In-Situ Data Compression for Flow Simulation in Porous Media

Henry Lehmann[1], **Bernhard Jung**[1]
[1]Virtual Reality and Multimedia Group, TU Bergakademie Freiberg, Freiberg (Saxony), Germany

Abstract— *Supercomputing simulations easily generate datasets in the range of tera or even peta bytes. However, writing datasets during simulation and reading them for post-processing implies a high I/O load and thus imposes a major bottleneck. In the computation phase, compute nodes stall at writing large amounts of data and the simulation slows down - scientists have to capture data less frequently losing important information. In the post-processing phase, the data has to be read into memory and visualization becomes a time consuming batch process. These problems can be addressed with in-situ data reduction techniques which reduce the I/O load significantly. Recently a novel in-situ data compression method, ISABELA, has been proposed [6]. It provides high compression rates at negligible run time overhead while being local, communication-free, and scalable. This contribution describes a variant of the ISABELA method and its application to a fluid simulation in porous media. Modifications to the original algorithm concern the specific properties of porous media as well as a general new method for exploiting temporal coherence in time-dependent fluid simulation data. Evaluation results concern the optimal choice of curve fitting parameters for data compression and give insight into the trade-off between desired error bounds and compression rates. Overall, the very high compression rate of the ISABELA method is confirmed by our experiments.*

Keywords: in-situ processing, lossy compression, high performance computing, b-spline, data-intensive application, temporal pattern

1. Introduction

As high performance computing resources and algorithms become generally available to scientists, the simulation of physical phenomena plays an increasingly important role in prototyping and knowledge discovery. However the growing computational power raises new problems and challenges to existing hard- and software [2]. With numerical simulations producing amounts of data in the order of tera or even peta bytes, storage devices are driven to peak performance on file systems not optimized for parallel operation [14]. Similarly, for later post-processing, large amounts of data have to be read into memory. This makes post-processing a time consuming batch process which hardly can be influenced by the user once being started [8].

In-situ data reduction is a promising approach to deal with the heavy I/O load of high-resolution supercomputing simulations. In in-situ processing, data compression techniques are directly integrated into the computation phase of the simulation run. Instead of forcing scientists to write out only a subset of the computed simulation data, in-situ processing offers the potential of storing simulation data in their full resolution [8], [6].

The work in this paper is specifically motivated by the need for in-situ data reduction in large-scale Lattice Boltzmann simulations [13] of flow in porous media. Porous media are characterized by highly complex geometries, which present special challenges to numerical simulations. In the Collaborative Research Center (CRC) 920 [1] at Technical University Bergakademie Freiberg (Germany), multifunctional filters for metal melt filtration are researched. The aim is to contribute towards the development of zero-defect materials improving safety of road and railway vehicles as well as aircrafts and other applications requiring highly stressable metal based components.

A further goal is to provide a basis for the design of a new real-time visualization framework using immersive Virtual Reality, replacing offline batch processing by interactive data exploration experiences. With interactive data visualization playing an increasingly important role in data analysis, in-situ data reduction is seen a key feature in CRC 920's evolving simulation and visualization frameworks.

Recently, a novel in-situ data reduction technique called ISABELA (for "In-situ Sort-And-B-spline Error-bounded Lossy Abatement") has been proposed [6]. It outperforms existing lossy and lossless compression techniques for scientific data and offers surprisingly high compression rates while introducing only a minor overhead in computation time into the simulation.

This paper reports on a variant of the ISABELA method (as no implementation details are given in [6]) and its application to fluid simulation data of flow within porous media. In Section 2, we review related work on lossless and lossy compression of scientific data. Section 3 presents our variant of the ISABELA compression method. Our ISABELA-variant takes advantage of differentiating between geometry and fluid cells when applied to voxel grids of porous media. We further describe an error quantization scheme and investigate a novel general method for exploiting temporal coherence in fluid simulation data to enhance com-

pression ratio. In Section 4, we present compression results and analyze different parameterizations of the compression algorithm. Finally, we conclude that ISABELA-like methods provide very good compression results for flow simulation in porous media as well. However, the optimal parameterization and implementation details for our melt simulation dataset varies from the suggested parameterization in the original ISABELA description.

2. Related Work

The complex internal structure of scientific data makes compression a challenge for existing compression methods. Lossless techniques for fast online compression of floating point data are presented e.g. in [10] and [3]. These techniques function by predicting data values and storing small differences between predicted and original data values. They also offer options to omit precision bits supporting lossy compression. However, at high compression rates the relative error increases significantly [6]. Moreover, the compression is not block-based making them unsuitable for random read/write applications.

In the context of compression of medical images, [11] introduces a lossless method based on wavelets improved by detection of global and local symmetries. However, this method induces significant computational effort to enable high compression rates and is thus not suited for in-situ data reduction.

Generally, scientific floating point data is commonly known to be inherently noisy and random-like, high-entropy data without repeating bit or byte patterns, thus commonly considered to be hardly compressible [15]. As argued in [6], standard lossless compression methods do not reach satisfying compression rates or otherwise introduce significant computational overhead.

Due to approximations and discretizations, simulation data is already flawed. Thus, lossy compression with a user-defined error bound is a reasonable choice for achieving high compression rates.

A lossy data reduction technique for "In-situ Sort-And-B-spline Error-bounded Lossy Abatement" (ISABELA) was recently proposed [6]. It outperforms existing lossy compression techniques, like Wavelets [5], and lossless compression techniques, like FPC [3], LZMA and Bzip. ISABELA offers surprisingly high compression rates of up to $\sim$85% while introducing only a minor overhead in computation time and ensuring a user-defined error bound. It is designed for local encoding and decoding of scientific data.

The main idea of ISABELA is to apply a pre-conditioner, which sets up a very strong signal-to-noise ratio by sorting data values. Due to monotonic and smooth behavior of the sorted data, a regression model approximates the data with significantly fewer coefficients compared to the overall amount of data [6], [4].

Relatively few studies have been conducted on the application of B-splines for compression of scientific data. In [4] and [7] B-splines are used to fit scattered data by optimizing positions of control points. ISABELA extends this concept and transforms the data to guarantee an accurate fitting model.

3. Compression Algorithm

In this section, a variant of the ISABELA-algorithm for in-situ data reduction is presented. As in the original proposal, a first pre-processing step is to subdivide, linearize and sort the data; in our variant, the algorithm further accounts for special properties of porous media, by differentiating between geometry and fluid cells. Then, the sorted data is fitted through cubic B-splines. A new variable width error quantization scheme is applied, which bounds the maximum relative error on a per-point basis. Finally a new method for exploiting the temporal coherence of fluid simulation data is integrated into the compression algorithm.

3.1 Subdividing, Linearizing and Sorting

As first pre-processing step, the data is subdivided into smaller contiguous blocks that can be compressed and decompressed independently from each other, thus supporting random access of data blocks. By linearizing the data, the three-dimensional compression problem is transformed into a one-dimensional one. Sorting of the linearized data results in smooth curves, that can be approximated much better through B-splines [6].

In figure 1 the procedure of subdividing, linearizing and sorting the data is sketched. In step 1 the entire data grid is subdivided into smaller blocks, which contain N data values. In step 2 the blocks are linearized in a fixed serialization order. This gives a scattered sequence $\alpha = (a_1, \ldots, a_N)$ of N data values for each block, which is shown in step 3. In step 4 the data is sorted in ascending order to get a monotonic sequence $\alpha' = (a'_1, \ldots, a'_N)$. In general, the method is independent of a particular block size N.

In simulations of porous media, the data grid contains geometry cells, which are represented by a constant data value $\bar{a}$ in α'. This offers a further opportunity for compression. Discarding data values belonging to geometry cells in the sorted sequence α' yields a new sequence of $n \leq N$ data values, which is denoted as $\beta = (b_1, \ldots, b_n)$. This enhances the performance of the compression, but keeps the possibility to reconstruct the data on per-block basis without decompressing the entire dataset. The sequence β then is applied to a regression model, which is able to represent the data accurately.

To be able to revert the sorting process and reconstruct α during decompression, one has to keep track of the indices, when sorting and shortening α to obtain β. The shortened sequence of indices of elements of β in α is denoted as $\tau = (t_1, \ldots, t_n)$. Remaining values with indices not being

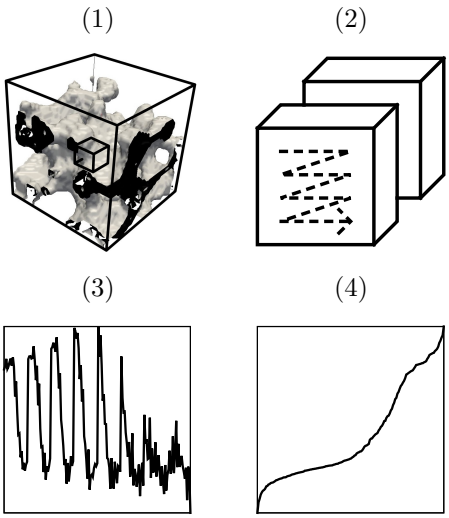


Fig. 1: Illustration of pre-processing the data for compression: *(1)* subdividing voxel grid containing porous media into smaller blocks, *(2) + (3)* linearizing data in blocks into scattered sequence, *(4)* sorting scattered sequence yields monotonic smooth data.

in τ, are set to the constant $\bar{a}$ in order to reconstruct the geometry cells omitted in β. In following an example of a particular instantiation of α, α', β and τ, with geometry cells having $\bar{a} = 0$ is given:

$$\begin{array}{lcccccccc} \alpha & : & 0.3 & 0.1 & 0.0 & 0.2 & 0.4 & 0.5 & 0.3 & 0.0 \\ \alpha' & : & 0.0 & 0.0 & 0.1 & 0.2 & 0.3 & 0.3 & 0.4 & 0.5 \\ \beta & : & 0.1 & 0.2 & 0.3 & 0.3 & 0.4 & 0.5 & & \\ \tau & : & 2 & 4 & 1 & 7 & 5 & 6 & & \end{array} \quad (1)$$

3.2 Cubic B-Spline Fit

The sequence β has a high signal-to-noise ratio and can be fitted by a cubic B-spline with few control points accurately [6].

Originally developed in the CAD and computer graphics fields, B-splines provide a flexible framework for representing complex shapes [9]. Cubic B-splines are constructed from a set of control points and a knot vector. The control points are used to specify the shape of the resulting curve, whereas the knot vector defines the influence of the control points on piecewise polynomial curve segments.

Cubic B-splines are denoted as a sum of M control points p_j, weighted by B-spline basis functions N_j^3 of degree 3:

$$S(u) = \sum_{j=1}^{M} N_j^3(u) \cdot p_j \quad (2)$$

The knot vector is chosen to have equally spaced knots, which means each control point has the same influence on the curve. B-splines are able to locally adapt the shape of a monotonic point cloud accurately, which makes them a very flexible and efficient tool in the present case of data compression [6], [4].

In order to fit the sequence β using the above setup, the n data values b_i are mapped to uniformly distributed values $x_i \in [0, 1]$. For each point a linear equation $S(x_i) = b_i$ according to eqn. (2) is set up, which gives a linear system

$$\mathcal{N}_n \cdot \mathcal{P} = \beta \quad (3)$$

$\mathcal{N}_n$ holds B-spline basis functions $N_j^3(x_i)$ for n equations and $\mathcal{P}$ holds control points p_j. The system is solved using the least squares method. This involves computing the Moore-Penrose Pseudoinverse $\mathcal{N}_n^+$. The control points, which minimize least square distances are given by $\mathcal{P} = \mathcal{N}_n^+ \cdot \beta$.

In the case of $n \leq M$, the observed block has a high amount of geometry cells. Then the system is underdetermined and the data will be stored directly, since the number of data values is less or equal to the number of control points.

Since the number n of data values per block is varying, the number of equations in the linear system 3 also varies. To address this problem, the matrices $\mathcal{N}_n$ and $\mathcal{N}_n^+$ are precomputed for $n = M+1, \ldots, N$ and the corresponding case is selected at compressing and decompressing respectively.

Once the data has been fitted, eqn. 2 is used to reconstruct the data from control points $\mathcal{P}$ and x_i values, which yields an approximation of β. The approximate sequence is denoted as $\beta' = (b'_1, \ldots, b'_n)$ and is obtained by $\beta' = \mathcal{N}_n \cdot \mathcal{P}$. In the next step the approximation error $b_i - b'_i$ is handled by applying error correction.

3.3 Error Quantization

Cubic B-spline regression ensures an accurate fitting model for the data. However, the approximation β' does not meet the precision needed for scientific data analysis. Thus an (unspecified) additional error quantization step is applied in ISABELA [6]. In the following, we describe the error quantization method developed for our ISABELA variant.

A user-defined error criterion is met by setting an upper bound p for the relative error ϵ_i on a per-point basis. The relative error is measured between sequence β and $\gamma = (c_1, \ldots, c_n)$:

$$\epsilon_i = \frac{(b_i - c_i)^2}{b_i^2 + \bar{\epsilon}} \leq p \quad (4)$$

γ denotes the sequence of data, reconstructed from the approximate sequence β' by applying an error correction step. Since the relative error can grow without limit for values near zero, the maximum relative error is bounded by introducing $\bar{\epsilon}$ into eqn. 4 [12]. The error correction step is given by the following quantization scheme

$$c_i = b'_i + \delta_i \cdot \bar{c} \cdot \Delta(b'_i) \quad (5)$$

where δ_i denotes integer quantization steps and $\bar{c} \cdot \Delta(b'_i)$ denotes the real valued width of the quantized value grid.

The quantization scheme in eqn. 5 uses a variable quantization width depending on b'_i, since a global width guaranteeing $\epsilon_i \leq p$ can become very small having δ_i to grow very high.

$\Delta(b'_i)$ describes the maximum tolerance, so that the relative error between b'_i and $b'_i \pm \Delta(b'_i)$ is less or equal to p. The variable quantization width follows from eqn. 4 and is given by

$$\Delta(b'_i) = p \cdot \sqrt{b'^2_i + \bar{\epsilon}} \tag{6}$$

Since original data values b_i from β are not known at decompressing, B-spline approximations b'_i near b_i are used to predict the quantization width. Since $\Delta(b'_i)$ does not bound the relative error with respect to b_i, a scaling factor $\bar{c}$ is applied to ensure, that

$$\bar{c} \cdot \Delta(b'_i) \leq \Delta(b_i) \tag{7}$$

holds, which guarantees eqn. 4 to hold for c_i computed from eqn. 5. The scaling factor is obtained by

$$\bar{c} = \min_i \frac{\Delta(b_i)}{\Delta(b'_i)} \tag{8}$$

Given β, β' and $\bar{c}$, the integer quantization steps are computed by

$$\delta_i = \left\lfloor \frac{b_i - b'_i}{\bar{c} \cdot \Delta(b'_i)} + \frac{1}{2} \right\rfloor \tag{9}$$

To store δ_i efficiently a stream of B_δ bit numbers is used. The number of bits needed to store δ_i is determined by

$$B_\delta = \left\lceil \mathrm{ld}\left(\max_i \delta_i\right) \right\rceil \tag{10}$$

Using τ the values c_i of γ are reordered into $\gamma' = (c'_1, \ldots, c'_N)$, which denotes an approximation of α meeting bounded relative error criterion. The reconstruction is accomplished for $i = 1, \ldots, n$ by setting $c'_{t_i} = c_i$. Remaining values with indices i not being in τ are set to $c'_i = \bar{a}$ to reconstruct geometry cells.

3.4 Exploiting Temporal Coherence

In ISABELA, temporal data patterns are exploited to enhance compression ratio. The improvement results from similarities in the index sequence τ, which changes only slightly over consecutive timesteps on a per block-basis. The pattern is applied by introducing a reference sequence τ^0 and delta sequences $\tau^1, \tau^2, \ldots$. Instead of storing the indices of delta sequences directly, the differences $\tau^1 - \tau^0, \tau^2 - \tau^0, \ldots$ are stored. This yields mainly small integer numbers, which are compressed using a lossless compression technique. Using this procedure the compression performance of ISABELA is improved by $\sim 3\%$.

In our implementation, we exploit another temporal pattern, which relies on the continuous nature of the data in time dimension. Since linearized sequences α of consecutive timesteps change smoothly, difference encoding of data values can be applied. For this, α^0 is considered a reference sequence and following sequences $\alpha^1, \alpha^2, \ldots$ are considered delta sequences. Temporal smoothness is sketched in figure 2.

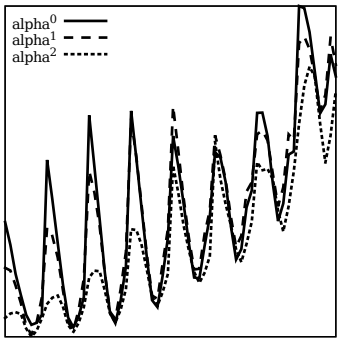


Fig. 2: Illustration of temporal smoothness of consecutive sequences α^0, α^1 and α^2 of u data for one block.

Instead of compressing the data values of delta sequences directly, differences of data values to decompressed reference sequence $\bar{\alpha}^1 = \alpha^1 - \gamma'^0, \bar{\alpha}^2 = \alpha^2 - \gamma'^0, \ldots$ are taken into account. The resulting values $\bar{\alpha}^k = (\bar{a}^k_1, \ldots, \bar{a}^k_N)$ of a delta sequence with index $k > 0$ are small. Applying sorting, linearizing and B-spline regression yields shortened sequences $\bar{\beta}$, $\bar{\beta}'^k$ and τ^k belonging to $\bar{\alpha}^k$.

Now, the relative error between $\bar{b}^k_i$ and $\bar{c}^k_i$ has to be bounded by p on a per-point basis. The error quantization is accomplished by

$$\begin{array}{rcl} c^k_i & = & c'^0_{t^k_i} + \bar{b}'^k_i \quad + \quad \delta_i \cdot \bar{c} \cdot \Delta(c'^0_{t^k_i} + \bar{b}'^k_i) \\ \underbrace{\bar{c}^k_i}_{c^k_i - c'^0_{t^k_i}} & = & \bar{b}'^k_i \quad + \quad \delta_i \cdot \bar{c} \cdot \Delta(c'^0_{t^k_i} + \bar{b}'^k_i) \end{array} \tag{11}$$

to meet the criterion. As $\Delta(c'^0_{t^k_i} + \bar{b}'^k_i) \approx \Delta(b'^k_i)$ and $\bar{b}^k_i - \bar{b}'^k_i < b^k_i - b'^k_i$, the integer quantization steps

$$\delta_i = \left\lfloor \frac{\bar{b}^k_i - \bar{b}'^k_i}{\bar{c} \cdot \Delta(c'^0_{t^k_i} + \bar{b}'^k_i)} + \frac{1}{2} \right\rfloor \tag{12}$$

become smaller. Thus, B_δ also decreases which reduces the amount of memory needed to reconstruct delta timestep k.

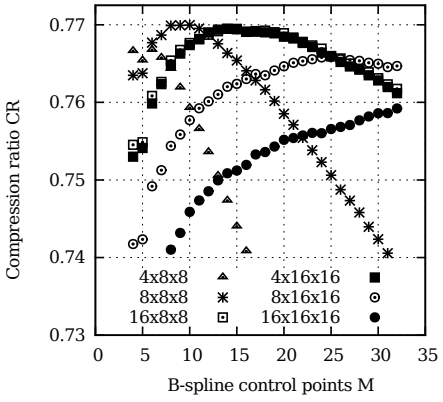
4. Results

The algorithm described in 3 has three parameters: block size N, number of B-spline control points M, and user-defined error threshold p. The parameters influence the compression ratio CR. Assuming 64 bit double precision floating point values, the compression ratio of one block is given by

$$\begin{array}{rcl} CR & = & 1 - \quad [64 \cdot (M+1) + \lceil \mathrm{ld}(N) \rceil \cdot \\ & & \quad (n+2) + B_\delta \cdot n] \cdot {}^1\!/_{(64 \cdot N)} \end{array} \tag{13}$$

Compression ratio is calculated by taking into account $M \cdot 64$ bits for B-spline control points, 64 bits for the scaling factor $\bar{c}$, $\lceil \mathrm{ld}(N) \rceil \cdot (n+2)$ bits for storing τ as well as n and B_δ and $B_\delta \cdot n$ bits for storing error quantization steps.

In order to estimate performance parameters, the algorithm is applied to subgrids of 26 timesteps of simulation data extracted from a Lattice Boltzmann simulation of melt in porous media. The data contains velocity (u, v, w) and density (ρ) values in a $64 \times 64 \times 64$ grid. The entire dataset has a size of $212,992KB$, where each feature has a total size of $53,248KB$.



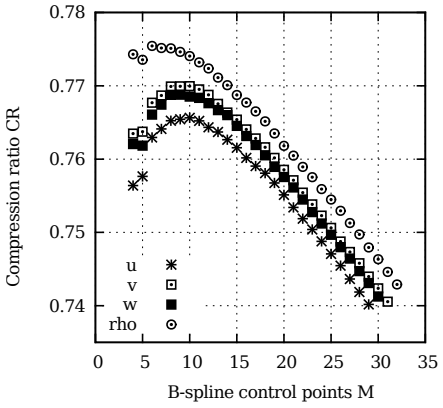


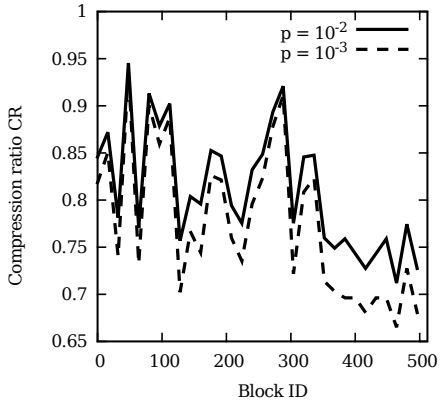
Fig. 3: *Top:* Mean compression rate achieved at compressing u plotted against number of B-spline control points for different block sizes. The optimal compression rate of about 77% was achieved with block size $N = 8 \times 8 \times 8$. *Bottom:* Performance of compressing u, v, w and ρ using block size $N = 8 \times 8 \times 8$. Best compression results over all attributes were achieved for $M = 10$.

4.1 Parameter Analysis

In the compression algorithm as described, the accuracy of cubic B-spline regression primarily depends on the parameters N and M. To resolve the impact of the parameters on the compression ratio, the algorithm was run with varying block sizes and varying number of control points. In figure 3 the results are depicted. We found that $N = 512$ and $M = 10$ yield the best overall compression ratio in our case. We also found that the use of non-cubic block shapes did not improve the compression ratio and we decided to use cubic $8 \times 8 \times 8$ blocks.

On one hand, $N = 512$ is a power of two, which allows to use all $\lceil \mathrm{ld}(N) \rceil$ bits to store the indices in τ optimally. On the other hand, there exists a trade-off between B_δ bits needed to encode the error and the number of control points M. If the data is fitted more accurately by the B-spline, less storage is needed to encode the error quantization steps.

In figure 4, the compression ratio for $N = 512$ and $M = 10$ is given for different user-defined relative error bounds $p = 10^{-2}, 10^{-3}, 10^{-4}$. In parentheses CR for setting $N := n$ is given, which corresponds to not considering geometry cells in the compression rate. In the whole data set 26.9% of the cells are geometry cells. Our experiments show that the algorithm yields high compression ratio ensuring the user-defined error bound. In particular, the algorithm clearly outperforms lossless methods like *gzip* (50.6%), *bzip2* (52.7%) and *lzma* (57.5%), ISABELA-variant algorithms achieve higher compression rates while being local,



p	u	v	w	ρ
10^{-2}	80.2 (71.2)	80.6 (71.7)	80.5 (71.6)	80.9 (70.1)
10^{-3}	76.5 (66.2)	76.9 (66.7)	76.8 (66.5)	77.4 (65.2)
10^{-4}	72.7 (61.0)	73.1 (61.5)	73.0 (61.3)	73.6 (60.1)

Fig. 4: *Top:* Variation of CR per block, implied by changing number of geometry cells and changing number of bits B_δ needed to encode error quantization steps. *Bottom:* Mean compression ratio CR achieved at compressing u, v, w and ρ using $M = 10$ and $N = 8 \times 8 \times 8$ (using variable amount n of data values per block) for different user-defined relative error bounds p. Numbers in parentheses indicate compression rates for non-geometry cells only.

scalable, communication-free and in-situ applicable [6].

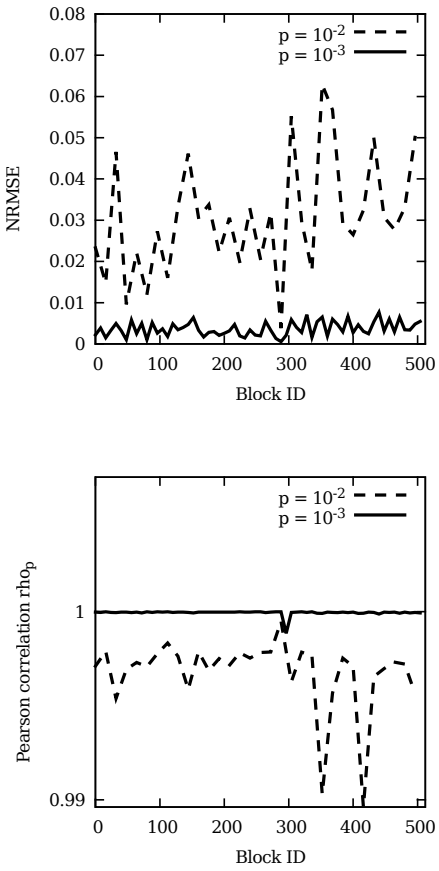


Fig. 5: Performance of the compression algorithm on per-block basis for user-defined relative error bound $p = 10^{-2}, 10^{-3}$. *Top:* normalized root mean square error $NRMSE(\alpha, \gamma')$. *Bottom:* Pearson Correlation $\rho_p(\alpha, \gamma')$.

4.2 Error Analysis

As in [6], normalized root mean square error $NRMSE(\alpha, \gamma')$ and Pearson correlation $\rho_p(\alpha, \gamma')$ are taken into account to observe the performance of the compression algorithm. $NRMSE(\alpha, \gamma')$ and $\rho_p(\alpha, \gamma')$ are given on per-block basis by

$$NRMSE(\alpha, \gamma') = \frac{\sum_{i=1}^{N} \sqrt{(a_i - c_i')^2}}{\max_j a_j - \min_j a_j} \quad (14)$$

$$\rho(\alpha, \gamma') = \frac{\mathrm{Cov}(\alpha, \gamma')}{\sqrt{\mathrm{Var}(\alpha) \cdot \mathrm{Var}(\gamma')}} \quad (15)$$

Error quantization and regression using the least squares method excellently reconstruct the data and yield high compression ratios even for low error threshold $p = 10^{-2}$. In figure 5, the performance is shown in terms of $NRMSE$ and Pearson Correlation ρ_p. $NRMSE$ shows low values, whereas ρ_p shows values being nearly one, which indicates excellent data reconstruction.

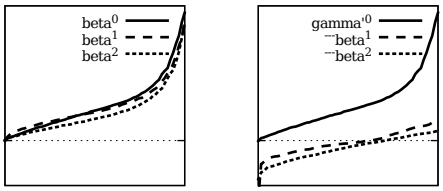


Fig. 6: Impact of difference encoding on the sorted shortened data to be compressed. *Left:* sequences β^0, β^1 and β^2 without difference encoding. *Right:* decompressed reference sequence γ'^0 and difference encoded delta sequences $\bar{\beta}^1, \bar{\beta}^2$. As can be seen, delta sequences are flatter and therefore need less error correction bits.

4.3 Analysis of Time Difference Encoding

The pre-conditioning step exploits the continuous nature of the data in spatial dimensions and produces smooth monotonic curves. Applying cubic B-spline regression with few control points accurately fits the data allowing high compression rates. By also taking advantage of the continuous nature of the flow data in temporal dimension, it is possible to further enhance compression ratio. Considering differences between reference sequences and delta sequences allows to reduce the amount of storage needed to encode error quantization steps. The impact of difference encoding data sequences is depicted in figure 6. For consecutive timesteps' sequences, differences are small and therefore, when being compressed with the same relative error bound as the original data, the amount of error correction data is reduced.

The performance of difference encoding K delta sequences $\alpha_1, \ldots, \alpha_K$ per reference sequence α_0 is shown at compressing 26 timesteps of simulation data. The mean compression ratio and the reduction of the number of bits B_δ needed to encode error quantization steps is shown in figure 7. Compression gain ranges from $\sim 2\%$ to $\sim 3\%$ which compares to the gain of the ISABELA method proposed in [6] exploiting temporal pattern in index sequences.

5. Conclusion

We presented a variant of the ISABELA method for in-situ compression of scientific data. One of our modifications to ISABELA addresses specific properties of flow in porous media. Another modification concerns the exploitation of temporal coherence of time-continuous datasets by compressing differences between reference and delta timesteps. For continuous smooth data, compressing differences yields a significant reduction of error correction data. As a (minor) drawback, difference encoding requires decompressing

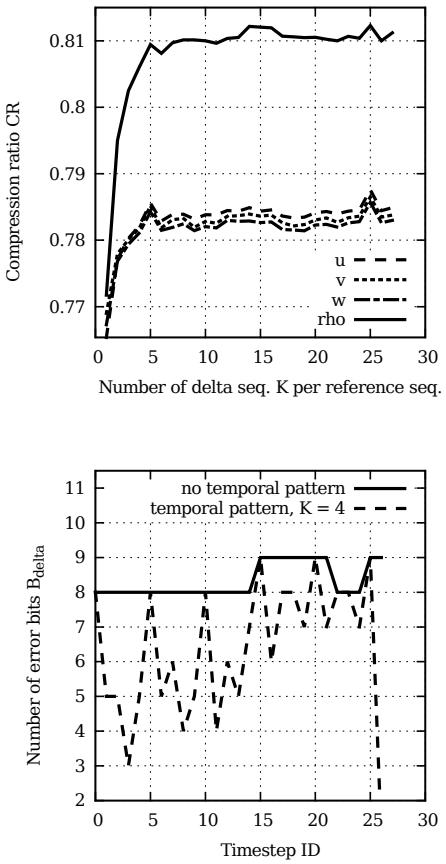


Fig. 7: *Top:* Compressing difference data in temporal dimension enhances mean compression ratio (CR). Plot shows the mean compression ratio achieved at compressing 26 timesteps with varying number of delta sequences K per reference sequence. *Bottom:* Showing reduction of number of bits B_δ needed to encode error quantization steps for one block of u data using difference encoding of $K = 4$ delta sequences.

blocks of reference timesteps. However, the underlying data structures involve B-splines which can be evaluated locally. Thus, local decompression is still possible without significant overhead. Improvements of the overall compression ratio compare to the delta-encoding in the original ISABELA method. Through the combination of both methods, the overall compression rate may be improved even more.

Our results on a melt simulation dataset confirm that high compression rates can be achieved with ISABELA-like algorithms also in the case of porous media. Further, compression lossiness can be controlled comfortably by providing the algorithm with a user-defined error bound. Results were presented that relate several error bound choices to achieved compression rates.

The analysis of our ISABELA variant applied to flow in porous media suggests an optimal block size of $8 \times 8 \times 8 = 512$ data cells. Further, our results indicate that the optimal number of B-spline control points is 10. These results differ from the original ISABELA publication [6] where a block size of 1024 and 30 control points are suggested as optimal. Future work is needed to clarify the optimal parametrizations of ISABELA-like compression methods on different types of datasets.

Acknowledgements

This research was supported by the Deutsche Forschungsgemeinschaft (DFG).

References

[1] Collaborative Research Center 920. Multi-functional filters for metal melt filtration - a contribution towards zero defect materials. TU Bergakademie Freiberg, Freiberg (Saxony), Germany. http://tu-freiberg.de/ze/sfb920/index.en.html.

[2] James Ahrens, Kristi Brislawn, Ken Martin, Berk Geveci, C. Charles Law, and Michael Papka. Large-scale data visualization using parallel data streaming. *IEEE Comput. Graph. Appl.*, 21(4):34–41, July 2001.

[3] Martin Burtscher and Paruj Ratanaworabhan. FPC: A high-speed compressor for double-precision floating-point data. *IEEE Trans. Comput.*, 58(1):18–31, January 2009.

[4] Jin J. Chou and Les A. Piegl. Data reduction using cubic rational b-splines. *IEEE Comput. Graph. Appl.*, 12(3):60–68, May 1992.

[5] Jill R. Goldschneider. *Lossy Compression of Scientific Data via Wavelets and Vector Quantization*. PhD thesis, University of Washington, 1997.

[6] Sriram Lakshminarasimhan, Neil Shah, Stéphane Ethier, Scott Klasky, Robert Latham, Robert B. Ross, and Nagiza F. Samatova. Compressing the incompressible with ISABELA: In-situ reduction of spatio-temporal data. In Emmanuel Jeannot, Raymond Namyst, and Jean Roman, editors, *Euro-Par (1)*, volume 6852 of *Lecture Notes in Computer Science*, pages 366–379. Springer, 2011.

[7] Seungyong Lee, George Wolberg, and Sung Yong Shin. Scattered data interpolation with multilevel b-splines. *IEEE Transactions on Visualization and Computer Graphics*, 3:228–244, 1997.

[8] Kwan-Liu Ma, Chaoli Wang, Hongfeng Yu, and Anna Tikhonova. In-situ processing and visualization for ultrascale simulations. *Journal of Physics: Conference Series*, 78(1):012043+, July 2007.

[9] Les Piegl and Wayne Tiller. *The NURBS book*. Springer-Verlag, London, UK, UK, 1995.

[10] Paruj Ratanaworabhan, Jian Ke, and Martin Burtscher. Fast lossless compression of scientific floating-point data. In *Proceedings of the Data Compression Conference*, DCC '06, pages 133–142, Washington, DC, USA, 2006. IEEE Computer Society.

[11] Victor Sanchez, Rafeef Abugharbieh, and Panos Nasiopoulos. 3d scalable lossless compression of medical images based on global and local symmetries. In *Proceedings of the 16th IEEE international conference on Image processing*, ICIP'09, pages 2497–2500, Piscataway, NJ, USA, 2009. IEEE Press.

[12] John Z. Sun and Vivek K. Goyal. Scalar quantization for relative error. *Data Compression Conference*, 0:293–302, 2011.

[13] A. J. Wagner. A practical introduction to the lattice boltzmann method. Department of Physics, North Dakota State University, 2008.

[14] Feng Wang, Qin Xin, Bo Hong, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Tyce T. Mclarty. File system workload analysis for large scale scientific computing applications. In *Proceedings of the 21st IEEE / 12th NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 139–152, 2004.

[15] T. A. Welch. A technique for high-performance data compression. *Computer*, 17(6):8–19, June 1984.