# Understanding Effectiveness of Multi-error-bounded Lossy Compression for Preserving Ranges of Interest in Scientific Analysis

Yuanjian Liu*, Sheng Di†, Kai Zhao‡, Sian Jin§, Cheng Wang†,
Kyle Chard*, Dingwen Tao§, Ian Foster*, Franck Cappello†
* University of Chicago, Chicago, IL, USA
† Argonne National Laboratory, Lemont, IL, USA
‡ University of California, Riverside, CA, USA
§ Washington State University, Washington, USA

***Abstract*—Lossy compression frameworks have been proposed as a method to reduce the size of data produced by scientific simulations. However, they do so at the expense of precision and existing compressors apply a single error bound across the entire dataset. Varying the precision across user-specified ranges of scalar values appears to be a promising approach to further improve compression ratios while retaining precision in specific areas of interest. In this work, we investigate a specific compression method, based on the SZ framework, that can set multiple error bounds. We evaluate its effectiveness by applying it to real-world datasets which have concrete precision requirements. Our results show that the multi-error-bounded lossy compression can improve compression ration by 15% with negligible overhead in compression time.**

## I. Introduction

Scientific applications such as the Community Earth System Model (CESM) [1] and Hardware/Hybrid Accelerated Cosmology Code (HACC) [2] can generate vast volumes of data. Storing these data in their raw format for analysis may be expensive and ineffective. Error-bounded lossy compressors [3], [4], [5], [6], [7], [8], [9] can significantly reduce data sizes and thus enable more effective storage and use of such data. For instance, SZ [4], [10], ZFP [3], and MGARD [11] allow users to specify an absolute error bound which limits the maximum compression error for each data point with a constant. Thus, the compressor can reduce the data size by allowing acceptable error for each data point such that they may be predicted by neighboring data points. Climate researchers have shown that the lossy reconstructed data are acceptable for post hoc analysis [12], [13], [14].

Most compression algorithms treat every part of the data in a uniform manner, so that researchers cannot customize the configuration to allow different error bounds for different data ranges. Yet in environmental science datasets, for example, different values in different ranges may have different significance to the post hoc analysis. For instance, to trace a hurricane's trajectory, researchers desire higher precision only when the elevation of the water surface is above a warning sealevel (such as 1 meter), since these points indicate the movement of a hurricane. Similarly, the Nyx cosmological simulation [15] determines the construction of dark matter halos primarily by values in a specific value range: $[81, 83]$. Making values in this range more precise than others can maintain post analysis result accuracy with only modest increase to the file size. Thus, there is a pressing need to develop new algorithms that treat different ranges of values with different precision.

In this paper, we investigate a novel compression method that extends the SZ error-bounded lossy compression framework to support different error bounds in terms of value ranges. We leverage visualization and quantitative measurements to demonstrate that the multi-error-bounded lossy compression can significantly improve compression quality according to user's diverse requirements on different value ranges.

We summarize our contributions as follows.

- We formalize the problem of setting different error bounds for different value ranges so that both prediction-based and transformation-based algorithms can share the same prerequisite and optimize for the same goal.
- We present a multi-error-bounded compression algorithm that can adapt to various required details via multiple error bounds in different value ranges.
- We evaluate our approach on real-world datasets with different post hoc analyses and verify the effectiveness of the multi-error-bounded compression. Experimental results show that better visual qualities can be reached and higher compression ratios can be obtained with appropriate configurations.

The rest of the paper is organized as follows. In Section II, we discuss related work. In Section III we formulate the problem of defining different ranges of interest. In Section IV we present the (de)compression algorithm that supports multiple error bounds for different value ranges. In Section V, we present our evaluation results on two real-world datasets. In Section VI, we summarize our work.

## II. Related Work

Current error-bounded lossy compressors offer different types of error bounds to address diverse user requirements. The most common error-bounding approach uses an absolute error

bound, which ensures that the pointwise difference between the original raw data and reconstructed data is confined within a constant threshold. Many compressors such as SZ [4], [10], [16], ZFP [3], [17], and MGARD [11] support absolute error bounds. Other error-bounding approaches have also been exploited to adapt to diverse user requirements. For instance, SZ supports pointwise relative error bound [18]; and Digit Rounding [19], Bit Grooming [20], zfp [3], and FPZIP [21] support a type of precision mode that allows users to specify the number of bits to be truncated in the end of the mantissa, to control the data distortion at different levels.

Recent research has explored methods for respecting specific metrics in order to satisfy user demands on a specific quality of interest. For example, Tao et al. [22] developed a formula that can link the target peak signal-to-noise ratio (PSNR) metric to an absolute error bound setting in SZ such that SZ can use a user-specified PSNR metric. MGARD [11] supports various norm error metrics and linear quantities of interest in its multigrid compression method. However, none of the existing compression methods allow users to define different error bounds for different data based on value ranges. There is certainly an opportunity to further optimize compression quality by allowing less precision in non-interesting ranges.

## III. Problem Formulation

In this section, we first describe what we consider preserving ranges of interest. We then formulate the target and constraints of this research problem.

In practice, scientists may not have the same precision needs for all the values in the global range. Instead, they often have specific value ranges of interest because of particular features on which they wish to analyze. Thus, they may wish to set different error bounds in various value ranges of the dataset.

We formulate the multi-error-bounded lossy compression problem as follows. Given a scientific dataset $D$ composed of $N$ floating-point values (either single precision or double precision), the objective is to develop an error-bounded lossy compressor that can respect a set of user-defined error bounds for different value ranges.

There are three important assessment metrics. The first two are compression speed $s_c$ and decompression speed $s_d$, respectively. They are usually measured in the unit megabytes per second: i.e., the size (in MB) of the original dataset processed (either compressed or decompressed) per time unit. The third evaluation metric is compression ratio (denoted by $\rho$) which is defined as follows:

$$\rho = \frac{N \cdot sizeof(dataType)}{Size_{compression}}, \tag{1}$$

where $dataType$ can be either float or double and $Size_{compression}$ is the total size after compression.

Our goal can be formulated as follows:

$$\begin{aligned} &\text{Maximize } \ \rho \\ &subject\ to \ \ |d_i - \hat{d}_i| \leq e(d_i) \end{aligned} \tag{2}$$

where $d_i$ denotes the $i$th data point in the original dataset $D$, $e(d_i)$ denotes the corresponding error bound at data point $d_i$ (e.g., in terms of user-specified multi-error-bounded error bounds), and $\hat{d}_i$ is the decompressed value of that data point. Unlike the prior constant-error-bounded lossy compression, the error bound here becomes a function of the value range in which the corresponding data point is located.

The following examples further illustrate how we formulate the research problem. The Hurricane Katrina simulation is used to track the location of the hurricane by calculating the height of the water surface. Accordingly, the researchers focus on the data whose values are greater than a threshold (e.g., 1 meter). Based on Formula (2), the target is to maximize the compression ratio while making sure the relatively higher values have lower error bound (e.g., if $d_i \geq 1$, then $e(d_i) = 0.01$; otherwise, $e(d_i)$=0.1). The Nyx cosmological simulation computes dark matter halo cells based on a threshold located in the range of $[81, 83]$ (according to the Nyx analysis code [15]). Thus, for any data point $d_i$ in $[81, 83]$, their error bounds $e(d_i)$ should be lower than $e(d_j)$, where $d_j$ refers to the data points that fall outside of the critical range $[81, 83]$. Setting multiple error bounds can completely eliminate the distortion of halo cells calculated by the reconstructed data with the unchanged compression ratios.

## IV. Multi-Error-Bounded Lossy Compression Framework

We extend the SZ model to support multi-error-bounded lossy compression in different value ranges. We first introduce the SZ compression model and then describe how we designed the multi-error-bounded compression method.

### A. SZ Compression Model

The classic error-bounded lossy compression model implemented by SZ is illustrated in Figure 1. As is shown in the figure, the SZ compression framework is composed of four key stages: prediction, quantization, Huffman encoding, and lossless compression. Given a set of raw data, SZ scans the entire dataset (either pointwise [4], [10] or blockwise [16]) to predict the data values. In a 1D dataset, the prediction method is simply a first-order Lorenzo predictor [10], which uses only the preceding value to approximate the current data point. In a 2D or 3D dataset, SZ adopts a hybrid data prediction method combining the first-order Lorenzo predictor (using three nearby values in the 2D Lorenzo and 7 nearby values in the 3D Lorenzo) and a linear-regression-based predictor [16]. Lorenzo predictor uses the "decompressed data" with certain data loss instead of the "original data" in its prediction (otherwise the compression and decompression stages cannot be synchronized), so the prediction accuracy would be degraded significantly when the error bound is relatively large.

The second stage in SZ uses a linear quantization method to convert the distance between the predicted value and original value to an integer value (called a quantization code or quantization number) for each data point. A customized Huffman encoder is then applied to compress the integer quantization codes, followed by a lossless dictionary coding (Zstd [23] by default in SZ).

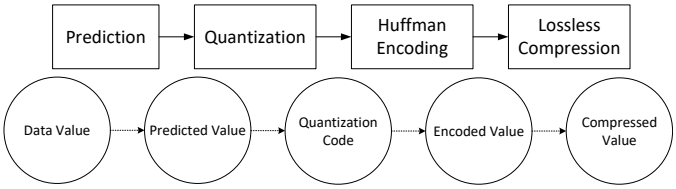


Fig. 1. **SZ Compression Framework**: The compression process is separated into four stages: (1) The prediction stage computes a value using nearby data points and can only be precise when data satisfy the properties of certain predictors; (2) The quantization stage moves those imprecise predicted data back to the error bounded region; (3) Huffman Coding can significantly reduce the file size when values cluster and therefore serves well for quantization code; (4) The final lossless compression stage is important because our processing outputs many repeated patterns and the lossless compression can significantly improve the compression ratio.

### *B. Preserving Multivalue-Range-Based Error Bounds*

As the name implies, the multivalue-range error-bound model is defined by an array of triplets, each containing the *low*, *high*, and *error bound*. The fundamental idea is to apply different quantization bins (whose length is twice the error bound) to different ranges. The algorithm will calculate the total number of varied-length quantization bins involved between the predicted value and the raw value during the compression, and identify the quantization bin based on the multi-range error bounds during the decompression.

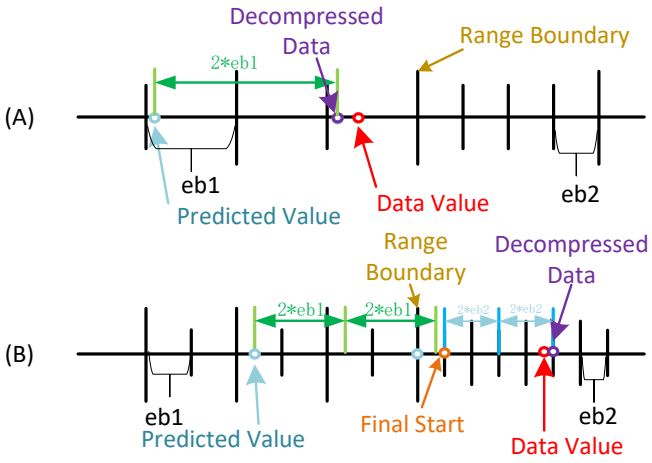


Fig. 2. **Multirange error-bound model** Situation A: when the predicted value and raw value are in the same range, it is identical to the single range's case except for some edge conditions; Situation B: when the predicted value and raw value are in different ranges, we use quantization bins to infer in which range the value falls and use the final range's starting point to compute the decompressed data.

Before describing the detailed methodology, we first review the notation used in this paper (see Table I). Let $d_i$ denote the original data value at position $i$. Let $p_i$ denote the predicted value for $d_i$. We use $R$ to denote the radius of the quantization bin (for instance, if there are 65,536 quantization bins, the radius $R$ is equal to 32,768). Let $\hat{d}_i$ denote the decompressed data value. Let $r(x)$ be a function that returns a value range index based on a given data value $x$=$d_i$. The range indexes go from 0 to the number of ranges minus 1, which is a simple map from the ranges of values to integers. Let $e(x)$ be a function that returns the user-specified error bound based on a given value range ($x$=$r(d_i)$). Let $l(x)$ denote the length of some value range based on the range index ($x = r(d_i)$). Let $low(r(x))$ and $high(r(x))$ denote the low boundary and high boundary of the value range $r(x)$, respectively. Let $q$ denote the quantization code, and let $q_s$ denote the shifted quantization number.[1]

TABLE I
KEY NOTATIONS

| Notation | Description |
|---|---|
| $d_i$ | original data value at position $i$ |
| $p_i$ | predicted value of $d_i$ |
| $\hat{d}_i$ | reconstructed data value after decompression |
| $r(x)$ | the value range of data point $x$ ($d_i$) |
| $e(x)$ | specified error bound based on a value range ($x$=$r(d_i)$) |
| $l(x)$ | length of some value range ($x = r(d_i)$) |
| $low(r(x))$ | lower boundary of value range $r(x)$ |
| $high(r(x))$ | higher boundary of value range $r(x)$ |
| $q$ | quantization index (a.k.a., quantization code) |
| $q_s$ | shifted quantization number |

The key part of the algorithm design is to calculate the total number of quantization bins that go through multiple ranges and make sure the decompression stage can synchronize with the compression stage. The algorithm needs to ensure it arrives in the correct range and gets the correct quantization number before calculating the decompressed value. Algorithm 1 presents the pseudocode of the multi-range-based quantization in the compression stage.

**Algorithm 1** MULTIRANGE QUANTIZATION IN COMPRESSION STAGE

**Input**: user-specified ranges and error bounds $\varepsilon$
**Output**: compressed data stream in form of bytes

**for** each data point $d_i$ **do**
  Use the composed prediction that combines Lorenzo predictor and linear regression predictor to obtain a prediction value $p_i$.
  $I_p \leftarrow r(p_i)$. /*Obtain range index of $p_i$*/
  $I_d \leftarrow r(d_i)$. /*Obtain range index of $d_i$*/
  **if** $I_d == I_p$ **then**
    $q \leftarrow round(\frac{(d_i - p_i)}{2e(I_d)})$. /*Quantized distance between $d_i$ and $p_i$.*/
  **else if** $I_d > I_p$ **then**
    $t = \sum_{i=I_p+1}^{I_d-1} \frac{l(i)}{2e(i)}$. /*Count bins for middle ranges.*/
    $t_p = round(\frac{high(I_p) - p_i}{2e(I_p)})$. /*Count quantized distance for $I_p$.*/
    $t_d = round(\frac{d_i - low(I_d)}{2e(I_d)})$. /*Count quantization distance for $I_d$.*/
    $q = t + t_p + t_d$. /*Get the logic quantization code.*/
  **else**
    $t = \sum_{i=I_d+1}^{I_p-1} \frac{l(i)}{2e(i)}$. /*Count bins for middle ranges.*/
    $t_p = round(\frac{high(I_d) - d}{2e(I_d)})$. /*Count quantized distance for $I_d$.*/
    $t_d = round(\frac{p_i - low(I_p)}{2e(I_p)})$. /*Count quantization distance for $I_p$.*/
    $q = t + t_p + t_d$. /*Get the logic quantization code.*/
  **end if**
  $q_s \leftarrow q + R$. /*Shift quantization code.*/
**end for**

[1]Since the C/C++ array has no negative index in an array, the logic quantization bins [-R,R] need to be shifted to [0, $2R$] in our implementation.

For each data point, there are two situations to address—whether the original data and predicted data are in the same range, as illustrated in Figure 2.

**Situation A** (line 5∼6): If its original raw value $d_i$ and its predicted value $p_i$ fall in the same range ($i.e., r(d_i) = r(p_i)$), the quantization problem falls back to the traditional linear-scale quantization [10]. Specifically, we compute the quantization code and decompressed data as follows.

$$q = round(\frac{d_i - p_i}{2e(r(d_i))}) \tag{3}$$

$$\hat{d}_i = p_i + 2e(r(d_i)) \cdot q \tag{4}$$

We use an example to illustrate how the linear-scale quantization works. Suppose the error bound (i.e., $e(r(d_i))$) is 20 and we have $d_i$=−74, $p_i$=−95. Then $d_i - p_i = 21$ and $q = round(21/40) = 1$. The decompressed value $\hat{d}_i$ is $-55$, whose distance to the raw value is less than the error bound.

**Situation B** (line 7∼17): When the raw value $d_i$ and its predicted value $p_i$ fall in different ranges (i.e., $r(d_i) \neq r(q_i)$). In the following text, we describe only the situation with $r(d_i) > r(q_i)$ (i.e., line 7∼11 shown in the algorithm); the other situation is similar.

When the counter passes into a different range, we need to continue adding the quantization bins from the boundary of the new range. Obviously, the decompressed data value is determined by the last value range and its quantization bin size. The formula for reconstructing the decompressed value is given below (we assume the raw data is greater than the predicted value, without loss of generality):

$$q_t = round(\frac{d_i - low(r(d_i)) - e(r(d_i))}{2e(r(d_i))}) \tag{5}$$

$$\hat{d}_i = low(r(d_i)) + e(r(d_i)) + 2e(r(d_i)) \cdot q_t. \tag{6}$$

We briefly describe the decompression in Algorithm 2. The core idea is to execute the similar operations in the compression stage reversely to get the decompressed data from a predicted data value and the corresponding quantization bins. As shown in the pseudocode, we first calculate the number of quantization bins for each value range (line 3∼5). We then decompress each data point based on the multivalue-range quantization (lines 6∼34). If the raw data value is lower than the predicted value (i.e., $q_j < 0$), the code will scan all the involved value ranges downward (lines 10∼29). Lines 25∼29 refer to the situation that the predicted value and original raw value fall in the same range. Lines 13∼23 deal with the other situation where the two values fall in different ranges.

Note that in the real implementation, we need to consider several edge cases. For instance, when the original data are near the high or low bound of a range, the quantization value in this final range might be equal to $quantRange[i]$, causing the decompressed value to be in the next range. In this case, we shift the quantization by 1 in the compression stage to ensure the decompressed data and original data are in the same range.

**Algorithm 2** MULTIRANGE QUANTIZATION IN DECOMPRESSION

**Input**: compressed data stream
**Output**: decompressed data stream in the form of bytes

1: Read ranges and error bounds in the header & initialize multirange quantizer.
2: Read the quantization bins and unpredictable data.
3: **for** each range index $I_i$ **do**
4: $\hat{l}_i = \frac{l(I_i)}{2e(I_i)}$. /*Calculate the number of quantization bins for each range*/
5: **end for**
6: **for** each decompressed data position $j$ **do**
7: Use the composed prediction that combines Lorenzo predictor and linear regression predictor to obtain a prediction value $p_j$.
8: $q_j = q_s - R$. /*Get the logic quantization code $q_j$*/.
9: $I_p \leftarrow r(p_j)$. /*Obtain range index of $p_j$*/
10: **if** $q_j < 0$ **then**
11: $\hat{\Delta} \leftarrow p_j - low(I_p)$ /*Compute $p_j$'s distance to the low boundary*/
12: $\hat{\Delta} \leftarrow round(\Delta/2(e(I_p)))$ /*Compute the quantized distance*/
13: **if** $q_j + \hat{\Delta} < 0$ **then**
14: **for** $i$ from $I_p - 1$ to 1 **do**
15: **if** $q_j + \hat{l}_i \geq 0$ **then**
16: $\hat{d}_j \leftarrow high(i) - e(i) + (q_j + 1) \cdot (2 \cdot e(i))$. /*Get decompressed data*/
17: **if** $\hat{d}_j < low(i)$ **then**
18: $\hat{d}_j \leftarrow low(i) + e(i)$. /*Correct the decompressed data*/
19: **end if**
20: **else**
21: $q_j \leftarrow q_j + \hat{l}_i$. /*Add quantization length for further search*/
22: **end if**
23: **end for**
24: **else**
25: $\hat{d}_j \leftarrow p_j + q_j \cdot 2 \cdot (e(I_p))$. /*Compute decompressed value*/
26: **if** $\hat{d}_j < low(I_p)$ **then**
27: $\hat{d}_j \leftarrow low(I_p) + e(I_p)$. /*Perform correction to avoid undesired boundary-crossing*/
28: **end if**
29: **end if**
30: **else if** $q_j == 0$ **then**
31: $\hat{d}_j \leftarrow p_j$. /*The prediction is accurate, directly use the predicted value*/
32: **else if** $q_j > 0$ **then**
33: Calculate the decompressed data using similar methods. /*Ignore the details here. It is similar to the case when $q_j < 0$, with just a few changes to the low bound and high bound and some calculation differences.*/
34: **end if**
35: **end for**

## V. EXPERIMENTAL EVALUATION

In this section, we first compare compression time on six datasets to explore the overhead of our approach. We then evaluate the compression ratio and post-analysis quality on two real-world simulation datasets: Nyx [15] and Hurricane Katrina [24], both of which have needs for extremely precise data values in specific ranges.

### A. Evaluation of Compression Time Overhead

We compare the compression time between our approach and the SZ compressor on six datasets: QMCPACK [25], Nyx [15], Miranda [26], Hurricane Isabel [27], RTM, and CESM [1] as shown in Table II.

We set five ranges for each multi-range compression task because in most cases users have requirements for only a small

TABLE II
BASIC DATASET INFORMATION

| Dataset | # Fields | Dimensions | Science |
|---|---|---|---|
| QMCPACK | 1 | 33120*69*69 | electronic structure of atoms, molecules, and solids |
| RTM | 1 | 449*449*235 | Electronic |
| Miranda | 7 | 256*384*384 | hydrodynamics code for large turbulence simulations |
| CESM | 79 | 1800*3600 | Climate |
| Nyx | 6 | 512*512*512 | Cosmology |
| Hurricane Isabel | 13 | 100*500*500 | Weather |

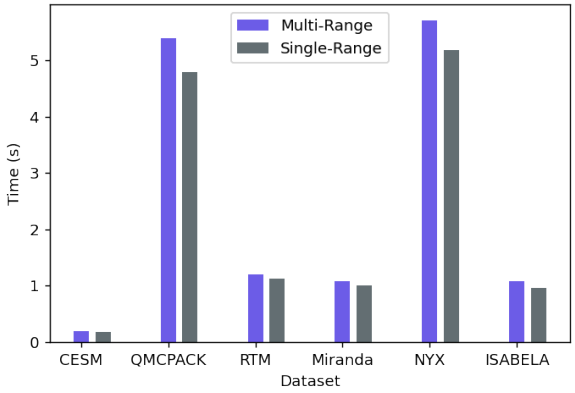


Fig. 3. Compression Time Overhead Comparison: The reference point is the SZ's single range version, which means using a global error bound for all data points

number of ranges (often one or two). Thus, evaluating with five ranges can roughly serve as a worst case scenario for compression time.

Figure 3 shows that our multi-range algorithm indeed adds overhead to the compression time, but we can see the overhead negligible in most cases and less than than 10% in all cases.

### B. Nyx Cosmological Simulation

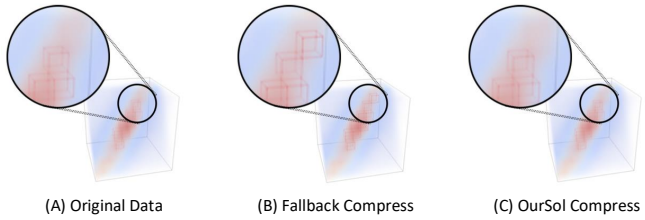


Fig. 4. Nyx halo cell visualization: The fallback method sets a global error bound to be 0.5 and the compression ratio is 75. Our solution (C) sets three ranges, [min, 81) with error bound set to 1, [81, 83) with error bound set to 0.01, and [83, max) with error bound set to 1, and the compression ratio is 78. In the visualization, our solution (C) has cells almost identical to original data's result, while the fallback method (B) shows some distortion and the cells' position and number is not identical to (A).

We consider compression of the Nyx cosmological simulation with a specific quantity of interest (i.e., dark matter halo cell information). Dark matter halos play an important role in the formation and evolution of galaxies and consequently in cosmological simulations. Halos are overdensities in the dark matter distribution and can be identified by using different algorithms; in this instance, we use the Friends-of-Friends algorithm [28]. For the Nyx simulation, which is an Eulerian simulation rather than a Lagrangian simulation, the halo-finding algorithm uses density data to identify halos [29]. When decompressing data there is a risk that some of the information (e.g., halo cells and halo mass) can be distorted from the original.

Figure 4 shows visualizations of halo cells using the original data and two different compression methods: a fallback method with global error bound set to 0.5 and our approach with three different ranges and a smaller error bound (0.01) applied to the range of $[81, 83]$. The figure shows that setting different error bounds for different value ranges can preserve the features of interest (i.e., halo cells) better than global-range error-bounded compression. The key reason is that according to the Nyx halo analysis code, the values in the range of $[81, 83]$ need to be extremely precise (due to the sophisticated physics, which we do not describe here).

Table III presents the error for two different post hoc analysis results using both the fallback compression method with a single global bound (set to 0.01 and 0.5) and our multirange approach with three ranges ($[min, 81) = 1$, $[81, 83) = 0.01$, and $[83, max) = 1$). We present RMSE (Root Mean Square Error) of cell number differences and mass differences of halos when compared with results obtained from the raw data. Specifically, when passed through the post hoc analysis, our solution can lead to lower RMSE for cell number and halo mass, which are 7% and 31%, respectively, compared with the RMSE under the global-range error-bounded compression.

TABLE III
COMPARISON OF DIFFERENT RANGE SETTINGS. FALLBACK SETS ONLY A GLOBAL ERROR BOUND (HERE 0.01 AND 0.5), AND MULTI-1-0.01-1 USES OUR MULTIRANGE ERROR-BOUNDED COMPRESSION WITH THREE ERROR BOUNDS ([MIN, 81)=1, [81, 83)=0.01, AND [83,MAX)=1).

| Method | RMSE of cell number | RMSE of halo mass |
|---|---|---|
| Fallback-0.01: | 0.089, | 125.84 |
| Fallback-0.5: | 2.820, | 429.26 |
| Multi-1-0.01-1: | 0.198, | 135.41 |

### C. Hurricane Katrina Simulation

Hurricane Katrina was one of the most devastating storms and the deadliest hurricane in the history of the United States because of its high storm surge (over 10 meters on the Mississippi coast) and high velocity. We use the hourly water elevation data downloaded from the ADCIRC website in this study, and the water elevation contour map with a 1-meter interval at four times—3:00 am and 17:00 pm UTC August 28 and 3:00 am UTC and 14:00 pm UTC August 29—was plotted for illustrative comparison.

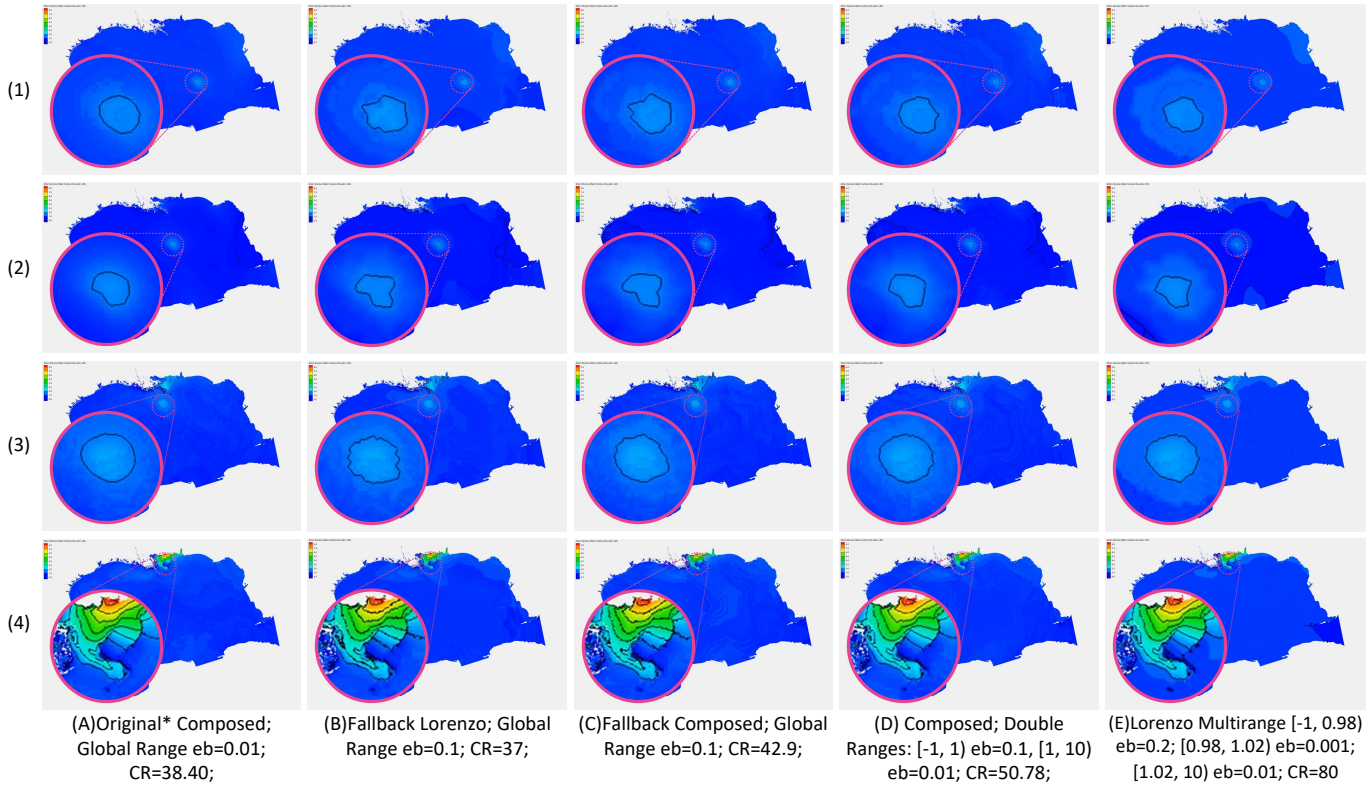


Fig. 5. Each row is a frame of the Hurricane Katrina simulation: (1) is frame 106, (2) is frame 120, (3) is frame 130, and (4) is frame 141. Each column represents a different setting of ranges and error bounds. Most of the blue data points in the graphs are close to zero. By applying a global range with error bound to be 0.01 with our solution, the visualization is almost identical to the original data, and therefore we use one column (A) to demonstrate the visualization result as a reference. The fallback version shown in (B) and (C) uses the original 1D SZ compressor, which does not handle the irrelevant data; thus it has the lowest compression ratio even with a higher error bound 0.1. "Composed" in (C) and (D) means we use a composed Lorenzo and linear regression predictor to predict values. "Lorenzo" in (E) means we use only the Lorenzo predictor with no linear regression. Comparing (B) and (C), our solution wins on the global range test by handling the irrelevant data and using the composed predictor (both Lorenzo and linear regression). Comparing (C) and (D), our multirange solution wins in both the compression ratio and visualization result. Comparing (D) and (E), we can further improve the compression ratio by using the Lorenzo predictor only and allowing some distortion in the deep blue area.

Katrina caused water elevation, and we consider elevations above 1 meter to be significant and therefore preserving a higher precision in that range. By applying multirange error-bound configuration, the compression quality (as shown in Figure 5) can be improved significantly compared with the original compression quality under the state-of-the-art SZ 2.1; see Figure 5 (D) and (E) vs. (B) and (C).

In the Katrina dataset, we found there are irrelevant data with value -99999 mixed in the data points, which represents the shoreline but breaks the smoothness of normal data points. The "Fallback" term means using the original SZ compressor without handling the irrelevant data, while our solution fixes the irrlevant data points during the compression. This side effect causes that (D)'s compression ratio is even higher than (B)'s although with a range of higher precision. We may further investigate the influence of the irrelevant data in the future, but in this case at least we can see that by modifying the setting of ranges and multiple error bounds, we can reach a better visual quality wihout sacrificing the compression ratio.

## VI. Summary

We investigated a novel compression approach that allows users to set different error bounds in various value ranges. Our approahc allows users to set the overall lossy compression quality that can be tolerated to meet their data fidelity requirement. Based on our evaluation using real-world simulation datasets, we report the following key findings.

- Multirange error-bound-based compression can significantly improve the visual quality for particular regions with the same or even higher compression ratios than traditional compression methods.
- In the Nyx cosmology simulation, the multirange error-bounded lossy compression can preserve the halo cells perfectly with a high compression ratio up to 78, while the global-range error-bounded compression suffers prominent distortion of cells.
- In the Hurricane Katrica simulation, multirange error-bounded compression can improve the compression ratio from 37 (based on SZ) to 80 (an improvement of 116%), even with higher data fidelity in maintaining the shape of Hurricane Katrina.

## 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, under contract DE-AC02-06CH11357, and supported by the National Science Foundation under Grant SHF-1617488, OAC-2003709 and OAC-2003624/2042084. We acknowledge the computing resources provided on Bebop, which is operated by the Laboratory Computing Resource Center at Argonne National Laboratory.

## REFERENCES

[1] J. Kay, C. Deser, A. Phillips, A. Mai, C. Hannay, G. Strand, J. Arblaster, S. Bates, G. Danabasoglu, J. Edwards *et al.*, "The community earth system model (CESM), large ensemble project: A community resource for studying climate change in the presence of internal climate variability," *Bulletin of the American Meteorological Society*, vol. 96, no. 8, pp. 1333–1349, 2015.

[2] S. Habib, V. Morozov, N. Frontiere, H. Finkel, A. Pope, K. Heitmann, K. Kumaran, V. Vishwanath, T. Peterka, J. Insley *et al.*, "HACC: extreme scaling and performance across diverse architectures," *Communications of the ACM*, vol. 60, no. 1, pp. 97–104, 2016.

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

[4] S. Di and F. Cappello, "Fast error-bounded lossy HPC data compression with SZ," in *IEEE International Parallel and Distributed Processing Symposium (IEEE IPDPS)*. IEEE, 2016, pp. 730–739.

[5] F. Cappello, S. Di, and et al., "Use cases of lossy compression for floating-point data in scientific data sets," *The International Journal of High Performance Computing Applications*, vol. 33, no. 6, pp. 1201–1220, 2019.

[6] X.-C. Wu, S. Di, E. M. Dasgupta, F. Cappello, H. Finkel, Y. Alexeev, and F. T. Chong, "Full-state quantum circuit simulation by using data compression," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. NY, USA: Association for Computing Machinery, 2019.

[7] D. Tao, S. Di, X. Liang, Z. Chen, and F. Cappello, "Improving performance of iterative methods by lossy checkponting," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 52–65.

[8] S. Jin, S. Di, X. Liang, J. Tian, D. Tao, and F. Cappello, "Deepsz: A novel framework to compress deep neural networks by using error-bounded lossy compression," in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '19. New York, NY, USA: ACM, 2019, pp. 159–170. [Online]. Available: http://doi.acm.org/10.1145/3307681.3326608

[9] D. Tao, S. Di, Z. Chen, and F. Cappello, "In-depth exploration of single-snapshot lossy compression techniques for n-body simulations," in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 486–493.

[13] A. H. Baker, H. Xu, J. M. Dennis, M. N. Levy, D. Nychka, S. A. Mickelson, J. Edwards, M. Vertenstein, and A. Wegener, "A methodology for evaluating the impact of data compression on climate simulation data," in *HPDC'14*, 2014, pp. 203–214.

[10] ——, "Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization," in *IEEE International Parallel and Distributed Processing Symposium (IEEE IPDPS)*. IEEE, 2017, pp. 1129–1139.

[11] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data—the univariate case," *Computing and Visualization in Science*, vol. 19, no. 5, pp. 65–76, Dec 2018.

[12] N. Sasaki, K. Sato, T. Endo, and S. Matsuoka, "Exploration of lossy compression for application-level checkpoint/restart," in *2015 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2015, pp. 914–922.

[14] A. H. Baker, D. M. Hammerling, and T. L. Turton, "Evaluating image quality measures to assess the impact of lossy data compression applied to climate simulation data," *Computer Graphics Forum*, vol. 38, no. 3, pp. 517–528, 2019.

[15] NYX simulation, https://amrex-astro.github.io/Nyx, online.

[16] 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 *2018 IEEE International Conference on Big Data*. IEEE, 2018.

[17] J. Diffenderfer, A. Fox, J. Hittinger, G. Sanders, and P. Lindstrom, "Error analysis of zfp compression for floating-point data," *SIAM Journal on Scientific Computing*, 02 2019.

[18] J. Zhang, X. Zhuo, A. Moon, H. Liu, and S. W. Son, "Efficient encoding and reconstruction of HPC datasets for checkpoint/restart," in *Proceedings of the 35th International Conference on Massive Storage Systems and Technology (IEEE MSST19)*, 2019.

[19] X. Delaunay, A. Courtois, and F. Gouillon, "Evaluation of lossless and lossy algorithms for the compression of scientific datasets in netcdf-4 or hdf5 files," *Geoscientific Model Development*, vol. 12, no. 9, pp. 4099–4113, 2019.

[20] C. S. Zender, "Bit grooming: statistically accurate precision-preserving quantization with compression, evaluated in the netcdf operators (nco, v4.4.8+)," *Geoscientific Model Development*, vol. 9, no. 9, pp. 3199–3211, 2016.

[21] 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.

[22] D. Tao, S. Di, X. Liang, Z. Chen, and F. Cappello, "Fixed-psnr lossy compression for scientific data," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, 2018, pp. 314–318.

[23] Zstd, https://github.com/facebook/zstd/releases, online.

[24] N. W. Scheffner, D. J. Mark, C. A. Blain, J. J. Westerink, and J. R. A. Luettich, "Adcirc: An advanced three-dimensional circulation model for shelves, coasts, and estuaries. report 5. a tropical storm database for the east and gulf of mexico coasts of the united states," *DRP Technical Report DRP-92-6*, 1994.

[25] QMCPack, https://qmcpack.org/, online.

[26] Miranda, https://wci.llnl.gov/simulation/computer-codes/miranda.

[27] Hurricane ISABELA Simulation Datasets, http://vis.computer.org/vis2004contest/data.html.

[28] M. Davis, G. Efstathiou, C. S. Frenk, and S. D. White, "The evolution of large-scale structure in a universe dominated by cold dark matter," *The Astrophysical Journal*, vol. 292, pp. 371–394, 1985.

[29] B. Friesen, A. Almgren, Z. Lukić, G. Weber, D. Morozov, V. Beckner, and M. Day, "In situ and in-transit analysis of cosmological simulations," *Computational Astrophysics and Cosmology*, vol. 3, no. 1, pp. 1–18, 2016.