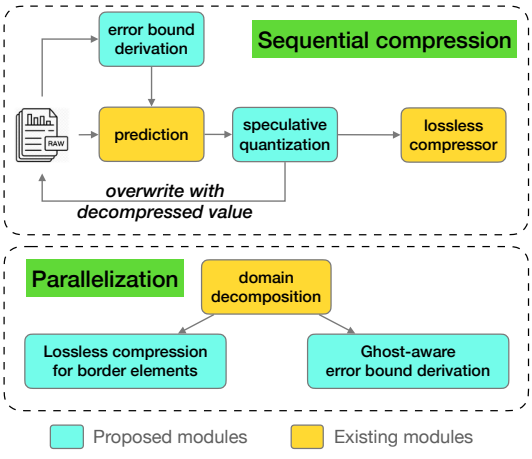


Fig. 1. Overview of the proposed compression framework.

We now present the overview of our framework in Fig. 1, which is built on the compression scheme mentioned above but heavily extends the functionality to preserve critical points detected by robust methods, provide a substantial improvement on compression ratios, and enable efficient parallelization. We highlight our proposed modules compared to the prior compression scheme using cyan boxes. Specifically, we propose a new error bound derivation module inherited from the sign of determinant preservation, which is well suited for preserving both critical points and other features. We then adopt a speculative quantization module that allows for flexible trade-offs between compression ratios and speed, making it adaptable to a wide range of use cases. To preserve critical points belonging to border cells in a distributed system, we propose two novel modules which can complement each other in different cases. The first module is a rather simple one, which compresses border elements losslessly, and the other one is driven by a parallelization strategy that integrates a more effective compression method that performs error bound derivation using ghost elements. These two modules have different pros and cons in terms of compression ratios and speed, thus, can be adopted in an adaptive fashion during runtime.

IV. Theoretical Foundation

In this section, we first derive the theories for preserving the sign of determinant, which is the foundation for many problems. After that, we show how to apply them to preserve features using critical points in vector fields as a case study.

Theorem 1: Given the target $(n+1) \times (n+1)$ matrix Λ, a sufficient absolute error bound to perturb the values in the m-th row while preserving the sign of determinant is:

$$\Psi(\Lambda) = \begin{cases} 0 & \textit{when} \quad \det(\Lambda) = 0 \\ \frac{|\det \Lambda|}{\sum_{i=0}^{n} |\det A_{mi}|} & \textit{Otherwise} \end{cases} \tag{2}$$

where A_{mi} is the submatrix obtained by removing the m-th row and the i-th column of Λ.

Proof:

We focus on proving the non-degenerative case since the degenerative case holds automatically. Without loss of generality, we assume that the last row $(x_{n0}, x_{n1}, \ldots, x_{nn})$ is perturbed by an error ε_i in the position i such that $\forall i, |\varepsilon_i| \leq \Psi(\Lambda)$. Then, the new determinant can be computed as follows:

$$\begin{aligned} \det \Lambda' &= \begin{bmatrix} x_{00} & x_{01} & \cdots & x_{0(n-1)} & x_{0n} \\ x_{10} & x_{11} & \cdots & x_{1(n-1)} & x_{1n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n0}+\varepsilon_0 & x_{n1}+\varepsilon_1 & \cdots & x_{n(n-1)}+\varepsilon_{n-1} & x_{nn}+\varepsilon_n \end{bmatrix} \\ &= \sum_{i=0}^{n} (-1)^{n+i} (x_{ni} + \varepsilon_i) \det A_{ni} \\ &= \sum_{i=0}^{n} (-1)^{n+i} x_{ni} \det A_{ni} + \sum_{i=0}^{n} (-1)^{n+i} \varepsilon_i \det A_{ni} \\ &= \det \Lambda + \sum_{i=0}^{n} (-1)^{n+i} \varepsilon_i \det A_{ni} \end{aligned}$$

When $\det\Lambda > 0$, we have:
$\det\Lambda' \geq \det\Lambda - \sum_{i=0}^{n}|\varepsilon_i||\det A_{ni}| > \det\Lambda - \sum_{i=0}^{n}\Psi(\Lambda)|\det A_{ni}| = \det\Lambda - \det\Lambda = 0$.
When $\det\Lambda < 0$, we have:
$\det\Lambda' \leq \det\Lambda + \sum_{i=0}^{n}|\varepsilon_i||\det A_{ni}| < \det\Lambda + \sum_{i=0}^{n}\Psi(\Lambda)|\det A_{ni}| = \det\Lambda - \det\Lambda = 0$. As such, $\det\Lambda$ and $\det\Lambda'$ always have the same sign, which corresponds to the same orientation.

Lemma 1: If the last column of Λ is an all-one vector, the sufficient bound can be optimized to:

$$\Psi(\Lambda) = \begin{cases} 0 & \textit{when} \quad \det(\Lambda) = 0 \\ \frac{|\det\Lambda|}{\sum_{i=0}^{n}|\det A_{ni}|} & \textit{Otherwise} \end{cases} \tag{3}$$

This theorem (and the lemma) identifies sufficient error bounds to preserve the sign of determinants in general cases, which directly leads to the following lemma and theorem for feature preservation.

Lemma 2: A sufficient absolute error bound for preserving the relative intersection position of a value f and an edge (f_0, f_1) is:

$$\begin{aligned}\Psi(f_0, f_1; f) &= \min\left(\Psi(\begin{bmatrix} f_0 & 1 \\ f & 1 \end{bmatrix}), \Psi(\begin{bmatrix} f_1 & 1 \\ f & 1 \end{bmatrix})\right) \\ &= \min(|f - f_0|, |f - f_1|).\end{aligned}$$

This reduces to the derivation theory for isosurface preservation in [37]. In the following, we show that Theorem 1 can be used to preserve critical points extracted from the robust point-in-simplex test (see Algorithm 1), which is the focus of this paper.

Theorem 2: A sufficient absolute error bound for the n-th point x_n in a $(n+1)$-simplex $\mathcal{S} = \{x_0, x_1, \ldots, x_n\}$ (x_i has n components) to keep the result of critical point detection is:

$$\Psi(\mathcal{S}) = \min(\Psi(\Lambda), \min_{0\leq i\leq n-1}\Psi(A_{in})) \tag{4}$$

where Λ is the orientation matrix for $\mathcal{S}$ and A_{ni} is the submatrix obtained by removing the i-th row and the n-th column of Λ.

Proof: Let Λ_i be the orientation matrix after replacing $x_i = (x_{i0}, x_{i1}, \cdots, x_{i(n-1)})$ with $x = (0, 0, \ldots, 0)$. According to Algorithm 1, preserving the signs of $s = \det\Lambda$ and $s_i = \det\Lambda_i$ is sufficient to preserve the outcome of the point-in-simplex test for critical point detection. Thus, a sufficient solution for this problem is $\Psi(\mathcal{S}) = \min(\Psi(\Lambda), \min_{0\leq i\leq n-1}\Psi(\Lambda_i))$. Meanwhile, we have:

$$\begin{aligned}\det\Lambda_i &= \begin{bmatrix} x_{00} & x_{01} & \cdots & x_{0(n-1)} & 1 \\ x_{10} & x_{11} & \cdots & x_{1(n-1)} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{(i-1)0} & x_{(i-1)1} & \cdots & x_{(i-1)(n-1)} & 1 \\ 0 & 0 & \cdots & 0 & 1 \\ x_{(i+1)0} & x_{(i+1)1} & \cdots & x_{(i+1)(n-1)} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n0} & x_{n1} & \cdots & x_{n(n-1)} & 1 \end{bmatrix} \\ &= (-1)^{n+i}\det A_{in}\end{aligned}$$

Therefore $\Psi(\Lambda_i) = \Psi(A_{in})$ and this completes the proof.

According to Theorem 2, we can directly derive sufficient error bounds for preserving the result of point-in-simplex test in 2D and 3D spaces using the following lemmas.

Lemma 3: A sufficient absolute error bound for preserving the critical point test in a simplex $\mathcal{S} = \begin{pmatrix} u_0 & v_0 \\ u_1 & v_1 \\ u_2 & v_2 \end{pmatrix}$ under a 2D vector field while perturbing (u_2, v_2) is:

$$\Psi(\mathcal{S}) = \min\left(\Psi(\begin{bmatrix} u_0 & v_0 & 1 \\ u_1 & v_1 & 1 \\ u_2 & v_2 & 1 \end{bmatrix}), \Psi(\begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix}), \Psi(\begin{bmatrix} u_0 & v_0 \\ u_2 & v_2 \end{bmatrix})\right). \tag{5}$$

Lemma 4: A sufficient absolute error bound for preserving the critical point test in a simplex $\mathcal{S} = \begin{pmatrix} u_0 & v_0 & w_0 \\ u_1 & v_1 & w_1 \\ u_2 & v_2 & w_2 \\ u_3 & v_3 & w_3 \end{pmatrix}$ under a 3D vector field while perturbing (u_3, v_3, w_3) is:

$$\begin{aligned}\Psi(\mathcal{S}) = \min(&\Psi(\begin{bmatrix} u_0 & v_0 & w_0 & 1 \\ u_1 & v_1 & w_1 & 1 \\ u_2 & v_2 & w_2 & 1 \\ u_3 & v_3 & w_3 & 1 \end{bmatrix}), \Psi(\begin{bmatrix} u_1 & v_1 & w_1 \\ u_2 & v_2 & w_2 \\ u_3 & v_3 & w_3 \end{bmatrix}), \\ &\Psi(\begin{bmatrix} u_0 & v_0 & w_0 \\ u_2 & v_2 & w_2 \\ u_3 & v_3 & w_3 \end{bmatrix}), \Psi(\begin{bmatrix} u_0 & v_0 & w_0 \\ u_1 & v_1 & w_1 \\ u_3 & v_3 & w_3 \end{bmatrix})).\end{aligned} \tag{6}$$

Note that the error bounds provided in Lemmas 3 and 4 only preserve the location of critical points (i.e., eliminating FN and FP). As such, we losslessly compress all nodes of a cell when a critical point is present. In addition, such error bounds keep the signs of each determinant, which over-preserve FN cases because the outcome of the point-in-simplex test will hold when $s \neq s_i$ for any i (see line 6 in Algorithm 1). We relax the error bounds for some special cases to accommodate this situation. In particular, it is well-known that critical points will not exist in a cell where at least one component of the vector field for that cell has the same sign, so we revise the sufficient error bounds accordingly, which is detailed in Algorithm 2 (to be introduced in the next section).

V. Implementation and Optimizations

A. Algorithm and implementation

As mentioned in Section III, we follow the coupled compression scheme proposed in [21] to implement our algorithm. The detailed steps are presented in Algorithm 2. The idea is to derive the error bound and perform error-bounded compression on the fly for each data point. Compared to that approach, our key innovation is to leverage the new error bound derivation mechanism proposed in the last section (line 5-17). In particular, we use $\Psi(j)$ derived in Lemmas 3 and 4 to determine the error bound for a data point with respect to any simplex j containing that point (line 10). We also pre-compute the existence of critical points for each cell (line 1-3) to avoid re-computation in line 7. This is important for the proposed method as the point-in-simplex test is more expensive than the numerical methods. In addition, we perform relaxation of the derived error bound when some components in a cell have the same sign (line 11-15), which indicates no critical point in the cell. This is done by evaluating the signs of all vertices in the current cell for each component. Especially when the condition holds for any component, we will relax to the error bound to the maximal one between its current value and the absolute value of the corresponding component at the current

vertex because the latter is sufficient to preserve the sign of that component which ensures that no critical point will be present in the decompressed data. After that, we follow the algorithm in [21] to aggregate and quantize the derived error bound, which is then used to quantize the original data and compute the decompressed data for later use (line 18-22). To this end, the quantized integers of both data and error bounds are fed to lossless encoders, including Huffman encoder and ZSTD, to perform the actual data reduction (line 24).

Algorithm 2 CP-preserving lossy compression for 3D data

Input: Input fixed-point vector field $\{\mathbf{d}\} = \{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$, fixed-point error bound τ' transformed from user-specified error bound τ.
Output: Compressed bytes.

1: **for** $i \leftarrow 0 \rightarrow n_c - 1$ **do** ▷ pre-compute existence of critical point for each cell
2: $cp\_exists[i] = \texttt{point\_in\_simplex}(\mathbf{0}, i)$
3: **end for**
4: **for** $i \leftarrow 0$ to $n_v - 1$ **do** ▷ iterate vertices
5: **for** $j \in$ vertex_cells(i) **do** ▷ iterate cells connected to vertex i
6: $\{i_0, i_1, i_2, i\} \leftarrow$ cell_vertices(j) ▷ vertices of cell j; fix i as the last index
7: **if** $cp\_exists[j]$ **then**
8: $\xi_i^{(j)} \leftarrow 0$ ▷ set to lossless to preserve types
9: **else**
10: $\xi_i^{(j)} \leftarrow \min(\Psi(j), \tau')$ ▷ see Lemma 4
11: **for** $z \in \{u, v, w\}$ **do** ▷ Check if any component has the same size
12: **if** $sgn(z_{i_0}) == sgn(z_{i_1}) == sgn(z_{i_2}) == sgn(z_i)$ **then**
13: $\xi_i^{(j)} \leftarrow \max(\xi_i^{(j)}, |z_i|)$ ▷ Relax the sufficient error bound
14: **end if**
15: **end for**
16: **end if**
17: **end for**
18: $\xi_i \leftarrow \min_j \xi_i^{(j)}$ ▷ aggregate error bound for vertex i
19: $\widehat{\xi_i} \leftarrow \texttt{quant}(\xi_i)$ ▷ quantize error bound of vertex i
20: $q_i \leftarrow \texttt{lossy\_compress}(\mathbf{d_i}, \widehat{\xi_i})$ ▷ quantize vector values with error bounds
21: $\mathbf{d}_i' \leftarrow \texttt{decode}(\text{bytes}, \widehat{\xi_i})$ ▷ calculate decompressed value $\mathbf{d}_i'$ on the fly
22: $\mathbf{d}_i \leftarrow \mathbf{d}_i'$ ▷ replace the input value with the decompressed value
23: **end for**
24: **return** compress_losslessly($\{q_i\}$, $\{\widehat{\xi_i}\}$)

Similar to [21], the proposed algorithm has a theoretical computational complexity of $\mathcal{O}(n_v)$ for structured data where n_v is the number of vertices, but it is expected to have higher decompression speed due to the adoption of the derivation theory in Section IV with absolute error bound. This eliminates the expensive logarithmic transform on original data, which is required in [21] to perform effective compression with point-wise relative error bound [41]. In addition, error bound derivation in this algorithm is based on the point-in-simplex test that allows for robust critical point detection, which is very useful in resolving ambiguous cases in a wide range of analytics. Its memory complexity is also $\mathcal{O}(n_v)$, which is same as the coupled method in [21].

B. Speculative compression

To accommodate the variability of computation and I/O in different computing systems, we propose a novel method to provide better flexibility on the trade-off between compression speed and ratios. We called it "speculative compression", as it borrows the concept of speculative execution [42] in computing systems. Specifically, we will compress data with a relaxed error bound to allow for higher compression ratios and roll back if such an error bound leads to discrepant results in our preserving target. This is inspired by the fact that the derived error bounds are sufficient but not necessary, which leads to over-preservation in many cases.

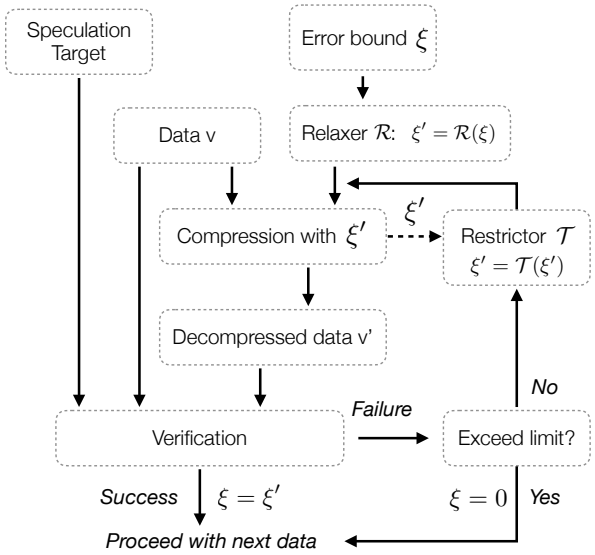


Fig. 2. Workflow of speculative compression.

Fig. 2 depicts the workflow for speculative compression given the speculation target, initial error bound ξ, and the data value v. The input error bound ξ is first relaxed to a higher one (denoted ξ') by a relax function $\mathcal{R}$, and then used to compress the data v and generate the decompression data v'. Then, the decompressed data is evaluated with the speculation target to see if all the conditions are met. If such verification is successful, we will use the relaxed error bound ξ' to compress our data; otherwise, we will restrict the current error bound ξ' to a lower one by a restriction function $\mathcal{T}$ and repeat the compression. To avoid extended execution time, we perform a hard cut-off if the number of failures exceeds a preset limitation n_l. In such case, we will set $\xi = 0$ for lossless compression of v, which ensures that no error will be introduced. In our implementation, we use simple relax function $\mathcal{R}(x) = 2^{n_l}x$ and restriction function $\mathcal{T}(x) = \frac{1}{2}x$.

Speculative compression is expected to deliver higher compression ratios at the cost of lower compression speed because it involves a trial-and-error process to figure out an error bound that is usually larger than the derived one. The essential trade-offs in between are highly related to the speculation targets. In the following, we investigate three speculation targets for Algorithm 2 and compare their impacts using a 2D vector field. We use $\mathbf{d_i}$ and $\mathbf{d'_i}$ to denote the i-th original and decompressed data (each with multiple components), respectively.

a) Speculation on derived error bound: This speculation relaxes the error bound to perform compression (line 18-20 in Algorithm 2), because the errors in the decompressed data may be much less than the specified error bound. The speculation target can be formulated as $||\mathbf{d}_i - \mathbf{d'}_i||_{L^\infty} \leq \xi_i$ (see line 18 in Algorithm 2 for ξ_i). It is the most lightweight solution because only the quantization of data and error bounds will be speculated.

b) Speculation on FN preservation: This speculation relaxes the derivation for preserving FNs (line 10-15 in Algorithm 2). The target can be formulated as $\texttt{point\_in\_simplex}(\mathbf{0}, j') == false$, where j' represents the vertex cell j after changing the current data point $\mathbf{d}_i$ to $\mathbf{d'}_i$. It introduces higher computational overhead because the verification needs to involve a point-in-simplex test for every cell containing the current data point.

c) Speculation on the entire critical point preservation procedure: This speculation further relaxes the entire derivation process (line 6-16 in Algorithm 2). The target can be formulated as $\texttt{critical\_point\_type}(j') == \texttt{critical\_point\_type}(j)$. It introduces the highest overhead because both critical point detection and its type identification need to be performed on every adjacent cell containing the current data point.

We summarize the speculation targets mentioned above and use an abbreviation for each of them in the later texts. We also formulate four speculation targets and analyze their correspondences to our compression algorithm in Table I. Generally speaking, aggressive speculation leads to high compression ratios but may suffer from low compression speed. For speculation on FN preservation, we evaluate two different values of n_l to investigate its impact on the speed and ratio.

TABLE I
SPECULATION TARGET AND IMPACT

Speculation Target	Abbr.	Correspondence in Algorithm 2	Speed	Ratio
None	NoSpec	-	Fast↑	Low↓
Derived error bound	ST1	line 18-20	Fast	Low
Preserving FN ($n_l = 1$)	ST2	line 10-15	Slow	High
Preserving FN ($n_l = 3$)	ST3	line 10-15	Slow	High
Preserving FN, FP, and FT	ST4	line 6-16	Slow↓	High↑

VI. PARALLELIZATION

In this section, we propose two methods to parallelize our algorithms in distributed environments. The key challenge in parallelization is to preserve critical points in edge and corner cells constituted by vertices across different processors, as depicted by the red and green regions in the 2D example in Fig. 3 (a). These cells are usually overlooked by traditional compression methods that are embarrassingly parallel, leading to incorrect critical point information within. While ghost elements [43] can be employed to exchange the values of the border elements, the adjacent elements across two processors cannot be compressed concurrently because the formula in Theorem 1 only allows for error introduction in one row.

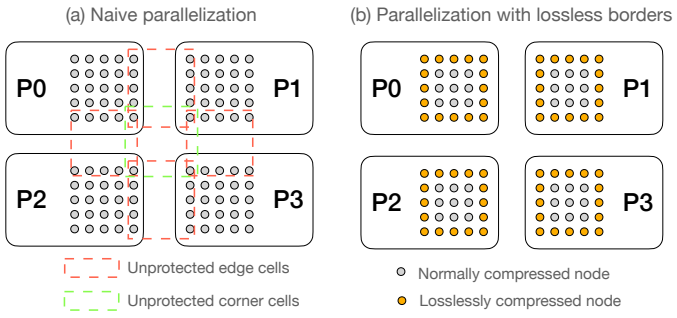


Fig. 3. Embarrassingly parallel strategies.

We first propose a simple yet effective parallelization strategy to preserve critical points in border cells with no communication cost. As illustrated in Fig. 3 (b), we use error bound 0 for all the border elements, which leads to lossless compression of those elements to ensure the same topology as that of the original data. However, this may have a negative impact on the compression ratios compared with naive parallelization, as more data points are encoded losslessly. Comparisons between these two methods are presented in Table II when $\tau = 0.01$, using the strong-scaling results with 1, 8, and 64 cores for the Nek5000 data. According to the table, it is observed that while being able to preserve all the critical points, parallelization with lossless border causes $25\% \sim 50\%$ degradation on the compression ratios, and the level of degradation increases with the number of cores used due to the increasing percentage of border elements. It is also observed that adopting a higher level of speculation leads to more degradation in the compression ratios. Due to the embarrassingly parallel design, both parallelization strategies yield almost linear speedup with around 100% parallel efficiency.

TABLE II
RESULT OF NAIVE PARALLELIZATION ON NEK5000

#Cores	Method	Speculation	TP	FP	FN	FT	Ratios	$S_c(MB/s)$	$S_d(MB/s)$
1	Naive parallelization	None	12,482	0	0	0	14.99	7.54	139.22
		ST4	12,482	0	0	0	19.01	3.50	141.34
	Lossless borders	None	12,482	0	0	0	14.26	5.06	93.07
		ST4	12,482	0	0	0	18.28	4.03	129.94
8	Naive parallelization	None	12,407	114	66	9	15.23	64.43	897.98
		ST4	12,301	582	169	12	19.41	25.83	903.53
	Lossless borders	None	12,482	0	0	0	12.20	65.09	892.63
		ST4	12,482	0	0	0	14.00	32.42	881.06
64	Naive parallelization	None	12,322	212	135	25	14.70	513.36	6638.57
		ST4	12,106	1,009	341	35	18.88	189.74	6776.97
	Lossless borders	None	12,482	0	0	0	9.34	515.32	6502.96
		ST4	12,482	0	0	0	10.35	258.67	6597.49

To address the limitation of parallelization with lossless borders, we propose another strategy that significantly reduces the number of elements requiring lossless representation with

low communication overhead. In particular, we pre-define the compression order of border elements and perform two-phase communication and compression as illustrated by a 2D example in Fig. 4. We start with the initial arrays on each processor, each with allocations for ghost elements but no values. During the first-phase communication, all the processors receive ghost elements from their left and top neighbors. This provides opportunities for each processor to normally compress data in the top left corner, which is done in the first-phase compression. For instance, as P1 receives the ghost elements on the left edge, it can compress all the data except the last row; however, the last row and column in P0 cannot be compressed because of a lack of information on their neighbors. Note that we use decompressed data to overwrite the original data upon successful processing of a vertex, which is required for both accurate prediction [7] and error bound derivation [21]. Also, note the vertices in the corners are always losslessly compressed to eliminate complex diagonal communication. In the second-phase communication, all the processors will receive the ghost elements in the form of decompressed data from their right and bottom neighbors to provide neighborhood information for the unprocessed vertices. In the last step, second-phase compression is performed to compress the rest vertices. This strategy also generalizes to 3D cases, where the exchange of ghost elements needs to be performed for each surface of the data cube, and vertices located on the edges of the ghost cube are compressed losslessly.

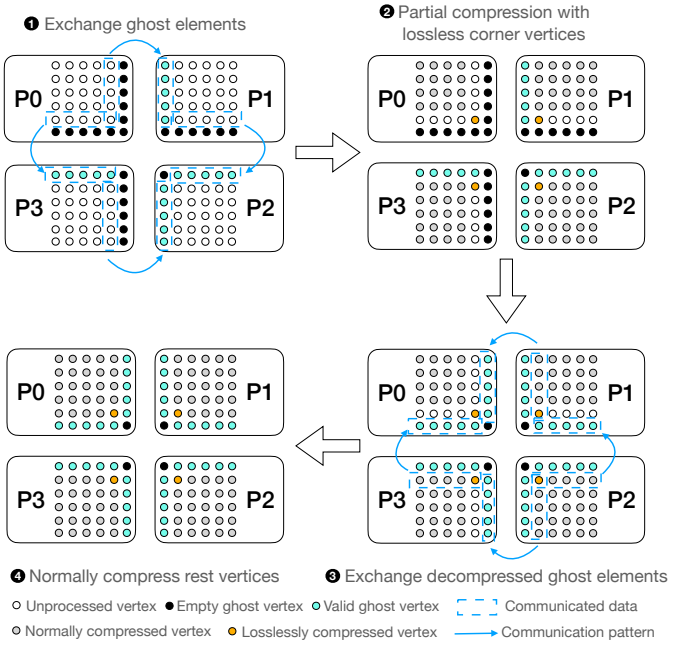


Fig. 4. Ratio-oriented parallel strategy.

Efficiency and complexity: We then analyze the efficiency and complexity of the two methods via the percentage of lossless compressed border elements and communication overhead. Without loss of generality, we assume all data are in single-precision floating-point format (4 bytes per data point), and λ and β are the message passing latency and bandwidth, respectively. We also assume that $n_1 \times n_2$ vertices are evenly distributed in a $\sqrt{p} \times \sqrt{p}$ processor grid in 2D cases and $n_1 \times n_2 \times n_3$ vertices are evenly distributed in a $\sqrt[3]{p} \times \sqrt[3]{p} \times \sqrt[3]{p}$ processor grid in 3D cases.

Parallelization with lossless borders: Since this method losslessly compresses all border elements, the percent of lossless compressed border elements is $\frac{2(n_1+n_2)\sqrt{p}}{n_1 n_2}$ in 2D cases and $\frac{2(n_1n_2+n_1n_3+n_2n_3)\sqrt[3]{p}}{n_1n_2n_3}$ in 3D cases. It has no communication overhead since the process is embarrassingly parallel.

Ratio-oriented parallelization: In the 2D cases, at most four elements are losslessly compressed in each processor, leading to a percentage of less than $\frac{4p}{n_1 n_2}$. Each phase of communication transmits two messages, each containing $\frac{n_1}{\sqrt{p}}$ and $\frac{n_2}{\sqrt{p}}$ vertices with 2 components, respectively. This yields $2(2\lambda + \frac{8(n_1+n_2)}{\sqrt{p}\beta})$ communication cost in total. In the 3D cases, at most $\frac{4(n_1+n_2+n_3)}{\sqrt[3]{p}}$ of vertices are compressed in a lossless fashion in a processor, which corresponds to a percentage of $\frac{4(n_1+n_2+n_3)\sqrt[3]{p^2}}{n_1n_2n_3}$. As for the communication, three messages of $\frac{n_1n_2}{\sqrt[3]{p^2}}$, $\frac{n_2n_3}{\sqrt[3]{p^2}}$, and $\frac{n_1n_3}{\sqrt[3]{p^2}}$ vertices with 3 components are transmitted in each phase, respectively, leading to $2(3\lambda + \frac{12(n_1n_2+n_2n_3+n_1n_3)}{\sqrt[3]{p^2}\beta})$ communication cost in total. However, there would be additional computational overhead for this approach though, as it needs to derive the error bounds for all the data points in 3D cases, while the prior parallelization directly uses error bound 0 for border elements.

Based on these statistics, we can conclude that parallelization with lossless borders features high speed at the cost of ratio deduction, while ratio-oriented parallelization mitigates the ratio deduction with extra overhead. In Table III, we present the ratios obtained by this parallelization strategy with no speculation. According to this table, the ratio-oriented method leads to compression ratios of $14.21\times$ and $13.19\times$ on the Nek5000 data with respect to 8 and 64 cores while preserving all the critical point information, which are very close to the compression ratios provided ($15.23\times$ and $14.70\times$ respectively) by naive parallelization without critical point preservation. The parallelization efficiency is roughly 75%, where the overhead mainly comes from the stencil communication for ghost element exchanges.

TABLE III
RESULT OF RATIO-ORIENTED PARALLELIZATION ON NEK5000

#Cores	Method	Speculation	TP	FP	FN	FT	Ratios	$S_c(MB/s)$	$S_d(MB/s)$
1	Ratio-oriented parallelization	None	12,482	0	0	0	15.00	7.40	137.76
8			12,482	0	0	0	14.21	63.87	848.62
64			12,482	0	0	0	13.19	383.04	4800.0

VII. EVALUATION

We evaluate our methods with four real-world datasets from climate and CFD simulations and compare them with four state-of-the-art error-bounded lossy compressors, namely FPZIP [3], SZ3 [2], ZFP [4], and cpSZ [21]. We present both the quantitative results in terms of the number of erroneous

critical points defined in Section III, as well as qualitative results for both 2D and 3D data. To this end, we further evaluate our parallelization strategies and present a large-scale case with 768 GB of data. Throughout the evaluation, we use "CR" to denote compression ratio, "S_C"/"S_D" to represent compression/decompression speed in megabytes per second (MB/s), "#TP" for the number of preserved critical points, and "#FN/#FP/#FT" for the number of erroneous critical points.

A. Experiment Setup

We evaluate four scientific datasets from four applications:

- **Ocean**: A simulated dataset representing ocean currents.
- **Nek5000**: A fluid simulation generated by Nek5000 [44].
- **Hurricane**: A simulation of Hurriance-ISABEL from the National Center for Atmospheric Research [45].
- **Turbulence**: A direct numerical simulation of forced isotropic turbulence on a $4,096^3$ periodic grid [46].

The detailed information of the datasets is listed in Table IV. Here n_d stands for the number of components in the data, and n_v and n_c are the numbers of vertices and cells, respectively. Based on the size of the data, we will evaluate the quality of the compression methods using the three datasets and I/O performance using the last one.

TABLE IV
BENCHMARK DATASETS

Dataset	n_d	n_v	n_c	Size
Ocean	2	3600×2400	$2 \times 3599 \times 2399$	65.92 MB
Hurricane	3	$100 \times 500 \times 500$	$6 \times 99 \times 499 \times 499$	286.10 MB
Nek5000	3	$512 \times 512 \times 512$	$6 \times 511 \times 511 \times 511$	1.50 GB
Turbulence	3	$4096 \times 4096 \times 4096$	$6 \times 4095 \times 4095 \times 4095$	768.00 GB

All of our experiments are conducted on a high-performance cluster [47], where each compute node contains 2 AMD EPYC ROME 7702P processors with 128 cores and 512 GB memory in total. The system is interconnected by 100Gbps InfiniBand and is equipped with Lenovo GPFS parallel file system.

B. Rate-distortion and speculation

We first present the rate-distortion of our methods and investigate the impact of speculation targets on the quality of compression. We use Peak Signal-to-Noise Ratios (PSNR) as our distortion metric due to its wide acceptance in the community, and bit-rate in the X-axis represents average bits per compressed data, which can be computed by 32 over compression ratios for single-precision floating-point data. The rate-distortion graphs for the Ocean and Nek5000 data are shown in Fig. 6, with points in the graph generated by setting $\tau = 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001$, respectively. As we can see, more aggressive speculation generally leads to higher compression ratios (lower bit rates), especially when the global error bound is relatively high, which also exhibits better rate-distortion in those regions. Nevertheless, the differences in compression ratios and PSNR diminish as the global error bound decreases. Another interesting observation is that higher error bounds may not always lead to higher ratios, which is clearly shown by the first a few error bounds in the 3D plot with Nek5000 data. This is possibly caused by the Lorenzo predictor used in the prediction stage of the compression. As Lorenzo predictor requires the use of decompressed data for prediction, lower error bounds may have better prediction accuracy due to lower errors in the decompressed data. Based on this observation, we set $\tau = 0.01$ for all later experiments for all speculation targets.

C. Preservation of critical points

We then present the preservation of critical points with our methods both qualitatively and quantitatively using the three datasets mentioned above. Since FPZIP, ZFP, and SZ3 do not provide mechanisms to preserve critical points, we tune them to the same ratio as the one provided by our method with no speculation using the available options provided by them. For cpSZ, we use pointwise relative error bound 0.1 for 2D data and 0.05 for 3D data as suggested by authors [21]. Note that cpSZ only provides guaranteed preservation of critical points when they are extracted using numerical methods, so it may introduce a small number of erroneous cases in our evaluation because we use SoS [25], [26] for critical point extraction.

The results on 2D Ocean data are displayed in Table V. As shown in the table, while general error-controlled lossy compressors control the maximal absolute error and/or pointwise relative error, they cannot preserve critical points. FPZIP performs the best among the general compressors, probably because it preserves pointwise relative error. While cpSZ has good preservation results, both of its two schemes have limited compression ratios. As comparisons, all of our methods preserve all the critical points, and our method with the most aggressive speculation delivers a compression ratio that is $2.71\times$ of that with the coupled scheme in cpSZ. Note that both cpSZ and our methods yield lower compression speed compared to the general lossy compressors, as they need to integrate the topological information into the compression process. Nonetheless, their decompression speed is comparable to SZ3 and FPZIP. In particular, our methods have higher decompression speed than cpSZ because we avoid the use of pointwise relative error bound, which requires an expensive logarithmic transform on the data.

We then present the qualitative results on this dataset by visualizing all critical points, with surface Line Integral Convolution (LIC) [48] on the vector field as background. It is clearly observed that many false positives occur in the decompressed data of SZ3 and ZFP, especially in regions that are close to the land areas. Also, it is interesting to see that only SZ3 and our method with high levels of speculation affect data in the land areas. For the former, it is mainly caused by error prorogation in the hierarchical interpolation; for the latter, it is due to the loosened error bound by the speculation as no critical points exist in the land areas.

The quantitative results for the two 3D datasets are listed in Table VI and Table VII, respectively. The trends are pretty similar to those of 2D, where general lossy compressors introduce a large number of false positives. Note that both cpSZ and our methods exhibit lower compression speed compared to 2D

TABLE V
QUANTITATIVE RESULTS ON 2D OCEAN DATA

Compressors	Settings	CR_u	CR_v	CR_{all}	S_c	S_d	$\#TP$	$\#FP$	$\#FN$	$\#FT$
FPZIP	-P 11	17.26	16.52	16.88	154.28	131.7	23,107	1,100	1,054	1,018
ZFP	-P 8	16.62	16.79	16.71	489.94	539.42	15,621	61,405	8,002	1,556
ZFP	-A 4	18.56	19.98	19.25	522.73	597.60	10,525	48,899	12,884	1,770
SZ3	-A 0.1	19.03	19.46	19.24	136.24	369.25	17,576	72,910	6,755	848
cpSZ	decoupled -R 0.1	-	-	7.58	38.99	101.40	25,137	0	0	42
	coupled -R 0.1	-	-	11.83	31.56	92.73	25,179	0	0	0
Our method	NoSpec -R 0.01	-	-	19.57	27.21	174.14	25,179	0	0	0
	ST1 -R 0.01	-	-	20.82	26.7	161.07	25,179	0	0	0
	ST2 -R 0.01	-	-	25.38	19.11	182.34	25,179	0	0	0
	ST3 -R 0.01	-	-	25.56	18.9	169.2	25,179	0	0	0
	ST4 -R 0.01	-	-	32.11	15.45	168.33	25,179	0	0	0

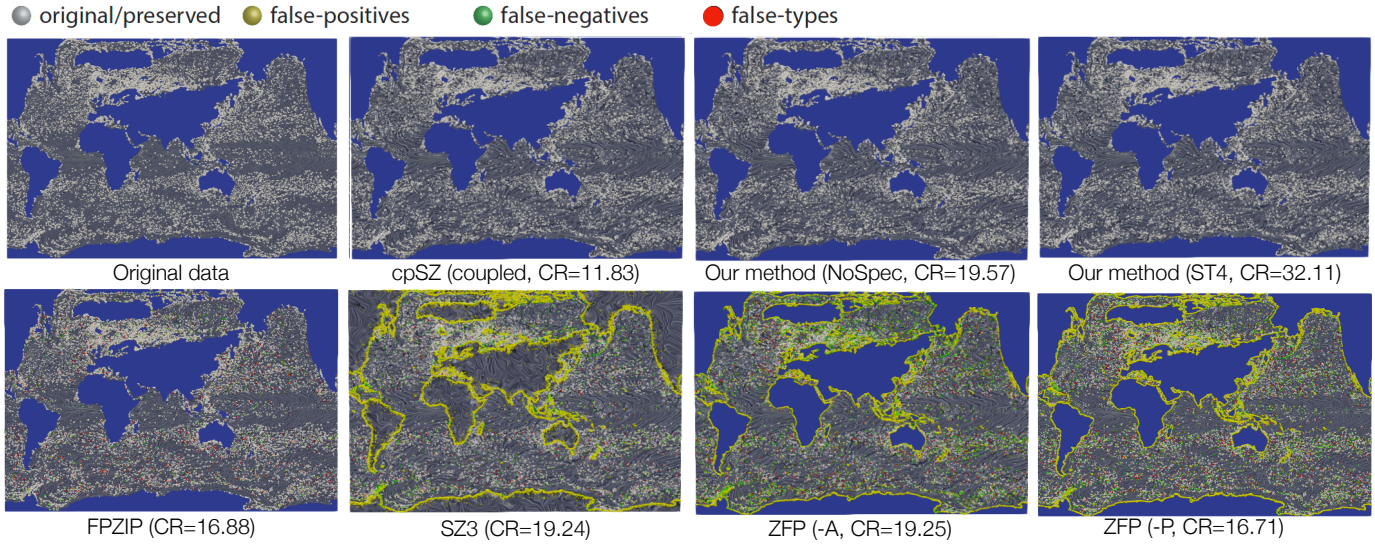
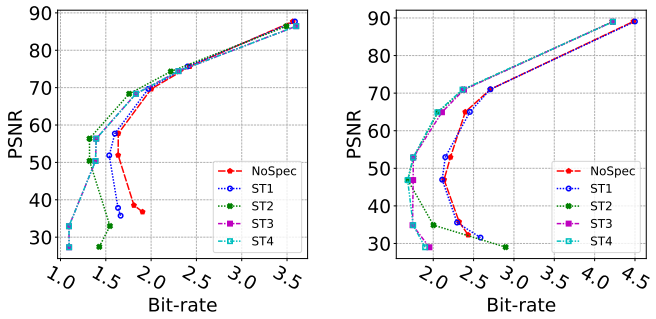


Fig. 5. Qualitative results on 2D Ocean data with surface LIC visualized as context.



(a) Ocean (b) Nek5000

Fig. 6. Rate-distortion of our method under different error bounds and speculation targets.

cases because each vertex has 24 adjacent cells to be analyzed for error bound derivation, whereas this number reduces to 6 in 2D data. Compared to the coupled scheme in cpSZ, our method with the most aggressive speculation yields 446% and 160% improvement on compression ratios in Hurricane and Nek5000, respectively.

We also present qualitative observations for the two datasets in Figs. 7 and 8, respectively, using streamlines [49] traced by points along the diagonal line as the context. We eliminate ZFP and SZ3 because the large number of false positives across the entire space makes it hard to visualize. It is observed that cpSZ leads to the best preservation of the global streamlines, but its compression ratio is pretty low. Our method with the most aggressive speculation provides better quality than FPZIP while providing much higher compression ratios.

D. Parallel I/O performance

We perform a parallel experiment to evaluate the two proposed parallelization strategies for the writing and reading time using the Turbulence data. The writing time is measured by the summation of time on compression and writing compressed data, while the reading time is computed by adding the time of reading compressed data to decompression time. We perform a strong-scaling test with 512 cores and $4,096$ cores, where the Turbulence dataset is divided into 512 blocks of $512\times512\times512$ grids and $4,096$ blocks of size $256\times256\times256$ grids, respectively, with each processor dealing with one block of data. For simplicity, we use "Simple" to denote the parallelization with lossless borders and "Ratio-oriented" to represent the parallelization with two-phase communication and compression. We also include GZIP [29] as the evaluation baseline because it is a lossless compressor that can preserve all the critical points under the distributed setting as our methods do, while none of existing lossy compressors can do that. The corresponding results are reported in Fig. 9.

TABLE VI
QUANTITATIVE RESULTS ON 3D HURRICANE DATA

Compressors	Settings	CR_u	CR_v	CR_w	CR_{all}	S_c	S_d	$\#TP$	$\#FP$	$\#FN$	$\#FT$
FPZIP	-P 9	34.14	40.11	10.77	20.40	183.22	142.74	645	442	334	106
ZFP	-P 9	30.63	33.09	11.08	19.59	322.18	673.32	608	27,243	407	70
ZFP	-A 1	16.56	16.80	36.94	20.41	310.71	655.11	425	21,856	620	40
SZ3	-A 0.04	17.43	17.52	60.30	22.90	146.38	380.51	487	9,562	550	48
cpSZ	decoupled -R 0.05	-	-	-	3.30	12.01	60.97	1,085	0	0	0
	coupled -R 0.05	-	-	-	7.24	6.85	85.29	1,085	0	0	0
Our method	NoSpec -R 0.01	-	-	-	22.78	9.53	142.08	1,085	0	0	0
	ST1 -R 0.01	-	-	-	24.11	9.57	129.49	1,085	0	0	0
	ST2 -R 0.01	-	-	-	36.26	4.16	145.33	1,085	0	0	0
	ST3 -R 0.01	-	-	-	36.98	4.15	142.49	1,085	0	0	0
	ST4 -R 0.01	-	-	-	39.55	4.36	139.74	1,085	0	0	0

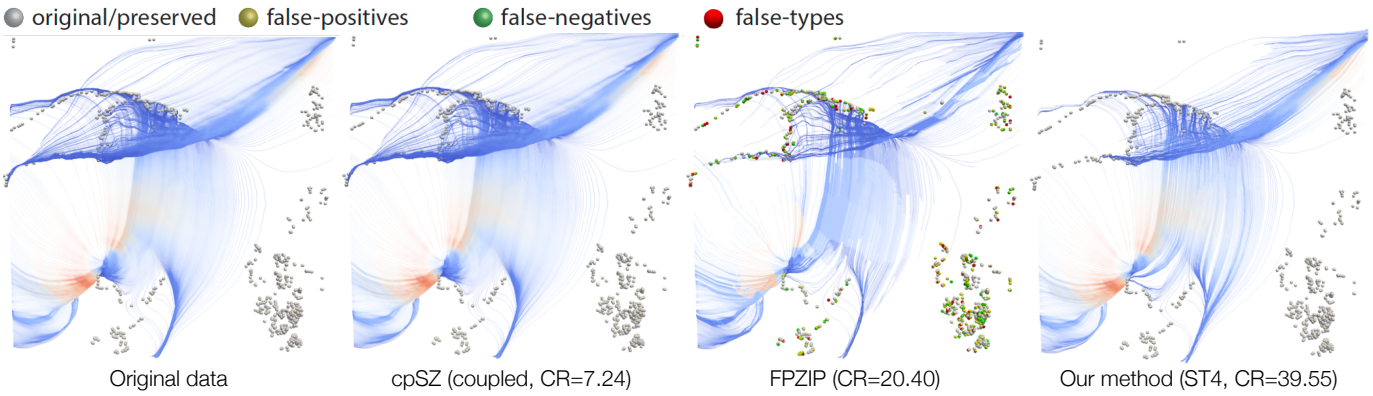


Fig. 7. Qualitative results on 3D Hurricane data with streamlines visualized as context.

TABLE VII
QUANTITATIVE RESULTS ON THE 3D NEK5000 DATA

Compressors	Settings	CR_u	CR_v	CR_w	CR_{all}	S_c	S_d	$\#TP$	$\#FP$	$\#FN$	$\#FT$
FPZIP	-P 11	14.0	13.56	15.34	14.26	140.08	127.89	9,932	2,328	2,084	466
ZFP	-P 10	13.64	13.20	14.57	13.78	285.76	534.53	7,253	232,235	4,848	381
ZFP	-A 4	14.74	15.06	14.78	14.86	294.48	558.47	5,649	192,363	6,345	488
SZ3	-A 0.13	14.28	14.95	15.62	14.93	48.04	22.68	7,313	114,448	4,762	407
cpSZ	decoupled -R 0.05	-	-	-	3.27	11.82	58.41	12,482	0	0	0
	coupled -R 0.05	-	-	-	7.30	6.62	83.13	12,469	7	9	4
Our method	NoSpec -R 0.01	-	-	-	15.00	8.64	141.93	12,482	0	0	0
	ST1 -R 0.01	-	-	-	15.16	8.62	119.10	12,482	0	0	0
	ST2 -R 0.01	-	-	-	19.00	4.75	143.37	12,482	0	0	0
	ST3 -R 0.01	-	-	-	18.27	4.75	142.52	12,482	0	0	0
	ST4 -R 0.01	-	-	-	19.02	5.06	143.99	12,482	0	0	0

The figure shows that all three strategies have negative impacts on writing data with 512 cores due to the slow compression performance. This is more obvious for ratio-oriented parallelization because it has higher computational and communication overhead than simple parallelization. However, benefits are observed for reading data with 512 cores due to the significantly reduced size ($15.17\times$ for the simple parallelization and $19.60\times$ for the ratio-oriented parallelization). This leads to more than 50% reduction in the reading time compared to reading the entire dataset without compression. For the large-scale evaluation with $4,096$ cores, significant improvements are observed in both writing and reading performance, as the reduced data size per core leads to greatly decreased compression/decompression time. While the resulting compression ratios ($13.29\times$ for the simple parallelization and $18.10\times$ for the ratio-oriented parallelization) are less than those of 512 cores, overall writing and reading performance improvements are more obvious. In absolute terms, our ratio-oriented strategy achieves $1.25\times$ and $4.38\times$ performance on writing and reading, respectively, compared with the vanilla approach with no compression. In contrast, lossless compression with GZIP yields minor performance gain compared with the vanilla approach because of its limited compression ratios. As scientific data is usually written once and read multiple times, we envision a broad use of the proposed methods for efficient data management due to its high reading performance.

VIII. CONCLUSION

In this paper, we develop a general theory for preserving signs of determinants and leverage it to implement a feature-preserving lossy compression framework. Unlike existing lossy compression frameworks, our framework can preserve

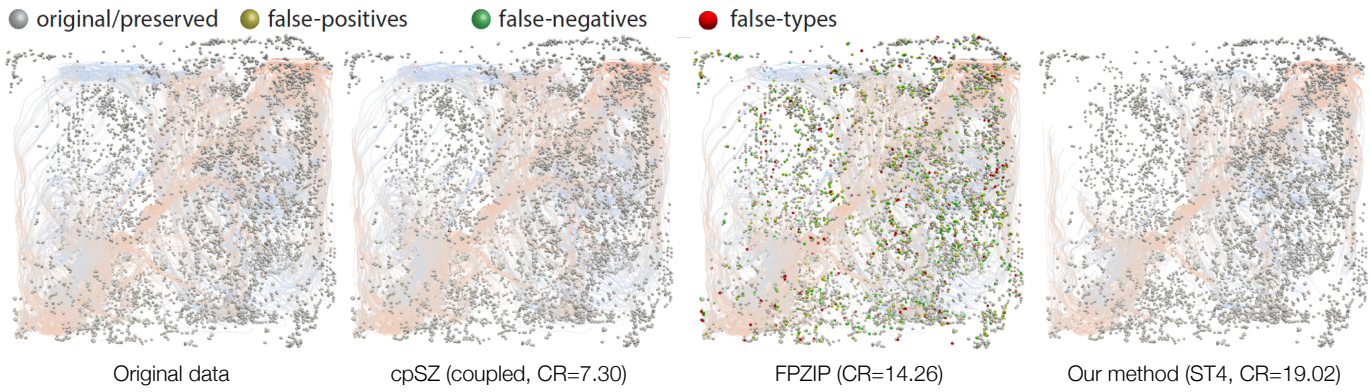
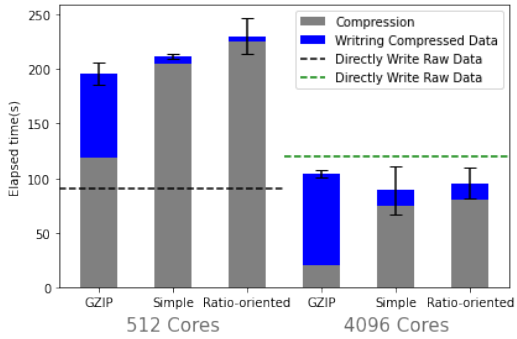
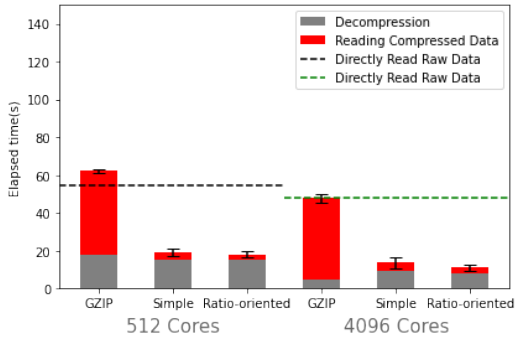


Fig. 8. Qualitative results on 3D Nek5000 data with streamlines visualized as context.



(a) Writing performance



(b) Reading performance

Fig. 9. Reading and writing performance with the proposed parallelization strategies using the Turbulence data on 512 and $4,096$ cores. Error bar encodes maximal and minimal time across 3 runs.

all critical points under robust detection algorithms. We further propose a speculative compression scheme that is able to obtain higher compression ratios with relaxed error bounds. In addition, we provide two strategies to parallelize our algorithm under distributed-memory settings. We evaluate our framework on four real-world datasets, with the largest dataset exceeding 700GB. Our experiments utilized up to $4,096$ cores across 50 nodes. Some key findings are listed below:

- Our framework provides guaranteed preservation of critical points while delivering compression ratios up to 440% better than the state of the arts.
- The proposed speculative compression significantly improves the compression ratios while retaining the critical points. Specifically, it results in compression ratio improvements of 54.23%, 26.80%, and 73.62% on the Ocean, Nek5000, and Hurricane datasets, respectively.
- Both of our parallelization strategies successfully preserve critical points during lossy compression in a parallel setting. In particular, the ratio-oriented parallelization leads to up to $1.25\times$ and $4.38\times$ speedup in writing and reading performance, respectively, compared to the vanilla approach with no compression on the Turbulent dataset using $4,096$ cores.
- Compared to cpSZ, although our framework is about $10\% - 25\%$ slower in compression speed, the decompression speed is $50\% - 100\%$ faster. Also, our framework achieves significantly higher compression ratios than cpSZ, while being able to preserve critical points extracted by robust algorithms.

In the future, we will extend this framework to preserve more features expressed by the sign of determinants. In addition, we will consider multiple ways to improve the efficiency of the proposed methods. For instance, we will leverage GPUs to improve the compression/decompression performance and investigate optimizations such as non-blocking message passing for better communication efficiency and asynchronous reading/writing for faster I/O operations.

ACKNOWLEDGMENTS

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations – the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, to support the nation's exascale computing imperative. The material was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (ASCR), under contract DE-AC02-06CH11357, and supported by the National Science Foundation under Grant OAC-2003709, OAC-2104023, OAC-2330367, OAC-2311756, and OAC-2313122. We acknowledge the computing resources provided on Bebop (operated by Laboratory Computing Resource Center at Argonne).

REFERENCES

[1] I. Foster, M. Ainsworth, B. Allen, J. Bessac, F. Cappello, J. Y. Choi, E. Constantinescu, P. E. Davis, S. Di, W. Di, H. Guo, S. Klasky, K. K. Van Dam, T. Kurc, Q. Liu, A. Malik, K. Mehta, K. Mueller, T. Munson, G. Ostouchov, M. Parashar, T. Peterka, L. Pouchard, D. Tao, O. Tugluk, S. Wild, M. Wolf, J. M. Wozniak, W. Xu, and S. Yoo, "Computing just what you need: Online data analysis and reduction at extreme scales," in *European conference on parallel processing*. Springer, 2017, pp. 3–19.

[2] K. Zhao, S. Di, M. Dmitriev, T.-L. D. Tonellot, Z. Chen, and F. Cappello, "Optimizing error-bounded lossy compression for scientific data by dynamic spline interpolation," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 1643–1654.

[3] P. Lindstrom and M. Isenburg, "Fast and efficient compression of floating-point data," *IEEE transactions on visualization and computer graphics*, vol. 12, no. 5, pp. 1245–1250, 2006.

[4] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE transactions on visualization and computer graphics*, vol. 20, no. 12, pp. 2674–2683, 2014.

[5] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data—the multivariate case," *SIAM Journal on Scientific Computing*, vol. 41, no. 2, pp. A1278–A1303, 2019.

[6] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola, "Tthresh: Tensor compression for multidimensional visual data," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 9, pp. 2891–2903, 2019.

[7] D. Tao, S. Di, Z. Chen, and F. Cappello, "Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization," in *2017 IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2017, pp. 1129–1139.

[8] X. Liang, S. Di, D. Tao, S. Li, B. Nicolae, Z. Chen, and F. Cappello, "Improving performance of data dumping with lossy compression for scientific simulation," in *2019 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2019, pp. 1–11.

[9] A. M. Gok, S. Di, Y. Alexeev, D. Tao, V. Mironov, X. Liang, and F. Cappello, "Pastri: Error-bounded lossy compression for two-electron integrals in quantum chemistry," in *2018 IEEE International Conference on Cluster Computing*. IEEE, 2018, pp. 1–11.

[10] F. P. Preparata and S. J. Hong, "Convex hulls of finite sets of points in two and three dimensions," *Communications of the ACM*, vol. 20, no. 2, pp. 87–93, 1977.

[11] H. Guo, D. Lenz, J. Xu, X. Liang, W. He, I. R. Grindeanu, H.-W. Shen, T. Peterka, T. Munson, and I. Foster, "Ftk: A simplicial spacetime meshing framework for robust and scalable feature tracking," *IEEE transactions on visualization and computer graphics*, vol. 27, no. 8, pp. 3463–3480, 2021.

[12] R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf, "The state of the art in flow visualization: Dense and texture-based techniques," in *Computer Graphics Forum*, vol. 23, no. 2. Wiley Online Library, 2004, pp. 203–221.

[13] R. S. Laramee, H. Hauser, L. Zhao, and F. H. Post, "Topology-based flow visualization, the state of the art," *Topology-based methods in visualization*, pp. 1–19, 2007.

[14] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen, "Over two decades of integration-based, geometric flow visualization," in *Computer Graphics Forum*, vol. 29, no. 6. Wiley Online Library, 2010, pp. 1807–1829.

[15] C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. De Floriani, G. Scheuermann, H. Hagen, and C. Garth, "A survey of topology-based methods in visualization," in *Computer Graphics Forum*, vol. 35, no. 3. Wiley Online Library, 2016, pp. 643–667.

[16] D. Matsuoka, F. Araki, Y. Inoue, and H. Sasaki, "A new approach to ocean eddy detection, tracking, and event visualization–application to the northwest pacific ocean," *Procedia Computer Science*, vol. 80, pp. 1601–1611, 2016.

[17] L. Yan, H. Guo, T. Peterka, B. Wang, and J. Wang, "Trophy: A topologically robust physics-informed tracking framework for tropical cyclones," *arXiv preprint arXiv:2307.15243*, 2023.

[18] P. Rautek, X. Zhang, B. Woschizka, T. Theusl, and M. Hadwiger, "Vortex lens: Interactive vortex core line extraction using observed line integral convolution," *IEEE Transactions on Visualization & Computer Graphics*, no. 01, pp. 1–11, 2023.

[19] T. K. Dey, J. A. Levine, and R. Wenger, "A delaunay simplification algorithm for vector fields," in *15th Pacific Conference on Computer Graphics and Applications (PG'07)*. IEEE, 2007, pp. 281–290.

[20] S. K. Lodha, J. C. Renteria, and K. M. Roskin, "Topology preserving compression of 2d vector fields," in *Proceedings Visualization 2000. VIS 2000 (Cat. No. 00CH37145)*. IEEE, 2000, pp. 343–350.

[21] X. Liang, S. Di, F. Cappello, M. Raj, C. Liu, K. Ono, Z. Chen, T. Peterka, and H. Guo, "Toward feature-preserving vector field compression," *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[22] K. Hormann and A. Agathos, "The point in polygon problem for arbitrary polygons," *Computational geometry*, vol. 20, no. 3, pp. 131–144, 2001.

[23] H. Edelsbrunner, *Algorithms in combinatorial geometry*. Springer Science & Business Media, 1987, vol. 10.

[24] F. Aurenhammer and H. Imai, "Geometric relations among voronoi diagrams," *Geometriae Dedicata*, vol. 27, no. 1, pp. 65–75, 1988.

[25] H. Edelsbrunner and E. P. Mücke, "Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms," *ACM Transactions on Graphics (tog)*, vol. 9, no. 1, pp. 66–104, 1990.

[26] H. Bhatia, A. Gyulassy, H. Wang, P.-T. Bremer, and V. Pascucci, "Robust detection of singularities in vector fields," in *Topological Methods in Data Analysis and Visualization III: Theory, Algorithms, and Applications*. Springer, 2014, pp. 3–18.

[27] J. Helman and L. Hesselink, "Representation and display of vector field topology in fluid flow data sets," *Computer*, vol. 22, no. 08, pp. 27–36, 1989.

[28] H. Theisel, C. Rössl, and T. Weinkauf, "Topological representations of vector fields," *Shape analysis and structuring*, pp. 215–240, 2008.

[29] P. Deutsch, "Gzip file format specification version 4.3," 1996.

[30] M. Burtscher and P. Ratanaworabhan, "Fpc: A high-speed compressor for double-precision floating-point data," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 18–31, 2008.

[31] F. Alted, "Blosc compressor," http://blosc.org/, online.

[32] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[33] M. Rabbani, "Jpeg2000: Image compression fundamentals, standards and practice," *Journal of Electronic Imaging*, vol. 11, no. 2, p. 286, 2002.

[34] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[35] Y. Collet, "Zstandard - real-time data compression algorithm," http://facebook.github.io/zstd/, online.

[36] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data-quantitative control of accuracy in derived quantities," *SIAM Journal on Scientific Computing*, vol. 41, no. 4, pp. A2146–A2171, 2019.

[37] P. Jiao, S. Di, H. Guo, K. Zhao, J. Tian, D. Tao, X. Liang, and F. Cappello, "Toward quantity-of-interest preserving lossy compression for scientific data," *Proceedings of the VLDB Endowment*, vol. 16, no. 4, pp. 697–710, 2022.

[38] H. Theisel, C. Rössl, and H.-P. Seidel, "Compression of 2d vector fields under guaranteed topology preservation," in *Computer Graphics Forum*, vol. 22, no. 3. Wiley Online Library, 2003, pp. 333–342.

[39] S. Koch, J. Kasten, A. Wiebel, G. Scheuermann, and M. Hlawitschka, "2d vector field approximation using linear neighborhoods," *The Visual Computer*, vol. 32, pp. 1563–1578, 2016.

[40] X. Liang, K. Zhao, S. Di, S. Li, R. Underwood, A. M. Gok, J. Tian, J. Deng, J. C. Calhoun, D. Tao *et al.*, "Sz3: A modular framework for composing prediction-based error-bounded lossy compressors," *IEEE Transactions on Big Data*, 2022.

[41] X. Liang, S. Di, D. Tao, Z. Chen, and F. Cappello, "An efficient transformation scheme for lossy data compression with point-wise relative error bound," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2018, pp. 179–189.

[42] F. Gabbay and A. Mendelson, *Speculative execution based on value prediction*. Citeseer, 1996.

[43] D. E. Culler, A. Dusseau, S. C. Goldstein, A. Krishnamurthy, S. Lumetta, T. Von Eicken, and K. Yelick, "Parallel programming in split-c," in *Supercomputing'93: Proceedings of the 1993 ACM/IEEE conference on Supercomputing*. IEEE, 1993, pp. 262–273.

[44] P. Fischer, J. Lottes, and H. Tufo, "Nek5000," Argonne National Lab.(ANL), Argonne, IL (United States), Tech. Rep., 2007.

[45] H. I. dataset, http://sciviscontest-staging.ieeevis.org/2004/data.html, online.
[46] K. Kanov, R. Burns, C. Lalescu, and G. Eyink, "The johns hopkins turbulence databases: an open simulation laboratory for turbulence research," *Computing in Science & Engineering*, vol. 17, no. 5, pp. 10–17, 2015.
[47] "Morgan Compute Cluster," https://www.ccs.uky.edu/, 2023.
[48] D. Stalling and H.-C. Hege, "Lic on surfaces," *Texture Synthesis with Line Integral Convolution*, pp. 51–64, 1997.
[49] A. Datta-Gupta and M. J. King, "Streamline simulation: theory and practice," 2007.