Estimating Potential Error in Sampling Interpolation

Megan Hickman Fulp, Dakota Fulp, Jon C. Calhoun
Holcombe Department of Electrical and Computer Engineering
Clemson University
Clemson, SC, USA
mlhickm@clemson.edu, dakotaf@clemson.edu, jonccal@clemson.edu

***Abstract*—As high-performance computing systems rapidly advance, the volumes of scientific data produced are a prominent issue. Data reduction methods, including compression and sampling, seek to alleviate these bottlenecks by significantly reducing overall size. Yet, research shows sampling yields lower overall reconstruction quality than lossy compression due to its inability to bound the error it introduces. This work presents and assesses the ability to estimate reconstruction error. We propose an absolute error estimation scheme using various metrics and evaluate its capabilities over several real-world datasets. Upon evaluation, our scheme is between 63.6% and 94.9% accurate at estimating absolute error and has an average 87.7% correlation with the error. Lastly, we implement and discuss applications of our error estimation, including a novel sampler enhancement that allows for user-specified targeted data quality and a process to achieve a user-specified storage constraint while yielding a better preservation of data than existing methods.**

***Index Terms*—Data Reduction, Data Sampling, Importance Sampling, Error Quantification, Feature Preservation**

I. Introduction

High-performance computing (HPC) systems have revolutionized the way scientists research by enabling them to solve previously intractable problems. While these systems have facilitated many groundbreaking findings, they also produce vast quantities of data. For instance, the Community Earth System Model (CESM) simulation is only possible due to these systems, as it is capable of generating 2.5 PB of data over 18 months [22]. However, it takes an additional 18 months to post-process and publish only 6.6% of the results, as the I/O subsystems cannot efficiently handle them.

Researchers alleviate these bottlenecks by using reduction methods to decrease the amount of data they transfer and store. Lossless compressors [9], [14] reduce data size with no loss in precision but operate poorly with floating-points due to the high entropy mantissa bits [28]. Lossy compressors achieve higher reduction ratios on floating-point data by introducing controlled approximations when encoding, which users control through a specified error bound. Data sampling achieves high reduction ratios by keeping specific data values with total precision, later used to reconstruct the entire dataset.

When reducing the Deep Water Impact Ensemble dataset [26], scientists must preserve essential features like the water spray for post-hoc analysis. Grosset et al. used Foresight [15] to analyze how well different data reduction schemes preserve the data at a 50× compression ratio. They found that data compression algorithms achieve a higher (better) peak signal-to-noise ratio and lower (better) absolute error than data sampling. SZ [18], in particular, was shown to preserve the details of the simulation much better than the most sophisticated data sampling method to date. When analyzing only the crater region of interest, using the signal-to-noise ratio and a visual comparison, the compression methods outperformed sampling again. This case study example has shown that at its current state, data sampling schemes can not perform at the same level as data compression algorithms, as they have both low data throughput and poor post-reconstruction quality.

This ineffectiveness is due to data sampling's inability to bound the error it introduces, like modern lossy compressors. The first step in combating this disadvantage is understanding how the error is introduced in the interpolation step of the sampling pipeline. This work details the shortcomings of existing sampling methods and proposes a scheme to estimate the amount of error introduced into the reconstructed dataset. Using this scheme, we aim to improve existing data reduction pipelines. Specifically, our contributions are as follows:

- We detail how error propagates during the linear interpolation reconstruction process and identify five metrics that influence potential error.
- We present our scheme for estimating the amount of error reconstruction introduces by using a weighted combination of error-affecting metrics.
- We examine the correlation between our error estimate and the true absolute error, finding an average correlation of 87.7% and accuracy ranging from 63.6% to 94.9%.
- We present two applications of our approach in existing data reduction pipelines: First, we propose a sampling process that achieves a target user-specified storage constraint while maintaining data quality better than existing methods. Second, we implement a first-of-its-kind sampling process that achieves a user-specified target data quality in terms of Peak Signal-to-Noise Ratio.

II. Background and Related Work

Scientists working with HPC applications use various data reduction strategies to reduce I/O bottlenecks that occur with large datasets [3]–[6], [9], [14], [18]–[20], [30]. Lossless compression algorithms, such as ZStd [9] or GZip [14], reduce data size with no loss in precision. Due to the high entropy found

978-1-6654-8045-1/22/$31.00 ©2022 IEEE

in floating-point mantissa bits, these algorithms are suboptimal for reducing HPC scientific data, only achieving reduction ratios of $1\times$ to $4\times$ when used on scientific data [28]. On the other hand, lossy compression algorithms, such as SZ [18] and ZFP [20], leverage data approximations and omissions to represent the input data with lower precision. When scientists want to retain regions of interest with higher quality, they often use various data sampling algorithms. Data sampling methods represent the input data by saving a small subset of data values. Unlike lossy compression, data sampling allows high reduction at fixed ratios. The following details the sampling methods studied in this work. Figure 2 visualizes the resulting sample set with varying sample method. Methods vary in the type of data distribution they return. Systematic and random methods yield an even distribution across the data. Importance-based sampling methods like value-based and multi-criteria cluster around regions of interests in the data. The following sections discuss each method in detail.

A. Random Sampling

Simple random sampling is a basic unbiased algorithm that gives each data value an equal opportunity to be included in the sample set. For each data point, a random number, ξ, is compared to a user-specified sampling percentage, α. In this case, $\xi, \alpha \in [0, 1]$ and if $\xi < \alpha$ then the algorithm includes the data value in the sample set. This algorithm achieves high throughput and maintains statistical quantities of the original dataset, such as mean and standard deviation, which makes it ideal for datasets with unknown distributions. However, it does not consider data point value and, as a result, does not always yield optimal sample sets.

B. Systematic Sampling

Systematic sampling is a method that results in a group of values gathered by an equally spaced sequence [32]. When taking systematic samples, we take a subset of data points that are equally spatially distanced from the previous sample. Specifically, given a user-specified sampling percentage, α, we calculate the spatial period of samples as $n = 1/\alpha$. Following this, we start at data location $i = 0$ and take every nth data point to be within our sample set.

Taking such a group of data points allows us to view a general overview of the data without over-representation or under-representation of any area of the data. The disadvantage of systematic sampling is that it can lead to false statistical overviews due to its periodic nature. However, for our error estimation purposes, having this even representation of the data allows us to visualize target high-error areas better, as we further discussed in Section IV-B.

C. Value-Based Sampling

Value-based sampling is a biased data sampling algorithm that prioritizes more important data values. Biswas et al.'s method biases rare data values without ignoring more common data values [4]. Their algorithm uses the distribution of data values to calculate an importance factor, I_F, for each data value such that more rare values have a greater I_F and more common values have a lower I_F. While this algorithm efficiently reduces data size and preserves data values within the data regions of interest, it does not consider areas of abrupt change or take into account local smoothness.

D. Multi-Criteria Based Sampling Algorithm

Multi-criteria-based sampling is another form of biased data sampling that prioritizes data values based on multiple data properties. For instance, Biswas et al.'s multi-criteria sampling method biases both more rare data values and data values in regions of significant change [3]. The algorithm prioritizes samples with a higher local gradient value during this selection process. This algorithm efficiently reduces data size while preserving data values within the data regions of interest and the edges of different regions. However, this algorithm does poorly in cases where there are too many edges or where there is not a centralized region of interest.

E. Reconstruction Algorithms

We must reconstruct the discarded data using the sample set to evaluate the quality. The nearest neighbor method reconstructs a data value by setting it equal to the value of the nearest saved point without consideration of neighboring points. However, this method often yields jagged, blocky results and produces the least-smooth reconstruction [11]. Linear interpolation is an approach that uses a curve-fitting method and linear polynomials to reconstruct missing data points. The interpolated value is within the discrete set of saved values. While there are various methods, triangulation with linear interpolation works best with uniform distributions of sample locations [31], like those used in our experiments, rather than sparse areas. Since linear interpolation yields the highest quality reconstruction for these datasets, we focus on estimating the error introduced while using this method.

The Delaunay triangulation [10] is traditionally used for piece-wise linear interpolation of data points. The algorithm constructs various triangles across the dataset, with vertices of sample locations, ensuring that another triangle intersects no triangle edges. The resulting set yields a patchwork of triangles across the grid, such that a triangular surface covers every grid point. Then, each missing data point is determined using linear interpolation within the triangle containing that point.

This reconstruction process introduces error into the data as it interpolates missing data values from the sample set. Estimating the error introduced is not trivial, as many factors affect the resulting error. For instance, Figure 1 illustrates the initial sample set, the Delaunay triangulation process, and the resulting reconstructed Asteroid Impact dataset. The interpolated dataset has many differences from the original data, yielding high amounts of error and low quality overall.

III. Estimating Error Propagation

Our novel error estimation scheme uses various mathematical metrics to predict the amount of reconstruction error that would occur. This process begins by taking an initial

Fig. 1: Samples, Triangulation and Interpolation of Impact.

Variable	Definition
P	All data points in the plane
S	All sample points $\in P$
V	Values at location P
$\hat{V}$	Interpolated values at location P
$DT(S)$	Delaunay triangulation of samples S
T_k	A triangle $\in DT(S)$
A, B, C	Points $\in S$ that form T_k
u	A point $\in P$, $\notin S$ to be interpolated
α, β, γ	Coefficients that hold the locational relationship between A, B, C, u true

TABLE I: Mathematical Variable Reference

set of samples for each 2D slice of a 2D or 3D dataset. For our initial study on error propagation, we contain the error to a 2D space for better visualization and explanation of mathematical correlations. Our process triangulates the samples using Delaunay triangulation before using several metrics to estimate the error at each data point. Finally, our process clusters the predicted error into sections, dividing the dataset into regions of interest and background data.

A. Initial Sample

Our scheme operates on regular structured grid datasets that consist of implicit data point locations. The first step in our scheme is to take an initial sample set for every 2D slice of a 2D or 3D dataset. After the initial study of error propagation is completed, we aim to expand this work to study the error propagation in the 3D space in future work. We examine various sample ratios and sampling methods, such as random [29], systematic [32], histogram-based [4], and multi-criteria [3] sampling. The initial sample aims to gain a general overview of the data so that we have the best understanding of the data. With a group of samples that are too clustered together, we have too much information in one area and too little in others, leading to a poor data summary. Therefore, we base our study on systematic sampling. Even though other sampling methods have been shown to achieve higher quality post-reconstruction [3], [4], that is not the goal of our analysis. We aim to achieve the best error prediction, thereby identifying regions of interest, features, or hard-to-preserve areas. However, we apply our error estimation process to all sampling methods in Section IV-B to further show that we still achieve satisfactory results regardless of method. The result of the sampling process leaves us with two sets: the locations of the samples, S, and the values of the samples, V. We save each value and location for future visualization and reconstruction purposes.

B. Delaunay Triangulation

A popular way of reconstructing a dataset is to consider it a piece-wise union of cells, subdividing the plane into multiple patches, each having an independent local interpolating function. Subdivision is obtained via triangulation. A Delaunay Triangulation [10] $DT(S)$ is the triangulation formed from a set of S discrete points such that no point in S is inside the circumcircle of any triangle in $DT(S)$, as demonstrated in Figure 1b. The minimum angle of each triangle in the triangulation is maximized to avoid sliver triangles, where the area is significantly smaller than its circumcircle.

A triangulation $DT(S)$ over a plane is defined as a set of multiple individual triangles, T_k, represented as $DT(S) = \{T_k\}$. Each triangle T_k consists of three distinct vertices, $A, B, C \in S$ such that $T_k = A\alpha, B\beta, C\gamma$, where

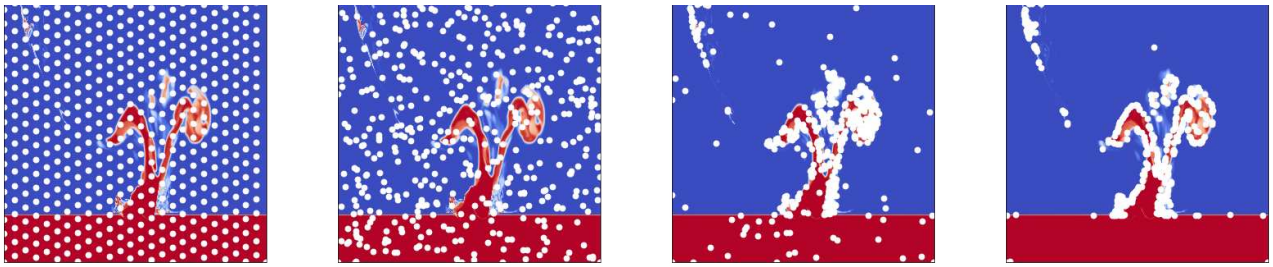
- $A, B, C \in S$
- $\alpha, \beta, \gamma \in [0, 1]$
- $\alpha + \beta + \gamma = 1$
- the union of T_k equals the convex hull of A, B, C

Since there may be multiple Delaunay Triangulations per S, there is no guaranteed unique solution for every set. The advantage is that Delaunay Triangulations optimize several geometric properties, including: smallest angle, largest circumdisk, and largest minimum containment disk (the smallest enclosed disk that includes the triangle) [7]. Long, skinny triangles are avoided, as they make interpolation less accurate as the points used to interpolate are spatially far apart. Smaller, more Equilateral triangles yield better interpolation since the vertices are spatially closer.

C. Linear Interpolation

Linear interpolation is a method of curve fitting using linear polynomials to build new data points within the range of a discrete set of known data points. It is used to estimate missing data using known values. Linear interpolation can also be applied to triangular meshes (a piece-wise planar surface formed by connected triangles). We utilize the vertices in the Delaunay Triangulation to interpolate the data values within each triangle using linear interpolation. For a triangle $T_k \in DT(S)$, with vertices at locations A, B, C, let a point $u \in T_k$, as demonstrated in Figure 3. The point u can be expressed as $u = A\alpha + B\beta + C\gamma$, an affine combination such that all coefficients sum to one, i.e. $\alpha + \beta + \gamma = 1$ and $\alpha, \beta, \gamma \in [0, 1]$.

An α, β, γ are found based on the locations of the data points, A, B, C, u in relation to the plane. In the 2D plane, the system of equations $u_x = A_x\alpha + B_x\beta + C_x\gamma$ and $u_y = A_y\alpha + B_y\beta + C_y\gamma$ are used. Once an α, β, γ that solve the system of equations are found, we use them to interpolate the value of u as $\hat{V}(u) = V(A)\alpha + V(B)\beta + V(C)\gamma$. Thus, the interpolated value is based upon the location of u, relative to its distance from vertices A, B, C. The closer u is to A, the more weight the value at point A has in u's interpolated value.



(a) Systematic (b) Random (c) Value-Based (d) Multi-Criteria

Fig. 2: White circles are samples gathered for the Impact Dataset with various sample methods at a reduction ratio of 100:1.

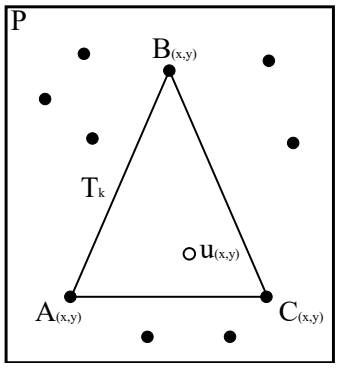


Fig. 3: Triangulation of point u ∈ triangle ABC.

D. Error-Effecting Metrics

Once our scheme collects an initial set of samples, it clusters the dataset into separate feature sections by estimating the absolute error due to interpolation. It begins by creating the Delaunay triangles from the initial sample set that it would use during true reconstruction. For each data point, it leverages the data point's value and the triangle vertices it is in to estimate the absolute error. Error is introduced into the reconstructed data when $\hat{V}(u) \neq V(u)$. In this section, we analyze the factors that affect how different $\hat{V}(u)$ is from $V(u)$.

1) Metric: Distance Out of Range: The value of u is calculated as the weighted sum of the contributing vertices' values. Values A, B, C are used in a weighted average with weights of α, β, γ, signifying the significance of each value in relation to $V(u)$. As seen in Lemma 1, the range of possible interpolated values is in the range of the minimum and maximum values of the vertices.

$$\textit{Lemma 1: } \hat{V}(u) = V(A)\alpha + V(B)\beta + V(C)\gamma$$
$$\alpha + \beta + \gamma = 1$$
$$\alpha, \beta, \gamma \in [0, 1]$$
$$\text{Thus, } \hat{V}(u) \in [min(V(A, B, C)), max(V(A, B, C))]$$

If a value to be reconstructed is outside the range of the triangle's vertices, then no method can reconstruct it accurately. The more out of range the value is, the higher the difference to its reconstructed value is, leading to a higher reconstructed error and lower resulting data quality. Metric 1: *Out of Range* quantifies how much out-of-range the data value is from the extrema of the triangle vertices. The range of this metric is the difference between the minimum and maximum values in the entire true dataset. In the worst scenario, we have a set of samples that only contain the minimum value in the set of all true values, V. Since there were no samples containing the true maximum value $\in V$, the linear interpolation would result in some error in locations where reconstructed data points were assigned an incorrect value of min(V) when it should have been max(V).

$$\textit{Metric 1: Out of Range: } m_1 =$$
$$max[|min(V(A, B, C) - V(u)|, |V(u) - max(V(A, B, C)|]$$
$$m_1 \in [-max(V) - min(V), max(V) - min(V)]$$

2) Metric: Difference from Nearest: When using linear interpolation to reconstruct the data values within each triangle, the resulting value is slightly biased towards the nearest triangle vertex sample. The weights $\alpha, \beta, \gamma \in [0, 1]$ are derived based on the global plane coordinates of the vertices at locations A, B, C a point $u \in T$. Thus, the distance from each vertex and u affects the resulting weight of that vertex in the overall interpolation. With this in mind, the more significant the difference between the value to be reconstructed and the value of the closest triangle vertex, the higher the error in the reconstructed data value. Metric 2: *Nearest Difference* quantifies the absolute difference between values of each point within the triangle and the nearest vertex.

$$\textit{Metric 2: Nearest Difference:}$$
$$N = min(\sqrt{(u_x - Z_x)^2 + (u_y - Z_y)^2}), \text{ where}$$
$$Z \in A, B, C.$$
$$m_2 = V(u) - V(N)$$
$$m_2 \in [min(V), max(V)]$$

3) Metric: Difference in Triangle Vertices: The difference of the values of each vertex affects the interpolated value.

$$\textit{Lemma 2: } \hat{V}(u) = V(A)\alpha + V(B)\beta + V(C)\gamma$$
$$\alpha + \beta + \gamma = 1$$
$$\alpha, \beta, \gamma \in [0, 1]$$
$$error = abs\hat{V}(u) - V(u)$$
$$error = abs(V(A)\alpha + V(B)\beta + V(C)\gamma) - V(u)$$
$$range = max(V(A, B, C)) - min(V(A, B, C))$$

Thus, as range increases, potential error increases.

The higher the difference between the values of each vertex, the harder it is to smooth out the values within the triangle, resulting in a lower interpolation quality. Unlike the other

two error estimation metrics, this metric remains constant for each point within a triangle, as it is not based on individual data values. Metric 3: *Vertex Difference* quantifies the average difference between the values of each triangle's vertices.

Metric 3: Vertex Difference:

$$m_3 = max(V(A,B,C)) - min(V(A,B,C))$$
$$m_3 \in [min(V), max(V)]$$

4) Metric: Triangle Area: If the triangle area is too large, there is more mathematical uncertainty. Triangles with a large area have more values that need to be reconstructed within the triangle. As area grows, the three triangle vertex values are less likely to represent the interior points sufficiently. Therefore, larger triangles have a higher probability of yielding lower quality in that region. We calculate Metric 4: *Triangle Area* as the number of data points within the triangle to be reconstructed rather than the mathematical area.

Metric 4: Triangle Area:

$$m_4 = V \in T_k$$
$$m_4 \in [3, \tfrac{V}{2}]$$

5) Metric: Circle Ratio: The circle ratio is the measure of the triangle's flatness. With this metric, an equilateral triangle's circle ratio is 0.5, and any triangle more flat is less than this. In theory, the flatter the triangle, the more it stretches over the data, and the more it could encompass a larger distribution of data values. Metric 5: *Circle Ratio* is calculated as the ratio of the incircle radius over the circumcircle radius.

Metric 5: Circle Ratio:

Given a set S that contains all n edge lengths of triangle T,

$$\text{Incircle Radius, } r = \Pi_{i=0}^{n} S_n$$
$$\text{Circumcircle Radius, } R = \sum_{i=0}^{n} S_n$$
$$m_5 = \tfrac{2r}{R}$$
$$m_5 \in [0, 1]$$

E. Error-Effecting Metrics Evaluation

As our goal is to predict the error in the reconstructed data without having to reconstruct it completely, our scheme uses the metrics outlined in Section III because there exists a mathematical correlation between these metrics and the error found in the linear interpolated values. Thus, these metrics enable us to predict the resulting reconstruction.

Correlation is the numerical representation of the linear relationship between two datasets. With it, we describe the rate of change of one variable in response to changes in another. Covariance is similar but only measures the positive or negative direction of the linear relationship between variables, while correlation measures both direction and magnitude. The Pearson Correlation Coefficient (PCC) is the ratio between the covariance of two variables and the product of their standard deviations, which normalizes it between -100% and 100%.

Here, we discuss the PCC between each individual metric and the true absolute error on the original dataset and reconstructed data. We report the correlation percentage by metric and dataset. The correlations range from -100%, a perfectly negative correlation, to 100%, a perfectly positive correlation, where 0% signifies no correlation. In general, any $|PCC| > 70\%$ indicates possible col-linearity, while $|PCC| < 70\%$ implies low or negligible relationship [23].

The data from scientific simulations can vary, resulting in some metrics being better for certain dataset types and worse for others. To account for this, we cluster datasets into two distinctions: centralized and decentralized feature datasets.

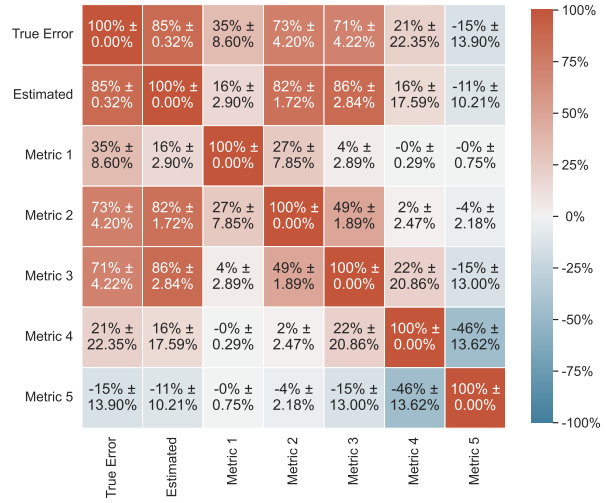
A *centralized* feature dataset has one clustered set of values as the region of interest to the end-user. Examples of this distinction include the impact and pressure datasets (see Table II). With the impact dataset, domain scientists are explicitly interested in studying the asteroid entry into the water [1], [27]. With the hurricane pressure dataset, scientists are interested in the hurricane eye. In both datasets, the feature scientists study is a single cluster of data values.

Conversely, a *decentralized* feature dataset has no single cluster of values. Instead, in these simulations, scientists are interested in multiple regions of interest or the entire dataset. These datasets have data that needs to be studied across the entire dataset. Examples include the precipitation and Nyx datasets, as they have multiple regions of interest or data evenly dispersed throughout the dataset. By considering dataset distinction, we achieve better prediction accuracy and minimize the standard deviation between correlations.

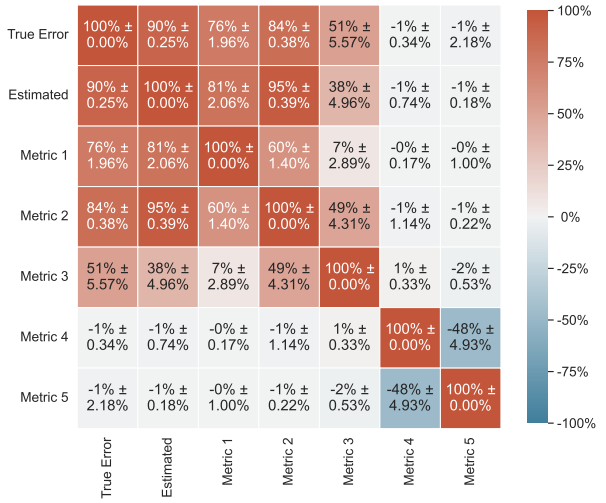
1) Metric 1: Distance Out of Range: Upon analyzing Figure 4, we find that Metric 1 has an average correlation to the true error of $35\% \pm 8.60\%$ when working with centralized feature datasets and an average correlation of $76\% \pm 1.96\%$ with decentralized feature datasets. This metric is ineffective with centralized feature datasets as the feature values within these datasets are clustered. Therefore, the probability of finding a value out of range is lower. With this in mind, our method chooses to use Metric 1 when predicting decentralized feature datasets, as this results in a correlation $> 70\%$.

2) Metric 2: Difference from Nearest: Metric 2 has an average correlation to the true error of $73\% \pm 4.20\%$ when working with centralized feature datasets and an average correlation of $84\% \pm 0.38\%$ with decentralized feature datasets. This metric is effective regardless of the dataset feature distinction because this metric mathematically aligns strongly with the interpolation process we use to reconstruct the data. Specifically, the reconstruction process utilizes a weighted average of the triangle vertices to rebuild each missing data point within the triangle. During this process, the weight of each coefficient is determined by the distance between the missing data location and each vertex. Therefore, our method utilizes this metric for all estimations regardless of the dataset.

3) Metric 3: Difference in Triangle Vertices: When assessing Metric 3's correlation, we find it has an average correlation to the true error of $71\% \pm 4.22\%$ when working with centralized feature datasets and an average correlation of $51\% \pm 5.57\%$ when working with decentralized feature datasets. This metric is highly effective with centralized feature datasets. If the difference between triangle vertex values is too high, this indicates that the triangle intersects with a region of high entropy. When this occurs, there is a much higher possibility of error occurring within this region. Conversely,



(a) Centralized feature



(b) Decentralized feature

Fig. 4: Correlations by dataset type

this metric is ineffective on decentralized feature datasets as most of these datasets share a similar level of entropy across the entire data. Hence, our method utilizes this metric for centralized feature dataset estimations.

4) Metric 4: Triangle Area: Assessing the correlation of Metric 4, we find it has an average correlation to the true error of $21\% \pm 22.35\%$ when working with centralized feature datasets and an average correlation of $-1\% \pm 0.34\%$ when working with decentralized feature datasets. In both dataset distinctions, we find this metric has a low correlation. While mathematically, if the area of a triangle is too big, there should be more uncertainty, we use systematic sampling. This sampling approach leads to the area of most triangles being the same. This result is also visible when using other basic sampling algorithms. As a result, the size of the triangles cannot correlate with the error enough for this metric to be useful in our estimation. Consequently, we choose not to utilize this metric in our estimation process.

5) Metric 5: Circle Ratio: Upon assessing the correlation of metric 5, we find it has an average correlation to the true error of $-15\% \pm 13.90\%$ when working with centralized feature datasets and an average correlation of $-1\% \pm 2.18\%$ when working with decentralized feature datasets. Similar to Metric 4, this metric is ineffective since we utilize systematic sampling. By using this initial form of sample gathering, most of the triangles are of the same shape, meaning the circle ratio cannot correlate well enough with the overall error for it to be useful. As a result, we choose not to utilize this metric as well in our estimation process.

TABLE II: Datasets used in experiments.

NAME	VARIABLE	SHAPE	ROI	RANGE
Impact [26]	V02	$300 \times 300 \times 300$	Central.	[0.0, 1.0]
Isabel [16]	Pressure	$500 \times 500 \times 100$	Central.	[-3412, 3224]
Isabel [16]	Precipitation	$500 \times 500 \times 100$	Decentral.	[0.0, 0.008]
Nyx [21]	Dark Matter ρ	$512 \times 512 \times 512$	Decentral.	[0.0, 13779]

IV. Error Metric Evaluation

We evaluate each metric to determine which can be appropriately used to estimate error. We leverage the metrics with a high correlation to the true reconstruction error to build our proposed scheme and evaluate the accuracy of its estimations against the true reconstruction error.

A. Experimental Setup

1) Hardware and Software: We conduct trials on Clemson's Palmetto Cluster [24]. Specifically, we use nodes that are 40 core Intel Xeon CPUs with 372 GB of memory. On these machines, we use gcc 8.3.1. We use the Systematic sampling method in our experiments, except where specified otherwise.

2) Datasets: We evaluate the accuracy of our scheme on several real-world HPC single-precision simulation datasets, detailed in Table II. The specified time-steps contain a predefined region of interest (ROI) inside the domain.

B. Error-Estimation

We use a weighted average of the error metrics to estimate the error in the dataset if we were to reconstruct. We set these weights based on each metric's correlation to the actual error. Based on the metric correlations discussed in the previous section, we evenly set the weights of the highly correlated metrics according to whether the dataset contains a centralized or decentralized feature. For the centralized features, we estimate a data value's error as 50% Metric 2 (Nearest Difference) and 50% Metric 3 (Vertex Difference), as these metrics have the highest correlation with the true error that are also $> 70\%$. For decentralized features, Metric 3 has a correlation $< 70\%$, signifying that it has little relation to the true error. However, Metric 1 (Out of Range) shifts to be $> 70\%$. Therefore, for these dataset types, we estimate error as 50% Metric 1 and 50% Metric 2. While the remaining metrics could be considered in our error estimation, these three lead to the best estimation of error overall, as they have the highest correlations to the true error.

TABLE III: Average accuracy and correlations of error estimations with varying dataset and initial method over all slices.

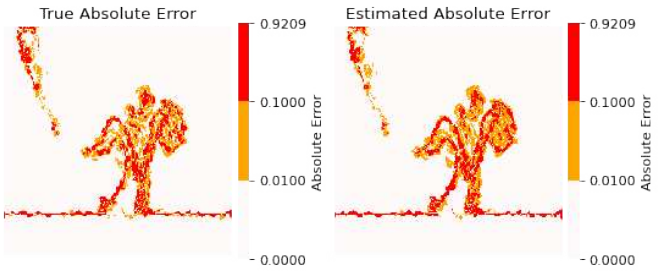
DATASET	METHOD	ACCURACY	CORRELATION
Impact (VO2)	Systematic	**87.11**%	85.1%
	Random	86.45%	86.8%
	Value-Based	80.41%	94.2%
	Multi-Criteria	85.68%	92.4%
Isabel (Pres.)	Systematic	88.14%	85.7%
	Random	85.46%	81.0%
	Value-Based	86.63%	88.6%
	Multi-Criteria	**91.92**%	80.9%
Isabel (Prec.)	Systematic	**94.88**%	89.7%
	Random	93.89%	90.2%
	Value-Based	83.35%	84.5%
	Multi-Criteria	93.89%	87.2%
Nyx (ρ)	Systematic	**65.61**%	90.2%
	Random	59.94%	90.8%
	Value-Based	40.61%	80.9%
	Multi-Criteria	41.09%	80.7%

1) Evaluation: To evaluate our proposed estimation, we visualize and quantify the difference between our estimated and the true errors that occur when reconstructing. Figure 5 illustrates both our estimated error and the true absolute error interpolated from a 10% sample. We categorize the error distribution into low, medium, and high errors based on the overall range of the data values. This enables the separation of background values and regions of interest. Specifically, if the initial sample set adequately represents a data value, it is a more common value that generally is of little importance to users. However, if a value has a high predicted error, the sample set cannot sufficiently represent it, as the value is rare. Therefore, the data points with the highest predicted error are most likely the most critical regions.

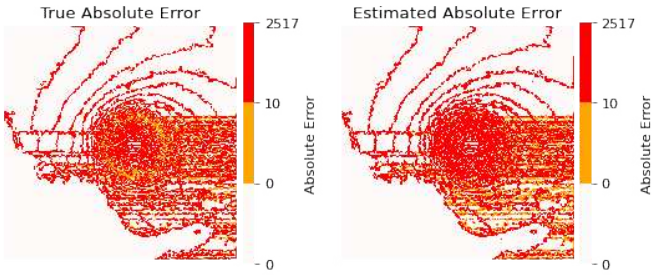
Visually, the results in Figure 5 find the difference between estimated and actual absolute error nearly identical. They also demonstrate the strength of the correlation between the factors we use and the error present in the reconstructed dataset. Specifically, our estimation has an average 90.11% correlation with the actual error across all datasets. When quantifying the error on a per-point basis to determine the accuracy of our scheme, we consider our estimated error to be accurate if it is within $0.01\times$ the range of the dataset (i.e., value-range relative error bound). This approach enables us to normalize the accuracy based on individual dataset attributes. Upon evaluating each point, we find our scheme has an average accuracy of 77%. Overall, our scheme is sufficiently capable of accurately predicting the error interpolation introduces.

Table III details the accuracy and correlation of our error estimation. Since systematic sampling yields an even distribution of information across the dataset, we are provided a better understanding of the data. However, we show that we still have high accuracy and correlations with the other methods.

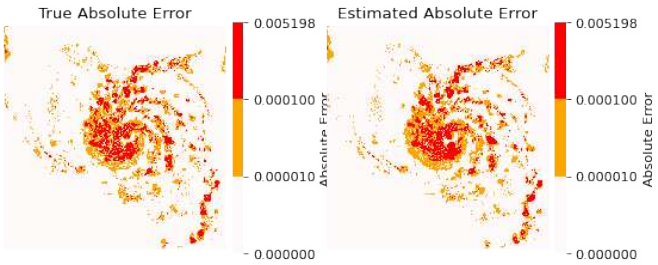
2) Performance: Error estimation is highly parallelizable for distributed systems, as it consists of triangle-based and point-based qualities. Metric 3 (Difference in Triangle Vertices) is a triangle-based quality where all points in the triangle share the same metric value. This metric must only be calculated once per $T_k \in T$. The second component is the point-based metrics, 1 (Distance Out of Range) and 2 (Difference from Nearest), where a unique value is calculated for each point in P. We run the calculations of each metric in parallel, as they can each be calculated independently. For our parallelization, we use PyMP [17], a tool that adds a parallel iterator to Python by creating a shared array structure and usage of the Unix system `Fork()`. In Table IV, we provide the time to sample each dataset with various sampling methods. For our solution, we provide the time it takes to sample sequentially and with 12 processors, which yields an average speedup of $44\times$.



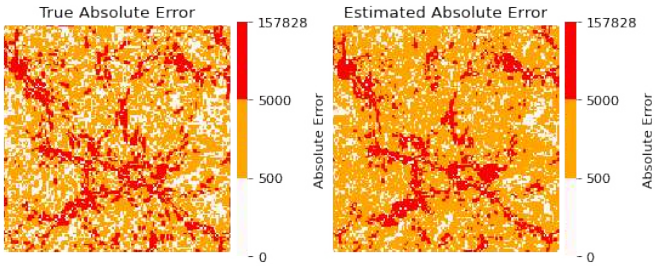
(a) Impact (Accuracy: 87.11%, Correlation: 85.1%)



(b) Pressure (Accuracy: 88.14%, Correlation: 85.7%)



(c) Precipitation (Accuracy: 94.88%, Correlation: 89.7%)



(d) Nyx (Accuracy: 63.61%, Correlation: 90.2%)

Fig. 5: Predicted absolute error categorized by error bound.

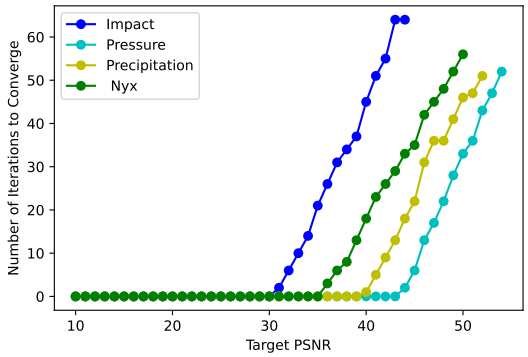


Fig. 6: Error iterations needed to converge to target PSNR.

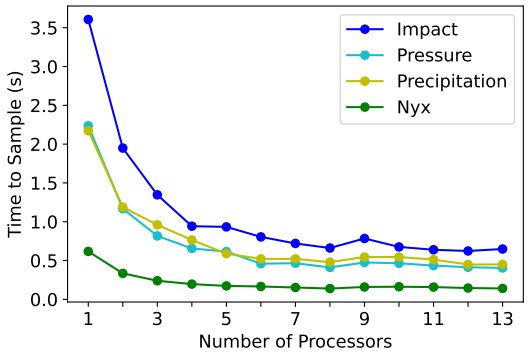


Fig. 7: Time to complete targeted 10:1 sampling process with varying number of processes. Sampling time is improved by 5.08× on average when using 12 processors versus one.

V. Applications

A. Error Minimization

One standard method of quantifying the amount of error introduced into data is the peak signal-to-noise ratio (PSNR) (Eq. 2). PSNR is a ratio between the original and reconstructed data, representing the amount of noise found when comparing them. Specifically, the higher the PSNR, the higher the quality and lower the data distortion. PSNR is calculated using mean-square error (MSE), as shown in Equation 1, where n is the number of data points, Y_i is the original value and $\hat{Y}_i$ is the reconstructed value. We use the resulting MSE to calculate PSNR in Equation 2, where max_val and min_val are the maximum and minimum of the original dataset.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \quad (1)$$

$$PSNR = 20 * log10((max\_val - min\_val)/\sqrt{MSE}) \quad (2)$$

Since PSNR is calculated using MSE between true and reconstructed values, we leverage our error prediction as a substitute for MSE, resulting in a predicted PSNR. Figure 5 illustrates the 51.50% to 89.54% accuracy of our predicted MSE. We use this to identify regions of high absolute error such that we can take new samples at these locations to improve PSNR. By estimating error and adding new samples through an iterative process, we are capable of achieving the desired PSNR. This progressive set of samples yields a better set than a non-iterative method but at lower speeds due to multiple error estimations. Figure 6 illustrates the number of error estimations we must make to achieve various target PSNRs. To our knowledge, this is the first sampling pipeline capable of achieving a user-specified target PSNR. While other works exist on reducing the size of computer graphics, they rely on mesh simplification, and only work on a single, topologically–sound mesh [8]. Our method can work with any existing sampling method, including those with cross-variable [12] and cross-timestep [13] capabilities, resulting in a more general solution. However, each dataset is limited in the PSNR it can achieve. In Figure 6, the right-most data point of each dataset represents the highest PSNR achievable before reaching a sample ratio of 100%, meaning all data points are saved, and no reduction is achieved.

B. Targeted Sampling

In situ visualization allows the user to view a reduced-size representation of a simulation's state while the simulation is ongoing. As the I/O bottleneck becomes a consistent problem, in situ visualization is critical for exascale data analysis [2]. The constraints to in situ visualizations and analysis lie in performance relative to the storage available. As such, the time the domain scientist spends waiting for the production of visualizations must be minimized. The usefulness of the visual is the trade-off between the information gained and the time the domain scientist spends to reach that insight [25]. Scientists need to start analyzing the data as quickly as possible and spend less time waiting for the data. Waiting for large, exascale simulation output files to be exported to local memory for visualization can take a long time due to the I/O bottlenecks. Therefore, data sampling is an effective solution where the end user can study a summary of the data rather than analyzing the memory-intensive, high-resolution dataset.

Data sampling balances the amount of data reasonably available to be analyzed with the amount of data desired. The higher the sampling rate, the higher resolution of the data, the more storage and time to read into memory for post-hoc analysis, and the longer it takes the end user to study the data. While higher-resolution data are necessary for presentations and visualizations, it is essential for end users to be able to quickly determine whether a feature is of scientific interest by working with smaller, more quickly stored, loaded, and visualized intermediate data. With sampled data, the end user can spend more time analyzing the data than waiting for the data to be transferred, loaded, and rendered. [25] found that very low sampling rates can be used to quickly and effectively make visualizations that display the features of interest across time-steps of the simulation. The most important of these visuals can then be downloaded at higher resolutions.

By identifying the areas of high potential error, we extract the dataset's features. We use the clustered error in Figure 5

TABLE IV: Quality (PSNR) and time to sample each dataset with varying sampling method. Sample time for our solution is provided in a parallel (12 processors) and sequential scheme.

DATASET	METHOD	RATIO	PSNR	TIME	PROCESSORS
Impact (VO2)	Systematic	50:1	15.9	0.0026s	1
	Random	50:1	17.2	0.0086s	1
	Value-Based	50:1	13.6	0.0227s	1
	Multi-Criteria	50:1	12.1	0.0820s	1
	Our Solution	50:1	**20.4**	0.628s	12
	Our Solution	50:1	**20.4**	17.80s	1
Isabel (Pres.)	Systematic	50:1	31.8	0.0039s	1
	Random	50:1	32.6	0.0176s	1
	Value-Based	50:1	30.8	0.0335s	1
	Multi-Criteria	50:1	27.0	0.0974s	1
	Our Solution	50:1	**35.9**	0.470s	12
	Our Solution	50:1	**35.9**	46.88s	1
Isabel (Prec.)	Systematic	50:1	31.3	0.0031s	1
	Random	50:1	29.5	0.0020s	1
	Value-Based	50:1	18.7	0.0102s	1
	Multi-Criteria	50:1	20.6	0.0559s	1
	Our Solution	50:1	**31.4**	0.433s	12
	Our Solution	50:1	**31.4**	13.42s	1
Nyx (ρ)	Systematic	50:1	26.1	0.0007s	1
	Random	50:1	**26.4**	0.0013s	1
	Value-Based	50:1	18.0	0.0070s	1
	Multi-Criteria	50:1	16.9	0.0134s	1
	Our Solution	50:1	24.5	0.1485s	12
	Our Solution	50:1	24.5	2.66s	1

to label data as background values and regions of interest. If the initial sample set adequately represents a value, it is a more typical value that generally is of little importance to users. However, if a value has a high predicted error, the sample set cannot sufficiently represent it. Therefore, the data points with the highest predicted error are most likely critical. This enables our scheme to provide insight into where regions of interest lie in the dataset. We leverage this insight to include better samples, yielding a more optimal data summary. After identifying the top most error-induced values, we can incorporate them into our sample to gain a better overall set.

For our sampling application, we first take a 0.1% sample ratio using systematic sampling to get an overview of the data. Then we include the areas of highest predicted error into our sample until we reach the overall target ratio of 1%. Figure 8 shows that this method yields a better overview of samples and a final reconstructed view of the datasets. We repeat these experiments on other datasets, with results shown in Table IV. The majority of improvement is with centralized ROI datasets. We find less improvement with decentralized datasets, as they include fine details that are difficult to capture. Statistically, we achieve a higher PSNR with the Nyx dataset just by sampling the four corner points (PSNR = 25.4) than we do by attempting to achieve an accurate sample with $40\times$ more samples.

The process of estimating error is highly parallelizable, as the error for each point is independent. Figure 7 shows the time it takes to perform the error estimation and targeted sample with varying number of processes and a reduction ratio of 10:1 (5% sample ratio). With more processes, we can estimate multiple errors at a time. We achieve an average speedup of $5.07\times$ using 12 processors compared to a single processor. When using a smaller sample ratio of 1%, as shown in Table IV, we achieve a $44\times$ improvement.

VI. Conclusion

Due to the increasing size of data produced by scientific applications, reduction schemes have become prominent. Even the most sophisticated sampling algorithms still introduce more error than domain scientists tolerate. Our work studies the error introduced by the linear interpolation of samples. We estimate the absolute error introduced with 63.6% to 94.88% accuracy and 87.68% correlation with the true error. This work is highly applicable to improving current sampling algorithms and building new data reduction pipelines, as demonstrated in two specific applications. First, we applied the error estimation process to targeting a user-specified data quality, which no other sampling process has achieved, to our knowledge. Second, we incorporated error estimation into a sampling process that guarantees a storage constraint to be met while retaining a higher data quality than existing methods.

In our future work, we aim to improve our scheme further. Specifically, we have shown that our method is parallelizable and is capable of estimating error of multiple data points across processes. To further reduce temporal overhead, we aim to implement our scheme on accelerated hardware such as GPUs. Next, we aim to extend our scheme to further domains and investigate more use cases. Lastly, we aim to extend our error-estimation process to estimate error in the full 3D space. Currently, we estimate error based on 2D slices. By adding the 3D component, linear interpolation uses more samples to estimate the missing data. In the future, we aim to estimate how error propagates in the 3D space.

Acknowledgment

Clemson University is acknowledged for generous allotment of compute time on Palmetto cluster. This material is based upon work supported by the National Science Foundation under Grant No. SHF-1910197 and SHF-1943114.

References

[1] J. Ahrens, S. Jourdain, P. O'Leary, J. Patchett, D. H. Rogers, and M. Petersen. An image-based approach to extreme scale in situ visualization and analysis. In *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 424–434. IEEE, 2014.

[2] A. C. Bauer, H. Abbasi, J. Ahrens, H. Childs, B. Geveci, S. Klasky, K. Moreland, P. O'Leary, V. Vishwanath, B. Whitlock, et al. In situ methods, infrastructures, and applications on high performance computing platforms. In *Computer Graphics Forum*, volume 35, pages 577–597. Wiley Online Library, 2016.

[3] A. Biswas, S. Dutta, E. Lawrence, J. Patchett, J. C. Calhoun, and J. Ahrens. Probabilistic data-driven sampling via multi-criteria importance analysis. *IEEE Transactions on Visualization and Computer Graphics*, 27(12):4439–4454, 2021.

[4] A. Biswas, S. Dutta, J. Pulido, and J. Ahrens. In situ data-driven adaptive sampling for large-scale simulation data summarization. In *Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization - ISAV '18*, page 13–18. ACM Press, 2018.

[5] J. Calhoun, F. Cappello, L. N. Olson, M. Snir, and W. D. Gropp. Exploring the feasibility of lossy compression for pde simulations. *The International Journal of High Performance Computing Applications*, 33(2):397–410, 2019.

[6] F. Cappello, S. Di, S. Li, X. Liang, A. M. Gok, D. Tao, C. H. Yoon, X.-C. Wu, Y. Alexeev, and F. T. Chong. Use cases of lossy compression for floating-point data in scientific data sets. *The International Journal of High Performance Computing Applications*, 33(6):1201–1220, 2019.

[7] S.-W. Cheng, T. K. Dey, J. Shewchuk, and S. Sahni. *Delaunay mesh generation*. CRC Press Boca Raton, 2013.

[8] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1):37–54, 1998.

[9] Y. Collet and M. Kucherawy. Zstandard Compression and the application/zstd Media Type. RFC 8478, Oct. 2018.

[10] B. Delaunay et al. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2, 1934.

[11] E. Di Bella, A. Barclay, R. Eisner, and R. Schafer. A comparison of rotation-based methods for iterative reconstruction algorithms. *IEEE transactions on nuclear science*, 43(6):3370–3376, 1996.

[12] S. Dutta, A. Biswas, and J. Ahrens. Multivariate pointwise information-driven data sampling and visualization. *Entropy*, 21(7):699, 2019.

[13] M. H. Fulp, A. Biswas, and J. C. Calhoun. Combining spatial and temporal properties for improvements in data reduction. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 2654–2663. IEEE, 2020.

[14] J. L. Gailly. Gzip, 1992.

[15] P. Grosset, C. M. Biwer, J. Pulido, A. T. Mohan, A. Biswas, J. Patchett, T. L. Turton, D. H. Rogers, D. Livescu, and J. Ahrens. Foresight: Analysis that matters for data reduction. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, page 1–15, Nov 2020.

[16] Hurricane ISABEL Simulation Data. http://vis.computer.org/vis2004contest/data.html, 2019. Online.

[17] C. Lassner. Pymp. https://github.com/classner/pymp, 2022.

[18] 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 (Big Data)*, pages 438–447, 2018.

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

[20] P. Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE transactions on visualization and computer graphics*, 20(12):2674–2683, 2014.

[21] Z. Lukic. Nyx cosmological simulation data, 2019. Available: http://dx.doi.org/10.21227/k8gb- vq78.

[22] S. Mickelson, A. Bertini, G. Strand, K. Paul, E. Nienhouse, J. Dennis, and M. Vertenstein. A new end-to-end workflow for the community earth system model (version 2.0) for the coupled model intercomparison project phase 6 (cmip6). *Geoscientific Model Development*, 13(11):5567–5581, 2020.

[23] M. M. Mukaka. A guide to appropriate use of correlation coefficient in medical research. volume 24, pages 69–71, 2012.

[24] Palmetto Cluster, Clemson University. http://citi.clemson.edu/palmetto/, 2021. [Online; accessed 24-June-2021].

[25] J. Patchett and J. Ahrens. Optimizing scientist time through in situ visualization and analysis. *IEEE computer graphics and applications*, 38(1):119–127, 2018.

[26] J. Patchett and G. Gisler. Deep water impact ensemble data set. Technical report, Los Alamos National Laboratory, 2017. LA-UR-17-21595.

[27] J. Patchett, F. Samsel, K. C. Tsai, G. R. Gisler, D. H. Rogers, G. D. Abram, and T. L. Turton. Visualization and analysis of threats from asteroid ocean impacts. *Los Alamos National Laboratory*, 2016.

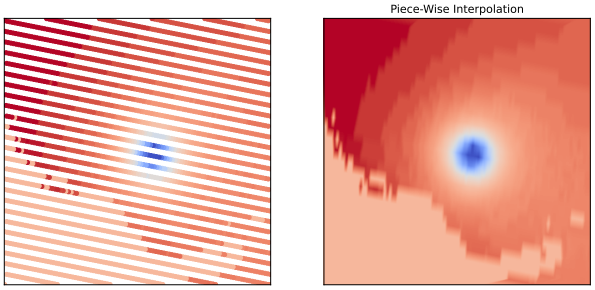
[28] S. W. Son, Z. Chen, W. Hendrix, A. Agrawal, W. keng Liao, and A. Choudhary. Data compression for the exascale computing era - survey. *Supercomputing frontiers and innovations*, 1(2), 2014.

[29] Y. Tillé and M. Wilhelm. Probability sampling designs: principles for choice of design and balancing. *Statistical Science*, pages 176–189, 2017.

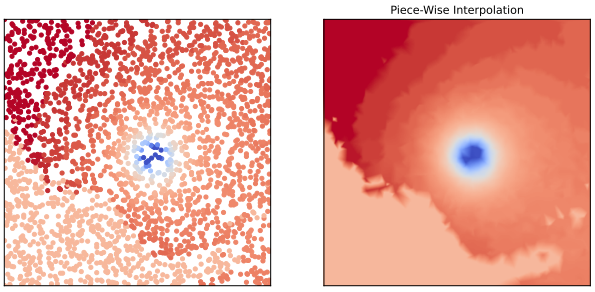
[30] R. Underwood, S. Di, J. C. Calhoun, and F. Cappello. Fraz: A generic high-fidelity fixed-ratio lossy compression framework for scientific floating-point data. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 567–577. IEEE, 2020.

[31] C.-S. Yang, S.-P. Kao, F.-B. Lee, and P.-S. Hung. Twelve different interpolation methods: A case study of surfer 8.0. In *Proceedings of the XXth ISPRS congress*, volume 35, pages 778–785, 2004.

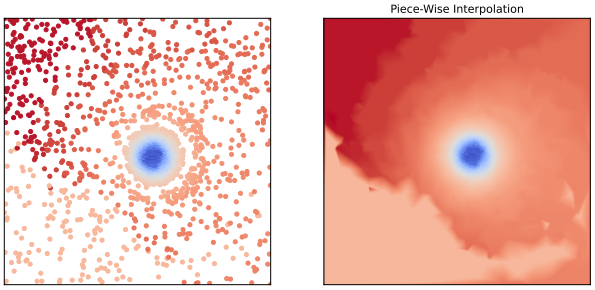
[32] F. Yates. Systematic sampling. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 241(834):345–377, 1948.



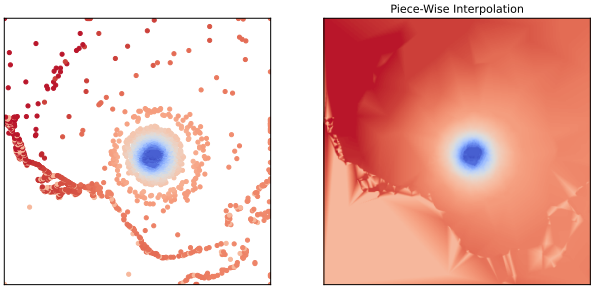
(a) Systematic (PSNR: 31.8)



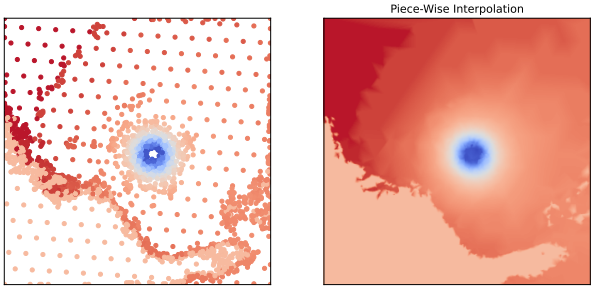
(b) Random (PSNR: 32.6)



(c) Value-Based (PSNR: 30.8)



(d) Multi-Criteria (PSNR: 27.0)



(e) Our Systematic Error-Minimization (PSNR: 35.9)

Fig. 8: Samples and Reconstructed Quality of varying methods with the Hurricane Pressure Dataset at a 1% sample ratio.