Data Sci. Eng.
DOI 10.1007/s41019-017-0042-4

Optimal Compressed Sensing and Reconstruction of Unstructured Mesh Datasets

Maher Salloum[1] · Nathan D. Fabian[2] · David M. Hensinger[2] · Jina Lee[1] · Elizabeth M. Allendorf[3] · Ankit Bhagatwala[4] · Myra L. Blaylock[1] · Jacqueline H. Chen[1] · Jeremy A. Templeton[1] · Irina Tezaur[1]

Received: 18 November 2016 / Revised: 2 June 2017 / Accepted: 31 July 2017

Abstract Exascale computing promises quantities of data too large to efficiently store and transfer across networks in order to be able to analyze and visualize the results. We investigate compressed sensing (CS) as an in situ method to reduce the size of the data as it is being generated during a large-scale simulation. CS works by sampling the data on the computational cluster within an alternative function space such as wavelet bases and then reconstructing back to the original space on visualization platforms. While much work has gone into exploring CS on structured datasets, such as image data, we investigate its usefulness for point clouds such as unstructured mesh datasets often found in finite element simulations. We sample using a technique that exhibits low coherence with tree wavelets found to be suitable for point clouds. We reconstruct using the stagewise orthogonal matching pursuit algorithm that we improved to facilitate automated use in batch jobs. We analyze the achievable compression ratios and the quality and accuracy of reconstructed results at each compression ratio. In the considered case studies, we are able to achieve compression ratios up to two orders of magnitude with reasonable reconstruction accuracy and minimal visual deterioration in the data. Our results suggest that, compared to other compression techniques, CS is attractive in cases where the compression overhead has to be minimized and where the reconstruction cost is not a significant concern.

✉ Maher Salloum
mnsallo@sandia.gov

[1] Sandia National Laboratories, Livermore, CA, USA

[2] Albuquerque, NM, USA

[3] University of California, Los Angeles, CA, USA

[4] Pilot AI Labs, Redwood City, CA, USA

Keywords Compressed sensing · Tree wavelets · Compression · In situ · Large-scale simulation · Unstructured mesh · Compression ratio · Reconstruction · Optimal · Gradient

1 Introduction

Large-scale computing platforms are challenged by the growing size of computational datasets generated by simulations. On future exascale platforms, the datasets are expected to be too large to be efficiently stored and transferred across networks to analysis and visualization workstations, thus impeding data exploration and knowledge discovery. Several in situ data compression schemes are available to reduce the size of simulation datasets at each time step as they are generated. The compressed version of the dataset is transferred to any workstation and reconstructed by scientists and engineers. Some conceivable usages of the compressed data in data science and engineering are:

1. *Data visualization* where loss in the quality and resolution in the reconstructed data is usually acceptable which indicates that large compression rates can be performed on the original data.
2. *Resiliency in high-performance computing* where a simulation can be restarted from previously compressed data snapshot rather than the original one in order to save storage space. In this case, it is expedient that the reconstructed data should be as accurate as possible compared to the original data such that no error can grow and propagate in the remaining part of the simulation.
3. *Generating low-dimensional bases* for example in uncertainty analysis studies where multiple snapshots

Published online: 09 August 2017

of the data are taken and can be compressed due to the large redundancies they contain which allows large compression rates and moderately acceptable loss in the data.

A suitable compression scheme is one that has a small impact on the running simulation. In other words, the code should not be significantly altered and the overhead cost should be very low. There are two major types of scientific data compression methods: lossless and lossy methods. In lossless methods, the compression ratio (CR), defined as the ratio between the raw and compressed data sizes, depends on the data entropy [1] which is associated with the randomness in the numerical precision representation of the data variables. Data entropy is very large in data produced by scientific computations, making lossless methods suffer from low compression ratios ($2 \leq \mathrm{CR} \leq 10$). However, lossy methods perform compression by discovering redundancies due to trends and features in the data regardless of entropy. Scientific data are often smooth in space and time which signifies that it would be highly compressible using lossy methods. We propose compressed sensing (CS) as a lossy method to compress and reconstruct datasets.

CS is known to be a fast data compression technique. It works by sampling the data on the computational cluster using any sampling technique within an alternative function space such as wavelet bases described later in this paper. Its compression rates are relatively low in images and videos because such data exhibit a great deal of discontinuities in their color fields. However, data obtained from computational simulations are quite different; they are fairly smooth since they are obtained from the discretization of smooth partial differential equation operators [2]. By virtue of this smoothness, this kind of scientific data has low information density, and starting from this hypothesis, any such data field can be represented in an alternative function space and exhibit a decaying magnitude in its spectrum. Therefore, CS is expected to achieve compression ratios in scientific data that are larger than in images.

CS has several advantages over other compression methods. First, it is data agnostic, i.e., it works as a "black box" since it does not require any knowledge of the simulation data type, structure or the features contained therein, as is the case with wavelet compression (WC). Second, unlike WC, it does not require the selection of a basis type and order during the compression in situ. Suitable wavelets bases are selected and computed during the post-processing stage allowing interactive reconstruction and visualization according to the required accuracy and quality. Finally, CS can be implemented in practice in a non-intrusive manner which means that existing simulation codes and their interfaces will not be significantly altered when data compression is invoked.

On the other hand, the data reconstruction procedure is relatively slow. Thus, it is unlikely that CS would find widespread application in data compression, as there are methods with faster reconstruction, e.g., vector quantization (VQ) [3] and floating-point array compression [4]. Nevertheless, there may be instances where the data reconstruction speed is not a significant concern. For example, in exploratory research, the interactive sparse reconstruction is made available on dedicated analysis and visualization machines. In another example, in situ computations may themselves be fast but produce large data that need to be compressed with a low overhead. However, it will be difficult to devise a data compression technique based on CS for such cases without a thorough understanding of its performance, reconstruction accuracy, practical implementation and generality.

Conventional CS theory, developed for image compression, is based on the representation of lattice data using the so-called first-generation wavelets (FGW) [5]. We are unaware of any literature on the applicability of compressed sensing on point cloud (irregularly spaced points) data. In this paper, we extend the CS theory to encompass second-generation wavelets and tree wavelets (TW) that can be described on point clouds, e.g., an unstructured mesh. To our knowledge, this extension of CS to point cloud data has not been explored yet. FGWs and TWs are described in Sect. 2.2.

1.1 Literature Review

In situ reduction of large computational datasets has recently been the subject of multiple research efforts. In the ISABELA project [1, 6], the dataset is sorted as a vector and encoded with B-splines while computing the resulting associated errors. DIRAQ is a more recent effort by the same research group [7]. It is a faster parallel in situ data indexing machinery that enables efficient query of the original data. However, ISABELA and DIRAQ suffer from low compression ratios ($\sim 4-6\times$), which might not be sufficient for exascale datasets in which I/O is a critical bottleneck. The *zfp* project, [4], works by compressing 3D blocks of floating-point field data into variable-length bitstreams. The *zfp* approach results in compression ratios larger than 10 with very good accuracy but is limited by the bitstream length on how much it can compress without significant error loss. Adaptive compression based on the error for a given block can yield much higher compression rates for a smaller increase in error. Vector quantization [3, 8] is a traditional approach for compressing image data. Similar to *zfp*, it works by breaking up data into blocks, treating

them as vector quantities, clustering them into a number of groups based on the requested reduction size and replacing individual blocks by their centroid. Usually, it is only applicable to regular grid data. Bernardon et al. [8] applied VQ to unstructured grids by quantizing in time rather than in space. Most of the compression approaches are effective for temporal data but require accumulating several time steps data in memory which might be prohibitive on computational clusters. Therefore, we only examine compression at specific time steps in this paper.

The compression scheme of Zhao et al. [9] works similarly to *zfp* on spatio-temporal data, providing large compression ratios and fast decompression speeds but it is also limited to regular grid data. However, our CS approach compresses unstructured data and we expect to obtain compression ratios up to two orders of magnitude while keeping a reasonable accuracy in the reconstructed results. A very recent effort in scientific data compression relies on the Tucker tensor decomposition [10] to reduce the size of a large dataset gathered from different time steps, variables and computing nodes. Tucker compression results in compression ratios up to three orders of magnitudes, but it is unclear how it would perform in situ for data generated at one time step at a time. Furthermore, this technique is applicable only on regular grids and it is not clear how it would adapt to unstructured meshes. Finally, CS attempts for scientific data reduction and visual rendering are currently limited to data represented on regular grids [11–13].

Ongoing research efforts include in situ visualization [14] and feature extraction [15] using combustion simulation datasets from the S3D simulator developed at Sandia National Labs (SNL). Selected simulation outputs are analyzed and visualized in situ while data analysis occurs at separate computational nodes incurring a significant overhead. Moreover, such in situ techniques require preselected outputs and cannot be used interactively since most clusters are operated in batch mode. Similar techniques have been implemented in the ParaView coprocessing library [16].

Sampling techniques are also employed in situ for data reduction. Woodring et al. [17] devised an approach for a large-scale particle cosmological simulation. A major feature in this work is the low cost of the offline data reconstruction and visualization. However, the compression ratios are low and the technique requires skipping levels in the simulation data. Sublinear sampling algorithms have also been proposed for data reduction [18] in the analysis of stationary graphs. An ongoing effort attempts to transfer sublinear sampling theory into practice with focus on large combustion datasets.

1.2 Proposed Work and Approach

In this paper, we present a compression/decompression method based on CS for scientific data represented on irregular geometries and unstructured meshes. The data reconstruction (decompression) is based on Alpert tree wavelets [19, 20], suitable for irregularly spaced data. We use the stagewise orthogonal matching pursuit (StOMP) algorithm [21] as the sparse data reconstruction method. We also present adaptive algorithms that alleviate the method's computational cost and enhance its accuracy in high-performance computing (HPC) engineering applications.

This paper is organized as follows. In Sect. 2, we describe the mathematical tools of CS, tree wavelets and sparse data reconstruction used in the compression/decompression method we are proposing. We also propose two improvements to the StOMP algorithm by automating its choice of a threshold parameter and one linear algebra operation in order to alleviate decompression complications and obtain a systematic reasonable accuracy. In Sect. 4, we present our preliminary results of the method for a small-scale toy problem in order to validate its operation. In Sect. 5, we develop a correlation that predicts the optimal compressibility of a dataset according to a given reconstruction accuracy. In Sect. 6, we apply the method along with its improvements to some large-scale datasets and present its performance in terms of accuracy and compression and decompression speeds on datasets obtained from turbulent combustion simulations and land-ice modeling. Finally, in Sect. 7, we summarize the findings in this paper and describe the limitations and propose future directions of our study.

2 Mathematical Tools

In this section, we describe the mathematical tools required for our compression/decompression algorithms. These are compressed sensing, wavelets and sparse reconstruction solvers. They operate in the same way as in the case when CS is applied to image data. The vector representation of the data is exactly the same in the case of point clouds and unstructured meshes. In fact, any data can be gathered in a vector form regardless of whether it is represented on a regular grid or not. The major difference is in the way the wavelets are computed, but their matrix form is the same as in traditional CS.

2.1 Compressed Sensing

Compression using CS can be cast mathematically as the sparse sampling of the dataset in transform domains.

Compressed samples are obtained by projecting the data on random vectors [22],

$$\boldsymbol{y} = [\Phi]\boldsymbol{f}, \tag{1}$$

where $\boldsymbol{y} \in \mathbb{R}^M$ are the compressed samples, $\boldsymbol{f} \in \mathbb{R}^N$ is the signal we are sampling and $[\Phi] \in \mathbb{R}^{M \times N}$ is the sampling matrix. Here $\boldsymbol{f}$ represents the data field to be compressed, which can be represented on a regular grid or point cloud. The compressed version of the data field is the vector $\boldsymbol{y}$ that is shorter than $\boldsymbol{f}$ ($M < N$). To this end, information about the mesh connectivity is not needed. If multiple data fields are considered for compression, they are gathered in columns of a matrix and the compression in Eq. 1 becomes a matrix–matrix product. Theoretically, only $C \cdot K \cdot \log_2(\mathrm{N}/\mathrm{K})$ samples are necessary for reconstruction, where K is the number of active wavelet modes considered and C is a constant that depends on the data [23].

In practice, reconstruction is posed as a linear inverse problem for the wavelet coefficients $\boldsymbol{s} \in \mathbb{R}^N$ which are the representation of the data $\boldsymbol{f}$ in the wavelet space following:

$$\boldsymbol{f} = [\Psi]\boldsymbol{s}, \tag{2}$$

with $[\Psi]$ being the wavelet basis matrix. Combining Eqs. 1 and 2, the inverse problem is stated as the following underdetermined system,

$$\boldsymbol{y} = [\Phi][\Psi]\boldsymbol{s} = [A]\boldsymbol{s}, \tag{3}$$

and we must find $\boldsymbol{s}$ from samples $\boldsymbol{y}$ obtained by compression. Regularization is provided by a L_1 norm of the wavelet coefficients, which also enforces sparsity. The random vectors in $[\Phi]$ are designed to be incoherent with the chosen wavelets. Incoherence ensures that the amount of wavelet information carried by the compressed samples is high. Different types of sampling matrices $[\Phi]$ can be used to perform compression [22] (see Eq. 1). Once $\boldsymbol{s}$ is computed, the data $\boldsymbol{f}$ can be recovered by an inverse wavelet transform. Data loss occurs mainly in the sampling procedure in Eq. 1. Another source of data loss is the error caused by the approximate numerical solution of Eq. 3.

2.2 Tree Wavelets

Wavelets can represent a basis set that can be linearly combined to represent multiresolution functions. Wavelets have compact support and encode all the scales in the data (j) as well as their location (k) in space. First-generation wavelets (FGW) are the most commonly used in compressing images and videos. To date, all CS work of which we are aware of is based on FGWs which only accommodate data defined on regular grids. FGWs are characterized by scaling functions $\phi(x)$ to perform approximations and $\psi(x)$ to find the details in a function f [24]. These are defined in Eqs. 4 and 5 at all levels of j in the hierarchy and span all locations k in the regular grid. They are computed in terms of the so-called mother wavelet ψ_0 which provides the approximation at the largest level $j = 0$. At each level j, the scaling functions are defined as,

$$\phi_j(x) = \sum_{k \in \mathbb{Z}} a_k \phi_{j-1}(2x - k), \tag{4}$$

and the wavelets are given by:

$$\psi_j(x) = \sum_{k \in \mathbb{Z}} b_k \phi_{j-1}(2x - k), \tag{5}$$

where x is the spatial coordinates and a_k and b_k are coefficients usually constrained to maintain basis orthogonality. Each new basis function models a finer resolution detail in the space being spanned. Regular grids are dyadic and the maximum number of levels $j_{\max}$ is equal to $\log_2(N)$ where N is the number of grid points in each dimension.

In this work, we examine compression on unstructured mesh data or point clouds which are not well represented by FGWs [25] and require non-traditional wavelet bases such as second-generation wavelets [25, 26], diffusion wavelets [27] or tree wavelets (TW) [19, 20]. There are two major differences between FGWs and TWs. The first difference in TW is that the maximum number of levels $j_{\max}$ is computed by recursively splitting the non-dyadic mesh into different non-overlapping groups, thereby forming a multiscale hierarchical tree [25]. Since TWs do not require dyadic grids, they can accommodate finite intervals and irregular geometries. The second difference is that the scaling functions and the wavelets themselves depend on the mesh because they are defined by the discrete set of points, x_k, as opposed to FGWs where the regular grid coordinates do not enter in the wavelets calculation.

In our work, we use the TWs of Alpert et al. [19, 28], referred to as multiwavelets. The major advantage of these TWs is that their computation does not require any special treatment of the domain boundaries or any holes present therein (e.g., by converting the domain to a simply connected rectangle via cuts and mapping) and they avoid them by construction [20]. The discrete Alpert wavelets $\psi_j(x_k)$ are polynomials, are quick to compute and are represented in the matrix, $[\Psi]$.

2.2.1 Building the Wavelets Matrix

Computing a wavelet matrix recursively using Eqs. 4 and 5 is not practical. Therefore, a discrete methodology is employed to build the Alpert wavelet matrix. Here, we briefly present this methodology for the case of one-dimensional (1D) non-

dyadic mesh for a polynomial order w. Consider a 1D mesh with N points $x_1 < \cdots < x_i < \cdots < x_N$.

The first step is to subdivide the mesh into P almost equally sized bins such that the number of points n per bin is $2w \geq n \gtrsim w$. This subdivision operation is obvious and not necessary in the case of a dyadic mesh where FGW are used.

The second step is to compute the so-called initial moment matrices $[M]_{1,1\leq p\leq P} \in \mathbb{R}^{n\times w}$ for each bin:

$$[M]_{1,p} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{w-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{w-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^{w-1} \end{bmatrix} \tag{6}$$

The content of the $[M]_{1,p}$ is the wavelet functions ψ in Eq. 5, assumed to be polynomials in Alpert multiwavelets [19]. If $[M]$ is not a square matrix, then its first n columns are selected to make it suitable for orthogonalization. For each bin, we also compute the matrices $[U]_{1,p} \in \mathbb{R}^{n\times n}$ as the orthogonalized (e.g., using a QR operation) and transposed version of the matrices $[M]$.

The third step is to assemble the matrices $[U]_{1,p}$ in a large matrix $[V]_1 \in \mathbb{R}^{N\times N}$ similarly to Fig. 1 (left column). The columns of the matrix $[V]$ cover the N points in the mesh and the N rows cover the N wavelet coefficients where the detail level increases at the lower regions in $[V]$. Here, $w = 3$ and $n = 4$; therefore, the $w = 1$ rows in the lower part of $[U]_{1,p}$ are assigned to the lower part of $[V]$ since they involve larger polynomial orders hence finer details. The upper part ($n - w = 1$ row in this case) is assigned to the upper part of $[V]$.

The fourth step is to account for the finer detail levels of the mesh. The P bins obtained in the mesh subdivision constitute a tree hierarchy of the details found in the mesh. The maximum number of detail levels is computed as $j_{\max} = \log_2(\mathrm{P}) + 1$. Matrices $[V]_{2\leq j\leq j_{\max}}$ have to be computed similarly to $[V]_1$ in order to compute the wavelet matrix as:

$$[\Psi] = \prod_{j=1}^{j_{\max}} [V]_j \tag{7}$$

When performing the matrix products, the contribution of the coarse details has to be maintained. Thus:

$$[V]_{2\leq j\leq j_{\max}} = \begin{bmatrix} [I]_{N'(j)\times N'(j)} & 0 \\ 0 & [\tilde{V}]_{P'(j)\times P'(j)} \end{bmatrix} \tag{8}$$

in which $P'(j) = Pw/2^{j-2}$ and $N'(j) = N - P'(j)$. The submatrices $[\tilde{V}]_j$ are assembled by submatrices $[U]_{j,1\leq p\leq P/2^{j-1}}$ similarly to the third step. The new matrics $[U]_{j,p}$ are obtained by orthogonalizing new matrices $[M]_{j,p}$, which in turn are computed recursively from $[U]_{j-1}$ and $[M]_{j-1}$. More details on this step are given in [19, 20].

The above procedure can be generalized to D-dimensional meshes with the following differences:

1. The domain decomposition into almost equal bins is performed using algorithms such as k-means clustering.

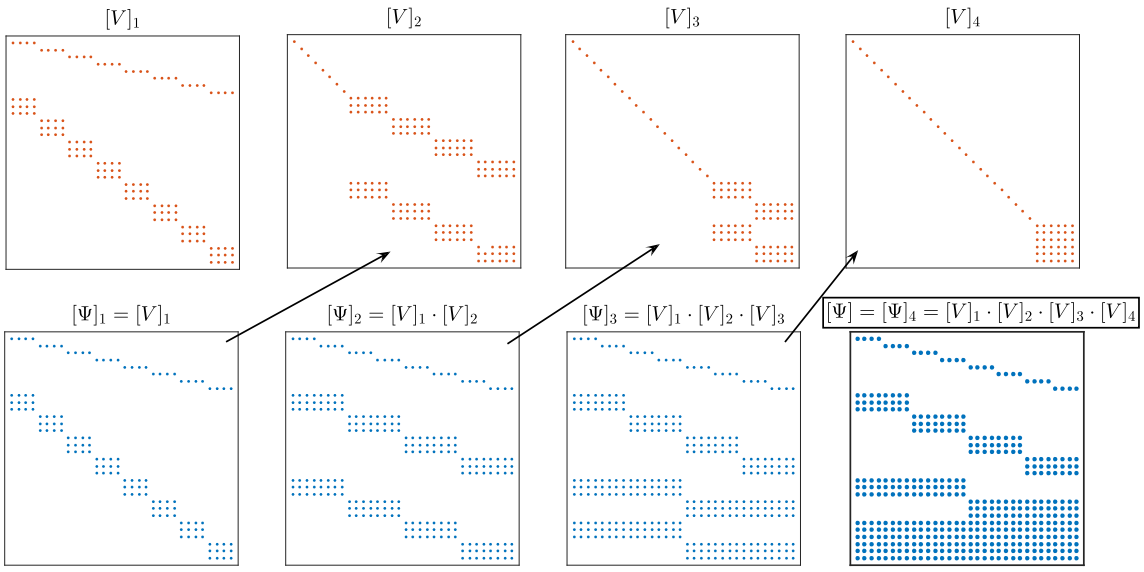

Fig. 1 Representations of the sparse Alpert wavelet matrices at different detail levels j. For each level j, the wavelet matrix $[\Psi]_j$ is the cumulative product of the matrices $[V]_{i\leq j}$. Shown are matrices computed for a one-dimensional mesh containing $N = 32$ points where the wavelet order is $w = 3$ such that the mesh is divided into $P = 8$ bins and exhibits $j_{\max} = 4$ detail levels. The mesh is not necessarily dyadic

2. The polynomial order $w_{1 \le d \le D}$ is assigned for each dimension d such that the effective order is $w = \prod_{d=1}^{D} w_d$.
3. The entities $m_{\alpha,\beta}$ in the matrix $[M]_{1,p}$ are computed as:

$$m_{\alpha,\beta} = \prod_{d=1}^{D} x_{d,\alpha}^{\beta_d - 1}, \tag{9}$$

where x_d are the coordinates in the dimension d and $1 \le \beta_d \le w_d$.

Remarks

1. The procedure described above to build the wavelet matrix $[\Psi]$ guarantees that it is orthonormal such that its matrix inverse is equal to its transpose, $[\Psi]^{-1} = [\Psi]^T$
2. For $w = 1$, the Haar wavelets are recovered and the domain is decomposed to the mesh point level such that $P = N$ and $n = 1$. In this case, the moment matrices reduce to a scalar equal to one.

2.3 Reconstruction

Much of the work in CS is handled by the reconstruction phase which uses wavelet basis to reconstruct the dataset from the compressed samples. Different algorithms exist in the literature. Here, we use a greedy algorithm called stagewise orthogonal matching pursuit (StOMP) [21] that has been empirically demonstrated to be very efficient.

The reconstruction process can be described as follows. Given the underdetermined linear system in Eq. 3, we need to find the sparsest vector $\boldsymbol{s}$ satisfying that equation. In other words, the StOMP algorithm computes an approximate solution to the problem:

$$\text{minimize} \quad ||\boldsymbol{s}||_1 \quad \text{such that} \quad \boldsymbol{y} = [\Phi][\Psi]\boldsymbol{s} = [A]\boldsymbol{s} \tag{10}$$

If $[\Phi]$ and $[\Psi]$ exhibit low mutual coherence and $\boldsymbol{s}$ is sparse, i.e., has few nonzero elements, then StOMP can efficiently provide a solution. StOMP finds the nonzero elements of $\boldsymbol{s}$ through a series of increasingly correct estimates. Starting with an initial estimate $\boldsymbol{s}_0 = \boldsymbol{0}$ and an initial residual $\boldsymbol{r}_0 = \boldsymbol{y}$, the algorithm maintains estimates of the location of the nonzero elements of $\boldsymbol{s}$. We note that the input compression ratio defined as the ratio between the sizes of vectors $\boldsymbol{f}$ and $\boldsymbol{y}$ in Eq. 1 is not expected to match the sparsity of the output $\boldsymbol{s}$ of the StOMP algorithm.

StOMP repeats the operations below until convergence. It first finds residual correlations

$$\boldsymbol{c}_n = [A]^T \boldsymbol{r}_{n-1} / ||\boldsymbol{r}_{n-1}||_2, \tag{11}$$

and uses them to find a new set of nonzero entries,

$$J_n = \{j : |c_n(j)| > \theta\}, \tag{12}$$

where θ is a threshold parameter introduced in the original StOMP algorithm in order to assess which coefficients c_n have to be retained. θ is not meant to ensure the validity of the input compression ratio R. Elements above the threshold are considered nonzero entries and added to the set J_n. The new set of entries J_n is added to the current estimate of nonzero entries $I_n = I_{n-1} \cup J_n$ and is used to give a new approximation of $\boldsymbol{s}_n$,

$$\boldsymbol{s}_n = ([A]_{I_n}^T [A]_{I_n})^{-1} [A]_{I_n}^T \boldsymbol{y}, \tag{13}$$

and residual, $\boldsymbol{r}_n = \boldsymbol{y} - [A]\boldsymbol{s}_n$. Equations 12 and 13 are part of the original StOMP algorithm published in [21]. The matrix $[A]_{I_n} \in \mathbb{R}^{M \times card(I_n)}$ is the subset extracted from the columns I_n of the matrix $[A]$. These steps are summarized in Algorithm 1 along with the proposed improvements to the StOMP scheme.

2.3.1 *Improving the Evaluation of s_n*

In the original StOMP algorithm, there is no linear system solution in Eq. 13, it is just an evaluation of the vector $\boldsymbol{s}_n$ as a function other entities in the algorithm. We have improved this StOMP algorithm through a more accurate evaluation of $\boldsymbol{s}_n$ in Eq. 13, especially when the matrix $[A]_{I_n}^T [A]_{I_n}$ has a low condition number. Our improved algorithm first approximates the reciprocal condition number of $[A]_{I_n}^T [A]_{I_n}$ [29]. If it is larger than a tolerance we set to 2.2×10^{-16} then we evaluate $\boldsymbol{s}_n$ directly using Eq. 13, i.e., by inverting the matrix $[A]_{I_n}^T [A]_{I_n}$. Otherwise, the matrix is ill-conditioned so we evaluate $\boldsymbol{s}_n$ by solving the underdetermined system $[A]_{I_n}^T \boldsymbol{s}_n = \boldsymbol{y}$ using the least squares method which is equivalent to Eq. 13 but does not suffer from ill-conditioning and results in reasonable solutions. The solution of an underdetermined system using the least squares method is well documented in the literature [29]. The latter case might not occur frequently; it mostly happens for large compression ratios. This improvement we are providing keeps the algorithm robust in such cases to

result in reasonable reconstructions without breaking the code execution.

Algorithm 1 The automatically tuned Stagewise Orthogonal Matching Pursuit (StOMP) algorithm for the solution of underdetermined systems $[A]\boldsymbol{s} = \boldsymbol{y}$.

$\boldsymbol{s} = \boldsymbol{s}_0$ {Initial guess of the solution}
$\boldsymbol{r}_0 = \boldsymbol{y} - [A]\boldsymbol{s}_0$ {Initial residual}
$\rho_0 = \|\boldsymbol{r}_0\|_2$
$\theta = 3\|\boldsymbol{y}\|_2\|\boldsymbol{y}\|_1^{-1}R^{-0.5}$ {Optimal threshold parameter}
for $n = 1$ until convergence **do**
  $c_n = [A]^T\boldsymbol{r}_{n-1}/\rho_{n-1}$
  $J_n = \{j : |c_n(j)| > \theta\}$
  $I_n = I_{n-1} \cup J_n$ {The active wavelets indices}
  $[B] = [A]_{I_n}^T[A]_{I_n}$
  if rcond($[B]$) $< 2.2 \times 10^{-16}$ **then**
    Solve $[A]_{I_n}\boldsymbol{s}_n = \boldsymbol{y}$ {$\boldsymbol{s}_n$ is the least-squares solution if $[B]$ is poorly conditioned}
  else
    $\boldsymbol{s}_n = [B]^{-1}[A]_{I_n}^T\boldsymbol{y}$
  end if
  $\boldsymbol{r}_n = \boldsymbol{y} - [A]_{I_n}\boldsymbol{s}_n$
  $\rho_n = \|\boldsymbol{r}_n\|_2$
end for
Return $\boldsymbol{s}$ with $\boldsymbol{s}_{I_n} = \boldsymbol{s}_n$ and $\boldsymbol{s}_{i \notin I_n} = \boldsymbol{0}$

2.3.2 *Prediction of the StOMP Threshold Parameter*

All sparse reconstruction algorithms involve a number of "knobs" such as threshold and regularization parameters that have to be properly tuned in order to converge to a unique solution. This makes the use of such methods impractical for automated batch data reconstruction jobs. The threshold parameter, θ, of StOMP in Eq. 12 is usually chosen from a range of values [21] and provided as an input to the algorithm. An ad hoc choice of θ might result in a large error; it has to be within a restricted range as shown in Fig. 2 to grant a minimal error, i.e., a reasonably accurate result for the input compression ratio R regardless of the sparsity of the output $\boldsymbol{s}$. This might not be practical for batch jobs where different problems require different values of θ. Therefore, we propose an empirical technique similar to [30] to estimate θ inside the StOMP algorithm as a function of other inputs. This grants both a minimal error in the reconstructed data and an automated use of StOMP in batch reconstructions.

As mentioned previously, the role of the threshold θ is to assess which wavelet coefficients have to be retained at each iteration. We assume a dependency of θ on the L_1 and L_2 error norms ratio $||y||_2/||y||_1$ (or its inverse) that was first observed in the derivation found in [31] when maximizing the information retention in the wavelet dictionary. Moreover, it is known that the threshold is related to the sparsity of the solution $\boldsymbol{s}$ [30] expressed by its L_1 norm $||\boldsymbol{s}||_1$ which in turn is related to $||\boldsymbol{y}||_1$ following Eq. 1. According

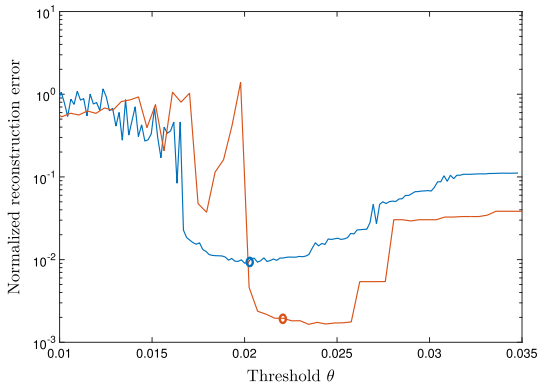

Fig. 2 Variation of the StOMP reconstruction error as a function of the threshold parameter θ for two different datasets (see Sect. 4.1, Fig. 4). For each dataset, θ was varied for the same compression ratio and reconstruction tolerance

to our terminology, θ is dimensionless. Thus, according to our first observation, we decided to normalized $||\boldsymbol{y}||_1$ by $||\boldsymbol{y}||_2$ in the correlation model we assume for θ. We also assumed a dependence of θ on the compression ratio R. In fact, a larger compression ratio implies the existence of fewer dominant modes which manifests a lower threshold. According to these *assumptions*, we propose the following model for θ:

$$\theta = k_0\left(\frac{\|\boldsymbol{y}\|_2}{\|\boldsymbol{y}\|_1}\right)^{k_1} R^{k_2} \tag{14}$$

and obtain the coefficients k_0, k_1 and k_2 by finding θ that grants reasonable reconstructions of given datasets of different sizes representing functions similar to f and g in Eqs. 16–17 (combinations of sines and cosines). These are representative of most PDE solution functions to the extent that they can be expanded into sines and cosines according to Fourier series. We omit the details of this fitting procedure in this paper for the sake of brevity, but report that the values $k_0 = 3, k_1 = 1$ and $k_2 = -0.5$ are obtained. This closed form solution is interesting and requires a more thorough theoretical proof that is beyond the scope of this study. Even though it was obtained empirically from a set of training data, we found that it is applicable to all other datasets we considered in this paper with no need for further tuning unlike the one reported in [30]. The values of θ estimated from Eq. 14 are highlighted as the circle markers in Fig. 2. They fall in the range where the reconstruction error is minimal.

3 Implementation

In order to test the compression/decompression procedure, we have built a processing pipeline that both samples and reconstructs data from a dataset. It allows us to experiment with existing datasets and assess reconstruction quality.

The library is split into two pieces. The in situ piece consists of a small sampling code base. It stores the samples, mesh points coordinates and the seed used to generate the Bernoulli sampling matrix $[\Phi]$. During the in situ processing, the sampling matrix is not constructed explicitly, but used implicitly to generate the sampled data. No wavelet computation is required at this stage.

The reconstruction side is responsible for rebuilding the sampling matrix and constructing the wavelet matrix from the mesh data. By providing those matrices to StOMP, we are able to reconstruct the wavelet sampled data, $\boldsymbol{s}$ in Fig. 3, and then inverse transform $\boldsymbol{s}$ to reconstruct the original data $\boldsymbol{f}$.

We have implemented the reconstruction procedure in two ways: (1) in MATLAB in order to experiment and produce some of the results shown in Sect. 4.1 and (2) a more production-oriented version written using the Trilinos sparse linear algebra library [32] and ParaView for visualization [33]. We have published an open source basic version of this library in SWinzip v1.0 [34] that does not require Trilinos or ParaView. SWinzip contains C++ and MATLAB versions of the library.

The current implementation acts on data distributed on many cores/processes in the following manner. It performs a local in situ compression on each process during a HPC simulation (step I) as illustrated in Fig. 3. This allows a communication-free compression resulting in a highly scalable compression method. The data reconstruction (step II) also takes place locally for each data subdomain corresponding to each core/process.

In our study, we assess the data reconstruction error using the mean squared error normalized by the data size and by the difference between the maximum and minimum value in the data. We refer to this metric as the normalized relative mean squared error (NRMSE). The NRMSE between a dataset given by a field f and defined on N points, and its reconstructed version f^r is computed as:

$$\text{NRMSE} = \frac{||f - f^r||_2}{\sqrt{N}[\max(f) - \min(f)]} \tag{15}$$

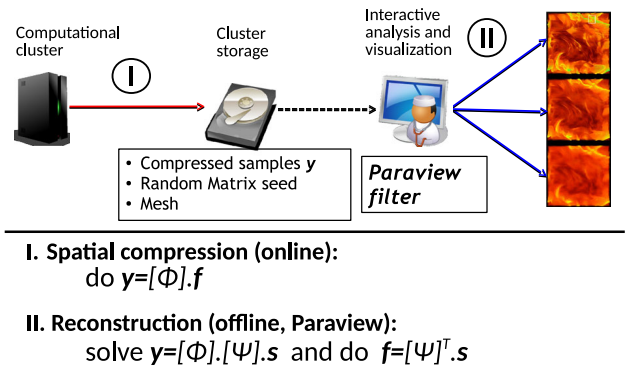

Fig. 3 A schematic showing the major steps during the in situ data compression using CS (*I*) and the offline reconstruction and visualization of the original dataset (*II*) using ParaView. The random seed, the mesh and the compressed samples $\boldsymbol{y}$ are transferred to the visualization platform where the reconstruction of $\boldsymbol{f}$ takes place

4 Results

In this section, we discuss the compression capability and reconstruction quality of our compression/decompression method. We also describe some aspects of its practical implementation. We consider two types of datasets. The first type is "toy problems," where we assume mathematical functions defined on an irregular two-dimensional geometry. These datasets are small, and we can compress and reconstruct them on one processor. The second type (reported in Sect. 6) is distributed larger datasets obtained from parallel simulations. In this case, we have both structured two-dimensional meshes where we can compare our CS results with other compression techniques such as WC and *zfp* and unstructured three-dimensional meshes which we compress only using the CS and WC methods.

4.1 Method Validation

We consider a square geometry with randomly chosen holes such that it constitutes an irregular geometry as shown in Fig. 4. We discretize it using an unstructured triangular mesh consisting of $N = 33{,}067$ points. We assume two functions f and g given in Eqs. 16 and 17 as the datasets represented on the obtained mesh. We choose these functions such that f exhibits multiple oscillations and reveals more features than g (see Fig. 4).

$$f = 48\sin(8\pi x)\sin(7\pi y)\sin(6\pi x) \tag{16}$$

$$g = 12\sin(2\pi x)[4\sin(2\pi x) - 4\sin(2\pi y)] \tag{17}$$

The motivation behind this choice is that we would like to explore the effect of data features on the number of samples required during compressed sensing, that are necessary to accurately reconstruct the original dataset. Representing f and g on such irregular geometry using traditional FGWs is problematic. One would consider interpolating the field from the unstructured mesh onto an assumed regular grid and then applying the usual CS and wavelet techniques for data compression. There are two major problems in this approach. First, the interpolation is an expensive procedure and might introduce a significant error. Second, it is not obvious how to treat the irregular parts of the geometry (e.g., the holes in Fig. 4) and what values to assume in their regions. Any assumed value will induce a discontinuity in the field, rendering the

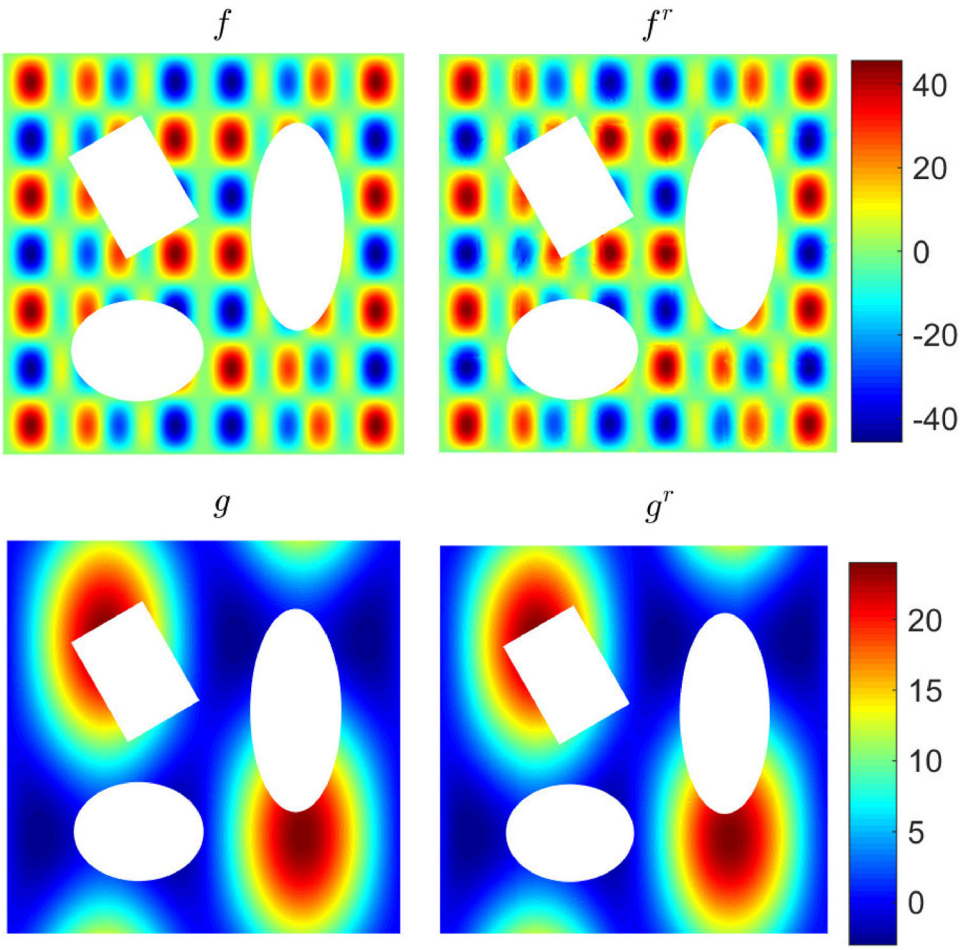

Fig. 4 Plots showing (*left column*) smooth datasets represented on a complex geometry and an unstructured mesh of 33,067 nodes, and (*right column*) their reconstructed version from compressed samples, at full wavelet detail. The *top* and *bottom rows* denote two different datasets given by Eqs. 16 and 17, respectively. The compression ratios are $R = 10$ and $R = 70$ for f and g, respectively. The bases for reconstruction are Alpert wavelets with an order $w = 5$

compression using CS or wavelets useless since a very low compression ratio will be necessary to accurately reconstruct the data.

Instead, we select an Alpert wavelet basis suitable for irregular geometries and meshes, compress the data $\boldsymbol{f} \in \mathbb{R}^N$ and $\boldsymbol{g} \in \mathbb{R}^N$ using a random Bernoulli matrix $[\boldsymbol{\Phi}] \in \mathbb{R}^{M \times N}$ as described in Sect. 2.1, and reconstruct f using the StOMP algorithm described in Sect. 2.3. Both compression and reconstruction are performed in a serial run. We denote the reconstructed fields by f^r and g^r, respectively. They are plotted in Fig. 4 which shows that the original and reconstructed datasets are visually indistinguishable. For the function f that reveals many features, a compression ratio of $R = 10$ was selected for an accurate reconstruction. However, g affords a higher compression ratio of $R = 70$ since it has fewer features which incur more data redundancy. This result indicates the possibility to predict the compression ratio in terms of the gradients in the data. This latter increases with the number of features in a given dataset. Such prediction of the compression ratio is beneficial in situ since the compressibility of the datasets is unknown a priori. We will explore this prediction in Sect. 5 below.

Despite the satisfactory visual aspect of the reconstruction shown in Fig. 4, we look at the spatial local error plotted in Fig. 5 in order to better understand its distribution. The plots suggest that the local error does not correlate with the trends and features in the data. The regions of large error (red color in the left column) indicate areas where noise might be visually noticeable. The reconstructed function f^r exhibits regions of larger error more than g^r since it contains more features as discussed previously.

4.1.1 *Effect of the Wavelet Order*

In order to obtain a quantitative description of the reconstruction accuracy, we evaluate the global NRMSE of difference between the original and reconstructed versions. Figure 6 shows this error for f and g as a function of the wavelet order for different compression ratios. As expected, the error is lower for low compression ratios.

The error decreases with the wavelet order w. The decreasing trend is reversed at larger values of w. We attribute this mainly to an overfitting of the function f by the high-order Alpert polynomial wavelets. It is therefore

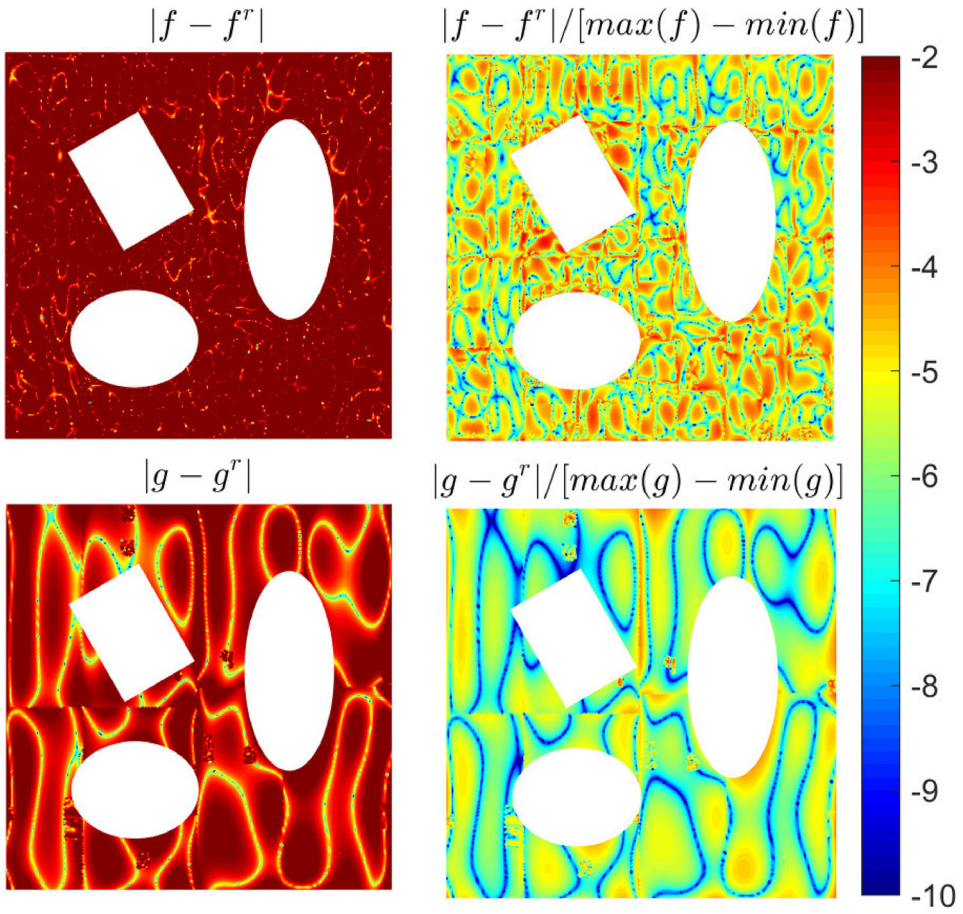

Fig. 5 Plots showing the local reconstruction errors for the fields f and g plotted in Fig. 4, as indicated. The *right column* shows the local errors in the *left column* normalized by the difference between the maximum and minimum values. All errors are plotted on a *logarithmic scale*. The compression ratios are $R = 10$ and $R = 70$ for f and g, respectively. The bases for reconstruction are Alpert wavelets with an order $w = 5$

preferred to choose lower orders. Empirically, and according to the plots in Fig. 6, we found that $w = 5$ is an optimum value for the wavelet order that minimizes the reconstruction NRMSE and prevents overfitting of the given function in the dataset.

4.1.2 *Effect of the Sampling Matrix*

Data compression using CS can be performed following Eq. 1 using different types of sampling matrices $[\Phi]$. For the two data fields f and g described previously in this section, we have performed the compression and reconstruction using various sampling matrices available in the literature [22]. We examine the data field f represented on the irregular geometry described previously in this section. We plot in Fig. 7 the reconstruction NRMSE (left vertical axis) and the coherence metric (right vertical axis) defined as [23]:

$$\mathcal{C} = \sqrt{N} \max_{ij} |[A]_{ij}| \tag{18}$$

where the matrix $[A]$ is product of the wavelet and sampling matrices (see Eq. 3). We notice that the coherence and the error are closely related to each other. The coherence metric measures the largest correlation between any wavelet-sampling vector pair. If it is large then the sampling technique is not encoding all information in the wavelet basis, which results in a larger reconstruction error.

Figure 7 implies that all the sampling matrices we considered except the Fourier matrix have satisfactory coherence requirements. In all other results shown in this paper, we have employed the Bernoulli sampling matrix since it resulted in the lowest error and due to its low storage requirements and simple construction method. Each row of a Bernoulli matrix is formed by N / 2 values of 1 and -1 randomly scrambled.

4.1.3 *Comparison with Other Reconstruction Algorithms*

We also compare our improved StOMP reconstruction with reconstructions obtained by other algorithms and codes that exist in the literature, namely ADMM [35], fixed point continuation (fpc) using Bregman iterations [36], CoSaMP [37] and Sparsa [38]. The results are reported for the f function in Fig. 8 which we consider as representative of typical PDE solution fields as discussed in Sect. 2.3.2. When StOMP is used, convergence is reached within 25 iterations and 7 s on a quad-core laptop operating at

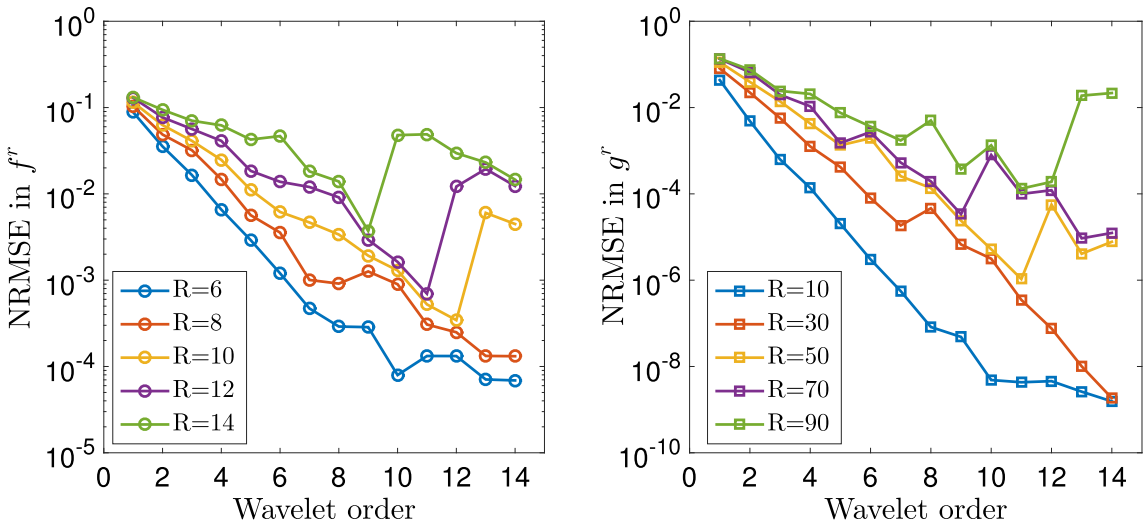

Fig. 6 Plots showing the NRMSE between the original and reconstructed datasets f^r (*left*) and g^r (*right*) plotted in Fig. 4 as a function of the wavelet order for different compression ratios, as indicated

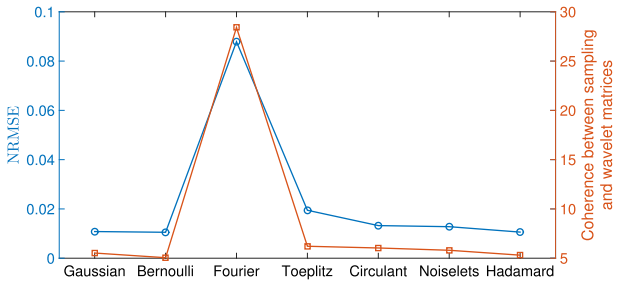

Fig. 7 Plots showing the reconstruction error (*left* y-axis, *blue line*) and the coherence metric (*right* y-axis, *red line*) for different sampling matrices $[\Phi]$. Results correspond to the StOMP reconstruction of the field f with a compression ratio $R = 10$. The wavelet bases for reconstruction are Alpert wavelets with an order $w = 5$

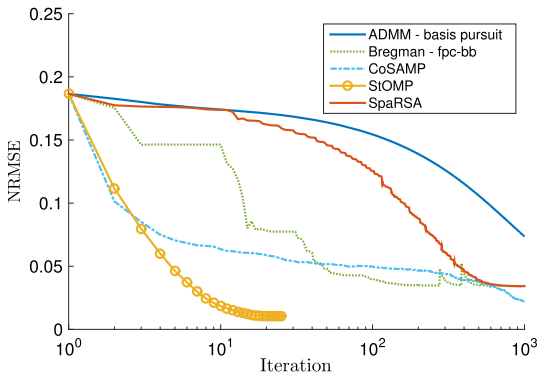

Fig. 8 Plots showing the decay of the normalized L_2 error (NRMSE) with iterations for different sparse reconstruction algorithms, as indicated. Results are reported for the function f plotted in Fig. 4 (*top row*) for $R = 10$ and $w = 5$

3.0 GHz, whereas all other methods would require more than 1000 iterations which takes more than 350 s. Furthermore, our improved StOMP algorithm does not require any tuning, unlike all other methods involving "knobs" that we manually tuned in our computations to give a reasonable reconstruction.

A reason behind the better performance we observe in the improved StOMP algorithm might be that the other methods are by nature not suitable for data represented on unstructured meshes and/or to cases using the Bernoulli-Alpert Matrix pairing for compressed sensing. It could be that the other methods would converge faster for images where more basic sampling techniques (e.g., Fourier sampling matrix) and more straightforward dictionaries are used. We do not perform comparisons for such cases for brevity and since they are outside the scope of this paper focusing on unstructured mesh data.

5 In Situ Prediction of the Compression Ratio

A major challenge in CS is its current limitation to fixed-rate compression preventing it from operating in a fixed accuracy mode. In other words, the compression ratio has to be chosen before performing an in situ compression and has to be the same for all computing processes in a parallel HPC application which correspond to the different subdomains in a distributed data field. If the data contains features expressed by large gradients in the field, the choice of the compression ratio might be low resulting in a poor reconstruction.

When lossy compression methods (e.g., compressive sensing, wavelet compression, etc.) are locally invoked during scientific computations on a multiprocessor cluster, it is beneficial, even crucial, to have an estimate of the compression ratio (R) that is suitable to the dataset

corresponding to the local computational subdomain in the cluster. We found in Sect. 4.1 that fields that exhibit many features and fluctuations require a relatively lower R to be accurately reconstructed. This also applies to realistic fields. For example, during a turbulent computational fluid dynamics (CFD) simulation with a uniform mesh, there could be regions where the flow field is smooth and other regions where turbulent fluctuations are present. The velocity field in the former case admits a relatively large R, whereas the latter requires a much lower R. The same disparity in the values of R can occur at different time steps, e.g., before and after an ignition or a turbulence onset. Thus, a constant R assigned to all processors might not usually be an optimal choice since it could result in inaccurate reconstructions in some regions in the domain and inefficient compression in others. The results of Sect. 4.1 imply that the optimal R is closely related to the field gradients. This implies that the compressibility of any scientific dataset can be predicted if a correlation is derived between R and the field gradients. This concept is similar to the data entropy often used in data workflow studies [39]. However, our concept is based on the redundancies in the visual data features and smoothness rather than the redundancies in the data precision and machine representation.

In this section, we develop a correlation that relates R to the gradient defined on each mesh point in the computational subdomain. We simplify the local variation in the gradient by computing its descriptive statistics, i.e., mean μ, standard deviation σ, median m and maximum $\mathcal{M}$. These statistics are normalized by the difference between the maximum and minimum value of the data and by the local mesh size in each element. We also consider the effect of the number N of data points on which the dataset is represented. N plays a role in assessing the data compressibility since a dataset can be represented on meshes of different refinements. In fact, fine (resp. coarse) meshes admit larger (resp. lower) R. According to these observations, one can devise different mathematical relationships between R and the gradient statistics. In this study, we *assume* the following correlation relating R to these descriptive statistics:

$$R = \eta_0 \mathcal{M}^{\eta_1} \sigma^{\eta_2} \mu^{\eta_3} |\mu - m|^{\eta_4} N^{\eta_5} \tag{19}$$

This assumed correlation of the exponential dependence of R on the gradient statistics is inspired by the trends observed in the plots shown in Fig. 9 below and generated from the data described in Sect. 5.1. This correlation accounts for all aspects of the gradient distribution in the dataset. It also accounts for its skewness in terms of the deviation between the gradient mean and median $|\mu - m|$. The parameters $\eta_{0 \le i \le 5}$ are to be inferred from training values of N, descriptive statistics and $\mathcal{R}$, the "true" R. Here $\mathcal{R}$ is obtained at a given data reconstruction accuracy ϵ defined as the NRMSE. An initial large value of R is selected. It is used to compress and then reconstruct the dataset. If the reconstruction error is larger than ϵ, R is decreased. This procedure is repeated in a systematic manner until the reconstruction error meets ϵ where the $\mathcal{R}$ is recorded. According to the trends observed in Fig. 9 and the dependence on N described above, we expect to obtain negative values $\eta_{1 \le i \le 4}$ and a positive value of η_5.

5.1 Training Data

We have obtained two datasets distributed among many processors. Using Alpert tree wavelets with an order $w = 5$, each of these subdatasets produces, for a reconstruction accuracy $\epsilon = 0.05$, a set of $\mathcal{R}$, N and the gradients' statistics. Based on these, we infer the values of η_i in Eq. 19.

5.1.1 Turbulent Combustion Data

We have obtained turbulent combustion data from a reactivity-controlled compression ignition (RCCI) simulation [40, 41] involving several time steps. Each time step consists of 116 variables encompassing the chemical species present during the combustion and other primitive variables such as temperature, velocities, etc. Each of these variables is defined on a large two-dimensional regular grid. We have subdivided this large grid into different subdomains and in different ways such that each subdomain could include $1600 \le N \le 25{,}600$ data points. Thus, we generated several subdomains from this spatial subdivision and from the different variables and time steps. We randomly selected subdomains for the training of the correlation 19 and others for its verification/validation.

5.1.2 Turbulent Flow Data

We have also obtained data from the simulation of three-dimensional high-speed internal flow in a flat channel [42]. The data involves several time step, each consisting of 8 primitive variables. The mesh is irregular since it is refined in some areas close to the walls in order to better capture the turbulent structures and features arising in these locations. For each variable and time step, the data are distributed among 40 processors resulting in different subdomains. Similarly to the turbulent combustion data, this allows us to randomly generate several samples to train and verify the correlation 19.

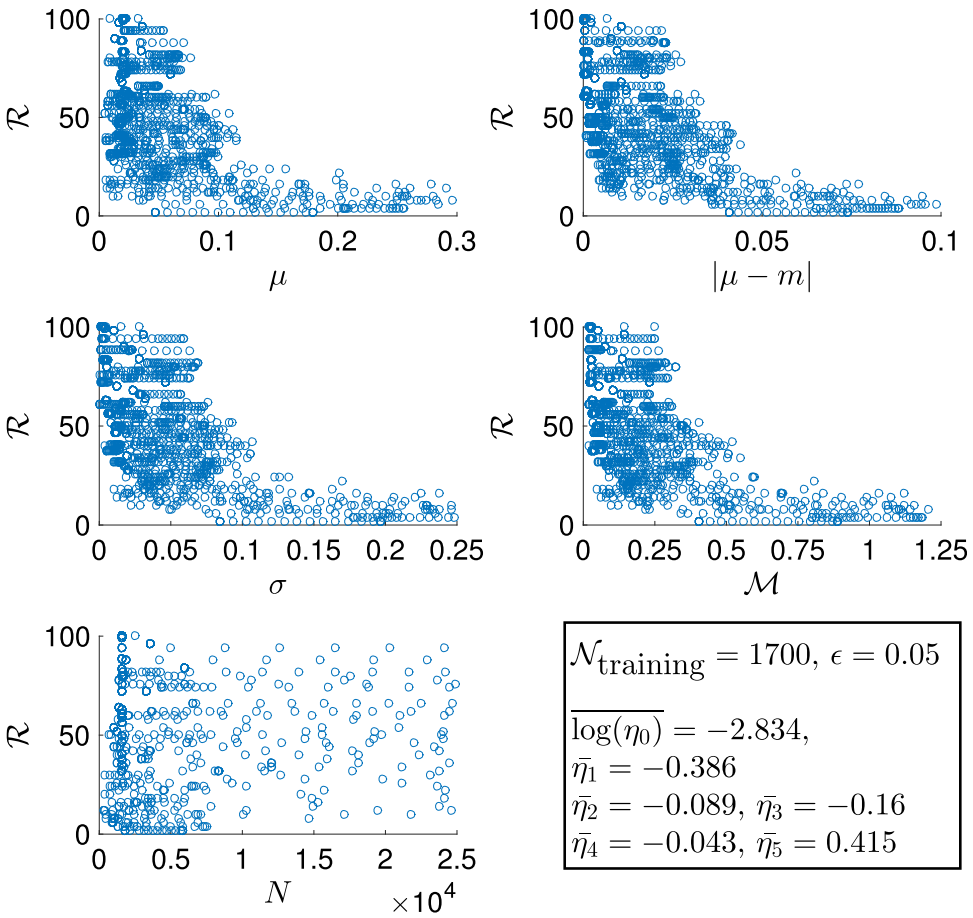

Fig. 9 Plots showing the "true" training compression ratio $\mathcal{R}$ as a function of the gradients statistics. Results are obtained from $\mathcal{N} = 1700$ training samples and $\mathcal{R}$ is computed for a NRMSE $\epsilon = 0.05$

5.2 Inference of the Correlation

Each dataset is constituted of different field variables, distributed among different processors and computed at many time steps. This results in a large number of $\mathcal{R}$ and gradient statistics sets. We divided these sets into two groups. One to infer the correlation and the other to validate and test it. We performed operations to minimize the inference set and eliminate redundancies and data clustering [43]. These operations are necessary for numerical stability as further described below. The effective amount of inference training data is $\mathcal{N} = 1700$. This results in the data plotted in Fig. 9 where $2 \leq \mathcal{R} \leq 100$ and an equal number of the descriptive statistics (μ, σ, etc.) is obtained for each value of $\mathcal{R}$. This latter exhibits a decreasing trend with the gradient statistics as expected.

We use a Bayesian regression method to perform the inference similar to the one used in [44]. By applying the "log" function to both sides of 19, the problem reduces to a linear multivariate inference problem which admits an analytical solution. For a each subdomain j and corresponding data extracted from the training data, we can write:

$$\begin{aligned}\log(\mathcal{R}_j) = \{1, \log(\mathcal{M}_j), \log(\sigma_j),\\ \log(\mu_j), \log(|\mu_j - m_j|), \log(N_j)\} \cdot \boldsymbol{\eta}\end{aligned} \tag{20}$$

where $\boldsymbol{\eta} = \{\log(\eta_0), \eta_1, \eta_2, \eta_3, \eta_4, \eta_5\}$. For all the $\mathcal{N}$ training data, we can write Eq. 20 in matrix–vector form:

$$\boldsymbol{d} = [Q] \cdot \boldsymbol{\eta} \tag{21}$$

where $\boldsymbol{d} = \{\log(\mathcal{R}_j)\}_{j=1}^{\mathcal{N}}$ and $[Q] \in \mathbb{R}^{\mathcal{N} \times 6}$ is a matrix containing the gradient statistics information. We derive the vector $\boldsymbol{\eta}$ in the form:

$$\boldsymbol{\eta} = \bar{\boldsymbol{\eta}} + [L]\boldsymbol{\xi} \tag{22}$$

where $\bar{\boldsymbol{\eta}}$ contains the nominal values of the coefficients in the correlation (see Fig. 9), $\boldsymbol{\xi}$ is a vector of random numbers drawn from a unit normal distribution, i.e. $\boldsymbol{\xi} \sim \mathcal{N}(0, 1)$ and $[L]$ is a lower-triangular matrix that accounts for the uncertainty in $\boldsymbol{\eta}$ due to limited amount of training data. Practically, one would using the nominal result $\bar{\boldsymbol{\eta}}$ to estimate the compression ratio. As mentioned previously, $\bar{\boldsymbol{\eta}}$ admits an analytical expression:

$$\bar{\boldsymbol{\eta}} = \left([Q]^T[Q]\right)^{-1}[Q]^T\boldsymbol{d} \tag{23}$$

Inverting the matrix in Eq. 23 requires the absence of data point clusters and thus the necessity of the data minimization and uniformity as previously described in this section. For $\mathcal{N} = 1700$, we derived the expression of the compression ratio as:

$$R = 0.059\,\mathcal{M}^{-0.386}\sigma^{-0.089}\mu^{-0.16}|\mu - m|^{-0.043}N^{0.415} \qquad (24)$$

where the numbers in the exponentials are the entities in the derived vector $\bar{\boldsymbol{\eta}}$. $\eta_{1\leq i\leq 4}$ are negative and η_5 is positive, as expected. We use Eq. 24 to predict the compression ratio R of a validation dataset different than the one given by $\boldsymbol{d}$ and $[Q]$. We plot the "true" $\mathcal{R}$ versus the predicted R in Fig. 10 which shows a good agreement between the measured and predicted values. The scattering in the results is mainly due to the fixed choice of the wavelet order $w = 5$. Such scattering and fluctuations due to the wavelet order have been observed in the preliminary results (see Fig. 6). Figure 10 also indicates that the compression ratio is slightly overpredicted at its lower range ($R < 20$). This is most probably due to the inadequacy of the assumed model in Eq. 19 for low compression ratios, i.e., large gradients in the data. Other correlations similar to Eq. 24 can also be inferred for accuracy metrics other than the NRMSE, e.g., the error metrics of Wang et al. [45].

Remark It is sufficient to compute an approximation of the gradient fields in order to compute the statistics needed to predict R. Approximate gradients are often computed on the fly in large-scale simulations for post-processing purposes. In such situations, computing the optimal compression ratio using Eq. 24 does not require any additional overhead on the simulation. In other situations, computing the local approximate field gradient all over the mesh might be an expensive additional necessary step. However, in order to compute the gradients statistics, it is sufficient to approximate the field gradient on a sample ($\sim 10\%$) of the mesh points according to the sublinear sampling theory [18, 46].

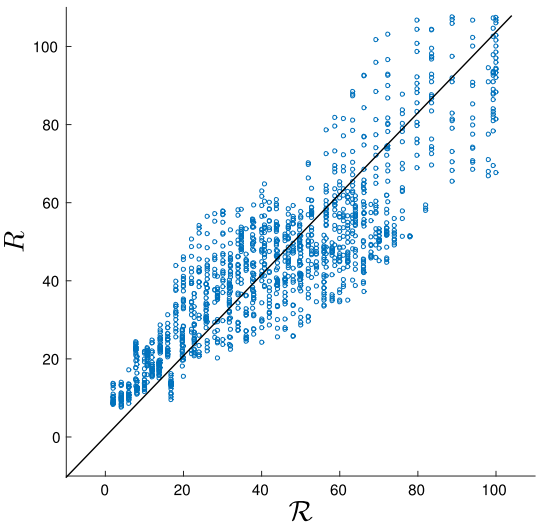

Fig. 10 Plot showing the compression ratio R predicted using Eq. 24 versus the "true" compression ratio $\mathcal{R}$. The data are obtained from a set of validation samples

6 Large-Scale Case Studies

In this section, we consider large datasets distributed among many processors and obtained from CFD and finite simulations ran on HPC clusters. We first consider data represented on a regular grid such that we can compare the results of our CS compression method with direct wavelet compression (WC) using Alpert bases described in Sect. 2.2.1 and with *zfp* [4], a recently developed compressor designed for structured data, in terms of accuracy and compression and decompression speed. We then consider datasets represented on an irregular geometry discretized using an unstructured three-dimensional mesh. Here, we show the effectiveness of our CS compression method with regards to its new feature that motivated our study that is the ability of CS to be applicable on unstructured mesh data.

6.1 Two-Dimensional Distributed Datasets

In this section, we consider larger two-dimensional datasets from a reactivity-controlled compression ignition (RCCI) combustion simulation called S3D [47] represented on a square geometry. These results are taken from few time steps over the course of the simulation run. The simulation involves a combustion ignition moment throughout its course. Before the ignition, the data fields are smooth with moderate spatial variations throughout the domain. After the ignition, the data fields exhibit large and sharp gradients revealing combustion flames, i.e., reaction fronts. Images of the data are shown in Figs. 11 and 13 below (left panel) for cases before and after ignition, respectively. Before the ignition, the data consist of 1600^2 mesh points distributed among 100 processes whereas after the ignition, the data consist of 3200^2 mesh points distributed among 400 processes.

We perform the compression and reconstruction locally in each process as described in Sect. 3. We consider two cases, where the compression ratio is fixed and assumed the same across all processes and where the CR is estimated locally in each process using the correlation described in Sect. 5. In the latter case, we define the average compression $\bar{R}$ as the mean value of the estimated values of the local CRs.

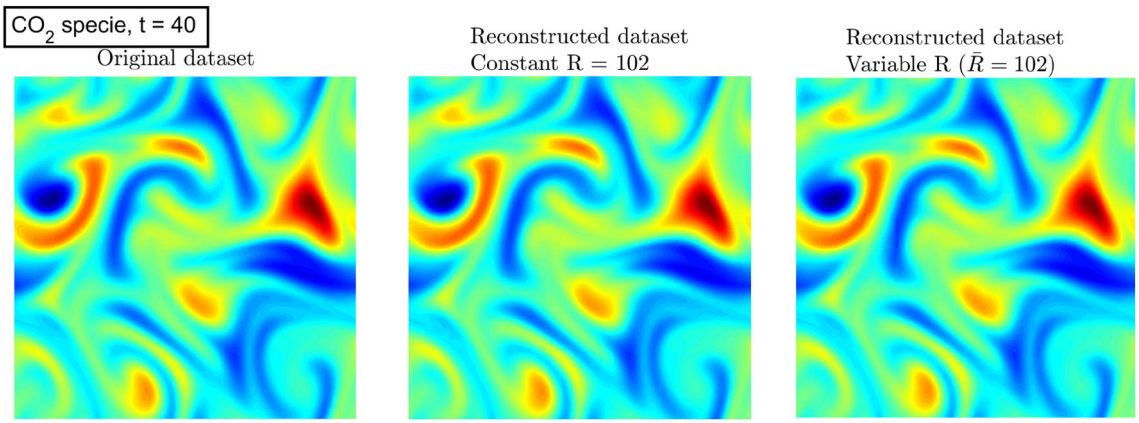

Fig. 11 Plots showing (*left*) the original carbon dioxide CO_2 field; the field is represented on 1600×1600 regular grid distributed on 10×10 processes, (*middle*) the reconstructed version using a constant compression ratio R throughout its processes and (*right*) the reconstructed version using a compression ratio R optimally computed and variable throughout its processes. Results are reported from a RCCI simulation at $t = 40$ compressed using CS and reconstructed using StOMP with a wavelet order $w = 5$

Figure 11 shows the carbon dioxide (CO_2) field at time step $t = 40$ prior to the ignition. The field does not exhibit any large variation. For a CR of about two orders of magnitude, the reconstructed data are visually indistinguishable from the original data in both constant and estimated CRs. This indicates that the in situ optimal estimation of the CR does not bring significant improvements to the reconstruction accuracy. This is not surprising since the data field is pretty smooth with spatial gradient magnitudes of the same order throughout the domain.

In addition to the visual aspect of the reconstruction, we quantify the error resulting from the lossy compression. We first investigate the local relative error between the original and reconstructed data. The simulation variables we investigated are nonzero entities; therefore, computing the relative error is a safe exercise from a numerical point of view. We present the local relative error in these results in terms of their probability density function (PDF) distribution as shown in Fig. 12 for the CO_2 field at different time steps. All PDF peaks occur at a relative error lower that 0.05%. We quantify the width of the distribution in terms of its full width at half maximum (FWHM) as shown in the figure for $t = 40$. After the ignition ($t = 72$ and 90), the PDF peaks occur at a lower error with lower FWHM which is not surprising due to the increased number of grid points in the domain and the relatively small region of the flame fronts of large gradients in the domain.

More details on the relative distributions are given in Tables 1 and 2 for the CO_2 and velocity fields, respectively. We notice that the velocity allows for larger CRs especially before the ignition which is due to its increased smoothness. In addition to the FWHM, these tables show the maximum relative error computed at the 95th percentile of the distribution in order to exclude the outliers, i.e., the very rare large spikes that might occur in the reconstructed

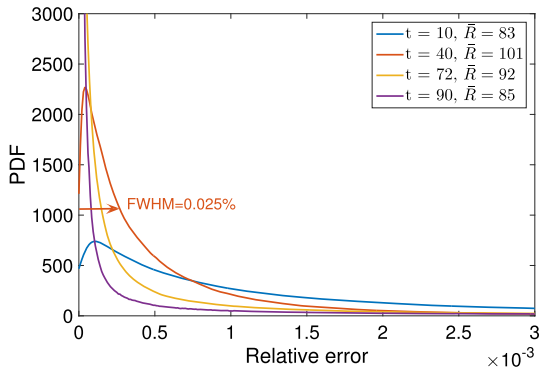

Fig. 12 Plots showing the probability density functions (PDF) of the relative error between the original CO_2 concentration field and its constructed value using StOMP. The CO_2 concentration field is represented on a 1600×1600 regular grid distributed on 10×10 processes for $t = 10$ and $t = 40$ (resp. 3200×3200 regular grid distributed on 20×20 processes for $t = 72$ and $t = 90$). Results are reported for a local StOMP reconstruction with a wavelet order $w = 5$ where the compression ratios were optimally estimated. The plot shows how the FWHM is computed, and the rest of these values are reported in Table 1

data. We notice that after the ignition, there is an increase in the maximum relative error due to the existence of the large gradients in the data. These can be seen in Fig. 13. However, the ranges of the maximum relative error in the field data fall in acceptable ranges as identified by a domain expert.

In combustion simulation, it is common to compute the field gradients for other data analysis tasks, e.g., computing the stress tensor. We investigate the effect of the lossy compression on the CO_2 and velocity gradients in terms of their FWHM and maximum values as shown in the right part of Tables 1 and 2. The maximum errors in the gradients using CS are significantly increased. This is due to the

Table 1 Compression ratio R, NRMS error and relative error statistics in the reconstructed CO_2 species concentration field and its gradient for different time steps of RCCI simulation

	$\bar{R}$	NRMS error	CO_2 field relative error		CO_2 field gradient relative error	
			FWHM (%)	Max (%)	FWHM (%)	Max (%)
$t = 10$	83	0.0032	0.069	0.81	10 [0.25]	700 [3.02]
$t = 40$	101	0.0016	0.025	0.23	7.4 [0.25]	171 [2.19]
$t = 72$	92	0.0072	0.0045	5.5	0.95 [0.05]	1900 [5.85]
$t = 90$	85	0.0081	0.004	4.87	1.1 [0.18]	2500 [4.72]

The CO_2 concentration field is represented on a 1600×1600 regular grid distributed on 10×10 processes for $t = 10$ and $t = 40$ (resp. 3200×3200 regular grid distributed on 20×20 processes for $t = 72$ and $t = 90$). Results are reported for a local StOMP reconstruction with a wavelet order $w = 5$ where the compression ratios were optimally estimated. The numbers in square brackets indicated values obtained from direct wavelet reconstruction. The maximum value (Max) is computed at the 95th percentile of the error distribution to remove the effect of the outliers

Table 2 Compression ratio R, NRMS error and relative error statistics in the reconstructed velocity magnitude U field and its gradient for different time steps of RCCI simulation

	$\bar{R}$	NRMS error	U field relative error		U field gradient relative error	
			FWHM (%)	Max (%)	FWHM (%)	Max (%)
$t = 10$	131	0.0024	0.006	1.84	0.6 [0.05]	839 [2.08]
$t = 40$	130	0.0025	0.02	1.01	1.6 [0.05]	741 [1.46]
$t = 72$	93	0.0052	0.0085	3.34	1 [0.15]	2176 [9.52]
$t = 90$	86	0.0061	0.043	3.75	4.5 [0.15]	2224 [13]

The U field is represented on a 1600×1600 regular grid distributed on 10×10 processes for $t = 10$ and $t = 40$ (resp. 3200×3200 regular grid distributed on 20×20 processes for $t = 72$ and $t = 90$). Results are reported for a local StOMP reconstruction with a wavelet order $w = 5$ where the compression ratios were optimally estimated. The numbers in square brackets indicated values obtained from direct wavelet reconstruction. The maximum value (Max) is computed at the 95th percentile of the error distribution to remove the effect of the outliers

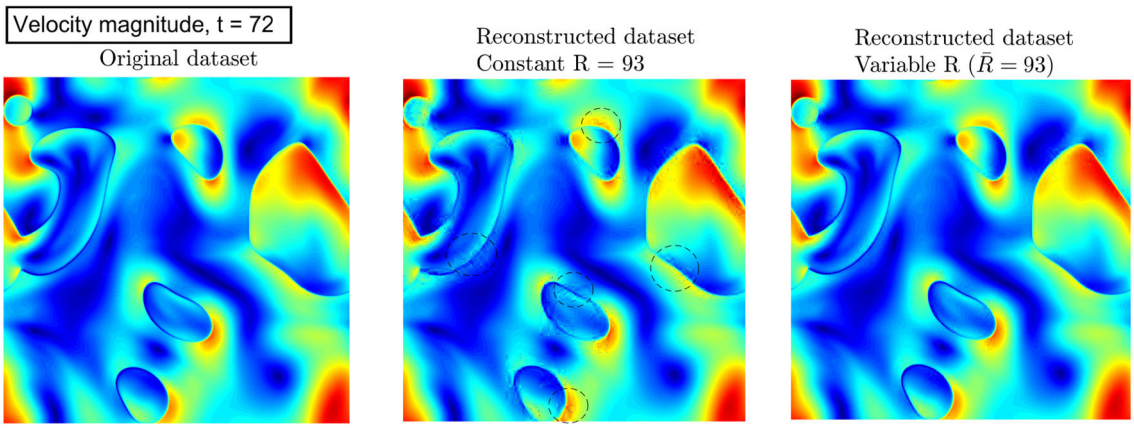

Fig. 13 Plots showing (*left*) the original velocity magnitude U field; the field is represented on 3200×3200 regular grid distributed on 20×20 processes, (*middle*) its reconstructed version using a constant compression ratio R throughout its processes and (*right*) its reconstructed version using a compression ratio R optimally computed and variable throughout its processes. Results are reported from a RCCI simulation at $t = 72$ compressed using CS and reconstructed using StOMP with a wavelet order $w = 5$

noise in the reconstructed data resulting from the stochastic nature of CS. However, the maximum errors obtained using direct WC using Alpert bases (shown in square brackets) are still within acceptable ranges as identified by a domain expert. This shows that if the gradients are required in the data analysis stage, CS should not be used as the compression method whereas direct WC is still a viable lossy data compression method.

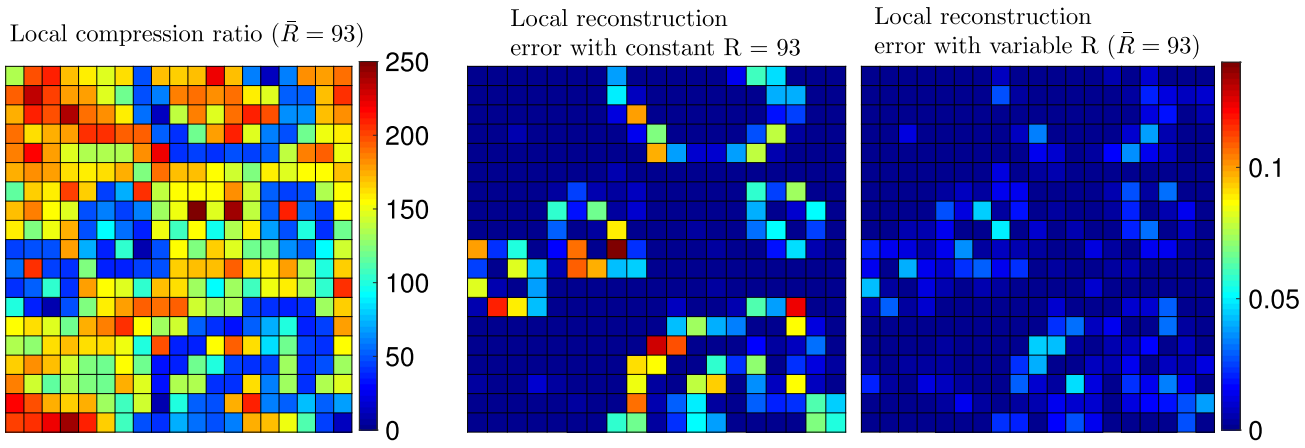

Fig. 14 Plots showing (*left*) the local optimal compression ratio R that is used to reconstruct the velocity magnitude U field shown in Fig. 13, (*middle*) the local NRMS error obtained when the same compression ratio is used all over the processes, and (*right*) the local NRMS error obtained when an optimal compression ratio is used throughout the processes

6.1.1 Impact of Optimized CS Compression

The results presented so far in this section are reported using compression ratios estimated for optimal accuracy. In this part, we show the importance of such estimation in the turbulent combustion data. Figure 13 shows the velocity field after the ignition at $t = 72$ along with its reconstructed version using a constant and estimated CR. When using a constant CR, the reconstructed data exhibits discrepancies and anomalies in the velocity field that are visually noticeable around the flame fronts where sudden variations in the velocity occur. These anomalies are almost fully eliminated when the CR is optimally estimated.

The estimated CRs used to generated the right-hand side plot of Fig. 13 are reported in Fig. 14 (left) as a spatial color plot. We can clearly see how that in the subdomains corresponding to the flame front region the estimated CR is significantly lower than other regions due to the large gradients. Since this estimation of the compression estimation occurs in situ during a simulation, it serves as an a priori prediction of the flame fronts as interesting features in the simulation data. In other words, Eq. 24 can be applied during a simulation for in situ feature detection which is beneficial during scientific workflows [46] which can feedback into the simulation triggering its control parameters, e.g., mesh size, data input/output frequency, etc.

Figure 14 also shows the local NRMSE in each subdomains. The plots illustrate the increase in this error in the areas of large gradients when a fixed CR is used and how these errors are significantly reduced when it is optimally estimated.

Finally, we compare the NRMSE resulting from the lossy compression using different methods. We consider CS, direct Alpert WC and the fixed-rate *zfp* compressor [4]

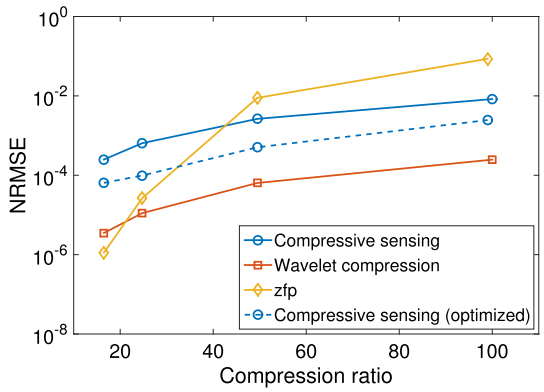

Fig. 15 Plots showing the global NRMSE between the original and reconstructed data fields using different scientific data compression methods as a function of the compression ratio. The solid lines indicate results obtained with a constant compression ratio throughout the subdomains, while the solid line indicate results for CS using the optimal compression ratio obtained by Eq. 24. Results are reported for all 120 field variables of a RCCI simulation at $t = 72$ where $N = 25{,}600$. A wavelet order $w = 5$ was used in compressed sensing and wavelet compression

and apply them to all the 120 field variables present in the simulation at a moment after the ignition $t = 72$. Results for other time steps do not differ significantly.

Direct Alpert WC works in the following manner. Rather than subsampling and reconstructing in the wavelet domain as is the case of CS, we transform the field directly and then zero out all of the transformed coefficients that fall below the threshold as decided by the compression rate, e.g., a compression ratio $R = 8$ means that the lowest 90% of coefficients in magnitude are zeroed out and only the largest $\frac{1}{10}$ are used to reconstruct the original field. The indices of the largest $\frac{1}{10}$ are also stored in order to reconstruct the data using an inverse wavelet transform. The second method we compare against is *zfp* [4]. It works by

compressing 3D blocks of floating-point data into a variable-length bitstream sorted in order of error magnitude. Compression occurs by dropping the low-order end of the bitstream. We have implemented its fixed-rate compression method in this paper, i.e., by specifying a constant compression rate as the input to its software code.

The errors are plotted in Fig. 15 as a function of the CR considered to be fixed along the subdomains for all three methods (solid lines). While *zfp* has an excellent accuracy at low CRs, it is not able to achieve good reconstruction accuracy after about 40× compression since it is not performing any adaptive compression in our case. We see that the best case for accuracy is the Alpert wavelet compression for all compression range. The accuracy of CS falls in between *zfp* and WC and remains in acceptable ranges even at $R = 100$ as shown in previous results. Figure 15 also shows that the accuracy of CS is improved by about 4.5 times when the optimally estimated CR is used.

6.1.2 Compression and Decompression Performance

Finally, we report the compression and decompression speeds of CS and compare them with those of WC and *zfp*. We have performed the tests for one time step of the RCCI combustion data for all 120 variables in the data involving $N = 25{,}600$ mesh points per process. This means the CS and WC were cast as matrix–matrix products while *zfp* was applied to each variable separately in a sequential manner. The experiments were performed on a Dell Laptop with 4 cores operating at 3.0 GHz. CS and WC were implemented in MATLAB 2015a through the open-source library SWinzip v1.0 [34] while *zfp* was compiled from its open-source code available online [48]. The speeds are plotted in Fig. 16 as a function of the CR assumed to be constant throughout the domain.

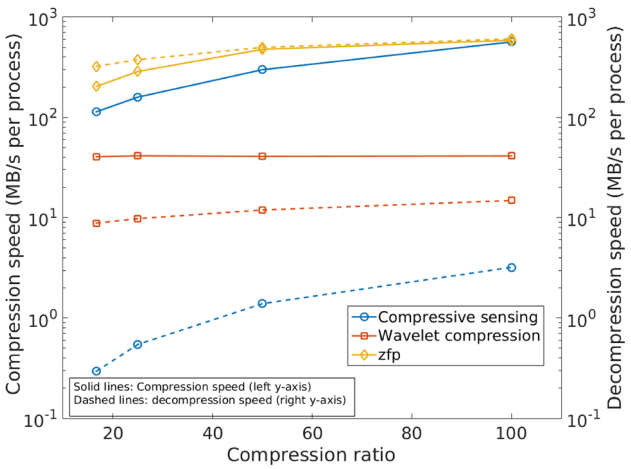

Fig. 16 Plots showing the compression and decompression speeds of different scientific data compression methods as a function of the compression ratio. A constant compression ratio was used throughout the subdomains. Results are reported from a RCCI simulation at $t = 72$. Results are reported from a RCCI simulation at $t = 72$ where $N = 25{,}600$. A wavelet order $w = 5$ was used in compressed sensing and wavelet compression

CS involves the pre-computation of the three matrices $[\Phi]$, $[\Psi]$ and $[A]$ in Eqs. 1–3, WC only involves the matrix $[\Psi]$, while *zfp* does not involve any matrices. In our experiments, we did not account for the time required to build these matrices since we assumed that it amortizes with the large number of variables and time steps present in the data.

We find that CS is the slowest in decompression which is expected since it involves a lengthy iterative sparse solution of an undertermined system (StOMP). On the other hand, *zfp* is the fastest in both compression and decompression. However, there is a reasonable trade-off between the speed and the increased error incurred using *zfp* at large compression ratios. CS comes in the second place in compression speed in our case. CS benefits from the MATLAB linear algebra operations that use the multicore laptop CPU architecture. In the absence of any multicore architecture, both CS compression and decompression speeds would decrease by about a factor of four if only one core is used during the computations. The speeds of WC fall within an intermediate range where decompression speed is significantly lower than the compression speed due to the less contiguous memory access in the irregularly spaced matrix row indices during the inverse wavelet transform. There is also a trade-off in WC considering its higher accuracy compared to CS and *zfp* as shown in Fig. 15.

6.1.3 Performance Prediction

When comparing the performance of different compression methods, it is difficult to devise a comparison strategy between different compression methods that allows easy predictions since lossy compression performance and accuracy are in general highly data dependent. We have reported our results for one of the case studies and one data size where the number of points per computing process is $N = 25{,}600$. Among the PDE simulation data we surveyed, we found that this is an average typical number of mesh points per process in finite elements and computational fluid dynamics that shows a balance between computing speed and memory use. Nevertheless, one might still want to predict how compression and decompression speeds might vary with N. Compression speed in CS is related to R/N since it corresponds to the one of a dense matrix–vector product similar to Eq. 1. However, when N varies, the data compressibility varies as well for a given reconstruction accuracy. As suggested in Sect. 5, R usually increases with N due to more redundancies in the data.

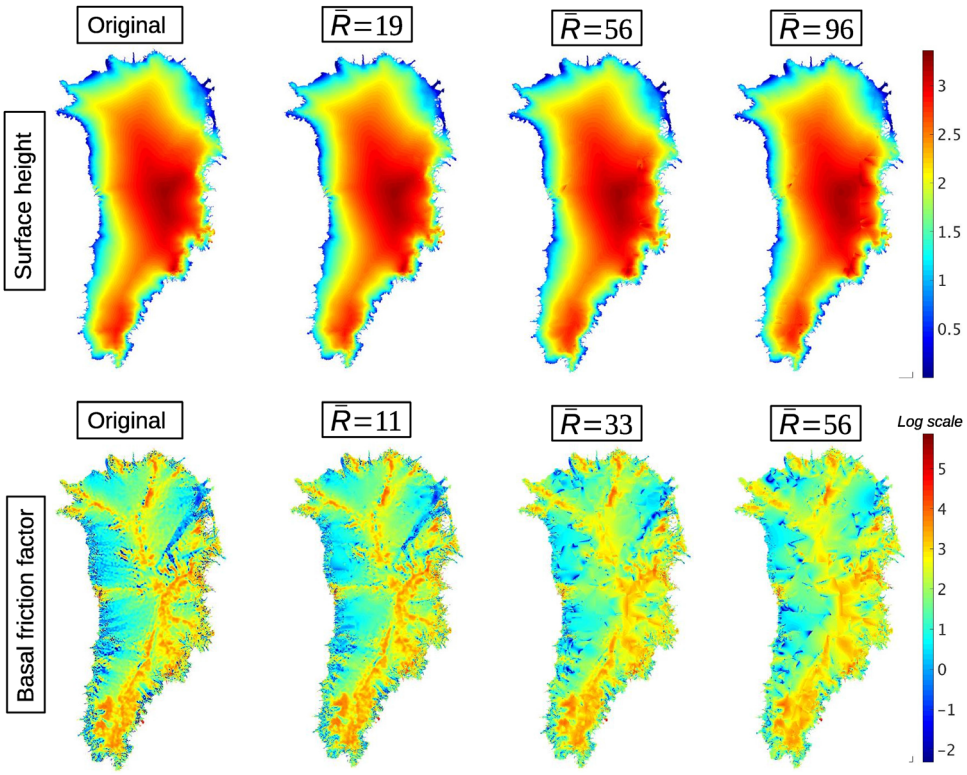

Fig. 17 Plots showing the original and reconstructed height surface (h) and basal friction (β) fields obtained from a Greenland ice sheet finite element simulation, at different compression ratios, as indicated. The fields are represented on an unstructured mesh with 2.8 million data points distributed on 64 processes. Results are reported for a local StOMP reconstruction from compressed samples with a wavelet order $w = 3$ where the compression ratios were optimally estimated

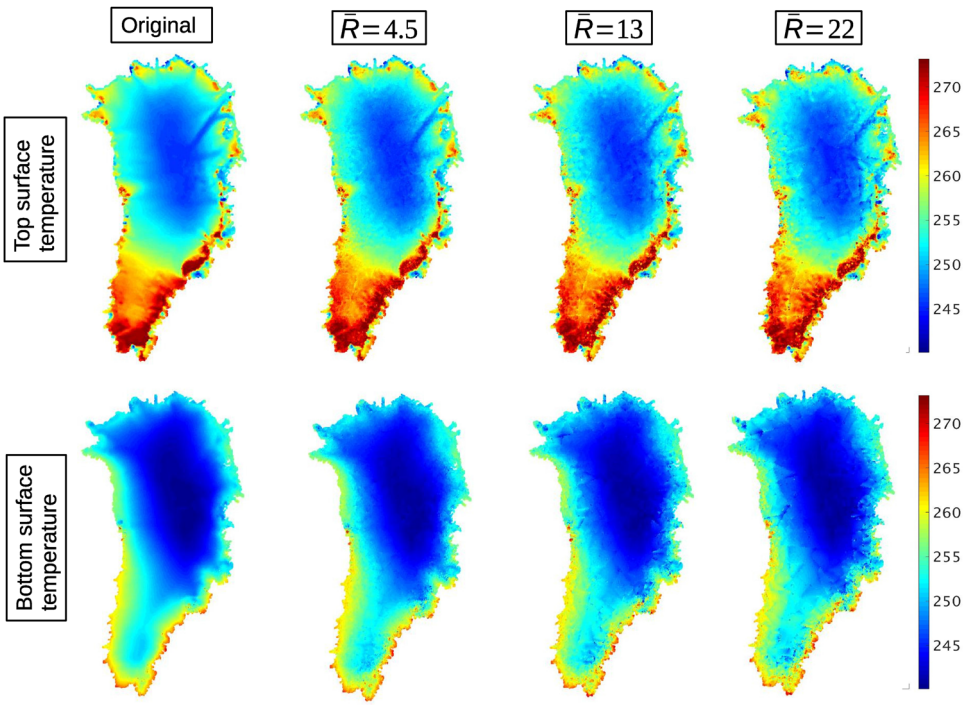

Fig. 18 Plots showing the original and reconstructed top and bottom ice temperature fields obtained from a Greenland ice sheet finite element simulation, at different compression ratios, as indicated. The fields are represented on an unstructured mesh with 2.8 million data points distributed on 64 processes. Results are reported for a local StOMP reconstruction from compressed samples with a wavelet order $w = 3$ where the compression ratios were optimally estimated

Accordingly, we do not expect that the CS compression speed would significantly vary with N unless reconstruction error requirements are relaxed. A similar conclusion can be drawn for the CS decompression noting that the StOMP algorithm converges faster in smoother fields which might increase the speed.

In WC, the compression ratio R does not play a major role in assessing the compression and decompression speeds since the major time is spent on the performing the sparse matrix–vector product in Eq. 2. The cost of this product is related to $N\log(N)$ [19, 28]. Thus, the speed of

WC is related to $1/\log(N)$ which signifies that it does not significantly vary with N.

The compression and decompression speeds of *zfp* increase in larger 3D data fields by virtue of the memory caching benefits [4] incurred in such cases. Increasing the compression in *zfp* might also increase the error in 3D data unless adaptive schemes are used.

6.2 Three-Dimensional Distributed Datasets

In this section, we test the accuracy of the CS compression on a tetrahedral unstructured mesh dataset based on an irregular geometry. We chose data obtained from a simulation of the Greenland ice sheet using the finite element land-ice solver known as Albany/FELIX [49] and applied our MATLAB-based algorithms of SWinzip v1.0 to perform the compression. The Greenland geometry is shown in Figs. 17 and 18 below. This geometry is discretized using an unstructured tetrahedral mesh with approximately 2.8 million nodes distributed among 64 processes. It has a large aspect ratio with respect to its thickness, with a length-to-thickness ratio of about 1000, yet its underlying mesh is still three dimensional. We report our results on three data fields: the surface height (h), the basal friction coefficient (β) and the ice temperature. The fields h and β are two-dimensional entities extruded in the thickness direction. We note that in this application, CS algorithms such as those described herein can also enable uncertainty quantification in land-ice modeling by providing a way to create a relatively low-dimensional representations of these (and other) high-dimensional fields.

We first explore the visual aspect of reconstructed results and then look at the error metrics. Figure 17 shows the original and reconstructed h and β fields at different CRs optimally estimated. h is a smooth field; thus, it affords large CRs up to 60 with a satisfactory visual quality in the reconstructed data. On the other hand, β is a more complicated field with values spanning different scales (the color bar is given on log scale) and sharp gradients. It is expected that it cannot handle large CRs. In fact, with $\bar{R} > 11$, many visual details are lost. However, the features depicting the large values of the β are preserved even at large CRs.

Figure 18 shows the original and reconstructed ice temperature fields at both sides of the Greenland geometry at different CRs optimally estimated. The achieved CRs are not large compared to the other cases considered in this paper. This is due to two main reasons. First, the top surface ice temperature is a complicated field with frequent large variations in the ice temperature values. Secondly and more importantly, the temperature varies sharply from the top to the bottom surface over the very small thickness of the Greenland island compared to its length. This disparity in the length scales hinders the CS compression quality.

In addition to the visual aspect of the results, we have computed the NRMSE for the h, β and ice temperature fields as a function of the average compression ratio. These plots are shown in Fig. 19. The NRMSE increase with the average CR at different rates. The increase rate corresponding to the ice temperature field is larger than the one of the β, which in turn is larger the NRMSE increase rate corresponding to the h. This result indicates that the compression quality is highly dependent on the data in hand. The data spatial smoothness plays a crucial role in assessing the compression amount and quality. Figure 19 also shows the NRMSE obtained with direct WC that is found to be about 4 times smaller than the one incurred by CS but with the same increasing trends as a function of the CR. WC has less computational steps and complications than CS which participate in enhancing its accuracy. We cannot compare the CS accuracy with *zfp* since the data in this case involves an unstructured mesh.

Finally, we report the distribution of the local relative error between the original and reconstructed data as shown in Fig. 20. Here again, the fields we consider are nonzero entities thus we are consider that our results are trustworthy. The distribution peaks increase with the CR for all fields as expected. However, the trends here are different than the ones observed in the NRMSE. While the ice temperature field results in the largest NRMSE (see Fig. 19), it exhibits the lowest relative error. This suggests that the choice of error metric plays a crucial role in assessing the capabilities of any lossy compression technique in terms of accuracy. In fact, the study of Wang et.al.

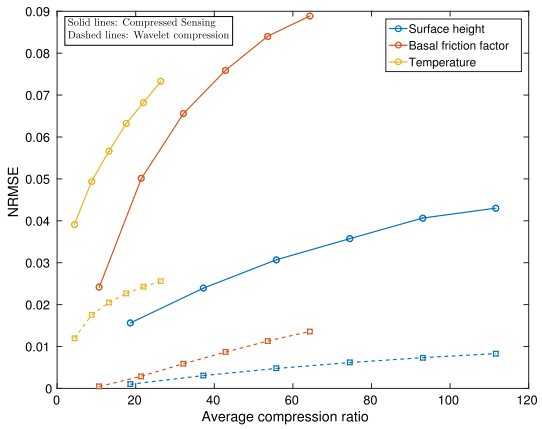

Fig. 19 Plots showing the variation of the global NRMSE between the original and decompressed indicated data fields as a function of the average compression ratio. All the fields are obtained from a Greenland ice sheet finite element simulation and are represented on an unstructured mesh with 2.8 million data points distributed on 64 processes. Results are reported for a local StOMP reconstruction from compressed samples and direct wavelet decompression with a wavelet order $w = 3$ where the compression ratios were optimally estimated

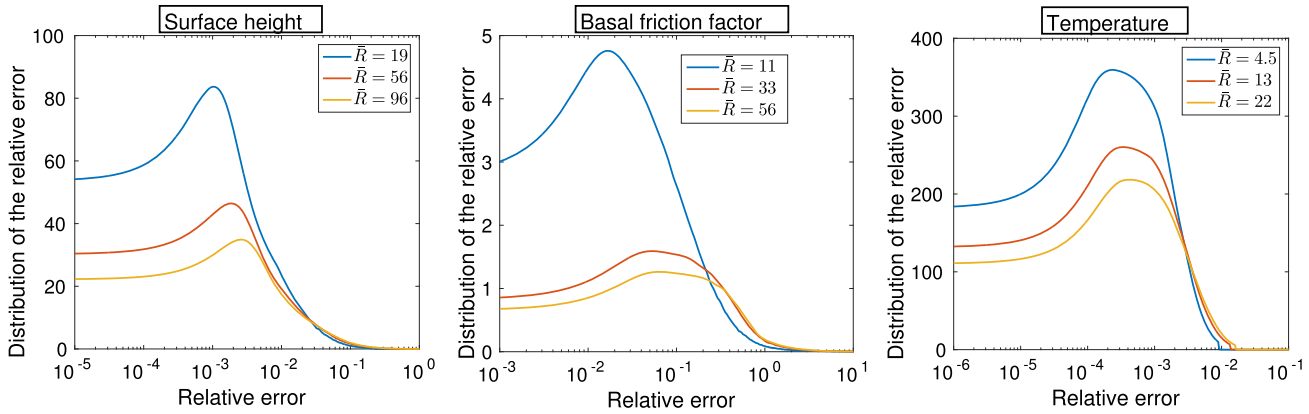

Fig. 20 Plots showing the probability density functions (PDF) of the relative error between the original and reconstructed versions of the indicated fields for different compression ratios. All the fields obtained from a Greenland ice sheet finite element simulation and are represented on an unstructured mesh with 2.8 million data points distributed on 64 processes. Results are reported for a local StOMP reconstruction with a wavelet order $w = 3$ where the compression ratios were optimally estimated

[45] suggests different errors metrics that cover the statistical and visual aspects of the difference between any two datasets. We will consider these error metrics in our future work on scientific data compression.

7 Conclusion

In this paper, we have demonstrated the application of compressed sensing to unstructured mesh data. We used Alpert tree wavelets to efficiently represent the irregularities present in the spatial geometries and meshes. We have devised an improved version of the StOMP algorithm to reconstruct the data more accurately and in an automated manner. We have also developed a formula to predict the compressibility of any dataset as a function of the spatial gradients before using compressed sensing which aids in reducing the reconstruction error. This in situ prediction of the compression ratio is also useful to detect interesting features and physical phenomena occurring in the data during a large-scale simulation. We found that CS has a reasonable performance characterized by a large data compression speed (~ 100 MB/s) which makes it attractive in cases were the compression overhead needs to be minimized. We also found that CS has the limitation of slow data reconstruction (~ 1 MB/s) which makes it useable in cases where data reconstruction speed is not a significant concern. CS also has the limitation that gradients fail to reconstruct accurately from data reconstructed from compressed samples.

We are able to achieve lossy compression ratios up to two orders of magnitude, depending on the oscillations and features present in the data. Deterioration in the reconstructed data at those ratios are reasonably minimal. The features in the data that are manifested by large gradients, and discontinuities in the data contribute in assessing the reconstruction quality. This is also the case for other compression methods we considered. Thus, scientific data compression is highly data and application dependent. Moreover, the choice of the error metric is crucial in assessing the required aspects of the data reconstruction properties.

For the time being CS does not seem an attractive unstructured mesh data compression method overall compared with the Alpert wavelet compression. Additional research is required in future work to extend the scope of compressed sensing in scientific data compression and strengthen its underlying methodology. Given that the reconstruction speed is relatively slow when using compressed sensing, time series models could be used to decrease the frequency of reconstruction, hence alleviating its overall cost in a transient simulation. It may also be the case that other wavelet and sampling matrix pairs and reconstruction algorithms will produce better results on some data. Finally, error metrics have to be developed in order to assess a more accurate a priori visual and statistical quality of the reconstructed data. We continue to explore all these ways to improve our algorithms.

Acknowledgements The authors would like to acknowledge Dr. Jaideep Ray and Dr. Keita Teranishi for providing valuable discussions and feedback that were helpful to accomplish this work. This work was supported by the Laboratory Directed Research and Development (LDRD) program at Sandia National Laboratories. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

References

1. Klasky S, Abbasi H, Logan J, Parashar M, Schwan K, Shoshani A et al (2011) In situ data processing for extreme scale computing. In: Proceedings of SciDAC 2011
2. Evans LC (2010) Partial differential equations, graduate studies in mathematics. American Mathematical Society, Providence
3. Gersho A, Gray RM (2012) Vector quantization and signal compression. Springer, New York
4. Lindstrom P (2014) Fixed-rate compressed floating-point arrays. IEEE Trans Vis Comput Graph 20(12):2674–2683
5. Donoho D (2006) Compressed sensing. IEEE Trans Inf Theory 52(4):1289–1306
6. Lehmann H, Jung B (2014) In-situ multi-resolution and temporal data compression for visual exploration of large-scale scientific simulations. In: IEEE 4th symposium on large data analysis and visualization (LDAV), Paris, France
7. Lakshminarasimhan S, Zou X, Boyuka DA, Pendse SV, Jenkins J, Vishwanath V, Papka ME, Klasky S, Samatova NF (2014) DIRAQ: scalable in situ data-and resource-aware indexing for optimized query performance. Clust Comput 17(4):1101–1119
8. Bernardon FF, Callahan SP, Comba JLD, Silva CT (2005) Rendering of time-varying scalar fields on unstructured meshes. Technical report, Lawrence Radiation Laboratory
9. Zhao K, Sakamoto N, Koyamada K (2015) Time-varying volume compression in spatio-temporal domain. J Adv Simul Sci Eng 1(1):171–187
10. Austin W, Ballard G, Kolda TG (2016) Parallel tensor compression for large-scale scientific data. Technical report. arXiv:1510.06689v2
11. Sen P, Darabi S (2011) Compressive rendering: a rendering application of compressed sensing. IEEE Trans Vis Comput Graph 17(4):487–499
12. Xu X, Sakhaee E, Entezari A (2014) Volumetric data reduction in a compressed sensing. Comput Graph Forum 33(3):111–120
13. Liu X, Alim UR (2015) Compressive volume rendering. Comput Graph Forum 34(3):101–110
14. Yu H, Wang C, Grout RW, Chen JH, Ma K (2010) In situ visualization for large-scale combustion simulations. IEEE Comput Graph Appl 30(3):45–57
15. Sauer F, Yu H, Ma K (2013) An analytical framework for particle and volume data of large-scale combustion simulations. In: Proceedings of the 8th international workshop on ultrascale visualization. ACM, New York, USA
16. Fabian N, Moreland K, Thompson D, Bauer AC, Marion P, Geveci B, Rasquin M, Jansen KE (2011) The paraview coprocessing library: a scalable, general purpose in situ visualization library. In: 2011 IEEE symposium on large data analysis and visualization (LDAV)
17. Woodring J, Ahrens J, Figg J, Wendelberger J, Habib S, Heitmann K (2011) In situ sampling of a large scale particle simulation for interactive visualization and analysis. SIAM J Math Anal 30(3):1151–1160
18. Bennett JC, Comandur S, Pinar A, Thompson D (2013) Sublinear algorithms for in-situ and in-transit data analysis at the extreme-scale. In: DOE workshop on applied mathematics research for exascale computing, Washington, DC, USA
19. Alpert B, Beylkin G, Coifman R, Rokhlin V (1993) Wavelet-like bases for the fast solution of second-kind integral equations. SIAM J Sci Comput 14(1):159–184
20. Pogossova E, Egiazarian K, Gotchev A, Astola J (2005) Tree-structured legendre multi-wavelets. In: Computer aided systems theory EUROCAST 2005. Volume 3643 of lecture notes in computer science. Springer, pp 291–300
21. Donoho DL, Tsaig Y, Drori I, Starck J-L (2012) Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit. IEEE Trans Inf Theory 58(2):1094–1121
22. Tsaig Y, Donoho D (2006) Extensions of compressed sensing. Signal Process 86(3):533–548
23. Candes E, Wakin M (2008) An introduction to compressive sampling. IEEE Signal Process Mag 25(2):21–30
24. Radunovic DM (2009) Wavelets: from math to practice. Springer, New York
25. Jansen M, Oonincx P (2005) Second generation wavelets and applications. Springer, New York
26. Sweldens W (1998) The lifting scheme: a construction of second generation wavelets. SIAM J Math Anal 29(2):511–546
27. Maggioni M, Bremer JC, Coifman RR, Szlam AD (2005) Biorthogonal diffusion wavelets for multiscale representations on manifolds and graphs. In: Proceedings of SPIE 5914, Wavelets XI, 59141M, San Diego, USA
28. Alpert BK (1993) A class of bases in L^2 for the sparse representation of integral operators. SIAM J Math Anal 24(1):246–262
29. Press WH, Teukolsky SA, Vetterling WT, Flannery BP (2007) Numerical recipes 3rd edition: the art of scientific computing. Cambridge University Press, Cambridge
30. Lodhi MA, Voronin S, Bajwa WU (2016) YAMPA: yet another matching pursuit algorithm for compressive sensing. In: Proceedings of SPIE 9857, compressive sensing V: from diverse modalities to big data analytics, 98570E, Baltimore, USA
31. Yin P, Esser E, Xin J (2014) Ratio and difference of L_1 and L_2 norms and sparse representation with coherent dictionaries. Commun Inf Syst 14(2):87–109
32. Heroux MA, Bartlett RA, Howle VE, Hoekstra RJ, Hu JJ, Kolda TG, Lehoucq RB, Long KR, Pawlowski RP, Phipps ET, Salinger AG, Thornquist HK, Tuminaro RS, Willenbring JM, Williams A, Stanley KS (2005) An overview of the Trilinos project. ACM Trans Math Softw 31(3):397–423
33. Ayachit U (2015) The paraview guide: a parallel visualization application. Kitware, Clifton Park
34. Swinzip v1.0 (2016) A Matlab and C++ library for scientific lossy data compression and reconstruction using compressed sensing and tree-wavelets transforms. Sandia National Laboratories. http://www.sandia.gov/~mnsallo/swinzip/swinzip-v1.0.tgz
35. Boyd S, Parikh N, Chu E, Peleato B, Eckstein J (2011) Distributed optimization and statistical learning via the alternating direction method of multipliers. Found Trends Mach Learn 3(1):1–122
36. Yin W, Osher S, Goldfarb D, Darbon J (2008) Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing. SIAM J Imaging Sci 1(1):143–168
37. Needell D, Tropp JA (2010) CoSaMP: iterative signal recovery from incomplete and inaccurate samples. Commun ACM 57(12):93–100
38. Wright SJ, Nowak RD, Figueiredo MAT (2009) Sparse reconstruction by separable approximation. IEEE Trans Signal Process 57(7):2479–2493
39. Lakshminarasimhan S, Zou X, Boyuka DA II, Pendse SV, Jenkins J, Vishwanath V, Papka ME, Klasky S, Samatova NF (2014) DIRAQ: scalable in situ data-and resource-aware indexing

for optimized query performance. Clust Comput 14(4):1101–1119
40. Kokjohn SL, Hanson RM, Splitter DA, Reitz RD (2011) Fuel reactivity controlled compression ignition (RCCI): a pathway to controlled high-efficiency clean combustion. Int J Engine Res 12:209–226
41. Bhagatwala A, Sankaran R, Kokjohn S, Chen JH. Numerical investigation of spontaneous flame propagation under RCCI conditions. Combust Flame **(under review)**
42. Safta C, Blaylock M, Templeton J, Domino S, Sargsyan K, Najm H (2016) Uncertainty quantification in LES of channel flow. Int J Numer Methods Fluids 83:376–401
43. Murphy KP (2012) Machine learning: a probabilistic perspective. MIT Press, Cambridge
44. Salloum M, Templeton J (2014) Inference and uncertainty propagation of atomistically-informed continuum constitutive laws, part 1: Bayesian inference of fixed model forms. Int J Uncertain Quantif 4(2):151–170
45. Wang C, Ma K-L (2008) A statistical approach to volume data quality assessment. IEEE Trans Vis Comput Graph 14(3):590–602
46. Salloum M, Bennett JC, Pinar A, Bhagatwala A, Chen JH (2015) Enabling adaptive scientific workflows via trigger detection. In: Proceedings of the first workshop on in situ infrastructures for enabling extreme-scale analysis and visualization, pp 41–45
47. Chen JH, Choudhary A, De Supinski B, DeVries M, Hawkes ER, Klasky S, Liao WK, Ma KL, Mellor-Crummey J, Podhorszki N et al (2009) Terascale direct numerical simulations of turbulent combustion using S3D. Comput Sci Discov 2(1):015001
48. zfp & fpzip (2015) Floating point compression. Lawrence Livermore National Laboratories. http://computation.llnl.gov/projects/floating-point-compression/download/zfp-0.4.1.tar.gz
49. Tezaur I, Perego M, Salinger A, Tuminaro R, Price S (2015) Albany/felix: a parallel, scalable and robust finite element higher-order stokes ice sheet solver built for advanced analysis. Geosci Model Dev 8:1–24