# High-performance Effective Scientific Error-bounded Lossy Compression with Auto-tuned Multi-component Interpolation

JINYANG LIU, University of California, Riverside, USA
SHENG DI*, Argonne National Laboratory, USA
KAI ZHAO, Florida State University, USA
XIN LIANG, University of Kentucky, USA
SIAN JIN, Indiana University Bloomington, USA
ZIZHE JIAN, University of California, Riverside, USA
JIAJUN HUANG, University of California, Riverside, USA
SHIXUN WU, University of California, Riverside, USA
ZIZHONG CHEN, University of California, Riverside, USA
FRANCK CAPPELLO, Argonne National Laboratory, USA

Error-bounded lossy compression has been identified as a promising solution for significantly reducing scientific data volumes upon users' requirements on data distortion. For the existing scientific error-bounded lossy compressors, some of them (such as SPERR and FAZ) can reach fairly high compression ratios and some others (such as SZx, SZ, and ZFP) feature high compression speeds, but they rarely exhibit both high ratio and high speed meanwhile. In this paper, we propose HPEZ with newly-designed interpolations and quality-metric-driven auto-tuning, which features significantly improved compression quality upon the existing high-performance compressors, meanwhile being exceedingly faster than high-ratio compressors. The key contributions lie as follows: (1) We develop a series of advanced techniques such as interpolation re-ordering, multi-dimensional interpolation, and natural cubic splines to significantly improve compression qualities with interpolation-based data prediction. (2) The auto-tuning module in HPEZ has been carefully designed with novel strategies, including but not limited to block-wise interpolation tuning, dynamic dimension freezing, and Lorenzo tuning. (3) We thoroughly evaluate HPEZ compared with many other compressors on six real-world scientific datasets. Experiments show that HPEZ outperforms other high-performance error-bounded lossy compressors in compression ratio by up to 140% under the same error bound, and by up to 360% under the same PSNR. In parallel data transfer experiments on the distributed database, HPEZ achieves a significant performance gain with up to 40% time cost reduction over the second-best compressor.

CCS Concepts: • **Information systems → Data compression**; • **Theory of computation → Data compression**; • **Mathematics of computing → Interpolation**.

*Corresponding author

Authors' addresses: Jinyang Liu, University of California, Riverside, Riverside, CA, USA, jliu447@ucr.edu; Sheng Di, Argonne National Laboratory, Lemont, IL, USA, sdi1@anl.gov; Kai Zhao, Florida State University, Tallahassee, FL, USA, kzhao@cs.fsu.edu; Xin Liang, University of Kentucky, Lexington, KY, USA, xliang@cs.uky.edu; Sian Jin, Indiana University Bloomington, Bloomington, IN, USA, sianjin@iu.edu; Zizhe Jian, University of California, Riverside, Riverside, CA, USA, zjian106@ucr.edu; Jiajun Huang, University of California, Riverside, Riverside, CA, USA, jhuan380@ucr.edu; Shixun Wu, University of California, Riverside, Riverside, CA, USA, swu264@ucr.edu; Zizhong Chen, University of California, Riverside, Riverside, CA, USA, chen@cs.ucr.edu; Franck Cappello, Argonne National Laboratory, Lemont, IL, USA, cappello@mcs.anl.gov.


2836-6573/2024/2-ART4 $15.00
https://doi.org/10.1145/3639259

Additional Key Words and Phrases: error-bounded lossy compression, interpolation, scientific database

**ACM Reference Format:**
Jinyang Liu, Sheng Di, Kai Zhao, Xin Liang, Sian Jin, Zizhe Jian, Jiajun Huang, Shixun Wu, Zizhong Chen, and Franck Cappello. 2024. High-performance Effective Scientific Error-bounded Lossy Compression with Auto-tuned Multi-component Interpolation. *Proc. ACM Manag. Data* 2, 1 (SIGMOD), Article 4 (February 2024), 26 pages. https://doi.org/10.1145/3639259

## 1 INTRODUCTION

The gigantic scale and exceptionally intense computation power of modern supercomputers have empowered the exascale scientific simulation applications to generate tremendous amounts of data in short periods, bringing up significant burdens for distributed scientific databases and cloud data centers. For instance, A one-trillion particle Hardware/Hybrid Accelerated Cosmology Code (HACC) [15] can harness approximately 22PB output data in a single simulation, and Community Earth System Model (CESM) [20] simulation may generate 2.5PB data for a simulation task [41]. To this end, error-bounded lossy compression techniques have been developed for those scientific data, and they have been recognized as the most proper strategy to manage the extremely large amount of data. The advantage of error-bounded lossy compression is primarily two-fold. On the one hand, it can reduce the original data to an incredibly shrunken size which is much smaller than the compressed data size generated by a lossless compressor. On the other hand, the error-bounded lossy compression can constrain the point-wise data distortion strictly upon the users' requirements. Existing state-of-the-art error-bounded lossy compressors in diverse archetypes, such as prediction-based model – SZ3 [32, 53] and QoZ [35], transform-based model – ZFP [33] and SPERR [27], and dimension-reduction-based model – TTHRESH [7], have been widely adopted in many use cases in practice.

Considering the abundant scope of related optimization strategies, we summarize the existing error-bounded lossy compressors as well as their pros and cons as follows. The orthogonal transform-based compressors like ZFP, exhibit high execution speeds but their compression ratios are limited to a certain extent because they focus on only local correlations (confined within $4^d$-blocks). The wavelet-based compressors such as SPERR and the Singular Value Decomposition (SVD) based compression such as TTHRESH, although can obtain quite high compression ratios, suffer from very low compression speeds attributed to their high-cost integrated data operation modules. Some prediction-based compressors (e.g. SZ3 and QoZ) deliver relatively high compression ratios with moderate running speeds, nevertheless, they may suffer from relatively low compression ratios in some cases. Recently, FAZ [36] attempted to create a hybrid framework taking advantage of heterogeneous compression techniques, however, its design fully orients the optimization of rate-distortion, so that its compression/decompression is much slower than the classic compressors such as SZ and ZFP.

For modern scientific databases and cloud data centers which often involve multiple sites over a wide area network (WAN), the extremely large amount of raw data costs an unacceptable time to transfer between machines. Therefore, data compressors are critical for efficient data transfer because transferring compressed data will significantly reduce the time cost, as confirmed by prior research [26, 38]. In this case, compression ratios and speeds are both critical for achieving high data transfer throughput. However, designing a versatile error-bounded lossy compressor that delivers high compression ratios with sufficient performance (i.e. speed) is quite challenging. On one hand, to reach a high compression performance, general techniques have to perform relatively simple data transform [33] or prediction within short-range areas [5, 30, 53], which cannot take advantage of long-range data correlations, thus leading to very limited compression ratios inevitably. On the other hand, to reach a high compression ratio, general techniques are applying sophisticated

techniques such as wavelet transform on the full data input [27, 36] or higher-order SVD [7], which suffer from very expensive operations inevitably, conflicting with our high-performance objective. As such, we must design more compact and effective data operation methods with relatively low computational costs, featuring high speed meanwhile yielding comparable compression ratios compared to the existing high-ratio compression techniques.

In order to design an error-bounded compressor that features both high compression ratios and satisfactory speeds, we propose an optimized quality-metric-driven error-bounded lossy compressor (called HPEZ) by developing a brand-new auto-tuning strategy and an anchor-based level-wise hybrid interpolation predictor. Integrating extensively optimized interpolation predictors and auto-tuning modules, HPEZ attains far better compression ratios and lower distortions than other high-performance error-bounded lossy compressors with limited compression speed degradation. HPEZ substantially outperforms high-ratio compressors in terms of speed. It achieves optimized throughput performance in a variety of use cases such as parallel data transfer for large (distributed) databases. We attribute our contributions as follows:

- Founded on theoretical analysis and algorithmic optimizations, we substantially upgrade the most critical step in the quality-oriented compression – interpolation prediction, leading to an immensely improved data prediction accuracy.
- We develop a series of optimization strategies including block-wise interpolation tuning, dynamic dimension freezing, and Lorenzo tuning, which can substantially improve the adaptability of the auto-tuning for the compression across a broad spectrum of inputs.
- We perform solid experiments using 6 real-world scientific datasets. HPEZ significantly outperforms state-of-the-art error-bounded lossy compressors in terms of rate-distortion, while still having a satisfactory speed. It preserves a leading speed compared to other high-ratio compressors. Consequently, it achieves the best throughput in distributed data transfer over WAN based on our experiments. HPEZ exhibits the least time cost in data transfer for most scientific datasets with up to 40% time reduction.

The remainder of this paper is organized as follows: Section 2 introduces related works. Section 3 provides the research background and the research problem formulation. Section 4 demonstrates the overall framework of HPEZ. The new designs of interpolation predictors in HPEZ are illustrated in detail in Section 5, and our designed auto-tuning blocks are proposed in Section 6. In section 7, the evaluation results are presented and analyzed. Finally, Section 8 concludes our work and discusses future work.

## 2 RELATED WORK

In general, scientific data compression techniques can be divided into two categories - lossless compression and lossy compression. Examples of existing lossless compressors for databases are Gorilla [40] and AMMO [48] for time-series data, and traditional lossy data compression methods include ModelarDB [18, 22] for time-series data and [13, 25, 28, 51] for Geology spatial-temporal data. Besides that, error-bounded lossy compression has been preferred and crafted to serve various scientific data reduction applications [9] and scientific databases. To meet the requirement of scientists, the error-bounded lossy compression needs to constrain the point-wise compression errors within a certain value, which differs from compression techniques for traditional data such as JPEG-2000 [44] for image data and h.265 [42] for video data. The error-bounded scientific compressors are classified into four main categories: prediction-based, transform-based, dimension-reduction-based, and neural-network-based. They also essentially utilize approaches to manage the data distortion in line with user-specified error bounds.

The prediction-based compressors use data prediction techniques, like linear regression [30] and dynamic spline interpolations [53], to anticipate the data points. Well-known examples are SZ2 [30] and SZ3 [32, 53]. Transform-based compressors, on the other hand, use data transformations to decorrelate the data, then switch to compress the more compressible transformed coefficients. ZFP [33], for example, is a typical example that employs exponent alignment, orthogonal discrete transform, and embedded encoding. SPERR [27], a more recent work, leverages wavelet transform for data compression. Dimension-reduction-based compressors apply dimension reduction techniques, with (high-order) singular vector decomposition (SVD) being a case in point (for instance, TTHRESH [7]). Neural-network-based compressors [14, 17, 34, 37] utilize neural network models like the autoencoder family [8, 23, 24], however, the speeds of them and relatively quite slow.

The aforementioned compressors each have their strengths and weaknesses, depending on the nature of the input data and user needs. To enhance scientific error-bounded lossy compression, two emerging approaches are raised to further refine the specialization of the compressor or to boost its versatility. Regarding compressor specialization, MDZ [52], a prediction-based compressor, is specifically tailored for molecular dynamics simulation data. SZx [49] offers low-ratio lossy compression at incredibly high speeds. CuSZ [45], CuSZ+ [46] and FZ-GPU [50] delve into GPU-based scientific lossy compression to quicken the compression process. [19] aims at maintaining the quantities of interest (QoI) of the input data. When it comes to enhancing the versatility of lossy compressors, QoZ [35] integrates user-specified quality metric optimization targets and anchor-point-based level-wise interpolation auto-tuning into the SZ3 compression framework. This can effectively improve the compression quality with limited speed degradation. FAZ [36], a hybrid compression framework, combines diverse compression techniques and adaptively generates the compression pipeline for varying inputs, while suffering from low compression speed.

With all those evolving works taken into insight, there is still a lack of broad-spectrum scientific error-bounded lossy compressors that can achieve both top-tier compression quality and adequate compression speed. In this paper, our proposed solution endeavors to fill this gap: we pursue both high compression quality (by optimizing the rate-distortion) and high execution throughput across a wide range of scientific datasets.

## 3 PROBLEM FORMULATION AND ANALYSIS

In this section, we mathematically formulate our research target and then present the fundamental analysis for addressing the target. With those analyses, we can determine the best-fit archetype for the to-be-proposed compressor HPEZ.

### 3.1 Problem Formulation

The target of HPEZ is to jointly optimize the compression ratio and the user-specified quality metrics (PSNR, SSIM, etc.). Moreover, the proposed new compressor is expected to have relatively high compression and decompression speeds and be well-adapted to diverse types of input data (integer and floating point, single-dimensional and multi-dimensional, and so on).

Eq. 1 is the formulated research target in this paper. A compressor $C$ and a decompressor $D$ compose the error-bounded lossy compression framework, together with their configuration parameters (denoted by $\theta$). With the input data (denoted by $X$) and a user-specified absolute error bound $e$, the compression framework generates compressed data (denoted as $Z = C_\theta(X)$) and the decompressed data (denoted as $X' = D_\theta(Z)$), which should strictly respect the error bound (denoted $e$) point-wisely. Under those mandatory conditions, HPEZ determines $C$, $D$, and $\theta$ by optimizing the compression ratio under a user-specified quality metric requirement (denoted as $m_0$). Each quality metric corresponds to (and is calculated from) a function $M$, which can be chosen from PSNR, SSIM, a constant function (in case no quality but just compression ratio is concerned), etc.

Moreover, to ensure the applicability of our proposed compressor for various use cases, we would like the proposed compressor to become a high-performance compressor (including SZ3, QoZ, et al.) having an overall execution speed of at least comparable to SZ3.

$$\begin{aligned} C, D, \theta &= \arg\max_{C,D,\theta} \frac{|X|}{|Z|} \\ s.t.\ |x_i - x_i^{'}| &\leq e, \forall x_i \in X \\ M(X, X^{'}) &= m_0 \end{aligned} \tag{1}$$

### 3.2 Determining the Best-fit Compressor Archetype for HPEZ

As mentioned before, our proposed compressor should exhibit both good rate-distortion and relatively high speeds. To this end, we need to investigate existing scientific error-bounded lossy compressors to identify the best-fit compressor archetype for our design. The categorization of compressors is priorly discussed in Section 1 and Section 2, but to conduct a deeper analysis here we categorize the existing compressors into more types according to their designs:

- Hybrid-data-prediction-based: Applying multiple data predictors for data prediction and reconstruction, such as regressors and Lorenzo predictors [30, 55].
- Interpolation-based: Leveraging interpolations for prediction-based data compression [35, 53].
- Discrete-orthogonal-transform-based: Making use of block-wise Discrete Orthogonal Transform and embedded coding in the compression [33].
- Wavelet-transform-based: Combining wavelet transforms and coefficient encoding methods for compression [27, 36].
- SVD-based: In TTHRESH [7], high-order singular value decomposition is the core of its data processing techniques.
- Deep-learning-based: Quite a few deep-learning-based error-bounded lossy compressors have been proposed. Among them, there are autoencoder-based ones [17, 34] and coordinate-network-based ones [16, 39].

Several existing works [34–36] have also conducted systematic and thorough experimental analyses of those compressors in diverse types, having tested them in multiple aspects including and not limited to execution speeds, rate-distortion, and practical use cases (e.g. I/O throughput). We conclude their findings as follows:

- Despite their great potential in achieving high compression ratios, wavelets-based and SVD-based compressors suffer from low compression speeds due to high computational costs. With fixed data processing strategies, certain examples of them such as SPERR and TTHRESH also fail to perform well in terms of rate-distortion on some data inputs.
- Discrete-orthogonal-transform-based ZFP has a very high compression efficiency, but it only presents quite limited compression ratios.
- The practicality of current deep-learning-based compressors is also not satisfactory. The networks integrated into them either need per-data online training (for each compression task) or large sizes of training data from the same application for pre-training. This fact greatly damages the availability and efficiency of deep-learning-based compressors.
- Compared with others, prediction-based compressors (including hybrid-data-prediction-based and interpolation-based ones) have the advantage of achieving both good compression ratios and acceptable compression speeds. Among them, interpolation-based compressors such as SZ3 [32] and QoZ [35] optimize the compression rate-distortion. In the experiments carried out by [35], QoZ shows the best performance in the parallel I/O throughput tests.

According to our research target and the pros and cons of existing compressor archetypes, we develop a novel high-performance effective compressor namely HPEZ based on the interpolation-based compressor design. In Section 4, 5, and 6, we will fully demonstrate the design details of HPEZ, including the research background and newly proposed features.

## 4 HPEZ DESIGN OVERVIEW

In this section, we propose an overview of the HPEZ compressor. As an interpolation-based scientific error-bounded lossy compressor, HPEZ is designed for structured data grids in types of floating points and integers. HPEZ is adaptive to either one-dimensional (1D) or multi-dimensional (2D, 3D, 4D ...) inputs, and exploits the dimension-wise spatial correlations and smoothness of them. HPEZ also has the potential to be applied to other domains including image and video because those data are also formatted as (or can be transformed into) structured data grids. The compression framework of HPEZ is illustrated in Figure 1. HPEZ takes advantage of the SZ3 modular framework [32], which contains the auto-tuning module, data prediction module, error quantization module, Huffman encoding module, and the Zstd lossless module. The detailed demonstration of the HPEZ compression pipeline is as follows:

- **Step 1: Auto-tuning**. With a user-specified quality metric optimization target, HPEZ first auto-tunes its predictor configurations, which will be featured in Section 6.
- **Step 2: Data prediction**: HPEZ applies the auto-tuned data predictor on the whole input, acquiring the prediction errors.
- **Step 3: Linear quantization (error control)**: A linear error quantization module quantizes the data prediction errors in step 2 to control the element-wise decompression error. For example, for each data value $x$ and its prediction $x^{'}$, the original error is $e = x - x^{'}$ and the quantized error $e_q$ satisfies $|e_q - e| <= \epsilon$ ($\epsilon$ is the error bound). In this way, we can use $x^{'} + e_q$ as the decompression of $x$ which is bounded by $\epsilon$.
- **Step 4: Huffman encoding**: The quantized prediction errors acquired from Step 3 are further encoded with Huffman encoding. A more concentrated distribution of quantization errors will lower the encoded tree size, therefore the reduction of prediction error is key to improving the compression ratio.
- **Step 5: Lossless postprocessing**: The encoded quantized errors and other metadata are losslessly compressed by Zstd [12] to further reduce the compressed size.

HPEZ leverages existing modules in stereotype prediction-based error-bounded compression model (orange ones in Figure 1) and interpolation techniques (yellow ones in Figure 1). Most importantly, our HPEZ framework introduces several new modules and significantly improved components (as marked in blue and pink), including interpolation designs and auto-tuning techniques. In the data prediction module and the auto-tuning module, new designs have been incorporated in HPEZ to enhance the compression rate-distortion substantially. With those new designs, first, we have significantly improved the interpolation-based data predictors in HPEZ, introducing multiple refinements upon the existing dynamic spline interpolation; Second, the auto-tuning module of HPEZ has also been facilitated with new components for handling new interpolation configurations and boosting adaptability for more datasets. Third, the compression speed of HPEZ still maintains at a high level, empowering it to well-fit efficiency-oriented tasks. Those newly proposed designs will be demonstrated in Section 5 and Section 6.

## 5 HPEZ INTERPOLATION-BASED PREDICTOR

In this section, we describe the details of our fine-tuned multi-component interpolation-based data predictor for HPEZ. Compared to the existing interpolation-based predictors, the HPEZ

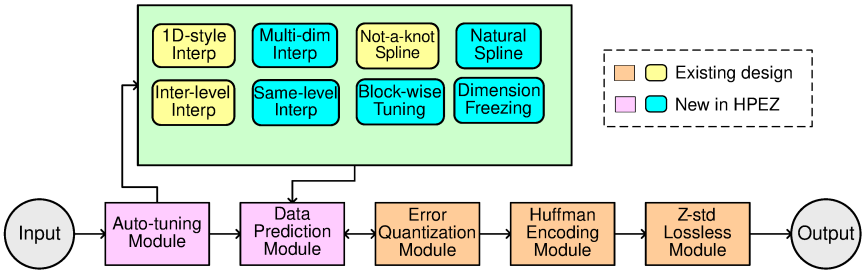



Fig. 1. HPEZ framework

interpolation-based predictor projects a significant improvement over them, attributed to several new components we designed and proposed. These components can obtain a significantly improved prediction accuracy, thus leading to much better rate distortions in the compression. Those new designs together with the existing interpolation designs will be described in the rest of this section and will get auto-tuned for optimization of compression quality (to be detailed in Section 6).

### 5.1 Overview of Interpolation-based Prediction

The interpolation-based data prediction and reconstruction in HPEZ follow the hierarchical anchor-based level-wise dynamic spline interpolation concept, whose prototype was first proposed in SZ3 [53] and then developed in QoZ [35]. Figure 2 presents the interpolation-based data prediction process in the QoZ compressor. Initialized with a sparse losslessly-saved grid, on each interpolation level, the predictor expands the predicted/reconstructed data grid by 2× (on each dimension), until all data points are predicted/reconstructed. The interpolations with larger strides are performed at higher levels, and the interpolation stride reduces (halved) as the level goes down. We refer the readers to read [35] for details. The key features of QoZ level-wize interpolation method include:

- Storing anchor points losslessly (with a fixed anchor stride);
- The interpolations are done hierarchically (level by level), from large strides (half of the anchor stride) to small strides (1).
- Each level may have different error bounds. Higher levels have smaller error bounds, and the last level always follows the input global error bound.
- Leveraging both linear (first-order) and cubic (third-order) 1-D spline interpolation;
- Performing the interpolation along each dimension;
- Selecting the best-fit interpolation method for each level;
- Auto-tuning and applying different error-bound values dynamically for different levels;

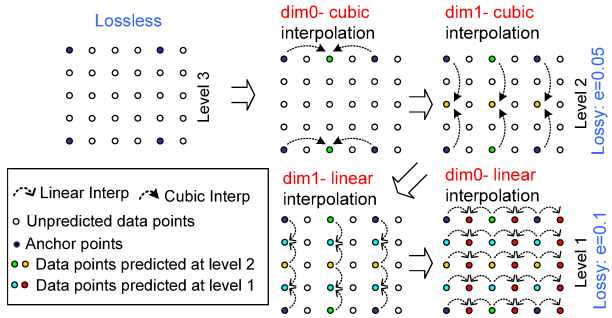



Fig. 2. The anchor-based level-wise dynamic spline interpolation.

Such an anchor-based level-wise interpolation prediction features three critical advantages. (1) The prediction has a very low time complexity: $O(N)$, where $N$ is the total number of data points in

the input dataset. This is because, for the prediction of each data point, the interpolation is executed just once with an upper-bounded number of neighbor points (e.g. for SZ3/QoZ the upper-bound is 4), and the quantization of its prediction error is also completed in constant time. (2) The level-wise design allows it to set various error bounds at different levels to minimize the negative impact of data compression errors in the data prediction. (3) The design of anchor points avoids inaccurate large-stride interpolations, maintaining its prediction accuracy at a relatively high level.

Although the interpolation-based prediction in HPEZ is built upon QoZ, HPEZ proposes several key improvements that significantly boost its prediction accuracy over QoZ, including:

- The natural cubic spline function;
- The multi-dimensional spline interpolation;
- Re-ordering of the interpolations.

Next, we will take a deep insight into the interpolation-based prediction in HPEZ, thoroughly demonstrating both the backgrounds and the new characteristics.

### 5.2 Spline Interpolation Formulas

All interpolations in HPEZ are based on certain spline interpolation formulas, which interpolate each data point with its neighbors along one dimension. As mentioned in Section 5.1, the spline interpolation formulas are categorized into linear spline interpolation and cubic spline interpolation. Illustrated in Figure 3, the data value $d_i$ on index $i$ is going to be predicted by a prediction $p_i$ with the known data points $d_{i-3}$, $d_{i-1}$, $d_{i+1}$, and $d_{i+3}$ in its neighbours. The linear spline interpolation just applies 2 of them with the following formula:

$$p_i = \tfrac{1}{2}d_{i-1} + \tfrac{1}{2}d_{i+1} \tag{2}$$

The cubic spline interpolation formulas leverage all the 4 neighbor points, and the formulas are deducted from 3 cubic spline functions ($f_1(x)$, $f_2(x)$, and $f_3(x)$):

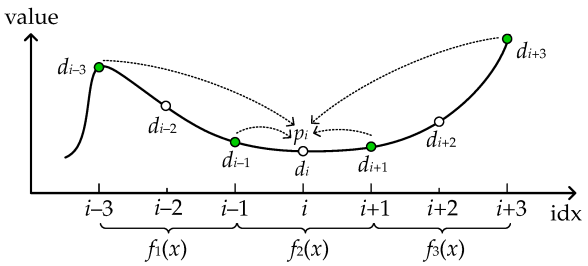



Fig. 3. Illustration of 1D cubic spline interpolation.

$$\begin{aligned} f_1(x) &= a_1(x-(i-3))^3+b_1(x-(i-3))^2+c_1(x-(i-3))+\delta_1 \\ f_2(x) &= a_2(x-(i-1))^3+b_2(x-(i-1))^2+c_2(x-(i-1))+\delta_2 \\ f_3(x) &= a_3(x-(i+1))^3+b_3(x-(i+1))^2+c_3(x-(i+1))+\delta_3 \end{aligned} \tag{3}$$

The spline functions $f_1$, $f_2$, and $f_3$ have scopes of $[i-3,i-1]$, $[i-1,i+1]$, and $[i+1,i+3]$, respectively. The zero-order, first-order, and second-order interpolation conditions are shown as follows:

$$\begin{aligned} & f_1(i-3) = d_{i-3};\ f_1(i-1) = d_{i-1} \\ & f_2(i-1) = d_{i-1};\ f_2(i+1) = d_{i+1} \\ & f_3(i+1) = d_{i+1};\ f_3(i+3) = d_{i+3} \\ & f_1^{'}(i-1) = f_2^{'}(i-1);\ f_2^{'}(i+1) = f_3^{'}(i+1) \\ & f_1^{''}(i-1) = f_2^{''}(i-1);\ f_2^{''}(i+1) = f_3^{''}(i+1) \end{aligned} \tag{4}$$

Since $f_1$, $f_2$, and $f_3$ have 12 coefficients in total and Eq. 4 only has 10 conditions, two more boundary conditions are needed. The traditional SZ3 and QoZ cubic spline interpolation [35, 53] applies the following 'not-a-knot' conditions:

$$f_1'''(i-1) = f_2'''(i-1);\ f_2'''(i+1) = f_3'''(i+1) \tag{5}$$

Then with Eq. 4 and Eq. 5, the prediction value of $p_i$ is:

$$p_i = f_2(i) = -\tfrac{1}{16}d_{i-3} + \tfrac{9}{16}d_{i-1} + \tfrac{9}{16}d_{i+1} - \tfrac{1}{16}d_{i+3} \tag{6}$$

However, there are other choices for the 2 boundary conditions, which may lead to different cubic spline interpolation formulas. We explore another set of boundary conditions: the natural spline condition, which is:

$$f_1''(i-3) = 0; f_3''(i+3) = 0 \tag{7}$$

Combining Eq. 4 and Eq. 7, the interpolation function for predicting $p_i$ would be written as:

$$p_i = f_2(i) = -\tfrac{3}{40}d_{i-3} + \tfrac{23}{40}d_{i-1} + \tfrac{23}{40}d_{i+1} - \tfrac{3}{40}d_{i+3} \tag{8}$$

Our experiments with multiple datasets under diverse error thresholds showed that Eq. 2, Eq. 6, and Eq. 8 have distinct advantages. In different cases, each of them is able to outperform others. Therefore, we employ all 3 of them and dynamically select from them for each task.

### 5.3 1D and Multi-dimensional Spline Interpolation

In traditional interpolation-based compressors, for each data point, the interpolation is performed along a single dimension, so we need to switch the interpolation directions during this process and arrange an order for those directions. In the following text, we call the interpolation method adopted by SZ3/QoZ *1D-style interpolation*. As an example, in Figure 4 (a), the 1D-style interpolation first proceeds interpolations along Dim0, then performs the rest of the interpolations along Dim1.

Actually, The existing 1D-style interpolation has not fully exploited the multi-dimensional continuity and smoothness of input data arrays, because all the interpolations are constricted in a single-dimensional direction. To address this limitation, we propose a new interpolation paradigm for HPEZ called multi-dimensional spline interpolation, which can take better advantage of data correlation across multiple dimensions. As shown in Figure 4 (b), the multi-dimensional spline interpolation initially performs the 1D interpolations for some data points as there are only 1D neighbors at the moment, then it performs 2D interpolations for the remaining data points that already have neighbors in two dimensions. The multi-dimensional spline interpolation is symmetric across all the dimensions, meaning that it does not need a selection of dimensional order.

With the main concept of the HPEZ multi-dimensional spline interpolation in mind, two questions remain: how should we carry out the multi-dimensional interpolations specifically, and why does it outperform the 1D-style interpolations?

We feature the HPEZ multi-dimensional interpolation as follows. For each data point $x$, suppose $X_i$ $(1 \le i \le n)$ are all the available 1D interpolation results for predicting $x$ (which can either be linear interpolation or cubic interpolation and are along all dimensions), the multi-dimensional interpolation result $X'$ is a linear-combination of $X_i$:

$$X' = \sum_{i=1}^{n} \alpha_i X_i \quad (\sum_{i=1}^{n} \alpha_i = 1) \tag{9}$$

Theorem 5.1. *With fine-tuned $\alpha_i$, $X'$ would have a no higher prediction error than that of the 1D-style interpolation $X_i$.*

Proof. Without loss of generality, we can regard $\{X_i\}$ and $X'$ as random variables, in which $\{X_i\}$ are independent with each other. When dealing with smooth data inputs, the $\{X_i\}$ can be thought of as no-biased estimations of $x$, i.e. $E(X_i) = x$.

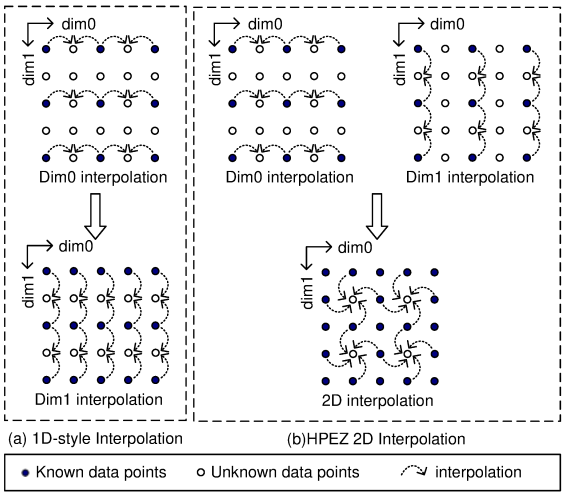



Fig. 4. Comparison of 1D-style interpolation and HPEZ multi-dimensional interpolation (an 2D example).

Now consider the $X^{'}$. Since $\sum_{i=1}^{n} \alpha_i = 1$, it is easy to know that $E(X^{'}) = x$, so $X^{'}$ is still a non-biased estimation of $x$. Because $X_i$ are independent with each other, $(X^{'} - x) = \sum_{i=1}^{n} \alpha_i (X_i - x)$ follows the distribution of $N(0, \sigma^2)$, in which:

$$\sigma^2 = \sum_{i=1}^{n} \alpha_i^2 \sigma_i^2 \tag{10}$$

With the Lagrange method, based on the constraint $\sum_{i=1}^{n} \alpha_i = 1$,

$$\min \sigma^2 = \frac{\prod_{i=1}^{n} \sigma_i^2}{\sum_{i=1}^{n} \pi_i} \leq \min\{\sigma_1^2, \sigma_2^2, ...\sigma_n^2\} \ (\pi_i = \frac{\prod_{j=1}^{n} \sigma_j^2}{\sigma_i^2}) \tag{11}$$

, and the minimum is obtained when:

$$\alpha_i^* = \frac{\pi_i}{\sum_{j=1}^{n} \pi_j} \tag{12}$$

As such, we have proved that, if the $\{\alpha_i\}$ is selected based on Eq. 12, the prediction error variance of the multi-dimensional interpolation $X^{'}$ will be no larger than each of the 1D-style interpolation $X_i$ according to Eq. 11. So, the average L-1 prediction error will also be minimized. □

How to determine $\alpha_i^*$ in HPEZ (i.e. how to estimate $\sigma_i^2$) will be detailed in Section 6.

### 5.4 Interpolation Re-ordering

After the proposal of natural cubic spline and multi-dimensional interpolation, HPEZ also introduces interpolation re-ordering, which improves both prediction accuracy and prediction speed. It includes two aspects: the fast-varying-first interpolation and same-level cubic interpolation.

*5.4.1 Fast-varying-first interpolation.* In the existing implementation of 1D interpolations, the interpolations are executed axis by axis on the input dataset, and along each axis, the interpolations are performed 'slice by slice'. The 'slice' here means a slice of the data array along an interpolation axis. Figure 5 (a) presents a 2D example for the order of interpolations adopted by QoZ (and also SZ3): the interpolations are performed in the sequence of numbers (①, ②, ③, $\cdots$). For the interpolation along Dim0 in QoZ, it follows dim0-major order: the interpolation is executed along Dim0 with a higher preference compared with Dim1. However, when Dim1 is the fastest-varying-dimension ,

this interpolation order may fall into a bad cache usage because it is successively accessing data points located distantly in the memory. To resolve this issue, HPEZ re-arranges the interpolation order, having the interpolations first move along the fast-varying dimension (the Dim1-major style as in Figure 5), as demonstrated in Figure 5 (b). The interpolation position first traverses through Dim1 and then moves along Dim0. In this way, the data points are accessed sequentially with shorter distances in the memory so that the cache usage can be optimized, greatly saving the memory access cost.

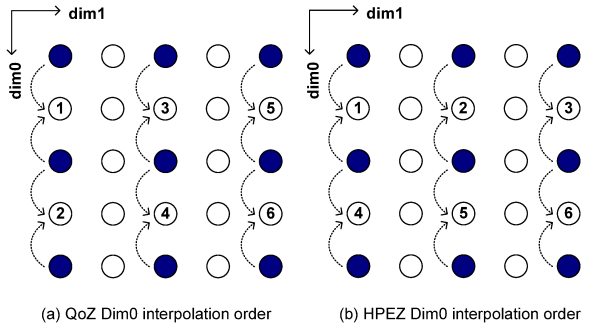



Fig. 5. Comparison of QoZ and HPEZ interpolation orders (Dim1 is the fastest-varying dimension)

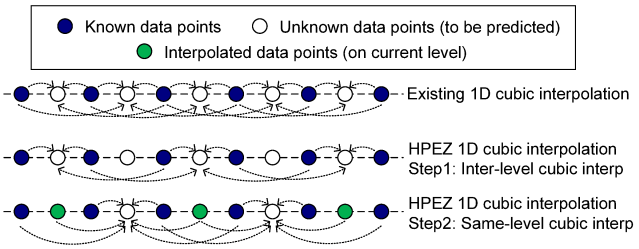



Fig. 6. Illustration of same-level cubic interpolation

*5.4.2 Same-level cubic interpolation.* We develop a new same-level cubic interpolation in HPEZ, which can further improve prediction accuracy. In the traditional interpolation design [35, 53], at each interpolation level, the neighbor points of each data point to be interpolated are limited on the higher levels (interpolation levels with larger strides). For the 1D cubic spline interpolation applied on a data point with stride $s$, 4 neighbor points with distance $s$ and $3s$ are used, which have been predicted on the higher interpolation levels. As shown in Figure 6 (note that $s$ is the distance between each closest hollow and solid point), the first row shows this interpolation method, in which all the hollow data points (on the current interpolation level) are predicted by the solid data points (on higher interpolation levels). If we are able to include more neighbors for each point (for example, the 2 white points with a distance of $2s$ to it), the prediction accuracy can be improved. As illustrated in the 2nd and 3rd rows of Figure 6, instead of traversing through all the white data points in one step, HPEZ splits the 1D cubic spline interpolation into 2 steps. In the first step (the second row of Figure 6), half of the white points are interpolated by inter-level interpolation (the existing interpolation) with 4 neighbor points. In the second round, the rest half of the white points are interpolated by the same-level interpolation with 6 neighbor points for each, including points interpolated on higher interpolation levels and the current interpolation level. With this new interpolation, half of the data points are predicted with two more neighbor points to achieve better prediction accuracy. Similar to the deductions in Section 5.2, for a data point $p_i$, with its 6 neighbor

points $d_{i-3}$, $d_{i-2}$, $d_{i-1}$, $d_{i+1}$, $d_{i+2}$, and $d_{i+3}$ the same-level cubic spline interpolation formula would be the following two. Eq. 13 is for the not-a-knot cubic spline and Eq. 14 is for the natural cubic spline. The same strategy can also be extended to the multi-dimensional interpolation, splitting it into 2 steps each with halved data points.

$$p_i = -\frac{1}{6}d_{i-2} + \frac{4}{6}d_{i-1} + \frac{4}{6}d_{i+1} - \frac{1}{6}d_{i+2} \tag{13}$$

$$p_i = \frac{3}{62}d_{i-3} - \frac{18}{62}d_{i-2} + \frac{46}{62}d_{i-1} + \frac{46}{62}d_{i+1} - \frac{18}{62}d_{i+2} + \frac{3}{62}d_{i+3} \tag{14}$$

## 6 HPEZ AUTO-TUNING MODULES

we developed an advanced auto-tuning module in HPEZ, which plays a critical role in preserving and optimizing the compression quality by making the best use of the abundant interpolation options offered by HPEZ which are discussed in Section 5. Figure 7 displays all the components and processes of the HPEZ auto-tuning module. This module inherits the interpolation error-bound tuning process from QoZ [35], while substantially upgrading the QoZ 'global' interpolation tuning process. Specifically, HPEZ exploits several brand-new processes: dynamic dimension freezing tuning, block-wise interpolation tuning, Lorenzo tuning, and a data sampling/statistical analysis process supporting those tuning processes. In the remainder of this section, we present the detailed design of the auto-tuning-related components in HPEZ.

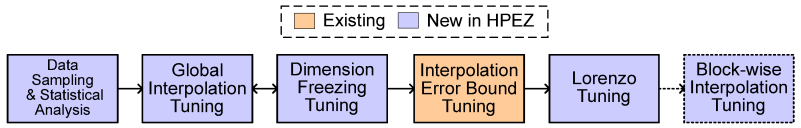



Fig. 7. HPEZ auto-tuning module

### 6.1 Data Sampling and Statistical Analysis

The data sampling and statistical analysis is an auxiliary process of the HPEZ auto-tuning module. In this process, HPEZ uniformly samples a small portion from the full data input (based on a hyper-parameter with the default sampling rate of 0.2%), and then it performs the 1D interpolation (both linear and cubic) on those data points with their neighbors along all dimensions. Afterward, the mean square errors (MSE) of the interpolations along different dimensions can serve as the estimations of the interpolation error variances ($\sigma_i^2$) described in Section 5.3. Thus, it can be used to determine the most non-smooth dimension in the data for dynamic dimension freezing (Section 6.3) by selecting the dimension with the largest interpolation MSE.

### 6.2 Global Interpolation Tuning

The global interpolation tuning process in HPEZ is derived from the predictor tuning process proposed in QoZ, which aims to select the best-fit interpolation configuration from different choices Specifically, at each interpolation level, the global interpolation tuning process makes the following selection for the input data:

- **Existing in QoZ**: The order of interpolation (linear or cubic);
- **Existing in QoZ**: The dimensional order (only for 1D-style interpolation);
- **New in HPEZ**: The type of cubic spline (not-a-knot or natural, only for cubic interpolation);
- **New in HPEZ**: The interpolation paradigm (1D-style or multi-dimensional);
- **New in HPEZ**: Applying inner-level interpolation or not (only for cubic interpolation);

Similar to QoZ, the sampled data are used for performing compression tests with all the available interpolation configurations. Then, HPEZ selects the interpolation configuration with the lowest average absolute prediction error as the final tuning result.

### 6.3 Dynamic Dimension Freezing

The dynamic dimension freezing in HPEZ is designed to avoid inaccurate interpolation predictions along non-smooth dimensions. For a multi-dimensional input data array, it may present fine smoothness along some of its dimensions but present bad smoothness along the other dimensions. In those cases, both the 1D-style and multi-dimensional interpolation will fail in achieving high prediction accuracy as they will involve interpolations along non-smooth directions. The dimension freezing is that, given one dimension, HPEZ sets anchor points along those dimensions with stride 1 (without intervals) and never performs interpolations along those dimensions. Figure 8 uses the interpolation on a 3D data block as an example of dimension freezing. For a clear view, only the 1D interpolations are shown. Figure 8 (a) is the normal 1D interpolations without a frozen dimension, and Figure 8 (b) is the 1D interpolations with a dimension frozen, in which no interpolations are made along the frozen dimension. With this dynamic strategy, HPEZ does not require data smoothness along all dimensions to optimize its compression ratio. According to our experimental results, compared to the highly improved prediction accuracy and greatly reduced quantization bin size, the storage overhead for additional anchor points is affordable. To determine whether to freeze a dimension and which dimension should be frozen, the auto-tuning module of HPEZ first specifies the most non-smooth dimension in the input data array in the statistical analysis (Section 6.1), then separately tunes 2 optimized interpolation configurations with/without this dimension frozen. If freezing this dimension presents a better compression ratio, HPEZ will freeze this dimension.

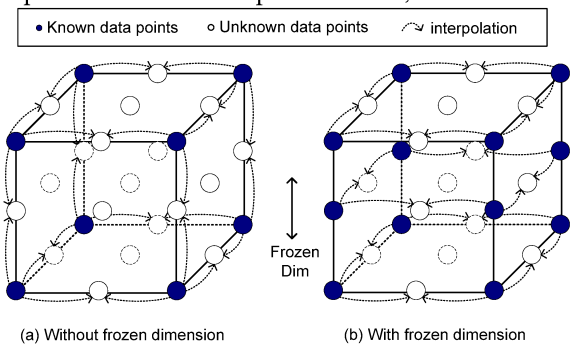



Fig. 8. Illustration of dimension freeze

### 6.4 Interpolation Error Bound Tuning

Previously indicated in Section 5.1, the HPEZ interpolations on each interpolation level follow a separate dynamically auto-tuned error bound. For the level-wise error-bound setting, HPEZ follows the same design as in QoZ [35]. The error bound for each level is computed by Eq. 15, in which $\alpha$ and $\beta$ are tunable parameters:

$$e_l = \frac{e}{min(\alpha^{l-1}, \beta)} \ (\alpha \geq 1 \ and \ \beta \geq 1) \tag{15}$$

In the auto-tuning process for determining $\alpha$ and $\beta$, HPEZ also leverages the module proposed in QoZ. We refer the readers to check [35] for details.

### 6.5 Tuning with Lorenzo Predictor

Leveraged in SZ3 but excluded by QoZ, the dynamic-order Lorenzo predictor designed in [55] is involved in HPEZ, as it is still an essential supplement of interpolation-based predictors for high-accuracy low-compression-ratio cases [32, 36, 53]. In the auto-tuning compression test process, after the auto-tuning module has acquired the optimized interpolation-based rate-distortion pair and its corresponding configuration, the auto-tuning module runs one more compression test with the Lorenzo predictor, then makes the selection between the interpolation-based predictor and the

Lorenzo predictor according to the pre-given optimization target. Following the design in [36], a multiplicative coefficient is applied to adjust the bit rate estimation of the Lorenzo predictor.

### 6.6 Block-wise Interpolation Tuning

If the interpolation predictor is finally selected after the Lorenzo tuning, the block-wise interpolation tuning will fine-tune the interpolation configuration separately on each data block. Various regions of the input data will exhibit different characteristics (such as dimension-wise smoothness), which makes them adapt to different interpolation configurations accordingly. To address this issue, HPEZ introduces the block-wise interpolation tuning process into its auto-tuning module, dedicated to identifying the best-fit interpolation configurations for diverse segments of the data. Figure 9 shows the details of the HPEZ block-wise interpolation tuning. First, after the auto-tuning has globally determined the level-wise interpolation error bounds (Figure 9 (a)), the input data array is split into blocks (Figure 9 (b)) of the same size. On each data block, a sub-block (in default has 4% of the full block size) is sampled out in the center of this block (Figure 9 (c)), and then the interpolation configuration for this block (Figure 9 (d)) is tuned by the compression tests performed on the sampled sub-block. The block size for block-wise interpolation tuning is a hyper-parameter in HPEZ, and after primary experiments, we use the default value of 32 for it.

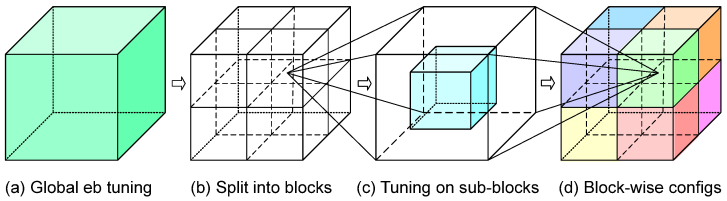



Fig. 9. Block-wise interpolation tuning

## 7 PERFORMANCE EVALUATION

To verify the effectiveness and efficiency of HPEZ, systematical evaluations of HPEZ together with six other state-of-the-art error-bounded lossy compressors are presented in this section.

### 7.1 Experimental Setup

*7.1.1 Experimental environment and datasets.* We conducted all the evaluation experiments on the Purdue Anvil supercomputer (for all experiments) and the Argonne Bebop supercomputer (for the Globus-based data transfer test). On the Anvil supercomputer, each computing node features two AMD EPYC 7763 CPUs with 64 cores having a 2.45GHz clock rate and 256 GB DDR4-3200 RAM. The computing node we used on the Bebop has the Intel Xeon E5-2695v4 CPU with 64 CPU cores and a total of 128GB of DRAM.

In order to evaluate the compressors more comprehensively and systematically, 8 real-world scientific applications from diverse scientific domains that have been frequently used for the evaluation of scientific data error-bounded lossy compression [54] are involved in the evaluation. The detailed information of the datasets is in Table 1. As suggested by domain scientists, some fields of the datasets listed above are transformed to their logarithmic domain before compression for better visualization. Among those 8 datasets, 6 are in the floating point type and 2 are in the integer type. Because floating point data are the very majority of scientific data and several of the existing scientific compressors only support floating point data, in the following experiments we mainly focus on the 6 floating point datasets and present the evaluations on the integer datasets as verification of HPEZ for its adaptiveness to scientific integer datasets and other integer datasets (natural images and videos).

Table 1. Information of the datasets in experiments

| App. | # files | Dimensions | Total Size | Domain | Type |
|---|---|---|---|---|---|
| RTM [21] | 37 | 449×449×235 | 6.5GB | Seismic Wave | Floating points |
| SEGSalt [4] | 3 | 1008×1008×352 | 4.2GB | Geology | Floating points |
| Miranda [1] | 7 | 256×384×384 | 1GB | Turbulence | Floating points |
| SCALE-LetKF [3] | 12 | 98×1200×1200 | 6.4GB | Climate | Floating points |
| CESM-ATM [20] | 33 | 26×1800×3600 | 17GB | Weather | Floating points |
| JHTDB [29] | 10 | 512×512×512 | 5GB | Turbulence | Floating points |
| NSTX-GPI [2] | 1 | 50000×80×64 | 977MB | Fusion | Integer |
| APS | 5 | 1792×2048 | 71MB | Material | Integer |

*7.1.2 Comparison of lossy compressors in evaluation.* In our experiments, we compare HPEZ with six other error-bounded lossy compressors, which have been verified to have good compression quality and/or performance in prior works [32, 35, 36, 53]. The six compressors can be categorized into **high-performance compressors** and **high-ratio compressors**. The high-performance compressors have relatively fast compression speeds with moderate compression ratios, including SZ3.1 [32], ZFP 0.5.5 [33], and QoZ 1.1 [35]. The high-ratio compressors achieve a high compression ratio/quality with advanced data processing methods, therefore having relatively low compression speeds. They are SPERR 0.6 [27], FAZ [36], and TTHRESH [7]. HPEZ should be categorized as a high-performance compressor because it exhibits comparable compression speed with modern high-performance compressors.

We didn't involve deep-learning-based compressors due to the following reasons: 1) Coordinate-network-based compressors suffer from extremely low compression speeds which are far from acceptable. 2) Autoencoder-based compressors also have low compression speeds (not comparable with high-performance compressors. For example, AE-SZ has similar speeds with SPERR [34]). Meanwhile, their compression ratios are lower than SZ3 as validated in [34].

*7.1.3 Experimental configurations and evaluation metrics.* In the compression experiments, the error bound mode we adopted is value-range-based error bound (denoted as $\epsilon$) [43], which is essentially equivalent to the absolute error bound (denoted as $e$), with the relationship of $e = \epsilon \cdot value\_range$. Since the value-range-based error bound can adapt to diverse amplitudes of datasets, it has been broadly used in the lossy compression community [30–32, 36, 55].

We perform the evaluation based on the following key metrics:

- Speeds: Check the compression and decompression speeds of compressors.
- Compression ratio (CR) under the same error bound: Compression ratio is the metric mostly cared for by the users. Given the input data $X$ and compressed data $Z$, the compression ratio $CR$ is: $CR = \frac{|X|}{|Z|}$ ( $|\ |$ is the size operator).
- *Rate-PSNR plots*: Plot curves for compressors with the bit rate of the compressed data and the decompression data PSNR.
- *Rate-SSIM plots*: Another rate distortion evaluation plotting bit rate and SSIM [47].
- Parallel throughput performance with compressors: Simulate and perform parallel data transfer tests on the distributed scientific database on multiple supercomputers.
- Visualization with the same CR: Comparing the visual qualities of the reconstructed data from different compressors based on the same CR.

## 7.2 Experimental Results

*7.2.1 Speeds.* To verify our categorization of compressors and examine the compression efficiency of HPEZ, in Table 2 we present the compression and decompression speeds of 6 comparison compressors and HPEZ (under error bound 1e−3, i.e., $10^{-3}$) on the Anvil machine. From the table, we can clearly observe that the high-performance compressors (SZ 3.1, ZFP 0.5.5, and QoZ 1.1)

have far better compression speeds than the high-ratio compressors (SPERR, FAZ, and TTHRESH) with the gap of 3×-10×. Having a speed of around 70% ~ 90% of QoZ 1.1, HPEZ can definitely be regarded as a high-performance compressor, achieving 2× ~ 6× performance improvement over SPERR/FAZ, and 4× ~ 17× performance improvement over TTHRESH. This relatively high speed ensures the advantages of HPEZ on efficiency-oriented and high-ratio-preferred compression tasks. Figure 10 presents the error bound-compression speed curves of each compressor on the 6 tested datasets ( the x-axis is the negative log10 of the error bounded and the y-axis is the compression speed). Those plots also prove that HPEZ is much more efficient than the high-ratio compressors (SPERR, FAZ, and TTHRESH) and has close performances to SZ3 and QoZ.

Table 2. Execution speeds (MB/s per CPU core) with $\epsilon$=1e-3

| Type | Dataset | SZ 3.1 | ZFP 0.5.5 | QoZ 1.1 | SPERR 0.6 | FAZ | TTHRESH | HPEZ |
|---|---|---|---|---|---|---|---|---|
| Compression | CESM | 219 | 331 | 215 | 49 | 58 | 10 | 140 |
| | RTM | 211 | 412 | 191 | 63 | 30 | 18 | 142 |
| | Miranda | 163 | 416 | 157 | 35 | 29 | 28 | 140 |
| | SCALE | 188 | 191 | 182 | 32 | 61 | 17 | 129 |
| | JHTDB | 140 | 225 | 122 | 33 | 28 | 23 | 105 |
| | SegSalt | 189 | 645 | 201 | 51 | 36 | 13 | 141 |
| Decompression | CESM | 661 | 584 | 689 | 92 | 101 | 53 | 513 |
| | RTM | 786 | 622 | 626 | 124 | 64 | 108 | 510 |
| | Miranda | 419 | 946 | 351 | 75 | 60 | 111 | 473 |
| | SCALE | 610 | 553 | 567 | 68 | 140 | 53 | 450 |
| | JHTDB | 376 | 425 | 243 | 70 | 59 | 60 | 330 |
| | SegSalt | 592 | 1060 | 629 | 108 | 65 | 97 | 485 |

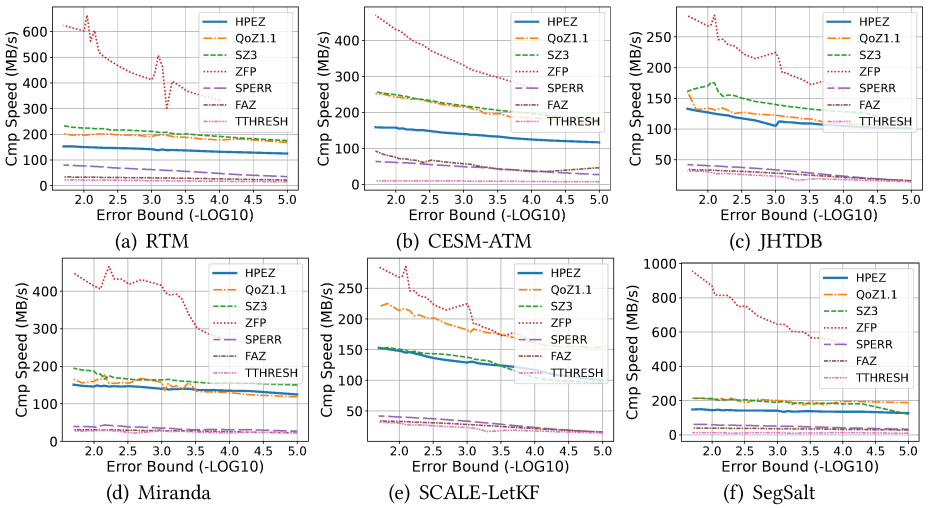



Fig. 10. Error bound-compression speed plots.

*7.2.2 Compression ratios with the same error bounds.* Compressing the datasets with the selected compressors under the same error bounds, we list all the compression ratios in Table 3 and 4. Table 3 is a comparison among the high-performance compressors (in which the compression ratio optimization targets are set for QoZ, FAZ, and HPEZ). HPEZ achieves the best compression ratio in all cases. On the SegSalt dataset, HPEZ has a 40% ~ 75% compression ratio improvement over the second-best compressor. On the RTM, Miranda, and JHTDB datasets, HPEZ achieves 20%-45% compression ratio improvements over the second-best. On the CESM-ATM dataset, under

Table 3. Compression Ratios of High-Performance Compressors (SZ, ZFP, QoZ and HPEZ)

| Dataset | $\epsilon$ | SZ 3.1 | ZFP 0.5.5 | QoZ 1.1 | HPEZ | Improve (%) |
|---|---|---|---|---|---|---|
| **RTM** | 1E-2 | 1764 | 62.9 | 2156 | **2701** | 25.3 |
| | 1E-3 | 249 | 26.2 | 285 | **395** | 38.6 |
| | 1E-4 | 55.3 | 14.3 | 58 | **71.1** | 22.6 |
| **Miranda** | 1E-2 | 574.6 | 46.6 | 977 | **1320** | 35.1 |
| | 1E-3 | 168 | 25.6 | 181 | **258** | 42.5 |
| | 1E-4 | 47.3 | 14.5 | 47.7 | **63.6** | 33.3 |
| **SegSalt** | 1E-2 | 856 | 59.1 | 1005 | **1484** | 47.7 |
| | 1E-3 | 140.6 | 24.9 | 151 | **260** | 72.2 |
| | 1E-4 | 38.2 | 14.9 | 35.9 | **61.7** | 61.5 |
| **SCALE** | 1E-2 | 167.3 | 14.5 | 160 | **186** | 11.2 |
| | 1E-3 | 40.4 | 7.8 | 41.5 | **52.9** | 27.5 |
| | 1E-4 | 14.1 | 4.6 | 13.4 | **15.4** | 9.2 |
| **JHTDB** | 1E-2 | 528.2 | 22.3 | 647 | **838** | 29.5 |
| | 1E-3 | 73.2 | 9.8 | 77.8 | **101** | 29.8 |
| | 1E-4 | 15.8 | 5 | 15.9 | **20.6** | 29.6 |
| **CESM-ATM** | 1E-2 | 373 | 18.2 | 263 | **675** | 81.0 |
| | 1E-3 | 64.9 | 9.6 | 59.4 | **153** | 135.7 |
| | 1E-4 | 22.9 | 5.8 | 21.7 | **38.9** | 69.9 |

Table 4. Compression Ratios of HPEZ and high-ratio compressors (SPERR, FAZ, and TTHRESH)

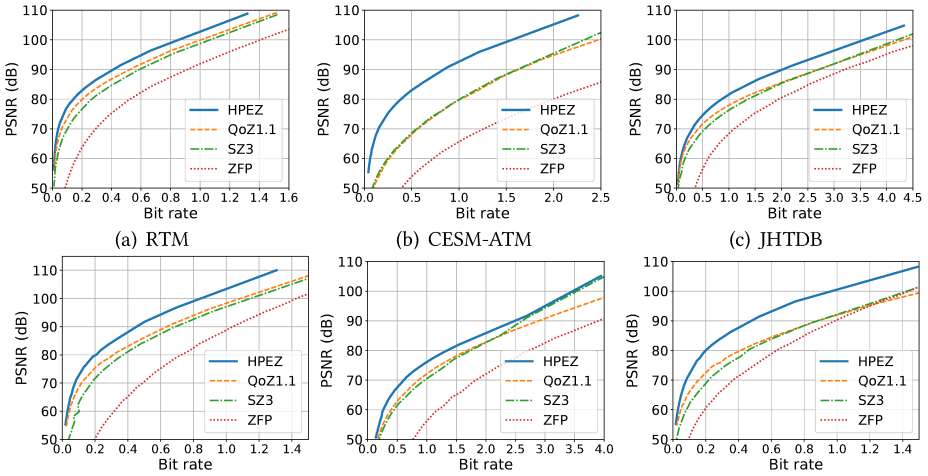
| Dataset | $\epsilon$ | SPERR 0.6 | FAZ | TTHRESH | HPEZ |
|---|---|---|---|---|---|
| **RTM** | 1E-2 | 2187 | 2695 | 782 | **2701** |
| | 1E-3 | 440 | **642** | 71.4 | 395 |
| | 1E-4 | 84.1 | **119** | 23.7 | 71.1 |
| **Miranda** | 1E-2 | 971.4 | 996.5 | 447 | **1320** |
| | 1E-3 | 243.9 | **263.5** | 142 | 258 |
| | 1E-4 | 74.5 | **93.6** | 55.1 | 63.6 |
| **SegSalt** | 1E-2 | 1219.4 | **1639.6** | 291 | 1484 |
| | 1E-3 | 228.9 | **388.9** | 99.5 | 260 |
| | 1E-4 | 61.3 | **117.3** | 28.8 | 61.7 |
| **SCALE** | 1E-2 | 103.5 | 177.9 | 80.0 | **186** |
| | 1E-3 | 35.5 | 51.8 | 18.9 | **52.9** |
| | 1E-4 | 15 | **16.8** | 8.4 | 15.4 |
| **JHTDB** | 1E-2 | 639.8 | 726 | 373 | **838** |
| | 1E-3 | 89.3 | 90.7 | 65.1 | **101** |
| | 1E-4 | 19.9 | 20.2 | 17.1 | **20.6** |
| **CESM-ATM** | 1E-2 | **1221** | 292 | 83.5 | 675 |
| | 1E-3 | 150 | 77.4 | 20.4 | **153** |
| | 1E-4 | 35 | 26.3 | 8.7 | **38.9** |

the error bound of 1e-3, HPEZ has a compression ratio of about 2.36× as high as the second-best (SZ3.1). With these considerable improvements, we can assert that HPEZ is the best choice among high-performance compressors regarding optimizing the error-bound-fixed compression ratio.

We also compare the compression ratios of HPEZ with the ones from the high-ratio compressors in Table 4. It shows that HPEZ can obtain even higher compression ratios than them in certain cases (e.g. on SCALE-LetKF and JHTDB). Note that the speed of HPEZ is substantially faster than the high-ratio compressors, making it quite competitive over them in speed-concerned use cases.

*7.2.3 Compression rate-distortion.* In this section, we mainly present the evaluations of the compression rate-distortion with HPEZ and other high-performance compressors. The high-ratio compressors are capable of achieving excellent compression rate-distortion by spending much more time cost than high-performance compressors, therefore the comparison of rate-distortion would be fair if and only if we exclude the high-ratio compressors, making it within the scope of high-performance compressors to clearly examine how HPEZ has improved the compression quality meanwhile maximally preserving the compression efficiency.

In Figure 11, the bit rate-PSNR curves of 4 high-performance compressors on 6 datasets are plotted and displayed (in which the rate-PSNR optimization targets are set for QoZ, FAZ, and HPEZ). Apparently, HPEZ has dominated this evaluation term, achieving the best PSNR under all bit rates on each dataset. This implies that, among the high-performance compressors, HPEZ can always provide the best quality of decompressed data (in terms of PSNR) under the same compression ratio, or can always yield the most compact compressed data for a certain PSNR constraint. On the CESM-ATM dataset, under PSNR of 70, HPEZ reaches around 360% compression ratio improvement over the second-best QoZ 1.1. On the SegSalt dataset, under PSNR of 80 HPEZ achieves about 100% compression ratio improvement over the second-best QoZ 1.1. There are approximately 20% ∼ 80% same-PSNR compression ratio improvements achieved by HPEZ on the other 4 datasets.




(a) RTM (b) CESM-ATM (c) JHTDB

(d) Miranda (e) SCALE-LetKF (f) SegSalt

Fig. 11. Rate-distortion (PSNR) plots of high-performance compressors.

To evaluate the HPEZ compression quality with more quality metrics, we also checked the SSIM of the decompressed results of each high-performance compressor, and those results are illustrated in Figure 12. Same as the PSNR, HPEZ undoubtedly presented the best SSIM under the same compressed size over all other evaluated high-performance compressors. Under the same compression bit rate, on multiple datasets including RTM, JHTDB, SCALE-LetKF, and SegSalt, there are 20% ∼ 40% SSIM improvements from HPEZ over the second-best QoZ 1.1. The SSIM improvements can be even much larger in the case of the CESM-ATM dataset.

In our analysis, the outstanding compression quality of HPEZ as a high-performance compressor is attributed to 3 aspects: First, the advanced interpolation techniques described in Section 5 have significantly raised the interpolation-based prediction accuracy for smooth datasets such as RTM, Miranda, SegSalt, and JHTDB. Next, avoiding interpolations along non-smooth directions, the compression for datasets with non-smooth dimensions (e.g. SCALE-LetKF and CESM-ATM) have been obviously enhanced by the dynamic dimension freezing technique (Section 6.3). Third, the

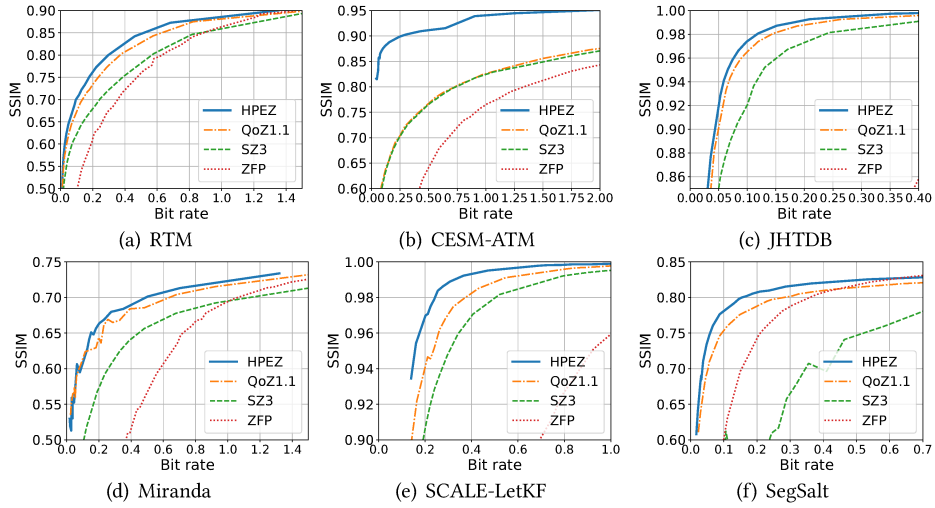



Fig. 12. SSIM of high-performance compressors.

block-wise interpolation tuning (Section 6.6) fine-tunes the interpolation on each data block, further optimizing the overall compression. In Section 7.2.7, we will feature the contribution of each HPEZ design component with more experimental results and in-depth analysis.

Lastly, we would like to claim that, in several cases, the compression quality (i.e. rate-distortion) of HPEZ can be at least comparable with the high-ratio compressors. In the comparisons between HPEZ and high-ratio compressors displayed in Figure 13, although on the Miranda dataset (Figure 13 (b)) HPEZ has quality gaps to the SPERR and FAZ, Figure 13 (a), (c) and (d) indicate that HPEZ may achieve close or even similar rate-distortion to the high-ratio compressors, with a compression speed far higher than them.

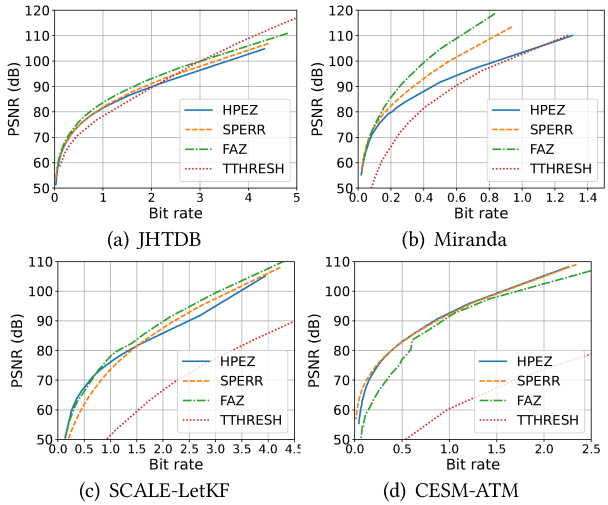



Fig. 13. Rate-PSNR of HPEZ and high-ratio compressors (HPEZ's speed is 2x-17x of high-ratio compressors).

*7.2.4 Parallel data transfer test on the distributed database.* In Section 7.2.3, we have analyzed how HPEZ over-performs other high-performance compressors in terms of compression quality.

Table 5. Compression-based parallel data transfer throughput time (in seconds, 2048 cores, under PSNR=80). Inter-machine speed is the transfer speed of compressed data between 2 machines.

| Dataset | Direction | Inter-machine Speed (GB/s) | SZ3 | ZFP | QoZ 1.1 | SPERR 0.6 | FAZ | TTHRESH | HPEZ | Improve (%) |
|---|---|---|---|---|---|---|---|---|---|---|
| CESM-ATM (41TB) | Anvil to Bebop | 0.79 ~0.91 | 1934 | 3221 | 1812 | 1560 | 1586 | 7752 | **1005** | 35.6 |
| | Bebop to Anvil | 0.95 ~1.19 | 1614 | 2695 | 1553 | 1522 | 1544 | 8560 | **916** | 39.8 |
| RTM (14TB) | Anvil to Bebop | 0.58 ~1.19 | 198 | 362 | **173** | 277 | 494 | 527 | 181 | -4.8 |
| | Bebop to Anvil | 0.47 ~1.04 | 189 | 524 | **166** | 296 | 474 | 560 | 182 | -9.5 |
| Miranda (2TB) | Anvil to Bebop | 0.46 ~1.04 | 49 | 84 | 44 | 72 | 87 | 121 | **39** | 11.3 |
| | Bebop to Anvil | 0.54 ~0.82 | 46 | 117 | 49 | 71 | 86 | 120 | **43** | 6.5 |
| SCALE-LetKF (13TB) | Anvil to Bebop | 0.88 ~0.94 | 873 | 1354 | 820 | 1037 | 782 | 2354 | **728** | 7.0 |
| | Bebop to Anvil | 1.05 ~1.15 | 745 | 1181 | 707 | 1007 | 670 | 2002 | **624** | 6.8 |
| JHTDB (10TB) | Anvil to Bebop | 0.83 ~1.15 | 567 | 826 | 527 | 645 | 583 | 835 | **417** | 20.9 |
| | Bebop to Anvil | 0.97 ~1.18 | 486 | 707 | 473 | 648 | 574 | 883 | **366** | 22.7 |
| SegSalt (8TB) | Anvil to Bebop | 0.63 ~1.18 | 163 | 289 | 174 | 221 | 251 | 393 | **137** | 15.9 |
| | Bebop to Anvil | 0.76 ~1.06 | 167 | 241 | 153 | 213 | 265 | 300 | **132** | 14.0 |

Furthermore, we will examine whether HPEZ can over-perform existing state-of-the-art error-bounded lossy compressors including high-ratio compressors in real-world use cases in which the compression and decompression time need to be taken into account. To this end, we have designed a real-world scale parallel data transfer experiment on the distributed scientific database. In this experiment, a distributed scientific database is established on multiple machines, and to accomplish the target of fast data transfer and access between the super-computers, instead of costing unacceptable time transferring the original exascale data, a lossy compressor compresses and decompresses the data in parallel on the source and destination machine, and only the compressed data with a highly-reduced size are transferred between the machines. The total time cost of this task is the accumulation of the local data I/O time, compression time, decompression time, and transfer time of the compressed data.

To convincingly prove the effectiveness of HPEZ for the parallel data transfer task, we conduct the corresponding experiments under a certain configuration. For a parallel test with $p$ cores, we augment the datasets by $p$ times then let each core compress and decompress the data in the original size. Using 2048 cores, we leveraged the 7 compressors to compress and transfer the datasets bidirectionally between the Anvil and Bebop supercomputer, constraining the decompressed data following the same distortion (PSNR = 80). The inter-machine data transfer is supported by the Globus Transfer Service [6, 10, 11], which is an efficient and widely adopted data transfer service in scientific research and education fields. Table 5 presents data transfer speed and the time cost with each compressor for each dataset. On most of the datasets tested (except for the RTM), HPEZ improves the optimal overall transfer time by 5% $\sim$ 40%, and in the worst case (on the RTM dataset), it is just slightly worse than QoZ 1.1. Therefore, the optimized balance of compression quality and efficiency of HPEZ does contribute to its utility in real-world large-scale parallel data transfer tasks.

Due to the computing resource limitation for executing the multi-core large-scale data transfer tests and repeating them with different datasets, compressors, and configurations, we have also designed a model for approximating the actual time costs in those tasks. For a specific core number $p$ and a data transfer speed $s$, we use the sequential compression/decompression speed of the compression/decompression with the same per-core data to estimate the parallel compression/decompression time cost, and the approximated data transfer time is just the compressed data size divides the transfer speed $s$. With those approximation methods, for each dataset, we approximate the time costs under a variety of compression error bounds, then plot and present the time cost-PSNR curves in Figure 14. The compressor speeds are acquired on the Anvil machine introduced in Section 7.1.1, the core numbers are 2048, and the data transfer speed is set to 1GB/s (according to the experimental results in Table 5). From the plots, we can claim that, for this task,

HPEZ has the potential to keep an advantage over the other existing compressors. On Miranda, CESM-ATM, and JHTDB datasets, with the approximations, HPEZ exhibits the minimized time cost for each fixed PSNR. On RTM, SCALE-LetKF, and SegSalt datasets, HPEZ may still always be the top-performing compressor and can have the optimized time cost in wide ranges of PSNR.

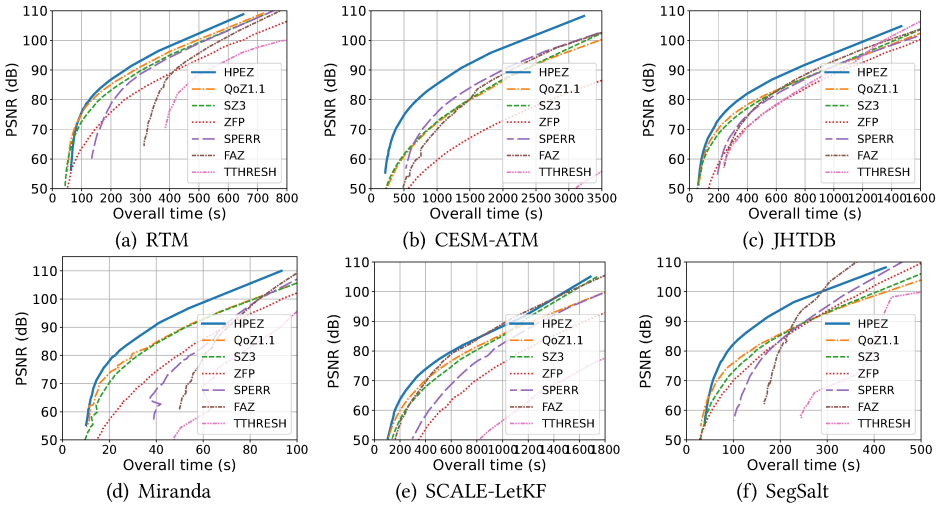
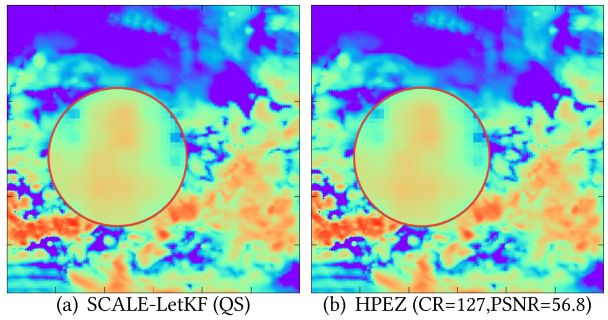



Fig. 14. Parallel data transfer time approximation and decompression PSNR (simulation on the Anvil supercomputer, $p = 2048$, $s = 1GB/s$).

*7.2.5 Case study: decompression visualizations.* As an example of the effectiveness of the HPEZ compression, in this section, we propose a case study of the compression tasks, visualizing the decompression outputs from various high-performance compressors. The example data input is the QS field (getting logarithmized in preprocessing) from the SCALE-LetKF dataset, and we compress it with HPEZ and 2 high-performance compressors: QoZ and ZFP (we omit SZ3 in this test because QoZ and SZ3 have close speeds and QoZ has better compression quality than SZ3) under similar compression ratios. The visualizations of 2-D slices from the original data and decompressed data are presented in Figure 15. In this case, among the decompression results with very close compression ratios, the decompression result of HPEZ (Figure 15 (b)) achieves the lowest data distortion with the highest PSNR (56.8). Moreover, regarding the magnified regions in Figure 15, compared to the decompression results of QoZ (Figure 15 (c), PSNR=52.7), HPEZ has better preserved the local data patterns in the original input (Figure 15 (a)). This case is an example to show the strong capability of HPEZ in providing high-quality compression results with high compression speed.

*7.2.6 Compression of HPEZ on integer datasets.* In this section, we propose the compression rate-distortion of HPEZ on the 2 integer datasets described in Section 7.1. Those datasets are scientific images and movies, therefore the experimental results with them can also reflect the potential of HPEZ to be leveraged on more integer-based datasets such as natural images and videos. Figure 16 contains the rate-PSNR curves from HPEZ and other integer-supportive high-performance compressors (SZ3 and QoZ). Apparently, HPEZ has comparatively excellent rate-distortion on the integer datasets as well as on the floating point datasets, presenting the optimized or near-optimized PSNR under the same bit rate.

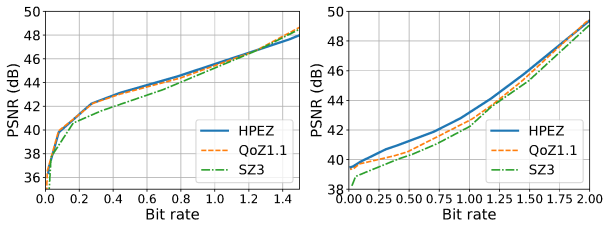
*7.2.7 Ablation study.* To better understand how HPEZ can generate high-quality compression outputs with comparatively fast speeds, we decompose the design of HPEZ, aggregating the design



(a) SCALE-LetKF (QS) (b) HPEZ (CR=127,PSNR=56.8)

(c) QoZ (CR=126,PSNR=48.4) (d) ZFP (CR=118,PSNR=21.0)

Fig. 15. Visualization of SCALE-QS field (logarithmized) and the decompressed data.

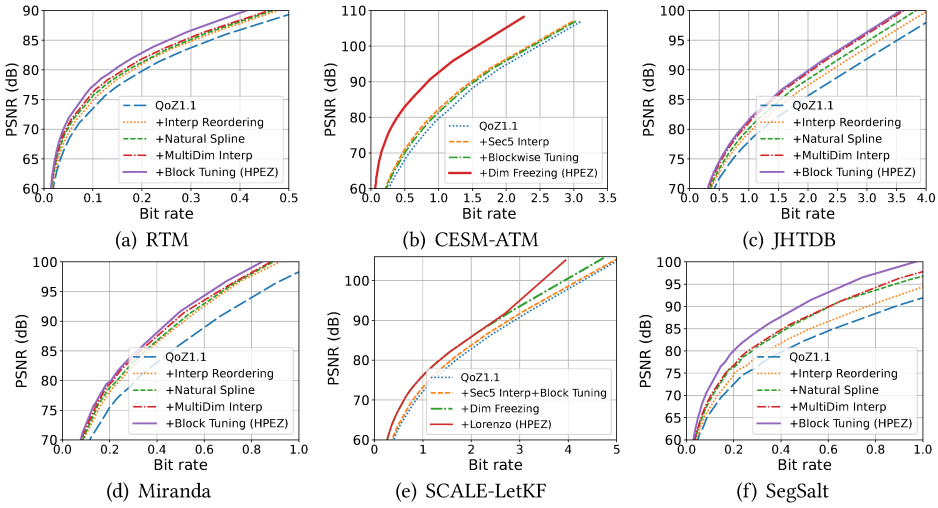



(a) APS (2D Image) (b) NSTX-GPI (3D Movie)

Fig. 16. Rate-PSNR on integer datasets.

components to QoZ 1.1 one by one for determining and quantifying the compression improvement brought by each component.

Figure 17 shows the rate-PSNR plots on the RTM dataset with QoZ 1.1, HPEZ, and the different accumulations of new design components between them. For example, in Figure 17 (a) representing the results of the RTM dataset, there is a curve showing the rate-distortion of QoZ 1.1, a curve showing the rate-distortion of QoZ 1.1 plus the interpolation re-ordering, a curve for the aforementioned one plus the natural cubic spline, and so on, eventually to the complete HPEZ. For the RTM, JHTDB, Miranda, and SegSalt datasets (Figure 17 (a) (c) (d) (f)), analyzing the rate-distortion curves we can easily find that the HPEZ interpolation designs, including interpolation re-ordering (Section 5.4), natural cubic spline (Section 5.2), and multi-dimensional interpolation (Section 5.3) all contribute to the improvement of rate-distortion. Additionally, block-wise interpolation tuning (Section 6.6) also plays an important role in optimizing the compression of their compression. Lastly, the effectiveness of multi-dimensional spline interpolation proved the generalization of Theorem 5.1 on diverse datasets and the integration of multiple interpolation optimization techniques.




(a) RTM (b) CESM-ATM (c) JHTDB

(d) Miranda (e) SCALE-LetKF (f) SegSalt

Fig. 17. Ablation study for rate-PSNR.

In Figure 17 (b) and (e) corresponding to the CESM-ATM and SCALE-LetKF datasets, we can verify the effectiveness of dynamic dimension freezing (Section 6.3) and the Lorenzo predictor (Section 6.5) The dashed curve in Figure 17 (b) and (e) integrates all the interpolation designs in Section 5. Nevertheless, compared with QoZ 1.1 they have not refined the compression sufficiently. In contrast, the dynamic dimension freezing itself (the solid curves in Figure 17 (b) and the dash-dotted curve in Figure 17 (e)) has solely boosted the rate-distortion to a remarkable extent for those 2 datasets. Furthermore, comparing the solid curve and the dash-dotted curve in Figure 17 (e), leveraging the Lorenzo predictor has quite enhanced the compression quality of HPEZ in low-error-bound (i.e. high bit rates) cases.

Last, to examine the acceleration by the fast-varying-first interpolation described in Section 5.4.1, in Table 6 we compare the sequential compression/decompression speeds of HPEZ between leveraging fast-varying-first interpolation or not (named as **HPEZ (w/o FVFI)**) in the table). Table 6 clearly shows that the fast-varying-first interpolation has appreciably contributed to the performance of HPEZ, especially on the Miranda and JHTDB datasets.

Table 6. Compression Speeds (MB/s) with and without fast-varying-first interpolation ($\epsilon$=1e-3, i.e., $10^{-3}$)

| Type | Dataset | CESM | RTM | Miranda | SCALE | JHTDB | SegSalt |
|---|---|---|---|---|---|---|---|
| Cmp | HPEZ (w/o FVFI) | 132 | 139 | 101 | 124 | 87 | 134 |
| | HPEZ | **140** | **142** | **140** | **129** | **105** | **141** |
| Dcmp | HPEZ (w/o FVFI) | 469 | 457 | 202 | 420 | 184 | 390 |
| | HPEZ | **513** | **510** | **473** | **450** | **330** | **485** |

## 8 CONCLUSION AND FUTURE WORK

In this paper, we propose HPEZ, an optimized interpolation-based error-bounded lossy compressor that supports quality-metric-driven auto-tuning and significantly improves compression ratio with low computation cost. The integration of advanced interpolation and auto-tuning designs in HPEZ has profoundly exploited the potential of the high-performance prediction-based compressor. In experiments, HPEZ achieves much better compression ratios and rate-distortion than existing high-performance error-bounded compressors with at most 140% or 360% compression ratio improvement under the same error bound or PSNR. HPEZ also over-performs existing error-bounded lossy

compressors in data throughput tasks. In parallel data transmission experiments for distributed databases, HPEZ can achieve at most 40% time cost reduction over the second bests, when compared with both high-performance and high-ratio error-bounded lossy compressors.

In the future, we plan to revise and develop HPEZ as follows: first, we will further optimize the speeds of HPEZ. Second, we will design more effective data prediction techniques for non-smooth data. Last, we will attempt to integrate compression techniques with a more flexible speed to adaptively tune the compression pipeline according to the requirements of compression speeds.

## ACKNOWLEDGMENTS

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations – the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, to support the nation's exascale computing imperative. The material was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (ASCR), under contract DE-AC02-06CH11357, and supported by the National Science Foundation under Grant OAC-2003709, OAC-2104023, OAC-2311875, OAC-2311877, and OAC-2153451. We acknowledge the computing resources provided on Bebop (operated by Laboratory Computing Resource Center at Argonne).

## REFERENCES

[1] [n. d.]. *Miranda application.* https://wci.llnl.gov/simulation/computer-codes/miranda
[2] [n. d.]. *NSTX-GPI.* https://w3.pppl.gov/~szweben/NSTX%20Blob%20Library/NSTXblobs.html
[3] [n. d.]. Scalable Computing for Advanced Library and Environment (SCALE) – LETKF. https://github.com/gylien/scale-letkf.
[4] [n. d.]. SEGSalt. https://wiki.seg.org/wiki/SEG/EAGE_Salt_and_Overthrust_Models.
[5] Mark Ainsworth, Ozan Tugluk, Ben Whitney, and Scott Klasky. 2018. Multilevel techniques for compression and reduction of scientific data—the univariate case. *Computing and Visualization in Science* 19, 5 (2018), 65–76.
[6] Rachana Ananthakrishnan, Kyle Chard, Ian Foster, and Steven Tuecke. 2015. Globus platform-as-a-service for collaborative science applications. *Concurrency and Computation: Practice and Experience* 27, 2 (2015), 290–305.
[7] Rafael Ballester-Ripoll, Peter Lindstrom, and Renato Pajarola. 2019. TTHRESH: Tensor compression for multidimensional visual data. *IEEE transactions on visualization and computer graphics* 26, 9 (2019), 2891–2903.
[8] Dor Bank, Noam Koenigstein, and Raja Giryes. 2020. Autoencoders. *arXiv preprint arXiv:2003.05991* (2020).
[9] Franck Cappello, Sheng Di, Sihuan Li, Xin Liang, Gok M. Ali, Dingwen Tao, Chun Yoon Hong, Xin-chuan Wu, Yuri Alexeev, and T. Frederic Chong. 2019. Use cases of lossy compression for floating-point data in scientific datasets. *International Journal of High Performance Computing Applications (IJHPCA)* 33 (2019), 1201–1220.
[10] Kyle Chard, Jim Pruyne, Ben Blaiszik, Rachana Ananthakrishnan, Steven Tuecke, and Ian Foster. 2015. Globus data publication as a service: Lowering barriers to reproducible science. In *2015 IEEE 11th International Conference on e-Science*. IEEE, 401–410.
[11] Kyle Chard, Steven Tuecke, and Ian Foster. 2016. Globus: Recent enhancements and future plans. In *Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale*. 1–8.
[12] Yann Collet. 2015. Zstandard – Real-time data compression algorithm. *http://facebook.github.io/zstd/* (2015).
[13] Ziquan Fang, Yuntao Du, Lu Chen, Yujia Hu, Yunjun Gao, and Gang Chen. 2021. E 2 dtc: An end to end deep trajectory clustering framework via self-training. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 696–707.
[14] Andrew Glaws, Ryan King, and Michael Sprague. 2020. Deep learning for in situ data compression of large turbulent flow simulations. *Physical Review Fluids* 5, 11 (2020), 114602.
[15] Salman Habib, Adrian Pope, Hal Finkel, Nicholas Frontiere, Katrin Heitmann, David Daniel, Patricia Fasel, Vitali Morozov, George Zagaris, Tom Peterka, et al. 2016. HACC: Simulating sky surveys on state-of-the-art supercomputing architectures. *New Astronomy* 42 (2016), 49–65.
[16] Jun Han and Chaoli Wang. 2022. Coordnet: Data generation and visualization generation for time-varying volumes via a coordinate-based neural network. *IEEE Transactions on Visualization and Computer Graphics* (2022).

[17] Lucas Hayne, John Clyne, and Shaomeng Li. 2021. Using Neural Networks for Two Dimensional Scientific Data Compression. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2956–2965.
[18] Søren Kejser Jensen, Torben Bach Pedersen, and Christian Thomsen. 2018. Modelardb: Modular model-based time series management with spark and cassandra. *Proceedings of the VLDB Endowment* 11, 11 (2018), 1688–1701.
[19] Pu Jiao, Sheng Di, Hanqi Guo, Kai Zhao, Jiannan Tian, Dingwen Tao, Xin Liang, and Franck Cappello. 2022. Toward Quantity-of-Interest Preserving Lossy Compression for Scientific Data. *Proceedings of the VLDB Endowment* 16, 4 (2022), 697–710.
[20] J. E. Kay and et al. 2015. 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* 96, 8 (2015), 1333–1349.
[21] Suha Kayum et al. 2020. GeoDRIVE – a high performance computing flexible platform for seismic applications. *First Break* 38, 2 (2020), 97–100.
[22] Søren Kejser Jensen, Torben Bach Pedersen, and Christian Thomsen. 2019. Scalable Model-Based Management of Correlated Dimensional Time Series in ModelarDB+. *arXiv e-prints* (2019), arXiv–1903.
[23] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[24] Soheil Kolouri, Phillip E Pope, Charles E Martin, and Gustavo K Rohde. 2018. Sliced Wasserstein auto-encoders. In *International Conference on Learning Representations*.
[25] Sriram Lakshminarasimhan, Neil Shah, Stephane Ethier, Scott Klasky, Rob Latham, Rob Ross, and Nagiza F. Samatova. 2011. Compressing the Incompressible with ISABELA: In-situ Reduction of Spatio-temporal Data. In *Euro-Par 2011 Parallel Processing*, Emmanuel Jeannot, Raymond Namyst, and Jean Roman (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 366–379.
[26] Sihuan Li, Sheng Di, Kai Zhao, Xin Liang, Zizhong Chen, and Franck Cappello. 2021. Resilient Error-Bounded Lossy Compressor for Data Transfer. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (St. Louis, Missouri) *(SC '21)*. Article 94, 14 pages.
[27] Shaomeng Li, Peter Lindstrom, and John Clyne. 2023. Lossy scientific data compression with SPERR. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 1007–1017.
[28] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S Jensen, and Wei Wei. 2018. Deep representation learning for trajectory similarity computation. In *2018 IEEE 34th international conference on data engineering (ICDE)*. IEEE, 617–628.
[29] Yi Li, Eric Perlman, Minping Wan, Yunke Yang, Charles Meneveau, Randal Burns, Shiyi Chen, Alexander Szalay, and Gregory Eyink. 2008. A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence. *Journal of Turbulence* 9 (2008), N31.
[30] Xin Liang, Sheng Di, Dingwen Tao, Sihuan Li, Shaomeng Li, Hanqi Guo, Zizhong Chen, and Franck Cappello. 2018. Error-Controlled Lossy Compression Optimized for High Compression Ratios of Scientific Datasets. In *2018 IEEE International Conference on Big Data*. IEEE.
[31] Xin Liang, Ben Whitney, Jieyang Chen, Lipeng Wan, Qing Liu, Dingwen Tao, James Kress, David R Pugmire, Matthew Wolf, Norbert Podhorszki, et al. 2021. MGARD+: Optimizing multilevel methods for error-bounded scientific data reduction. *IEEE Trans. Comput.* (2021).
[32] Xin Liang, Kai Zhao, Sheng Di, Sihuan Li, Robert Underwood, Ali M Gok, Jiannan Tian, Junjing Deng, Jon C Calhoun, Dingwen Tao, et al. 2022. SZ3: A modular framework for composing prediction-based error-bounded lossy compressors. *IEEE Transactions on Big Data* (2022).
[33] Peter Lindstrom. 2014. Fixed-rate compressed floating-point arrays. *IEEE transactions on visualization and computer graphics* 20, 12 (2014), 2674–2683.
[34] Jinyang Liu, Sheng Di, Kai Zhao, Sian Jin, Dingwen Tao, Xin Liang, Zizhong Chen, and Franck Cappello. 2021. Exploring Autoencoder-based Error-bounded Compression for Scientific Data. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 294–306.
[35] Jinyang Liu, Sheng Di, Kai Zhao, Xin Liang, Zizhong Chen, and Franck Cappello. 2022. Dynamic quality metric oriented error bounded lossy compression for scientific datasets. In *2022 SC22: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE Computer Society, 892–906.
[36] Jinyang Liu, Sheng Di, Kai Zhao, Xin Liang, Zizhong Chen, and Franck Cappello. 2023. FAZ: A flexible auto-tuned modular error-bounded compression framework for scientific data. In *Proceedings of the 37th International Conference on Supercomputing*. 1–13.
[37] Tong Liu, Jinzhen Wang, Qing Liu, Shakeel Alibhai, Tao Lu, and Xubin He. 2021. High-Ratio Lossy Compression: Exploring the Autoencoder to Compress Scientific Data. *IEEE Transactions on Big Data* (2021).
[38] Yuanjian Liu, Sheng Di, Kyle Chard, Ian Foster, and Franck Cappello. 2023. Optimizing Scientific Data Transfer on Globus with Error-bounded Lossy Compression. arXiv:2307.05416 [cs.DC]
[39] Yuzhe Lu, Kairong Jiang, Joshua A Levine, and Matthew Berger. 2021. Compressive neural representations of volumetric scalar fields. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 135–146.

[40] Tuomas Pelkonen et al. 2015. Gorilla: A Fast, Scalable, in-Memory Time Series Database. *Proc. VLDB Endow.* 8, 12 (Aug. 2015), 1816–1827.
[41] Tjerk P Straatsma, Katerina B Antypas, and Timothy J Williams. 2017. *Exascale scientific applications: Scalability and performance portability*. CRC Press.
[42] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on circuits and systems for video technology* 22, 12 (2012), 1649–1668.
[43] Dingwen Tao, Sheng Di, Hanqi Guo, Zizhong Chen, and Franck Cappello. 2019. Z-checker: A framework for assessing lossy compression of scientific data. *The International Journal of High Performance Computing Applications* 33, 2 (2019), 285–303. https://doi.org/10.1177/1094342017737147
[44] David S Taubman, Michael W Marcellin, and Majid Rabbani. 2002. JPEG2000: Image compression fundamentals, standards and practice. *Journal of Electronic Imaging* 11, 2 (2002), 286–287.
[45] Jiannan Tian et al. 2020. CuSZ: An Efficient GPU-Based Error-Bounded Lossy Compression Framework for Scientific Data. In *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques (PACT '20)*. 3–15.
[46] Jiannan Tian, Sheng Di, Xiaodong Yu, Cody Rivera, Kai Zhao, Sian Jin, Yunhe Feng, Xin Liang, Dingwen Tao, and Franck Cappello. 2021. cuSZ (x): Optimizing Error-Bounded Lossy Compression for Scientific Data on GPUs. *CoRR* (2021).
[47] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.
[48] Xinyang Yu et al. 2020. Two-Level Data Compression using Machine Learning in Time Series Database. In *36th IEEE International Conference on Data Engineering*. 1333–1344.
[49] Xiaodong Yu, Sheng Di, Kai Zhao, jiannan Tian, Dingwen Tao, Xin Liang, and Franck Cappello. 2022. SZx: an Ultra-fast Error-bounded Lossy Compressor for Scientific Datasets. *arXiv preprint arXiv:2201.13020* (2022).
[50] Boyuan Zhang, Jiannan Tian, Sheng Di, Xiaodong Yu, Yunhe Feng, Xin Liang, Dingwen Tao, and Franck Cappello. 2023. FZ-GPU: A Fast and High-Ratio Lossy Compressor for Scientific Computing Applications on GPUs. *arXiv preprint arXiv:2304.12557* (2023).
[51] Dongxiang Zhang, Mengting Ding, Dingyu Yang, Yi Liu, Ju Fan, and Heng Tao Shen. 2018. Trajectory simplification: an experimental study and quality analysis. *Proceedings of the VLDB Endowment* 11, 9 (2018), 934–946.
[52] Kai Zhao, Sheng Di, Perez Danny, Zizhong Chen, and Franck Cappello. 2022. MDZ: An Efficient Error-bounded Lossy Compressor for Molecular Dynamics Simulations. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*.
[53] Kai Zhao, Sheng Di, Maxim Dmitriev, Thierry-Laurent D. Tonellot, Zizhong Chen, and Franck Cappello. 2021. Optimizing Error-Bounded Lossy Compression for Scientific Data by Dynamic Spline Interpolation. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. 1643–1654. https://doi.org/10.1109/ICDE51399.2021.00145
[54] Kai Zhao, Sheng Di, Xin Lian, Sihuan Li, Dingwen Tao, Julie Bessac, Zizhong Chen, and Franck Cappello. 2020. SDRBench: Scientific Data Reduction Benchmark for Lossy Compressors. In *2020 IEEE International Conference on Big Data (Big Data)*. 2716–2724.
[55] Kai Zhao, Sheng Di, Xin Liang, Sihuan Li, Dingwen Tao, Zizhong Chen, and Franck Cappello. 2020. Significantly Improving Lossy Compression for HPC Datasets with Second-Order Prediction and Parameter Optimization. In *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing* (Stockholm, Sweden) *(HPDC '20)*. Association for Computing Machinery, New York, NY, USA, 89–100. https://doi.org/10.1145/3369583.3392688

Received July 2023; revised October 2023; accepted November 2023