# Toward Feature-Preserving 2D and 3D Vector Field Compression

Xin Liang [1*] Hanqi Guo [2†] Sheng Di [2‡] Franck Cappello [2§] Mukund Raj [2¶] Chunhui Liu [3‖]
Kenji Ono [4**] Zizhong Chen [1††] Tom Peterka [2‡‡]

1) Department of Computer Science and Engineering, University of California, Riverside, CA, USA
2) Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL, USA
3) Department of Mathematics, Faculty of Science, Kyoto University, Kyoto, Japan
4) Research Institute for Information Technology, Kyushu University, Fukuoka, Japan



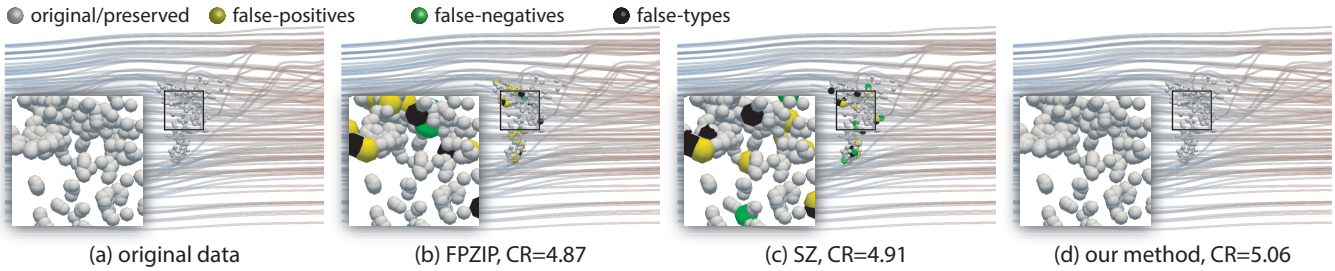



Figure 1: Original LES simulation dataset (a) its compressioned versions (b-d) with different compressors using similar compression ratios (CRs).

**ABSTRACT**

The objective of this work is to develop error-bounded lossy compression methods to preserve topological features in 2D and 3D vector fields. Specifically, we explore the preservation of critical points in piecewise linear vector fields. We define the preservation of critical points as, without any false positive, false negative, or false type change in the decompressed data, (1) keeping each critical point in its original cell and (2) retaining the type of each critical point (e.g., saddle and attracting node). The key to our method is to adapt a vertex-wise error bound for each grid point and to compress input data together with the error bound field using a modified lossy compressor. Our compression algorithm can be also embarrassingly parallelized for large data handling and in situ processing. We benchmark our method by comparing it with existing lossy compressors in terms of false positive/negative/type rates, compression ratio, and various vector field visualizations with several scientific applications.

**Keywords: lossy compression, critical points, vector field visualization.**

## 1 INTRODUCTION

Lossy compression of floating-point data has become a promising technique for data reduction, as the disparity between data generation rate and available I/O bandwidth continues to grow in today's and future supercomputers. Data generated from large-scale ocean, atmosphere, and fluid dynamics simulations can be compressed in situ, and then the decompressed data can be used both in situ and post hoc for data analysis and visualization. In order to preserve scientific insights, error-bounded lossy compressors such as SZ [13], ZFP [14], and FPZIP [15], as opposed to traditional JPEG [23] image compressors, are used to strictly guarantee the desired accuracy while acceptable compression ratio is achieved.

The motivation of this study is to preserve the accuracy of features extracted in error-bounded lossy compressed data. We examine critical points in 2D and 3D vector fields as an example of a feature. Critical points—locations where the vector field vanishes—are important because they are the key constituents of vector field topology and thus essentially determine the characteristics of flow visualizations based on geometry [17], texture [9], and topology [5,10]. The extraction of critical points leads to both locations and types (e.g., sources, sinks, and saddles), and both properties must be preserved in order to deliver authentic insights into the decompressed data.

With today's lossy compressors, failure to preserve critical points can result in false positives (FPs), false negatives (FNs), and false types (FTs). A false positive happens if a critical point is localized in the decompressed data but such a point does not exist in the same vicinity of the original data. A false negative means that the critical point is missed in the decompressed data. A false type occurs when the critical point type does not match in the original and decompressed data. For example, an attracting critical point may turn into a repelling one during compression/decompression.

In this work, we aim at improving error-bounded lossy compressors to preserve critical points in 2D and 3D piecewise linear vector fields. We define the preservation of critical points as, without any false positive, false negative, or type change in the decompressed data, (1) keeping each critical point in its original cell and (2) retaining the type of each critical point. The key to our method is to adapt a vertex-wise error bound for each grid point and to compress input data together with the vertex-wise error bounds using SZ [13], which is a prediction-based lossy compressor. We develop both decoupled and coupled compression pipelines. The decoupled approach estimates vertex-wise error bounds for all vertices and then compresses the vector field based on the error bounds. The coupled approach estimates the error bound and compresses the data on the fly during the lossy compression. The coupled method delivers higher compression ratios but is more computationally expensive than the decoupled method. We demonstrate that our methods

*e-mail: xlian007@ucr.edu
†e-mail: hguo@anl.gov
‡e-mail: sdi1@anl.gov
§e-mail: cappello@mcs.anl.gov
¶e-mail: mraj@anl.gov
‖e-mail: chunhui.liu@math.kyoto-u.ac.jp
**e-mail: keno@cc.kyushu-u.ac.jp
††e-mail: chen@cs.ucr.edu
‡‡e-mail: tpeterka@mcs.anl.gov

2020 IEEE Pacific Visualization Symposium (PacificVis)
978-1-7281-5697-2/20/$31.00 ©2020 IEEE

outperform existing lossy compressors in terms of FP, FN, and FT rates and compression ratio with several scientific applications.

Existing efforts to compress 2D vector fields based on topology [16,21] (1) are nontrivial to generalize to 3D, (2) do not guarantee local error bounds and may lead to large distortions, (3) require undetermined iterations to converge, or (4) are difficult to parallelize. More discussion on the differences between our method and previous efforts is in the following section. The contributions of this paper can be summarized as follows:

- Theoretical framework to preserve critical points in piecewise linear 2D/3D vector fields based on the vertex-wise error bound derivation
- Two feature-preserving compressor designs that enforce vertex-wise error bounds: a decoupled method optimized for speed and a coupled method optimized for storage

## 2 Background

We review the related work on error-bounded lossy compression and vector field compression and then formalize the critical point extraction problem.

### 2.1 Error-Bounded Lossy Compression

Data compression can be either lossless or lossy, and lossy compression can be further categorized into *non-error-bounded lossy* and *error-bounded lossy* methods. This study focuses on error-bounded lossy compressors, which guarantee local error within designated error bounds. Error-bounded lossy compressors usually deliver higher compression ratios than the lossless compressors such as FPC [3], while are more precise than non-error-bounded lossy compressors such as JPEG [23]. We refer to the literature [11, 18] for comprehensive reviews of scientific data compression; hence we focus mainly on error-bounded lossy compressors.

Error-bounded lossy compression can be prediction- or transformation-based. Prediction-based error-bounded compressors include FPZIP [15] and SZ [13]. FPZIP uses a Lorenzo predictor [7] with integer mapping on both the predicted and actual data for avoiding underflow, followed by an arithmetic encoding on the prediction residuals. SZ is a multialgorithm compressor with blockwise selection on the best-fit predictor, including both Lorenzo and regression-based predictors. Different from the arithmetic encoding used in FPZIP, SZ performs linear quantization on the prediction residuals and encodes the quantization integers by customized Huffman encoding and lossless compressors such as GZIP [1] and ZSTD [2]. An example of transformation-based error-bounded compressors is ZFP [14]. ZFP first performs the exponent alignment and fixed-precision points conversion and applies a fine-tuned orthogonal/inorthogonal transformation for each block and then encodes the coefficients for compression.

While a user-given single error bound is mandatory for all existing error-bounded lossy compressors, the key difference of our study is that we derive vertex-wise error bounds. Specifically, we construct vertex-wise error bounds that adapt the numerical tolerance in order to preserve features in the decompressed data. Formally, the relative error between the exact value $d \in \mathbb{R}_{\neq 0}$ and its approximation $d' \in \mathbb{R}$ is defined as

$$\delta(d,d') \stackrel{\text{def}}{=} |d-d'|/|d|. \tag{1}$$

The relative error bound $\Delta(\cdot)$ is an arbitrary value such that no relative error exceeds the error bound.[1] We generalize the notation of $\delta$ to represent the *maximal relative error* between the input data $\mathbf{d}$ and its approximation $\mathbf{d}'$ as $\delta(\mathbf{d},\mathbf{d}') \stackrel{\text{def}}{=} \max_i \delta(d_i,d_i')$, where $\mathbf{d}$ and $\mathbf{d}'$ are vectors of arbitrary dimensions in $\mathbb{R}$ and $i$ is the linear index of each element in $\mathbf{d}$ and $\mathbf{d}'$. We also use the notation $\Delta(\mathbf{d})$ to represent the relative error bound of each element in $\mathbf{d}$, and we use $||\Delta(\mathbf{d})||_{\max}$ as the maximal element of $\Delta(\mathbf{d})$.

[1] In this work, we limit relative error bounds to $[0,1]$, and in this case, $d'$ and $d$ always have the same sign.

### 2.2 Vector Field Compression

Vector field compression has been studied to preserve 2D topological features. Lodha et al. [16] used an iterative clustering method [20] to simplify and compress 2D vector fields. The iteration stops when all critical points remain identical to the original topology and the designated local error bound is met. Compared with the method, our method is noniterative, has fixed time complexity, and works for both 2D and 3D vector fields in regular and unstructured meshes. Theisel et al. [21] iteratively collapsed edges in the 2D mesh in order to guarantee topology preservation, but without local error control. Dey et al. [4] proposed a Delaunay simplification for the vector field based on error-bounded edge collapsing, but the simplification did not explicitly preserves topological features. Koch et al. [8] presented a segmentation-based compression approach based on region-wise linear approximation; a simplified mesh grid can be generated by iteratively adding new segmentations and testing topological equivalence.

To the best of our knowledge, this study is the first attempt to tailor error-bounded lossy compressors to preserve features in 2D and 3D vector field data. The following are the key differences between our method and the state of the art. First, existing error-bounded lossy compressors are not aware of important vector field features, whereas our method preserves critical points. By compressing individual vector components, existing compressors produce FP, FN, and FT critical points in the decompressed data. Second, existing topology-based vector field compression does not bound local error, whereas our method enforces local error bounds. For example, the iterative edge collapsing approach [21] does not control local error, which may lead to large distortions in the decompressed data. Third, existing vector field compression algorithms are challenging to generalize to 3D, whereas our technique applies to both 2D and 3D vector fields. Because 3D vector field topology is much more complicated than 2D, the generalization of clustering- [16], mesh simplification- [4,21], and segmentation- [8] based algorithms can be convoluted and computationally expensive. Fourth, most existing vector field compressors [4,8,16,21] are iterative, whereas our method compresses a vector field in a single pass with fixed-time complexity. In addition, our method can be easily parallelized and thus can (de)compress data for high-performance data storage, analysis, and visualization.

### 2.3 Critical Points in Piecewise Linear Vector Field

A critical point is defined as the location where the vector field vanishes. Formally, let $n_d$ be the dimensionality of the data and the vector field be $\mathbf{v}: \mathbb{R}^{n_d} \to \mathbb{R}^{n_d}$; the vector value $\mathbf{v}$ at a critical point $\mathbf{x}_c \in \mathbb{R}^{n_d}$ must be $\mathbf{0}$. In this study, we focus on non-degenerative (or first-order) critical points, where the determinant of the vector gradient tensor (Jacobian) of the vector field $|\mathbf{J}_\mathbf{v}(\mathbf{x}_c)| \neq 0$. The vector field $\mathbf{v}$ is defined on a simplicial (triangular or tetrahedral) mesh; the interpolation scheme for each cell is linear. Without loss of generality, we use 2D cases for illustration and derivation; the same techniques apply to 3D unless otherwise noted. We use the symbols of Fig. 2 in individual cells.

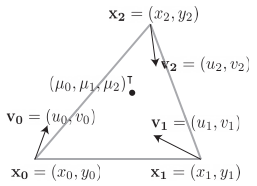



Figure 2: Symbols in individual 2D simplicial cells.

The extraction of critical points involves finding zero points in each linearly interpolated cell:

$$\mathbf{V}\cdot\boldsymbol{\mu} = \begin{bmatrix} u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \end{bmatrix} \begin{bmatrix} \mu_0 \\ \mu_1 \\ \mu_2 \end{bmatrix} = \mathbf{0} \text{ and } \mu_0+\mu_1+\mu_2 = 1, \tag{2}$$

where $\boldsymbol{\mu}$ is defined as the (normalized) barycentric coordinates of the critical point and $\mathbf{V}$ is a $2\times 3$ matrix consisting of all vector values for the vertices. If the condition $0 \le \mu_k \le 1$ holds for all $k \in \{0,1,2\}$, the critical point resides inside of the cell; otherwise the cell contains no critical point.

Critical points can be categorized into various types based on the signs of eigenvalues of the Jacobian $\mathbf{J}_\mathbf{v}(\mathbf{x}_c)$ [6, 22]. In general, negative and positive eigenvalues indicate attracting and repelling, respectively; eigenvalues with imaginary parts imply circulation behavior. The critical point can be determined analytically based on $\mathbf{J}$, because the *characteristic polynomial* $|\lambda\mathbf{I}-\mathbf{J}| = \lambda^2 - \mathrm{tr}(\mathbf{J})\lambda + |\mathbf{J}|$ is quadratic and thus has a closed-form solution, where $\mathbf{I}$ is the identity matrix and $\mathrm{tr}(\cdot)$ is the trace of a matrix. In 3D, the characteristic polynomial can be determined in closed form as well.

## 3 Problem Statement

We formulate the feature-preserving compression problem for the non-degenerative critical point extraction in piecewise linear vector fields. In general, original and decompressed 2D/3D vector fields $\mathbf{v}$ and $\mathbf{v}'$ respectively are defined on the same simplicial (triangular or tetrahedral) mesh; the vector values on the $i$th vertex are $\mathbf{v}_i$ and $\mathbf{v}'_i$, respectively; and the relative error bound of $\mathbf{v}_i$ is $\xi_i$. For each cell in the mesh, assuming the barycentric coordinates of the critical point in the original and decompressed data are $(\mu_0,\mu_1,\mu_2,[\mu_3])^\mathsf{T}$ and $(\mu'_0,\mu'_1,\mu'_2,[\mu'_3])^\mathsf{T}$, respectively, and the Jacobian eigenvalues are $(\lambda_0,\lambda_1,[\lambda_2])^\mathsf{T}$ and $(\lambda'_0,\lambda'_1,[\lambda'_2])^\mathsf{T}$, respectively, the objective of this study is to guarantee the following three conditions by finding proper relative error bounds $\xi_i$:

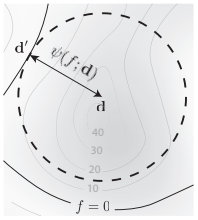
- **Non-FN**: If $\mu_k \in [0,1]$ holds for all $k$, $\mu'_k \in [0,1]$ holds for all $k$ as well;
- **Non-FP**: If there exists $k$ such that $\mu_k \notin [0,1]$, there exists $k'$ such that $\mu_{k'} \notin [0,1]$;
- **Non-FT**: The "non-FN" condition is met, and there exists a one-to-one mapping between $l$ and $l'$ such that $\mathrm{sgn}(\mathrm{Re}(\lambda_l)) = \mathrm{sgn}(\mathrm{Re}(\lambda_{l'}))$ and $\mathrm{sgn}(\mathrm{Im}(\lambda_l)) = \mathrm{sgn}(\mathrm{Im}(\lambda'_{l'}))$.

Here, $\mathrm{sgn}(\cdot)$ is the sign function; $\mathrm{Re}(\cdot)$ and $\mathrm{Im}(\cdot)$ are the real and imaginary part operators, respectively; and $k,k' \in \{0,1,\ldots,n_d\}$, $l,l' \in \{0,1,\ldots,n_d-1\}$.

To derive sufficient error bounds for these conditions, we introduce the *sign-preserving error function* (SPEF) of any given scalar function. Formally, denoting the ball $B(\mathbf{d},\gamma) = \{\mathbf{d}' \mid \delta(\mathbf{d},\mathbf{d}') \le \gamma\}, \gamma \in [0,1)$, we define the SPEF of any given scalar function $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ as

$$\psi(f;\mathbf{d}) \stackrel{\mathrm{def}}{=} \sup\{\gamma \mid f(\mathbf{d})f(\mathbf{d}') \ge 0, \forall \mathbf{d}' \in B(\mathbf{d},\gamma)\}. \tag{3}$$

As illustrated in the right figure, $\psi(f;\mathbf{d}) \in [0,1]$ is defined as the supremum of $\delta(\mathbf{d},\mathbf{d}')$ such that $f(\mathbf{d})$ and $f(\mathbf{d}')$ keep the same sign. In general, finding the closed form of $\psi$ functions is challenging. Thus we instead attempt to find a relaxed error bound $\Delta(\mathbf{d})$ that is less than or equal to $\psi(f;\mathbf{d})$, in order to preserve the sign of $f(\mathbf{d}')$.




Based on the definition of SPEF, a sufficient condition for non-FN, non-FP, and non-FT in 2D cases is

$$||\Delta(\mathbf{V})|| \le \psi(m_0,m_1,m_2;\mathbf{V}), \tag{4}$$

$$||\Delta(\mathbf{V})|| \le \max_{k\in\{k|\mu_k\notin[0,1]\}} \min\left(\psi(m_k), \psi(\sum_{k'\neq k} m_{k'};\mathbf{V})\right), \text{ and} \tag{5}$$

$$||\Delta(\mathbf{V})|| \le \begin{cases} \psi(|\mathbf{J}|;\mathbf{V}) & |\mathbf{J}| \le 0 \\ \psi(\mathrm{tr}^2(\mathbf{J}) - 4|\mathbf{J}|, \mathrm{tr}(\mathbf{J});\mathbf{V}) & |\mathbf{J}| > 0 \end{cases}, \tag{6}$$

**Algorithm 1** Error bound computation for critical point preservation for a triangular cell in either decoupled or coupled compression

**Input**: values $\mathbf{V} = \left(\begin{smallmatrix} u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \end{smallmatrix}\right)$ and coordinates $\mathbf{X} = \left(\begin{smallmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \end{smallmatrix}\right)$ of vertices
**Output**: maximal error bound $e_r \in [0,1)$ for $\mathbf{V}$

```
function EB_{DECOUPLED|COUPLED}(V, X)
    A ← [u0 u1 u2; v0 v1 v2; 1 1 1], b ← (0, 0, 1)^T
    if |A| = 0 then                                  ▷ check if the system is singular
        return 0;                                    ▷ use lossless compression for this cell
    else
        if μ = A^{-1}b ∈ [0,1]^3 then                ▷ check if critical point exists
            return min(eb_FN_{decoupled|coupled}(V),
                       eb_FT_{decoupled|coupled}(V, X))   ▷ Appendices B,C
        else
            return eb_FP_{decoupled|coupled}(V)      ▷ Appendices B,C
        end if
    end if
end function
```

respectively, where $\mathbf{V}$ is the 2x3 matrix defined in Eq. (2); $m_0$, $m_1$, and $m_2$ are *auxilary barycentrice coordinates* defined in Appendix A; and $k \in \{0,1,2\}$. Equations (4) and (5) can be directly generalized to 3D cases, and we use $||\Delta(\mathbf{V})|| = 0$ to guarantee the 3D non-FT condition. Proofs and error bound derivations are detailed in Appendices A, B, and C. The following sections focus mainly on compression algorithms.

## 4 Decoupled and Coupled Feature-Preserving Compression

This section presents both decoupled and coupled approaches to compress the vector field while preserving critical points. The decoupled method compresses data in a pipelined manner (Fig. 3 and Section 4.1), while the coupled method compresses data in a single pass (Fig. 4 and Section 4.2). Comparison between the two methods is in Section 4.3. We use $n_v$ and $n_c$ to represent the number of vertices and cells, respectively, in the descriptions.

### 4.1 Decoupled Compression

Figure 3 illustrates the pipeline of our decoupled compression method, which consists of cell-wise error bound computation, vertex-wise error bound aggregation, and vertex-wise compression.

**Cell-wise error bound computation** The pseudocode for cell-wise error bound computation is in Algorithm 1. The algorithm takes in the values and coordinates of the vertices in the given cell and returns a sufficient error bound to keep the critical point information in the cell. If the underlying linear system is deficient, we use zero as the error bound for the cell, and the vector values will be compressed losslessly in the compression stage. Otherwise, we check whether the critical point exists in the cell. If the critical point exists, we return a sufficient error bound to avoid FN and FT; otherwise, we return a sufficient error bound to avoid FP. Notice that the procedure for error bound computation is similar in the coupled compression method, and the mathematical derivation of sufficient error bounds is detailed in Appendices B and C.

**Vertex-wise error bound computation** We calculate vertex-wise error bounds based on cell-wise error bounds that are computed in the previous step. As shown in Algorithm 2, we iterate over each cell and compute the cell-wise error bound $\eta$ with Algorithm 1. For each vertex of the cell, we assign the minimum of the current error bound and $\eta$ as the updated error bound for the vertex.

**Vertex-wise compression** We use the vertex-wise error bound to guide the error-bounded lossy compression; the pseudocode is in Algorithm 3. The `quant()` in the pseudocode takes the derived error bound $\xi_i$ as input and returns the quantized value $\widehat{\xi}_i$, in order to reduce the storage of vertex-wise error bounds in the compres-

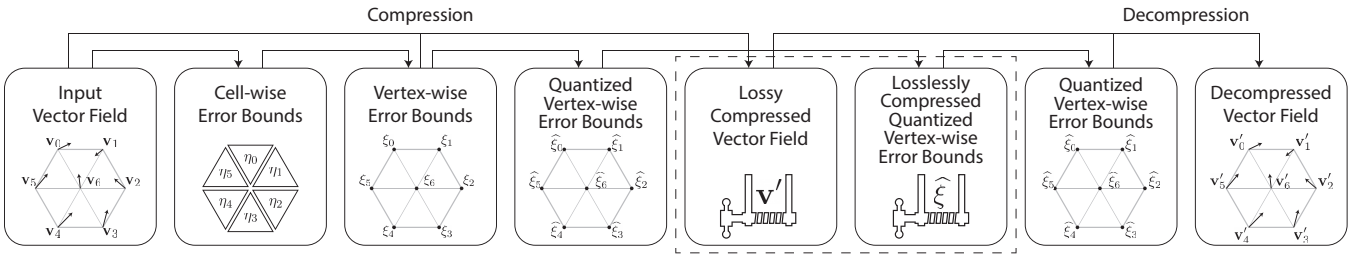



Figure 3: Decoupled compression pipeline for critical-point-preserving compression.

**Algorithm 2** Vertex-wise error bound computation based on cell-wise error bounds in decoupled compression.

**Input**: values $\{\mathbf{v}_i\}$ and coordinates $\{\mathbf{x}_i\}$ of all vertices
**Output**: vertex-wise error bounds $\{\xi_i\}$

```
function DECOUPLED_ERROR_BOUND_MAP_DERIVATION({v_i}, {x_i})
    {ξ_i} ← {1}                                   ▷ initialize each error bounds with 1
    for j ← 0 to n_c − 1 do                       ▷ iterate cells
        {i_0, i_1, i_2} ← cell_vertices(j)
        η ← eb_decoupled((v_{i_0}, v_{i_1}, v_{i_2}), (x_{i_0}, x_{i_1}, x_{i_2}))   ▷ see Alg. 1
        for i ∈ {i_0, i_1, i_2} do
            ξ_i ← min(ξ_i, η)                     ▷ vertex-wise error based on cell-wise error
        end for
    end for
    return {ξ_i}
end function
```

**Algorithm 3** Decoupled feature-preserving compression

**Input**: values $\{\mathbf{v_i}\}$, coordinates $\{\mathbf{x_i}\}$, and error bounds $\{\xi_i\}$ for all vertices
**Output**: compressed byte stream

```
buffer={∅}                                        ▷ buffer for vector field compression
for i ← 0 to n_v − 1 do
    ξ̂_i ← quant(ξ_i)                              ▷ quantize error bound for vertex i
    bytes ← lossy_compress(v_i, ξ̂_i)              ▷ compress v_i lossily while
guarantee the error bound ξ̂_i
    buffer.append(bytes)
end for
return compress_losslessly(buffer, {ξ̂_i})
```

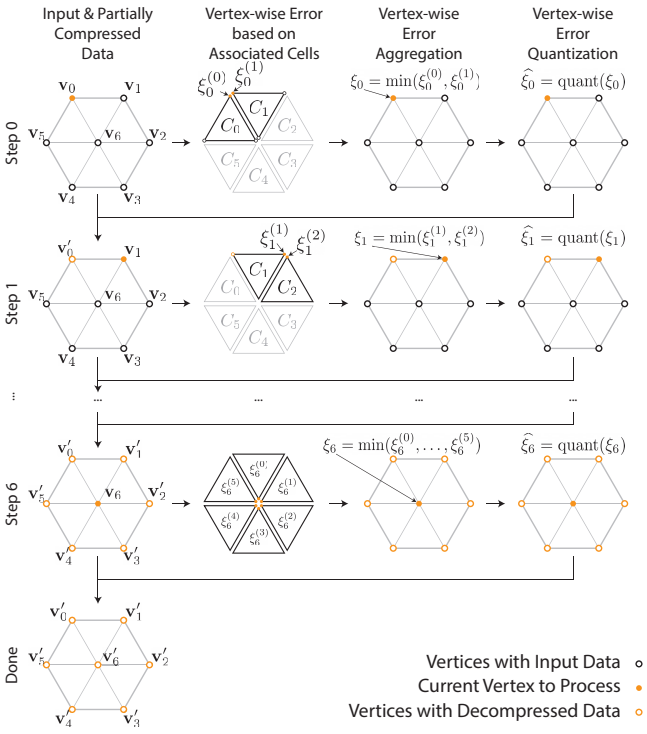



Figure 4: Illustration of coupled compression algorithm.

sion. The details on the quantization are discussed in Section 5.2. In the algorithm, we first internalize a byte buffer to stage quantized values of the vector field data. We then iterate each vertex by quantizing the error bound and compressing the data values. The `lossy_compress()` function lossily compresses the vector data by guaranteeing the given error bound and returns the compressed data in bytes. In our implementation, we use the prediction and quantization scheme in SZ for compression; more details are in Section 5. In the last step, the byte buffer and the quantized vertex-wise error bounds are losslessly encoded and compressed.

### 4.2 Coupled Compression

The coupled compression scheme couples error bound estimation and compression on the fly in the iteration of each vertex. The key to the coupled compression is to incorporate decompressed values, which are available during the process, in order to obtain more relaxed error bounds than that of decoupled compression.

As detailed in Algorithm 4, for each vertex $i$, we first compute the vertex-wise error bound $\xi_i^{(j)}$ based on each of its adjacent cells $j$. We then aggregate the vertex-wise error bound to $\xi_i$, quantize $\xi_i$ to $\widehat{\xi}_i$, and perform lossy compression on vector field data $\mathbf{v}_i$. The output bytes are appended to a preallocated buffer for further compression. The `quant()` and `lossy_compress()` functions are the same as those in the decoupled compression. Note that the coupled compression needs to use the `decode()` function to calculate the decompressed data $\mathbf{v}'_i$ on the fly and to overwrite the original $\mathbf{v}_i$.

Algorithm 4 can be proved by mathematical induction. We would like to show that in the $i$th iteration, if the dataset $\{\mathbf{v}'_0,\ldots,\mathbf{v}'_{i-1},\mathbf{v}_i,\ldots,\mathbf{v}_{n_v-1}\}$ preserves all critical points, $\{\mathbf{v}'_0,\ldots,\mathbf{v}'_{i-1},\mathbf{v}'_i,\ldots,\mathbf{v}_{n_v-1}\}$ preserves all critical points as well. Actually, we can obtain an error bound such that all adjacent cells of $i$ preserve critical points; critical points in nonadjacent cells remain unchanged. The base condition ($i=0$) holds as well because no decompressed data are available. This completes the proof.

Figure 4 illustrates the coupled compression on a tiny example ($n_v=7$ and $n_c=6$). At step 0, the error bound of vertex 0 is obtained per adjacent cell ($C_0$ and $C_1$). Then the error bound $\xi_0$ can be calculated by aggregation. The decompressed vector value is then calculated on the fly as $\mathbf{v}'_0$, which will be used in the error bound computation of $\mathbf{v}_1$, $\mathbf{v}_5$, and $\mathbf{v}_6$ in the next few iterations. Notice that we must use $\mathbf{v}'_0$ instead of $\mathbf{v}_0$ for the error bound derivation; otherwise, the use of the original value $\mathbf{v}_0$ violates the proof above.

**Algorithm 4** Coupled feature-preserving lossy compression

**Input**: values $\{\mathbf{v_i}\}$ and coordinates $\{\mathbf{x_i}\}$ of all vertices
**Output**: compressed byte stream

```
buffer={∅}                                          ▷ integer buffer for compression
for i ← 0 to n_v − 1 do                             ▷ iterate vertices
    for j ∈ vertex_cells(i) do                      ▷ iterate cells connected to vertex i
        {i_0, i_1, i_2} ← cell_vertices(j)          ▷ vertices of cell j
        ξ_i^(j) ← eb_coupled((v_{i_0}, v_{i_1}, v_{i_2}), (x_{i_0}, x_{i_1}, x_{i_2}))   ▷ see Alg. 1
    end for
    ξ_i ← min_j ξ_i^(j)                             ▷ aggregate error bound for vertex i
    ξ̂_i ← quant(ξ_i)                                ▷ quantize error bound of vertex i
    bytes ← lossy_compress(v_i, ξ̂_i)                ▷ quantize vector values with SZ
    v'_i ← decode(bytes, ξ̂_i)                       ▷ calculate decompressed value v'_i on-the-fly
    v_i ← v'_i                                      ▷ replace the input value with the decompressed value
    buffer.append(d)
end for
return compress_losslessly(buffer, {ξ̂_i})
```

### 4.3 Comparison between Decoupled and Coupled Compression

We compare our decoupled and coupled compression with regard to both complexity and compression ratio.

**Space and time complexities** The space complexities of both methods are identical ($O(n_v)$); the time complexity of the decoupled and coupled algorithm is $O(n_c)$ and $O(\sum_i \text{card}(\text{adj\_cells}(i)))$, respectively, where $\text{card}(\cdot)$ is the number of elements in a set. The $O(n_c)$ complexity of the decoupled compression algorithm is based on the cell-wise iteration in Algorithm 2. The complexity of the coupled compression algorithm is $O(n_v \cdot (\sum_i \text{card}(\text{adj\_cells}(i))/n_v))$, where the fraction is the average number of adjacent cells for each vertex. This expression can be reduced to $O(\sum_i \text{card}(\text{adj\_cells}(i)))$.

**Error bounds and compression ratio** Decoupled compression has a lower compression ratio than coupled compression does, because the coupled method delivers more relaxed error bounds than the decoupled method does. The reason is that the decoupled method attempts to control errors on three vertices simultaneously, while the coupled method achieves the same goal by controlling the error on one single vertex. We refer to Appendices B and C for more detail on error bound derivations.

## 5 Lossy Compressor Customization

We tailor the SZ lossy compressor to guarantee vertex-wise error bounds, and we quantify the vertex-wise bounds for efficient storage.

### 5.1 Baseline Compressor Selection

In general, any prediction-based compressor such as FPZIP and SZ can be customized for feature-preserving compression; we use SZ as an example in this study. Transform-based compressors may be used if the error can be bounded for individual vertices.

We review two important SZ features that are used in this study. First, SZ strictly guarantees the error bound. SZ uses linear quantization [19] on the difference between the original data and predicted value from a Lorenzo predictor for a strict absolute error bound guarantee. For relative error bound, SZ transforms it to absolute error bound by multiplying it with the original value and applies the linear quantization. Second, SZ uses a logarithmic transformation in its recent design [12] to transform a relative error compression problem in the original domain to an absolute error compression problem in the logarithmic domain. Specifically, SZ records the signs of the original data and transforms the original data to the logarithm of their absolute value. The transformed data there are compressed with the absolute error bound $\log(1+e_r)$ where $e_r$ is the relative error bound. During the decompression, the data are transformed back to the original data by the exponential function and the corresponding signs.

### 5.2 Efficient Vertex-wise Error Bounds Storage

We incorporate the following three optimizations to SZ and use these optimized functions in Algorithms 3 and 4.

**Logarithm-based error bound transform** We extend the logarithmic transformation for relative error bound compression in SZ to the vector field compression, in order to store one transformed error bound instead of $n_d$ absolute error bounds for different components. Note that we have to store the original data when the error bound is 0 or the logarithmic data need to be losslessly recorded, because the round-off error in the logarithmic and exponential function may lead to unexpected perturbations in the decompressed data.

**Exponential-scale error bound quantization** We use quantization to further compress the transformed relative vertex-wise error bounds. Instead of linear quantization, we use an exponential quantization for more aggressive size reduction. Specifically, we quantize the each error bound $e_r$ larger than $\varepsilon_0$ ($e_r$ less than $\varepsilon_0$ is quantized to 0) to an integer $q = \lfloor \log_b \frac{e_r}{\varepsilon_0} \rfloor$, where $b$ and $\varepsilon_0$ are two tuning parameters, and we apply Huffman encoding to the quantized integers. Unlike linear quantization, which quantizes $e_r$ to $\lfloor \frac{e_r}{\varepsilon_0} \rfloor$, exponential quantization has fewer values for the quantized integers, leading to higher compression ratios on the vertex-wise error bounds. In our experiments, we set $\varepsilon_0$ to machine precision and $b$ to 2, which leads to satisfactory performance.

**Global error bound restriction** We also use a global error bound as a strict restriction, such that any derived error bound greater than the threshold will be set to the threshold. The reason is that the precision of the Lorenzo predictor relies heavily on the precision of decompressed data, especially when the error bound of the current data is small compared with its neighbors. Limiting the error bound mitigates such problems and also decreases the range of the quantization index, reducing the size of the vertex-wise error bounds. However, this could reduce compression ratio, and the threshold needs to be tuned to achieve the best trade-off. In our implementation, we empirically set the global error bound to 0.1 for 1D and 2D data and to 0.05 for 3D data.

## 6 Evaluation

In this section, we evaluate our work using the three scientific datasets listed in Table 1, and we compare the results with three state-of-the-art error-bounded lossy compressors—FPZIP [15], SZ [13], and ZFP [14]—using different error bound configurations. We show both quantitative results, involving the exact number of FPs, FNs, and FTs reported by the critical point detection algorithm, and qualitative results, displaying the consequent visual difference in the local area of the changed critical points. All compressors we benchmarked are the latest releases as of September 12, 2019.

We use three datasets from ocean simulation, Nek5000 fluid simulation, and large eddy simulation (LES). Ocean and Nek5000 data are available in 2D and 3D regular grids, respectively, and we tessellate each 2D/3D cube into two triangles or six tetrahedra to construct piecewise linear vector fields. The LES dataset includes 71M tetrahedral cells and 110M wedges with 67.8M nodes. We consider only the tetrahedral cells for our test. All the experiments are conducted on an Intel Broadwell node with two Intel Xeon E5-2695 v4 processors and 128 GB of memory.

Table 1: Datasets for benchmarking

| Dataset | Size | $n_d$ | $n_v$ | $n_c$ |
|---|---|---|---|---|
| Ocean | 98.88 MB | 2 | $3600 \times 2400$ | $3599 \times 2399 \times 2$ |
| Nek5000 | 1.536 GB | 3 | $512^3$ | $511^3 \times 6$ |
| LES | 145.8 MB | 3 | 12.74 M | 71.19 M |

Table 2: Benchmark of (lossy) compressors on 2D ocean data: $e_a$, $e_r$ are the global absolute and relative error bound used by compressors, respectively; $CR_u$, $CR_v$, and $CR_{all}$ are the compression ratio (input size over output size) of $\mathbf{u}$, $\mathbf{v}$, and all components, respectively; $S_c$ and $S_d$ are the speed for compression and decompression, respectively; #TP is the number of true-positive (preserved) critical points; #FP, #FN, and #FT are the number of false critical points.

| Compressor | Setting | $e_a$ | $e_r$ | $CR_u$ | $CR_v$ | $CR_{all}$ | $S_c$ (MB/s) | $S_d$ (MB/s) | #TP | #FP | #FN | #FT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Our method | decoupled | - | - | - | - | 7.54× | 32.28 | 77.15 | 20,929 | 0 | 0 | 0 |
| Our method | coupled | - | - | - | - | 11.73× | 27.43 | 60.81 | 20,929 | 0 | 0 | 0 |
| FPZIP | -P 13 | - | 0.0625 | 11.48× | 11.00× | 11.23× | 122.23 | 102.86 | 20,416 | 310 | 244 | 269 |
| SZ | -A 0.05 | 0.05 | - | 11.19× | 11.50× | 11.35× | 131.07 | 214.97 | 18,350 | 43,880 | 1,913 | 666 |
| SZ | -P 0.07 | - | 0.07 | 11.34× | 11.06× | 11.20× | 90.53 | 149.33 | 19,680 | 630 | 601 | 648 |
| ZFP | -A 0.5 | 0.5 | - | 10.06× | 10.73× | 10.39× | 223.51 | 366.85 | 17,816 | 46,364 | 2,455 | 658 |
| ZFP | -P 10 | - | 0.125 | 11.11× | 11.18× | 11.14× | 228.04 | 359.04 | 18,685 | 49,207 | 1,593 | 651 |

### 6.1 Results with 2D Ocean Data

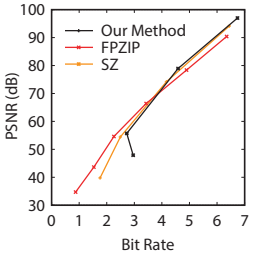
We first compare the number of FPs, FNs, and FTs by tuning all the lossy compressors to a similar compression ratio ($\sim 11.5\times$), as shown in Table 2. From this table, we can see that both of our approaches can be free of FPs, FNs, and FTs at high compression ratio, successfully preserving all the features, whereas existing general-purpose error-bounded lossy compressors have more or less altered critical points in their decompressed data. We also study the performance of both compression and decompression for all the compressors. Our methods are slower than existing compressors in terms of compression performance because of the higher time complexity—$n_c \approx 2n_v$ for the decoupled approach and $\sum_i \text{card}(\text{adj\_cells}(i)) \approx 6n_v$ for the coupled approach.

Table 3: Compression ratio for lossy compressors to avoid FP/FN/FT in 2D ocean data.

| Compressor | Settings | $e_a$ | $e_r$ | $CR_u$ | $CR_v$ | $CR_{all}$ |
|---|---|---|---|---|---|---|
| GZIP | -1 | 0 | 0 | 1.58× | 1.58× | 1.58× |
| FPZIP | -P 25 | - | $2^{-17}$ | 2.86× | 2.82× | 2.84× |
| SZ | -A $10^{-10}$ | $10^{-10}$ | - | 1.60× | 1.59× | 1.59× |
| SZ | -P $10^{-5}$ | - | $10^{-5}$ | 2.73× | 2.63× | 2.68× |
| ZFP | lossless | 0 | 0 | 1.90× | 1.88× | 1.89× |
| Our method | decoupled | - | - | - | - | 7.54× |
| Our method | coupled | - | - | - | - | 11.73× |

We also compare the compression ratios of different compressors by tuning compressors to be free of FPs, FNs, and FTs. To do so, we manually tune the error bound settings and detect critical points until no FPs, FNs, and FTs are present. The results are displayed in Table 3. Under such circumstances, existing compressors have to perform near lossless compression with compression ratios less than 3. In contrast, our decoupled and coupled approaches can lead to compression ratios of 7.54× and 11.73×, respectively, while automatically preserving critical points without manual intervention.

We then compare the global compression fidelity of SZ, FPZIP, and our method with the rate-distortion plot in the right figure. The plot is generated by first compressing data with different global error bounds, and then computing and plotting the peak signal-to-noise ratio (PSNR) and bit rate (average bits per compressed data sample). We can see that our method has comparable rate-distortion trends to those of SZ and FPZIP when the bit rate is high, because our derived error bounds are usually smaller than the global error bound. However, the bit rate of our method stops decreasing after it reaches 2.7, because the compression ratio is dominated by our derived error bounds instead of the global one.




We present the qualitative results by visualizing both the global view of critical point distribution and local topology. The global critical point distribution (including preserved, FPs, FNs, and FTs) of the different compressors is illustrated in Fig. 5. From this figure, we can see that all of the existing compressors have FPs, FNs, and FTs across the global region, whereas our methods preserve all the critical points. We also show the derived error bound and the final relative error of the two approaches in Figs. 5(c), (d), (g), and (h), with the range of $[0, 0.1]$. These figures indicate that the coupled approach indeed allows for higher error bound, leading to higher higher compression ratio than the decoupled approach can.

In Fig. 6, we further zoom into local regions to visualize the impact of critical point changes on local topology. We compare only with FPZIP for demonstration purpose, because it has the smallest number of FPs, FNs, and FTs. Specifically, we show the difference in the line integral convolution (LIC) of the local region near the critical points for FPs and FNs in case I and case II, where the gray spheres indicate the original/preserved critical points and the yellow spheres represent FPs. For example, the LIC patterns of the saddle and the focus in both the original data and decompressed data using our coupled approach can be observed in case I, while the patterns of the decompressed data of FPZIP shows no critical point. We also trace a streamline to show the impact of type change in case III, where an attracting focus in the original data is turned into a repelling focus in the decompressed data of FPZIP.

### 6.2 Results with Nek5000 Data

The quantitative results of the different lossy compressors are displayed in Table 4. We skip the absolute error bound mode because of its inefficiency in preserving critical point in previous experiments. Again, the other lossy compressors have critical point changes to achieve a similar compression ratio ($\sim 7.5\times$) to our coupled approach, which preserves all the critical points. Although GZIP can also preserve all the critical points, the compression ratio is only around 1.1×. Similarly, the compression performance of our methods is hindered mainly by the $n_c \approx 6n_v$ and $\sum_i \text{card}(\text{adj\_cells}(i)) \approx 24n_v$ complexity in the 3D regular grid.

We present the qualitative results by visualizing the critical point distribution and local topology in the Nek5000 data. The global views are displayed in Fig. 7, with traced streamlines from the same source. We also show the derived error bound and the resulting real errors in the coupled approach. We see in the figure that the streamlines generated from original data and decompressed data of different lossy compressors are almost the same. However, the changed critcal points in the decompressed data of FPZIP result in streamline changes in local regions, as shown in Fig. 8. For example, the changed critical point type in case III leads to an attracting effect instead of the repelling effect in the original data.

### 6.3 Results with LES Data

The unstructured LES data yield similar results, where our coupled approach preserves all the critical points while the other compressors introduce FPs, FNs, and FTs at the same compression ratio. In this case, the compression performance of our method is affected by both the 22.4× complexity ($\sum_i \text{card}(\text{adj\_cells}(i)) \approx 22.4n_v$) and the construction of unstructure grid.

Figure 1 visualizes the global critical point distribution with traced streamlines as context. Again, we see little change in all the global streamlines, but the other lossy compressors lead to FPs, FNs, and FTs in different locations.

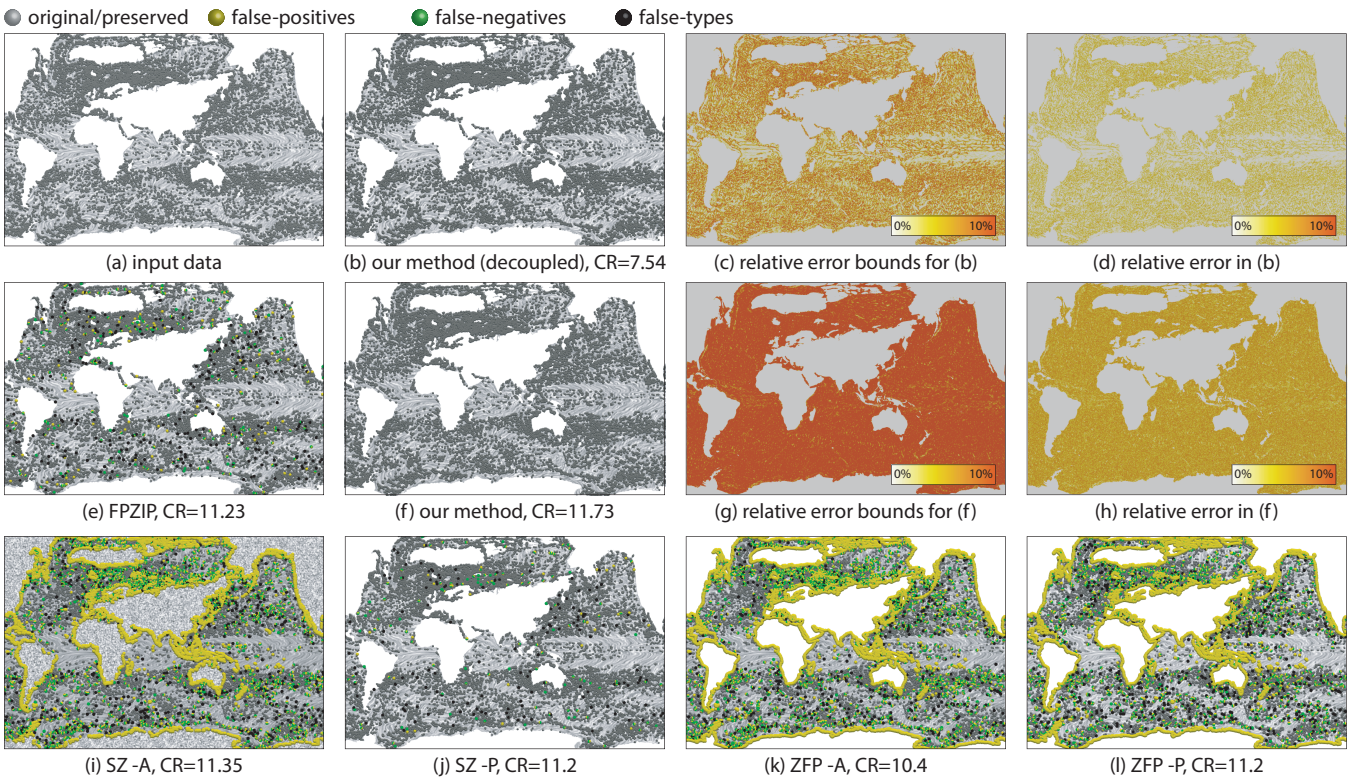



Figure 5: Visualizations of 2D ocean benchmark. More details are in Table 2

Table 4: Benchmark of (lossy) compressors on Nek5000 data.

| Compressor | Setting | $e_r$ | $CR_u$ | $CR_v$ | $CR_w$ | $CR_{all}$ | $S_c$ (MB/s) | $S_d$ (MB/s) | #TP | #FP | #FN | #FT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GZIP | -1 | 0 | 1.09× | 1.09× | 1.09× | 1.09× | 23.32 | 119.71 | 10,587 | 0 | 0 | 0 |
| Our method | decoupled | - | - | - | - | 3.27× | 8.97 | 41.46 | 10,587 | 0 | 0 | 0 |
| Our method | coupled | - | - | - | - | 7.48× | 6.38 | 62.14 | 10,587 | 0 | 0 | 0 |
| FPZIP | -P 16 | $2^{-7}$ | 6.87× | 6.73× | 7.59× | 7.04× | 94.90 | 82.76 | 10,499 | 98 | 72 | 16 |
| SZ | -P 0.015 | 0.015 | 7.14× | 6.74× | 7.75× | 7.19× | 97.27 | 148.06 | 10,199 | 358 | 326 | 62 |
| ZFP | -P 13 | 0.0625 | 6.59× | 6.47× | 6.95× | 6.66× | 129.29 | 306.60 | 9,927 | 695 | 566 | 94 |

Table 5: Benchmark of (lossy) compressors on unstructured LES Data.

| Compressor | Setting | $e_r$ | $CR_u$ | $CR_v$ | $CR_w$ | $CR_{all}$ | $S_c$ (MB/s) | $S_d$ (MB/s) | #TP | #FP | #FN | #FT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GZIP | -1 | 0 | 1.16× | 1.07× | 1.07× | 1.10× | 24.8 | 108.7 | 1,024 | 0 | 0 | 0 |
| Our method | coupled | - | - | - | - | 5.06× | 1.82 | 60.55 | 1,024 | 0 | 0 | 0 |
| FPZIP | -P 13 | 0.0625 | 6.78× | 4.27× | 4.27× | 4.87× | 81.95 | 73.88 | 992 | 31 | 25 | 7 |
| SZ | -P 0.02 | 0.02 | 6.44× | 4.39× | 4.38× | 4.91× | 121.06 | 238.82 | 984 | 13 | 29 | 11 |
| ZFP | -P 6 | 2 | 5.13× | 4.75× | 4.77× | 4.88× | 136.32 | 193.6 | 241 | 1870 | 708 | 75 |

### 6.4 Limitations

**Separatrix preservation** Our method does not theoretically guarantee the preservation of separatrices, but empirical studies show that our method outperforms general-purpose lossy compressors in preserving separatrices. As shown in Fig. 9(b), separatrices are changed in the FPZIP results because a saddle-source pair is missing. In Fig. 9(c), the separatrices are preserved in our decompressed data. We discussed the limitation with ocean climate researchers, and the preservation of critical point locations and types are important because they may imply features such as eddies. The preservation of separatrices may be achieved by iteratively reducing the global error bound until all separatrices are kept; we leave the separatrix preservation for future work.

**Eigenvector direction preservation** Our method introduces distortion to the eigenvalues and eigenvectors of critical points, which may in turn change the topology of separatrices. However, experiments show that the impact to eigenvector directions is usually minimal, as demonstrated in Fig. 9. Eigenvector directions can be strictly preserved by applying zero error bounds for corresponding cells, which may affect the compression ratio.

**Generalization to multilinear vector fields** Our compressor currently does not consider multilinear (bilinear and trilinear) vector fields. The error bound to avoid FN, FP, and FT may be derived with the technique presented in this paper but with much higher degrees of polynomials involved. Thus, closed-form error bounds may not be available, and numerical approximations are necessary. We will study feature preserving compression in multilinear and higher-order interpolated vector fields in future work.

## 7 Conclusions and Future Work

This paper introduces a theoretical framework to strictly preserve critical points by adapting vertex-wise error bounds in lossy com-

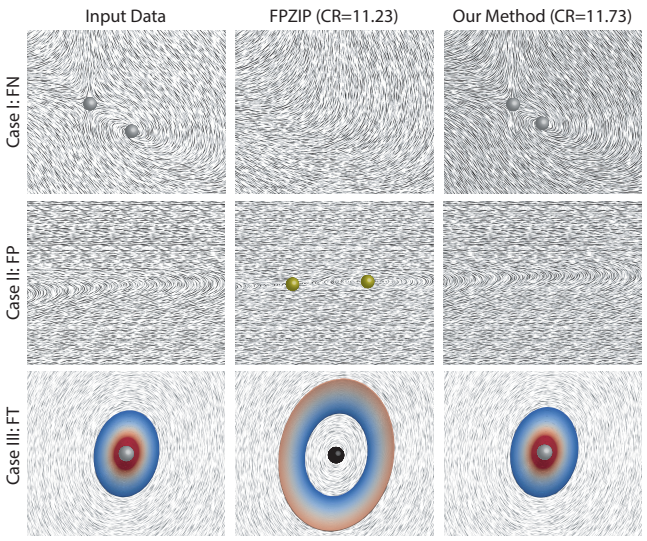



Figure 6: Visualizations of FN (case I), FP (case II), and FT (case III) in FPZIP decompressed data, compared with the original data and decompressed data of our method. In case III, the critical point type changed from attracting focus to repelling focus with FPZIP; each plot has one single streamline seeded from a fixed location near the critical point, and color encodes the integration time of streamlines.

pression. We also present two approaches to achieve feature-preserving compression: decoupled and coupled methods; the decoupled method is optimized for performance, while the coupled method has higher compression ratio. Experiments demonstrate that the proposed methods can preserve the critical points and local topologies are preserved in the visualization results.

We would like to further investigate preserving more topological features—topological skeletons, vortex core lines, and boundary surfaces—with error-bounded lossy compression. We would also like to generalize our method to multilinear, higher-order finite elements, and spectrum mesh elements in addition to simplicial cells. In addition to Lorenzo predictors, we will also investigate regression- and statistics-based predictors to further improve the compression quality.

## APPENDIX

We prove sufficient conditions to preserve critical points in Appendix A. Appendices B and C derive the error bounds for decoupled and coupled feature-preserving compression, respectively.

## A Sufficient Conditions to Preserve Critical Points

We prove Eqs. (4), (5), and (6) in this section. The derivations use the following properties of SPEFs:

$$\psi(fg;\mathbf{d}) \geq \min(\psi(f;\mathbf{d}),\psi(g;\mathbf{d})), \tag{7}$$

$$\psi(f+g;\mathbf{d}) \geq \min(\psi(f;\mathbf{d}),\psi(g;\mathbf{d})), \text{if } \operatorname{sgn}(f(\mathbf{d})) = \operatorname{sgn}(g(\mathbf{d})), \tag{8}$$

where $f$ and $g$ are two given scalar function $\mathbb{R}^{m\times n} \to \mathbb{R}$ that take the same arguments. We further generalize the notation of $\psi$ for multiple functions $f_0, f_1, \ldots, f_{n-1}$ to represent the maximal error bound to keep the sign of each function, and we have

$$\psi(f_0, f_1, \ldots, f_{n-1};\mathbf{d}) \stackrel{\text{def}}{=} \min_i \psi(f_i;\mathbf{d}). \tag{9}$$

We introduce *auxiliary barycentric coordinates* $(m_0, m_1, m_2)^\intercal$ to represent the 2D critical point solution in Eq. (2), such that in non-degeneracy cases, we have

$$\mu_k = m_k/(m_0+m_1+m_2) = m_k/M, k \in \{0,1,2\}, \tag{10}$$

where $M = \sum_k m_k \neq 0$, $m_0 = \left|\begin{smallmatrix} u_1 & u_2 \\ v_1 & v_2 \end{smallmatrix}\right|$, $m_1 = \left|\begin{smallmatrix} u_0 & u_2 \\ v_0 & v_2 \end{smallmatrix}\right|$, and $m_2 = \left|\begin{smallmatrix} u_0 & u_1 \\ v_0 & v_1 \end{smallmatrix}\right|$; We use similar notations to represent 3D critical point solutions with auxiliary barycentric coordinates. Notice that $m_0$, $m_1$, and $m_2$ are functions of $\mathbf{V}$, and we have the following.

**Proclaim 1.** *A sufficient condition to avoid FN in a triangular cell is* $||\Delta(\mathbf{V})||_{\max} \leq \psi(m_0, m_1, m_2; \mathbf{V})$.

*Proof.* The "non-FN" condition is equivalent to finding a proper maximal error bound $||\Delta(\mathbf{V})||_{\max}$ such that

$$||\Delta(\mathbf{V})||_{\max} \leq \psi(\mu_0, \mu_1, \mu_2, 1-\mu_0, 1-\mu_1, 1-\mu_2; \mathbf{V}) \stackrel{\text{def}}{=} \phi_{\text{FN}}(\mathbf{V}). \tag{11}$$

Based on Eqs. (10) and (7), we have

$$\begin{aligned}\phi_{\text{FN}}(\mathbf{V}) &\stackrel{(10)}{=} \psi\left(\frac{m_0}{M}, \frac{m_1}{M}, \frac{m_2}{M}, \frac{m_1+m_2}{M}, \frac{m_0+m_2}{M}, \frac{m_0+m_1}{M}; \mathbf{V}\right) \\ &\stackrel{(7)}{=} \psi(m_0, m_1, m_2, m_1+m_2, m_0+m_2, m_0+m_1, M; \mathbf{V}) \\ &\stackrel{(8)}{\geq} \psi(m_0, m_1, m_2; \mathbf{V}).\end{aligned} \tag{12}$$

The last inequality holds because $\operatorname{sgn}(m_0) = \operatorname{sgn}(m_1) = \operatorname{sgn}(m_2)$, which is deduced from $\mu_k = m_k/M \geq 0$ for all $k$. □

**Proclaim 2.** *A sufficient condition to avoid FP in a triangular cell is* $||\Delta(\mathbf{V})||_{\max} \leq \max_{k\in\{k|\mu_k\notin[0,1]\}} \min\big(\psi(m_k), \psi(\sum_{k'\neq k} m_{k'}; \mathbf{V})\big)$.

*Proof.* In the "non-FP" condition, there exists at least one $k$ such that $\mu_k \notin [0,1]$. Thus, a sufficient condition to avoid FP is to adapt $||\Delta(\mathbf{V})||_{\max}$ satisfying $\mu_k' \notin [0,1]$ in the decompressed data for any $k \in \{k \mid \mu_k \notin [0,1]\}$ such that

$$||\Delta(\mathbf{V})||_{\max} \leq \max_{k\in\{k|\mu_k\notin[0,1]\}} \psi(\mu_k(1-\mu_k); \mathbf{V}). \tag{13}$$

For $\psi(\mu_k(1-\mu_k); \mathbf{V})$, we have

$$\begin{aligned}\psi(\mu_k(1-\mu_k); \mathbf{V}) &= \psi\left(\frac{m_k}{M}\cdot\frac{M-m_k}{M}; \mathbf{V}\right) = \psi\left(\frac{m_k(\sum_{k'\neq k} m_{k'})}{M^2}; \mathbf{V}\right) \\ &\stackrel{(7)}{\geq} \min\big(\psi(m_k), \psi(\textstyle\sum_{k'\neq k} m_{k'}); \mathbf{V}\big),\end{aligned} \tag{14}$$

and thus the proclaim is proved. □

**Proclaim 3.** *A sufficient condition to avoid FT of noncenter critical points in a triangular cell is*[2]

$$||\Delta(\mathbf{V})||_{\max} \leq \begin{cases} \psi(|\mathbf{J}|; \mathbf{V}) & |\mathbf{J}| \leq 0 \\ \psi(\operatorname{tr}^2(\mathbf{J}) - 4|\mathbf{J}|, \operatorname{tr}(\mathbf{J}); \mathbf{V}) & |\mathbf{J}| > 0 \end{cases}. \tag{15}$$

*Proof.* Let $B = -\operatorname{tr}(\mathbf{J})$ and $C = |\mathbf{J}|$ for simplicity. The root of $\lambda^2 + B\lambda + C$ are $\lambda_0, \lambda_1 = (-B \pm \sqrt{B^2-4C})/2$. Consider the case of $C < 0$. The discriminant will be larger than 0; thus the roots are both real. Furthermore, the roots have to be one positive and one negative for $|B| < \sqrt{B^2-4C}$. Therefore, we need only to preserve the negativity of $\lambda_0\lambda_1$:

$$\psi(\lambda_0\lambda_1; \mathbf{V}) = \psi\Big(B^2 - (B^2-4C); \mathbf{V}\Big) = \psi(C; \mathbf{V}). \tag{16}$$

However, when $C > 0$, the sign of discriminant $B^2-4C$ has to be preserved because it determines the number of real roots. When the discriminant is greater than 0, we also need to preserve $B$ only because $\operatorname{sgn}(-B+\sqrt{B^2-4C}) = \operatorname{sgn}(-B-\sqrt{B^2-4C}) = \operatorname{sgn}(-B)$. The proclaim is thus proved. □

[2]The preservation of a center ($\operatorname{Re}(\lambda_0) = \operatorname{Re}(\lambda_1) = 0$ and $\operatorname{Im}(\lambda_0), \operatorname{Im}(\lambda_1) \neq 0$) requires $\operatorname{tr}(\mathbf{J}) = 0$, which leads to the error bound of 0. However, centers are seldom found because the floating-point representation of $\operatorname{tr}(\mathbf{J})$ is normally nonzero.

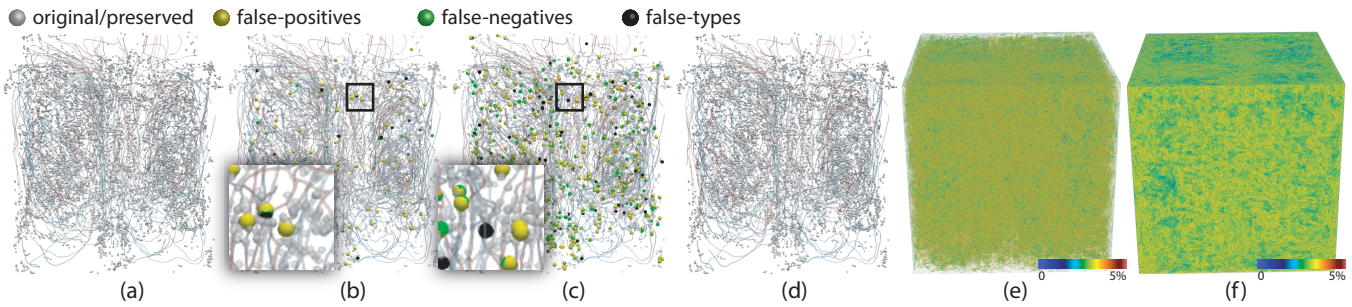



Figure 7: Visualization of Nek5000 simulation data: (a) original data with all critical points, (b) FPZIP decompressed data with false critical points, (c) SZ decompressed data with false critical points, (d) decompressed data from our method with all critical points, and (e) vertex-wise error bound derived and (f) vertex-wise error by our method. Streamlines are visualized as context

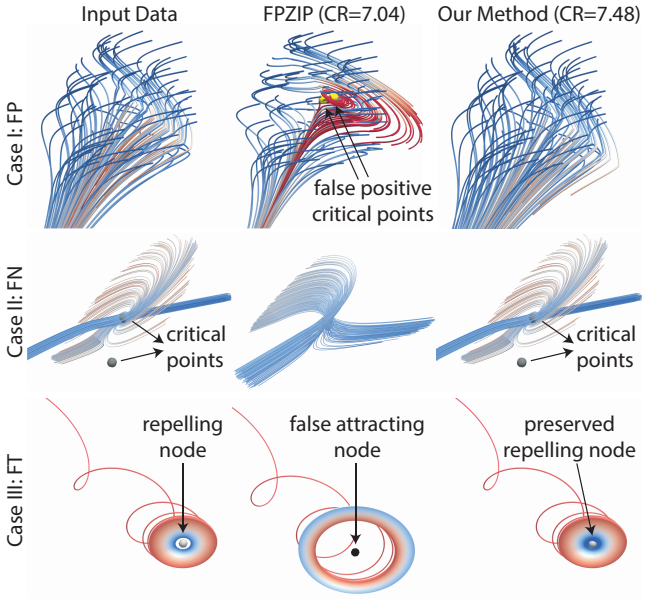



Figure 8: Visualizations of local streamline changes in FPZIP decompressed data, compared with the original data and decompressed data of our method. In case III, the streamline in each figure is seeded closely to the critical point in the original data, and colors encode the integration time of streamlines.

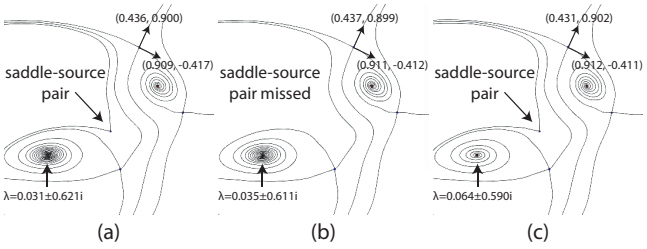



Figure 9: Zoomed visualization of separatrices in (a) original ocean data, (b) decompressed data with FPZIP (CR=7.04), and (c) decompressed data with our method (CR=7.48).

## B Error Bound Derivation in Decoupled Compression

This section derives `eb_FN_decoupled`, `eb_FP_decoupled`, and `eb_FT_decoupled` used in Algorithm 1 for decoupled compression. We use the following lemma for the error bound derivation in decoupled compression. The *positive ramp* $R^+(\cdot)$ and *negative ramp* $R^-(\cdot)$ used in the lemmas are defined as

$$\left(R^+(\mathbf{d})\right)_i = \max\{0, d_i\} \text{ and } \left(R^-(\mathbf{d})\right)_i = \min\{0, d_i\}, \tag{17}$$

respectively, for vector $\mathbf{d}$ of arbitrary dimensions, where $i$ is the linear index of the elements.

**Lemma 1.** *For $\mathbf{d} = \{d_{ij}\} \in \mathbb{R}^{m\times n}$, if $\sum_i \prod_j d_{ij} \neq 0$, we have*

$$\psi\left(\textstyle\sum_i \prod_j d_{ij}; \mathbf{d}\right) \geq \left| \frac{\sqrt[n]{\sum_i R^+\left(\prod_j d_{ij}\right)} - \sqrt[n]{-\sum_i R^-\left(\prod_j d_{ij}\right)}}{\sqrt[n]{\sum_i R^+\left(\prod_j d_{ij}\right)} + \sqrt[n]{-\sum_i R^-\left(\prod_j d_{ij}\right)}} \right|. \tag{18}$$

*Proof.* Consider the case of $\sum_i \prod_j d_{ij} \geq 0$. Assuming the adopted cell-wise error bound is $e_r$, we have

$$\begin{aligned} \textstyle\sum_i \prod_j d'_{ij} &= \textstyle\sum_i R^+\left(\prod_j d'_{ij}\right) + \sum_i R^-\left(\prod_j d'_{ij}\right) \\ &\geq (1-e_r)^n \textstyle\sum_i R^+\left(\prod_j d_{ij}\right) + (1+e_r)^n \sum_i R^-\left(\prod_j d_{ij}\right). \end{aligned} \tag{19}$$

Let the second line of Eq. (19) be greater than or equal to 0. The problem turns out to be solving

$$(1-e_r)^n \textstyle\sum_i R^+\left(\prod_j d_{ij}\right) + (1+e_r)^n \sum_i R^-\left(\prod_j d_{ij}\right) \geq 0. \tag{20}$$

By elementary calculation, we obtain $e_r$ in the form of Eq. (18) in this case. The case of $\sum_i \prod_j d_{ij} \leq 0$ can be proved similarly. □

We then derive the sufficient error bounds for all the SPEFs in Proclaims 1, 2, and 3, because all the functions in the SPEFs can be written in the form of $\mathbf{d} = \{d_{ij}\} \in \mathbb{R}^{m\times n}$. Notice that Proclaim 3 cannot be generalized to 3D; we use $||\Delta(\mathbf{V})||_{\max} = 0$ as the sufficient condition to guarantee "non-FT" due to the high complexity.

## C Error Bound Derivation in Coupled Compression

We introduce the derivation for `eb_FN_coupled`, `eb_FP_coupled`, and `eb_FT_coupled` used in Algorithm 1 for coupled compression in this appendix.

**Lemma 2.** *Let $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^n$, and $c \in \mathbb{R}$, if $R^+(\mathbf{a})^\intercal \mathbf{b} - R^-(\mathbf{a})^\intercal \mathbf{b} \neq 0$. Then we have*

$$\psi(\mathbf{a}^\intercal \mathbf{b} + c; \mathbf{b}) \geq \frac{|\mathbf{a}^\intercal \mathbf{b} + c|}{R^+(\mathbf{a})^\intercal \mathbf{b} - R^-(\mathbf{a})^\intercal \mathbf{b}}. \tag{21}$$

*Proof.* First, we consider the case of $\mathbf{a}^\intercal \mathbf{b} + c > 0$. The decompressed data $\mathbf{a}^\intercal \mathbf{b}' + c$ can be scaled as follows by assuming the adopted cell-wise error bound is $e_r$. Then we have

$$\begin{aligned} \mathbf{a}^\intercal \mathbf{b}' + c &= (R^+(\mathbf{a}) + R^-(\mathbf{a}))^\intercal \mathbf{b}' + c \\ &\geq R^+(\mathbf{a})^\intercal \mathbf{b}(1-e_r) + R^-(\mathbf{a})^\intercal \mathbf{b}(1+e_r) + c. \end{aligned} \tag{22}$$

Let the second line of Eq. (22) be larger than or equal to 0. Then we have

$$e_r \leq \frac{R^+(\mathbf{a})^\intercal\mathbf{b} + R^-(\mathbf{a})^\intercal\mathbf{b} + c}{R^+(\mathbf{a})^\intercal\mathbf{b} - R^-(\mathbf{a})^\intercal\mathbf{b}} = \frac{\mathbf{a}^\intercal\mathbf{b} + c}{R^+(\mathbf{a})^\intercal\mathbf{b} - R^-(\mathbf{a})^\intercal\mathbf{b}}. \quad (23)$$

Second, the case for $\mathbf{a}^\intercal\mathbf{b} + c < 0$ can be proved similarly. This completes the proof. □

The sufficient error bound to avoid FNs and FPs in coupled compresion (i.e., `eb_FN_coupled` and `eb_FP_coupled`) can be derived based on Proclaims 1 and 2, as well as Lemma 2.

Next, we explain how to derive the sufficient error bounds to avoid FTs for 2D piecewise linear vector field. The 2D piecewise constant Jacobian can be formulated as

$$\mathbf{J} = \widehat{\mathbf{X}}^{-1}\widehat{\mathbf{V}} = \begin{bmatrix} x_0 - x_2 & x_1 - x_2 \\ y_0 - y_2 & y_1 - y_2 \end{bmatrix}^{-1} \begin{bmatrix} u_0 - u_2 & u_1 - u_2 \\ v_0 - v_2 & v_1 - v_2 \end{bmatrix}. \quad (24)$$

Without loss of generality, we assume $\mathbf{v}_0 = (u_0, v_0)$ is the "current" value that is being processed in Algorithm 4 and $u_1, v_1, u_2, v_2$ are constants. By elementary calculation, $|\mathbf{J}|$ can be written as an affine function of $\mathbf{v}_0 = (u_0, v_0)$:

$$|\mathbf{J}|(u_0, v_0) = \alpha_0 u_0 + \alpha_1 v_0 + \alpha_2, \quad (25)$$

where $\alpha_0$, $\alpha_1$, and $\alpha_2$ are constants that can be derived from $u_1, v_1, u_2, v_2$, and $\widehat{\mathbf{X}}$. Likewise, the trace of $\mathbf{J}$ can be written as another affine function of $\mathbf{v}_0 = (u_0, v_0)$ with derived constants $\beta_0$, $\beta_1$, and $\beta_2$:

$$\mathrm{tr}(\mathbf{J})(u_0, v_0) = \beta_0 u_0 + \beta_1 v_0 + \beta_2. \quad (26)$$

Therefore, the sufficient error bounds for $\psi(\mathrm{tr}(\mathbf{J}); \mathbf{v}_0)$ and $\psi(|\mathbf{J}|; \mathbf{v}_0)$ can be directly derived by using Lemma 2.

We then derive the sufficient error bound for the discriminant $\mathrm{tr}^2(\mathbf{J}) - 4|\mathbf{J}|$. Assuming the adopted error bound is $e_r$, we denote the resulting relative errors in $u_0$ and $v_0$ are $\varepsilon_u \in (-e_r, e_r)$ and $\varepsilon_v \in (-e_r, e_r)$, respectively. We also have $u_0' = u_0(1 + \varepsilon_u)$ and $v_0'(1 + \varepsilon_v)$ for the decompressed value $u_0'$ and $v_0'$. Thus, the decompressed $\mathrm{tr}^2(\mathbf{J}') - 4|\mathbf{J}'|$ can be written as a quadratic function of $(\varepsilon_u, \varepsilon_v)$, which can be represented by the following quadratic form:

$$\mathrm{tr}^2(\mathbf{J}') - 4|\mathbf{J}'| = (\varepsilon_u, \varepsilon_v, 1)\mathbf{Q}(\varepsilon_u, \varepsilon_v, 1)^\intercal, \quad (27)$$

where $\mathbf{Q} \in \mathbb{R}^{3\times 3}$ are derived from $u_0, v_0, u_1, v_1, u_2, v_2$, and $\widehat{\mathbf{X}}$. Because $(\varepsilon_u, \varepsilon_v) \in (-e_r, e_r) \times (-e_r, e_r)$, it has an analytical infimum and supremum with respect to $e_r$. Therefore, let $\sup(\mathrm{tr}^2(\mathbf{J}') - 4|\mathbf{J}'|) < 0$ when $\mathrm{tr}^2(\mathbf{J}) - 4|\mathbf{J}| < 0$, and obtain a sufficient error bound. Likewise, we obtain a sufficient error bound when $\mathrm{tr}^2(\mathbf{J}) - 4|\mathbf{J}| > 0$ by letting the $\inf(\mathrm{tr}^2(\mathbf{J}') - 4|\mathbf{J}'|) > 0$.

Given the sufficient error bound for $\psi(\mathrm{tr}(\mathbf{J}); \mathbf{v}_0)$, $\psi(|\mathbf{J}|; \mathbf{v}_0)$, and $\psi(\mathrm{tr}^2(\mathbf{J}) - 4|\mathbf{J}|; \mathbf{v}_0)$, we obtain the `eb_FT_coupled` value required in Algorithm 4 based on Proclaim 3.

## ACKNOWLEDGMENT

We thank Dr. Jeffery Larson, Dr. Todd Munson, and Dr. Chongke Bi for useful discussions. Work by Chunhui Liu was supported by JSPS KAKENHI Grant Number JP17F17730 and JSPS grant (S) 16H06335. This material is based upon work supported by Laboratory Directed Research and Development (LDRD) funding from Argonne National Laboratory, provided by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-06CH11357. This work is also supported by the U.S. Department of Energy, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program.

## REFERENCES

[1] GZIP. `https://www.gzip.org`.
[2] ZSTD. `http://www.zstd.net`.
[3] M. Burtscher and P. Ratanaworabhan. FPC: A high-speed compressor for double-precision floating-point data. *IEEE Transactions on Computers*, 58(1):18–31, 2008.
[4] T. K. Dey, J. A. Levine, and R. Wenger. A Delaunay simplification algorithm for vector fields. In *Proc. Pacific Conference on Computer Graphics and Applications*, pp. 281–290, 2007.
[5] 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. *Comput. Graph. Forum*, 35(3):643–667, 2016.
[6] J. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *IEEE Computer*, 22(8):27–36, 1989.
[7] L. Ibarria, P. Lindstrom, J. Rossignac, and A. Szymczak. Out-of-core compression and decompression of large n-dimensional scalar fields. *Computer Graphics Forum*, 22(3):343–348, 2003.
[8] S. Koch, J. Kasten, A. Wiebel, G. Scheuermann, and M. Hlawitschka. 2D vector field approximation using linear neighborhoods. *The Visual Computer*, 32(12):1563–1578, 2016.
[9] R. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. Post, and D. Weiskopf. The state of the art in flow visualization: Dense and texture-based techniques. *Comput. Graph. Forum*, 23(2):203–222, 2004.
[10] R. S. Laramee, H. Hauser, L. Zhao, and F. H. Post. Topology-based flow visualization, the state of the art. In *Proc. Topology-Based Methods in Visualization*, pp. 1–19, 2007.
[11] S. Li, N. Marsaglia, C. Garth, J. Woodring, J. P. Clyne, and H. Childs. Data reduction techniques for simulation, visualization and data analysis. *Comput. Graph. Forum*, 37(6):422–447, 2018.
[12] 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 *CLUSTER'18: Proc. IEEE International Conference on Cluster Computing*, pp. 179–189. IEEE, 2018.
[13] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, and F. Cappello. Error-controlled lossy compression optimized for high compression ratios of scientific datasets. In *Proc. IEEE International Conference on Big Data*, pp. 438–447. IEEE, 2018.
[14] P. Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Trans. Vis. Comput. Graph.*, 20(12):2674–2683, 2014.
[15] P. Lindstrom and M. Isenburg. Fast and efficient compression of floating-point data. *IEEE Trans. Vis. Comput. Graph.*, 12(5):1245–1250, 2006.
[16] S. K. Lodha, J. C. Renteria, and K. M. Roskin. Topology preserving compression of 2D vector fields. In *Proc. IEEE Visualization 2000*, pp. 343–350, 2000.
[17] T. McLoughlin, R. S. Laramee, R. Peikert, F. Post, and M. Chen. Over two decades of integration-based, geometric flow visualization. *Comput. Graph. Forum*, 29(6):1807–1829, 2010.
[18] M. Rodríguez, E. Gobbetti, J. Guitián, M. Makhinya, F. Marton, R. Pajarola, and S. Suter. State-of-the-art in compressed GPU-based direct volume rendering. *Comput. Graph. Forum*, 33(6):77–100, 2014.
[19] 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 *Proc. IEEE International Parallel and Distributed Processing Symposium*, pp. 1129–1139, 2017.
[20] A. Telea and J. J. van Wijk. Simplified representation of vector fields. In *Proc. IEEE Visualization 1999*, pp. 35–42, 1999.
[21] H. Theisel, C. Rössl, and H.-P. Seidel. Compression of 2D vector fields under guaranteed topology preservation. *Comput. Graph. Forum*, 22(3):333–342, 2003.
[22] H. Theisel, C. Rössl, and T. Weinkauf. Topological representations of vector fields. In L. D. Floriani and M. Spagnuolo, eds., *Shape Analysis and Structuring*, pp. 215–240. Springer, 2008.
[23] G. K. Wallace. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992.