Version of Record: https://www.sciencedirect.com/science/article/pii/S0021999120304782
Manuscript_cb43ec025ef161beb2afca8c4d4637ca

Pass-efficient methods for compression of high-dimensional turbulent flow data

Alec M. Dunton[a], Lluís Jofre[b,c], Gianluca Iaccarino[b], Alireza Doostan[d,*]

[a] *Applied Mathematics, University of Colorado, Boulder, CO 80309, USA*
[b] *Center for Turbulence Research, Stanford University, Stanford, CA 94305, USA*
[c] *Dept. Fluid Mechanics, Technical University of Catalonia - BarcelonaTech, Barcelona 08019, Spain*
[d] *Smead Aerospace Engineering Sciences, University of Colorado, Boulder, CO 80309, USA*

Abstract

The future of high-performance computing, specifically on future Exascale computers, will presumably see memory capacity and bandwidth fail to keep pace with data generated, for instance, from massively parallel partial differential equation (PDE) systems. Current strategies proposed to address this bottleneck entail the omission of large fractions of data, as well as the incorporation of *in situ* compression algorithms to avoid overuse of memory. To ensure that post-processing operations are successful, this must be done in a way that a sufficiently accurate representation of the solution is stored. Moreover, in situations where the input/output system becomes a bottleneck in analysis, visualization, etc., or the execution of the PDE solver is expensive, the number of passes made over the data must be minimized. In the interest of addressing this problem, this work focuses on the utility of pass-efficient, parallelizable, low-rank, matrix decomposition methods in compressing high-dimensional simulation data from turbulent flows. A particular emphasis is placed on using coarse representation of the data – compatible with the PDE discretization grid – to accelerate the construction of the low-rank factorization. This includes the presentation of a novel single-pass matrix decomposition algorithm for computing the so-called interpolative decomposition. The methods are described extensively and numerical experiments on two turbulent channel flow data are performed. In the first (unladen) channel flow case, compression factors exceeding 400 are achieved while maintaining accuracy with respect to first- and second-order flow statistics. In the particle-laden case, compression factors of 100 are achieved and the compressed data is used to recover particle velocities. These results show that these compression methods can enable efficient computation of various quantities of interest in both the carrier and disperse phases.

Keywords: Big-data compression; interpolative decomposition; low-rank approximation; particle-laden turbulence; randomized algorithm; single-pass algorithm

1. Introduction

The design of modern high-performance-computing (HPC) facilities is constrained by the balance required between financial budget, computing power, and energy consumption. These constraints force system architects to make difficult trade-offs among supercomputer components, e.g., floating-point performance, memory capacity, interconnect speed, input/output (I/O), etc. As predicted by Moore's [1] and Kryder's [2] laws, many features of supercomputers have improved substantially over the past few decades. However, memory capacity and bandwidth have failed to keep pace with data generation capabilities. This trend is not reverting and will most likely augment in the near future. For example, it is expected that the Exascale

*Corresponding author
Email addresses: alec.dunton@colorado.edu (Alec M. Dunton), lluis.jofre@upc.edu (Lluís Jofre), jops@stanford.edu (Gianluca Iaccarino), alireza.doostan@colorado.edu (Alireza Doostan)

© 2020 published by Elsevier. This manuscript is made available under the Elsevier user license
https://www.elsevier.com/open-access/userlicense/1.0/

supercomputers to be deployed during the next decade will provide a 1,000-10,000× increase in floating-point performance but only a 100× increase in memory availability and access speed [3].

Flow solvers use random access memory (RAM), I/O, and disk space to store solution states at different times for subsequent restart and post-processing. As the gap between data generation and storage performance has increased, numerical solvers have typically adapted by saving their state less often, viz. temporal or spatial sub-sampling. This can lead to the loss of important data, rendering it less useful in post-processing operations. This problem is of particular importance in the case of turbulent flows, as the number of spatial and time integration resolutions required to capture all the flow scales in direct numerical simulation (DNS) increases exponentially with the Reynolds number, Re. Extrapolating this trend to future supercomputing settings, storage subsystems may become considerably underpowered with respect to the number-crunching capacity. In this scenario, the affordable resulting data storage frequency will not be sufficient for conducting meaningful analyses. A similar problem is encountered in outer-loop studies, such as inference, uncertainty quantification (UQ), and optimization, in which large ensembles of model evaluations for different input values are performed, resulting in a rapid growth of data storage requirements [4, 5, 6]. The storage capacity and bandwidth limitations also complicate the applicability of time-decoupled strong recycling turbulence inflow methods [7], in which flow data for several characteristic integral times, e.g., eddy-turnover time in homogeneous isotropic turbulence (HIT) or flow through time (FTT) in wall-bounded flows, are stored to disk to be reused later as inflow in spatially developing flow problems. If the prediction described above materializes, flow solvers will need to pursue new strategies in which the data size at each time slice is reduced before writing to disk, a process known as data compression.

Data compression can be divided into five main categories: lossless, near lossless, lossy, mesh reduction, and derived representations [8]. Focusing on the categories of lossless and lossy compression, the primary contrast between the two is that lossless algorithms guarantee reconstruction of compressed data without any loss of accuracy — within machine precision in the case of near lossless — whereas lossy algorithms do not. Due to their accuracy, lossless compression algorithms are more widely accepted in the scientific community for the purposes of data visualization, analysis, and compression. However, the guarantee of accuracy inherent in these methods comes at the cost of limited compression ratios [9, 10]. On the other hand, higher compression ratios can be obtained by using lossy data compression algorithms. This comes at the expense that the inverse transformation of the compressed data produces at best an approximation of the original data.

There are numerous existing methods in truly lossless compression, wherein the reconstructed data is bit-for-bit identical to the original [8]. Examples include the well known method *gzip* [11], as well as entropy-based coders [12, 13], dictionary-based coders [14, 15], and predictive coders, e.g., FPC [16], FCM [17], and FPZIP [18]. In a related class of methods, near-lossless compressers, reconstructed data is not identical to the original data due to floating-point round-off errors. Examples from this class of approaches include transformation methods such as lossless Fourier and wavelet transform schemes [19]. Because of the limited compression ratio attained by these methods coupled with the disk- and RAM-prohibitive magnitude of the data examined in Section 3, lossy compression methods for turbulent flow data are presented as a more appealing alternative. A more in-depth review of these strategies, as well as a more extensive list of sources can be found in [8].

As in the case of lossless compression techniques, there are several lossy compression approaches, including [20, 21, 22, 23, 24]. Notable lossy compression methods include bit truncation [8, 25], in which simulations are run using 64-bit floating point values but only 32-bit values are saved. Another lossy approach, quantization, entails floating-point values being converted into approximations with smaller cardinality, a fixed size following compression can be achieved, but without guarantees on error [8]. Predictive coding techniques rely on approximating data values using extrapolation from neighboring values. Examples from this class of methods include linear predictors such as *Compvox* [26], SZ [27, 28, 29], and *Lorenzo* [30], as well as spline-fitting predictors like *Isabela* [31, 20]. Transform-based compression methods involve computing the transform of the data, e.g., discrete Legendre transform [32, 33], discrete cosine transform (DCT) and wavelet transform, then storing the resulting coefficients in a manner that reduces memory footprint. Within this group of methods is ZFP [34], the Karhunen-Loéve transform [35], the Tucker decomposition for tensor data [36, 37, 23], and higher order methods based on it, e.g., [38, 39, 40].

1.1. Contribution of this work

This work is concerned with temporal compression of large-scale fluid dynamics simulation data via low-rank matrix decomposition algorithms. Standard low-rank factorization methods for temporal compression of large-scale fluid dynamics simulation data, such as standard proper orthogonal decomposition (POD) or principle component analysis (PCA), are not well suited to high-dimensional data due to their significant computational cost, high memory requirements, and limited parallel scalability [41]. In addition, they are not *pass*-efficient in that they need to access/read the data multiple times. Due to the memory bottlenecks inherent in large-scale simulations, methods which minimize the number of passes made over a data matrix are emphasized.

In this work, we examine the utility of four pass-efficient, low-rank factorization techniques for temporal compression of turbulent flow data. These include a blocked single-pass singular-value decomposition (SBR-SVD) [42], a single-pass and two-pass variants of interpolative decomposition (ID) [43]. The methods enable low-rank approximation of flows without scalability issues nor bottlenecks in extension to higher dimensions. In building two of the ID methods, we propose using coarse grid (or grid sub-sampled) representations, a.k.a *sketch*, of the data (as an alternative to random projections) in order to accelerate the construction of the factorization. This particular choice of sketch enables a single-pass implementation of ID, which, to the best of the authors' knowledge, is the first single-pass ID algorithm. In addition, we provide convergence analysis of the ID schemes relying on coarse grid data.

The pass-efficiency of single-pass ID (and SBR-SVD) becomes crucial when data sizes exceed memory available in RAM, as well as when the process of loading data into RAM or repeating the (PDE) simulations becomes a computational bottleneck. In such scenarios, SBR-SVD and single-pass ID have clear benefits over the two-pass algorithms; they require that the data be read only once. The single-pass ID algorithm presented in this work is limited to data matrices with an associated mesh, and can be less accurate than the two-pass algorithms depending on the desired compression factors. Our empirical observations indicate the overall superiority of SBR-SVD over single-pass ID in terms of the accuracy of the reconstructed data with similar compression factors. Additionally, SBR-SVD frequently leads to accuracy superior to that of the two-pass ID techniques. Practitioners may decide which scheme is best suited for their application and computing platform, balancing accuracy and performance.

Similar to this work, the authors of [44] present matrix decomposition methods as an effective compression technique for large-scale simulation data, though pass-efficiency is not emphasized in their work. In [45] and [46], the authors present online methods for maintaining a low-rank SVD approximation of simulation data via rank-one updates; this procedure requires significant computation at each step, however. Recent work by Tropp et al. [47] addresses this concern, though in their framework matrix updates arrive as sparse or rank-one linear updates; in the single-pass framework presented in this work, updates are assumed to arrive as row vectors into RAM. Moreover, in this work, the temporal (row) dimension of the PDE data matrix does not need to be known *a priori*.

As the focus of this paper is achieving temporal compression, methods designed for spatial compression in simulations of turbulent flow such as mesh reduction [48] and compressed sensing [49] are left for future investigation. Another interesting future work is a formal comparison of temporal compression techniques based on low-rank factorization with the aforementioned (non-matrix) techniques, such as SZ and FPZIP.

The rest of this manuscript is organized as follows. In Section 2, low-rank factorization strategies (Algorithms 1-6) for pass-efficient compression of high-dimensional data, including a novel single-pass ID, are described. Next, numerical results regarding the compression efficiency and reconstruction accuracy of Algorithms 2-6 for the computation of flow and particle statistics are discussed in Section 3. Finally, conclusions are drawn and future work is outlined in Section 4.

2. Low-rank decomposition methods for data compression

2.1. Review of QR and SVD

Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ denote the data matrix of interest whose rows are time snapshots of flow data, e.g., pressure or velocity, as depicted in Figure 1. While in the below discussions we refer to this matrix, we note

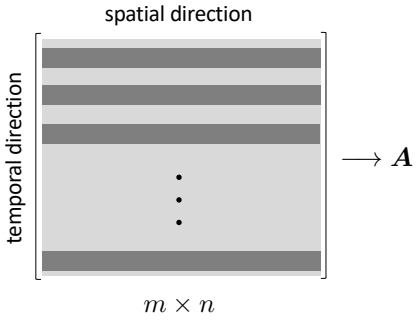

Figure 1: Schematic of a PDE data matrix $\boldsymbol{A}$ with m time solutions (rows) and n spatial degrees of freedom.

that in practice $\boldsymbol{A}$ is not formed explicitly due to the high-dimensionality of data, i.e., large m and/or n. Instead rows of $\boldsymbol{A}$ are *processed* one at a time. The matrix decomposition methods presented in this work involve at their core two canonical decompositions: the QR decomposition and the SVD. QR factorization yields a decomposition of $\boldsymbol{A}$ (more precisely $\boldsymbol{A}^T$) of the form $\boldsymbol{A} = \boldsymbol{QR}$, where $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ is a unitary matrix whose columns form an orthonormal basis for the column space of $\boldsymbol{A}$ [50]. This procedure is referred to as the range-finding step [41] in the algorithms described later in the paper. The three main approaches for computing this decomposition include pivoted Gram-Schmidt orthonormalization of the columns or rows of $\boldsymbol{A}$, Householder reflections, and Givens rotations [50]. Of particular interest in this work is the rank-revealing QR algorithm, which relies on the full-pivoted Gram-Schmidt procedure [51]. In the execution of this procedure, k pivot columns are selected to form an approximate basis for the range of $\boldsymbol{A}$. The selection of these columns induces the concept of a numerical rank [52]. A matrix $\boldsymbol{A}$ is said to be of numerical rank k for some $\epsilon > 0$ if there exists a matrix $\boldsymbol{A}_k$ of rank k such that $\|\boldsymbol{A} - \boldsymbol{A}_k\| \leq \epsilon$. This concept lies at the center of low-rank matrix decomposition methods.

Also crucial to the compression methods explored is the singular value decomposition (SVD), defined as $\boldsymbol{A} = \boldsymbol{USV}^T$ (the transpose is left unconjugated because the data in all applications is real in this work). The matrices $\boldsymbol{U} \in \mathbb{R}^{m\times n}, \boldsymbol{V} \in \mathbb{R}^{n\times n}$ are unitary and their columns form orthonormal bases for the column and row spaces of $\boldsymbol{A}$, respectively. The matrix $\boldsymbol{S} \in \mathbb{R}^{n\times n}$ is a diagonal matrix whose entries are the singular values of $\boldsymbol{A}$. To generate low-rank approximations of a matrix $\boldsymbol{A}$, one may employ a truncated SVD, which yields a decomposition of the form $\boldsymbol{U}_k\boldsymbol{S}_k\boldsymbol{V}_k^T$, with $\boldsymbol{U}_k \in \mathbb{R}^{m\times k}, \boldsymbol{V}_k \in \mathbb{R}^{n\times k}$ and $\boldsymbol{S}_k \in \mathbb{R}^{k\times k}$. In this decomposition, the column spaces of $\boldsymbol{U}_k$ and $\boldsymbol{V}_k$ are approximations of the k-dimensional row and column subspaces of the matrix $\boldsymbol{A}$, taken in correspondence to its k largest singular values, which are the entries $\sigma_1, ..., \sigma_k$ of the $k \times k$ diagonal matrix $\boldsymbol{S}_k$. The product $\boldsymbol{D} = \boldsymbol{U}_k\boldsymbol{S}_k\boldsymbol{V}_k^T$ forms a rank-k approximation of the matrix $\boldsymbol{A}$. By the Eckart-Young theorem [53], a truncated SVD is the theoretically best rank-k approximation of a matrix $\boldsymbol{A}$ in Frobenius norm, i.e.,

$$\inf_{\text{rank}(\boldsymbol{D})=k} \|\boldsymbol{A} - \boldsymbol{D}\|_F = \|\boldsymbol{A} - \boldsymbol{U}_k\boldsymbol{S}_k\boldsymbol{V}_k^T\|_F = \left(\sum_{j=k+1}^{\min(m,n)} \sigma_j^2\right)^{1/2}, \tag{1}$$

where σ_j is the j^{th} largest singular value of $\boldsymbol{A}$. An analogous statement may be made in terms of the spectral norm (induced 2-norm), in which case the lower bound is

$$\inf_{\text{rank}(\boldsymbol{D})=k} \|\boldsymbol{A} - \boldsymbol{D}\|_2 = \|\boldsymbol{A} - \boldsymbol{U}_k\boldsymbol{S}_k\boldsymbol{V}_k^T\|_2 = \sigma_{k+1}. \tag{2}$$

2.2. Randomized algorithms

The first methods developed for computing a low-rank approximation of a matrix, e.g., via the SVD, are often computationally expensive, can require $\mathcal{O}(k)$ passes over the input data matrix, lend themselves

to limited parallel scalablity, and are not designed to minimize memory movement [41]. Moreover, when such schemes are built in a purely deterministic framework, adversarial cases can be introduced, such as those presented by Kahan [54]. Developed in order to address these issues, randomized schemes have gained popularity in recent years in low-rank matrix factorizations. These methods rely on embedding the input matrix in a lower dimensional space via a random matrix $\boldsymbol{\Omega} \in \mathbb{R}^{n\times l}$, with $l \ll n$,

$$\boldsymbol{A\Omega}, \tag{3}$$

referred to as randomized projection [55, 56, 41, 57, 42]. The resulting matrix $\boldsymbol{A\Omega}$ is called a sketch matrix [41]. The effectiveness of randomized matrix algorithms relies on the utility of the sketch matrix $\boldsymbol{A\Omega}$. That is, they require that the column space of $\boldsymbol{A\Omega}$ approximately spans the column space of $\boldsymbol{A}$. The theoretical underpinning of random projections in numerical linear algebra is the Johnson-Lindenstrauss Lemma [58], which roughly speaking states that $\mathcal{O}(n)$ points in a Euclidean space may be randomly embedded in a $\mathcal{O}(\log(n))$-dimensional space such that pairwise distances between points are nearly preserved. This result precipitated the introduction of random matrices as dimension reduction tools in numerical linear algebra [41]. Matrices with i.i.d. Gaussian entries – used exclusively in this work – are a preeminent examples of such random projections, [55], though the literature is rife with other techniques for matrix sketching.

Benefits of using randomized methods for generating low-rank approximation, more of which are enumerated in [57], are:

- The cost of computing a k-rank approximation of $\boldsymbol{A}$ using deterministic methods, including some of those implemented in this work, requires $\mathcal{O}(mnk)$ operations. By using randomized methods this can be reduced to $\mathcal{O}(mn\log(k) + k^2(m+n))$ or better [41].
- Randomized methods require less communication than standard methods, which enables efficient implementation in low-communication environments such as graphics processing units (GPUs) [59].
- Of particular interest to this study is that randomized methods allow (in some implementations) for single-pass compression of matrices, which means that the matrix can be compressed as it is streamed and never has to be stored in RAM in its entirety [41, 42].

2.3. Randomized SVD and single-pass algorithms

A natural application of random projection is in the construction of SVD. Randomized SVD (R-SVD) methods approximate $\boldsymbol{A}$ in the form $\boldsymbol{A} \approx \boldsymbol{USV}^T$ for a given target rank k via two main stages. Notice that, for the interest of a simpler notation, we hereafter drop the subscript k from $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{S}$, but keep those in the Algorithms to facilitate their implementation. In the first stage of R-SVD, a basis $\boldsymbol{Q} \in \mathbb{R}^{m\times l}$ of the approximate column space of $\boldsymbol{A}$ is identified from the QR factorization of the sketch matrix $\boldsymbol{A\Omega}$, where $\boldsymbol{\Omega} \in \mathbb{R}^{n\times l}$ is, e.g., a Gaussian random matrix. Here $l = k + p$, where the so-called over-sampling parameter p is a small number, e.g., 10 or 20. In the second stage, the SVD of the smaller matrix $\boldsymbol{B} = \boldsymbol{Q}^T\boldsymbol{A} \approx \tilde{\boldsymbol{U}}\boldsymbol{SV}^T$ is computed. Recognizing that $\boldsymbol{A} \approx \boldsymbol{QB}$, the approximate SVD of $\boldsymbol{A}$ is given by $\boldsymbol{A} \approx (\boldsymbol{Q}\tilde{\boldsymbol{U}})\boldsymbol{SV}^T$, i.e., $\boldsymbol{U} = \boldsymbol{Q}\tilde{\boldsymbol{U}}$. The details of these steps are described in Algorithm 1 reported from [41].

More recently, multiple improvements of the R-SVD implementation in Algorithm 1 have been proposed to address pass and parallel efficiency of R-SVD, which is explained next. Firstly, Algorithm 1 requires two passes over the data matrix $\boldsymbol{A}$ (Steps 6 and 7), which makes it less attractive for the compression of large-scale PDE data that are expensive to store or re-generate. Halko et al. [41, Section 5.5] proposed a single-pass extension of R-SVD as follows:

(1) Generate Gaussian random matrices $\boldsymbol{\Omega} \in \mathbb{R}^{n\times l}$ and $\tilde{\boldsymbol{\Omega}} \in \mathbb{R}^{m\times l}$, where $k < l \ll \min(n, m)$
(2) Compute the products $\boldsymbol{A\Omega}$ and $\boldsymbol{A}^T\tilde{\boldsymbol{\Omega}}$ in a single-pass over $\boldsymbol{A}$
(3) Using these two products, compute the two QR decompositions $\boldsymbol{A\Omega} = \boldsymbol{QR}$ and $\boldsymbol{A}^T\tilde{\boldsymbol{\Omega}} = \tilde{\boldsymbol{Q}}\tilde{\boldsymbol{R}}$
(4) Solve for the matrix $\boldsymbol{B}$, given by the minimum residual solution to the relations $\boldsymbol{Q}^T(\boldsymbol{A\Omega}) = \boldsymbol{B}\tilde{\boldsymbol{Q}}^T\boldsymbol{\Omega}$ and $\tilde{\boldsymbol{Q}}^T(\boldsymbol{A}^T\tilde{\boldsymbol{\Omega}}) = \boldsymbol{B}^T\boldsymbol{Q}^T\tilde{\boldsymbol{\Omega}}$.

Algorithm 1 Basic Randomized SVD (R-SVD) $\boldsymbol{A} \approx \boldsymbol{U}_k \boldsymbol{S}_k \boldsymbol{V}_k^T$

procedure R-SVD($\boldsymbol{A} \in \mathbb{R}^{m \times n}$)
  $k \leftarrow$ target rank
  $p \leftarrow$ oversampling parameter
  $l \leftarrow k + p$
  $\boldsymbol{\Omega} \leftarrow randn(n, l)$
  $\boldsymbol{Q}\boldsymbol{R} \leftarrow qr(\boldsymbol{A}\boldsymbol{\Omega})$
  $\boldsymbol{B} \leftarrow \boldsymbol{Q}^T \boldsymbol{A}$
  $\tilde{\boldsymbol{U}}$, $\boldsymbol{S}$, $\boldsymbol{V} \leftarrow svd(\boldsymbol{B})$
  $\boldsymbol{U} \leftarrow \boldsymbol{Q}\tilde{\boldsymbol{U}}$
  $\boldsymbol{U}_k \leftarrow \boldsymbol{U}(:, 1:k)$; $\boldsymbol{S}_k \leftarrow \boldsymbol{S}(1:k, 1:k)$; $\boldsymbol{V}_k \leftarrow \boldsymbol{V}(:, 1:k)$
  return $\boldsymbol{U}_k$, $\boldsymbol{S}_k$, $\boldsymbol{V}_k$

(5) Compute the SVD of the small matrix $\boldsymbol{B}$, yielding $\boldsymbol{B} \approx \tilde{\boldsymbol{U}}\boldsymbol{S}\boldsymbol{V}^T$
(6) Form the matrix $\boldsymbol{U} = \boldsymbol{Q}\tilde{\boldsymbol{U}}$ and set $\boldsymbol{U}_k = \boldsymbol{U}(:, 1:k)$, $\boldsymbol{S}_k = \boldsymbol{S}(1:k, 1:k)$, $\boldsymbol{V}_k = \boldsymbol{V}(:, 1:k)$, to obtain the truncated SVD $\boldsymbol{A} \approx \boldsymbol{U}_k \boldsymbol{S}_k \boldsymbol{V}_k^T$ [41]

The main drawback in this method lies in step (3) above, where the typically ill-conditioned matrix $\boldsymbol{Q}^T\tilde{\boldsymbol{\Omega}}$ may lead to considerable accumulation of error compared to the double-pass method presented in Algorithm 1 [41]. In order to reduce the communication in the parallel implementation of R-SVD in Algorithm 1 and enable adaptive rank determination, a blocked formulation of the standard algorithm above is proposed in [59]. In this approach, the orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{m \times l}$ is separated into s blocks each of size $m \times b$ in the form

$$\boldsymbol{Q} = [\boldsymbol{Q}_1, \boldsymbol{Q}_2, \cdots, \boldsymbol{Q}_s], \tag{4}$$

where $s \times b = l$. The computation of the matrix $\boldsymbol{Q}$ is then decoupled and carried out on the smaller blocks $\boldsymbol{Q}_i$, with all blocks orthogonalized via Gram-Schmidt and concatenated at the end of the process. In doing so, the rank k can be determined so that the factorization admits a prescribed error ϵ. For the interest of clarity and completeness, the main steps of this blocked formulation from [59] are now reported, which constitute an alteration to Steps 5-7 of the standard R-SVD in Algorithm 1.

(1) For each block $i = 1, 2, 3, ..., s$ **do**
(2) Generate Gaussian matrix $\boldsymbol{\Omega}_i \in \mathbb{R}^{n \times b}$, where $s \times b = l$
(3) Compute the QR factorization of $\boldsymbol{A}\boldsymbol{\Omega}_i$ to obtain $\boldsymbol{Q}_i$
(4) Re-orthonormalize $\boldsymbol{Q}_i - \sum_{j=1}^{i-1} \boldsymbol{Q}_j \boldsymbol{Q}_j^T \boldsymbol{Q}_i$
(5) Compute $\boldsymbol{B}_i = \boldsymbol{Q}_i^T \boldsymbol{A}$
(6) Set $\boldsymbol{A} = \boldsymbol{A} - \boldsymbol{Q}_i \boldsymbol{B}_i$
(7) If $\|\boldsymbol{A}\| < \epsilon$ ***stop***
(8) Construct $\boldsymbol{Q} = [\boldsymbol{Q}_1, \boldsymbol{Q}_2, \cdots, \boldsymbol{Q}_s]$; $\boldsymbol{B} = \left[\boldsymbol{B}_1^T, \boldsymbol{B}_2^T, \cdots, \boldsymbol{B}_s^T\right]^T$

However, the above implementation of the blocking procedure increases the number of passes through $\boldsymbol{A}$ to $\mathcal{O}(s)$.

In a recent work, Yu et al. [42] proposed a single-pass formulation of R-SVD that is shown empirically to result in more accurate low-rank factorizations, as compared to the single-pass R-SVD of [41]. In more detail, the approach of [42] generates an approximate truncated SVD $\boldsymbol{A} \approx \boldsymbol{U}_k \boldsymbol{S}_k \boldsymbol{V}_k^T$ of rank k following the below steps:

(1) Generate Gaussian matrix $\boldsymbol{\Omega} \in \mathbb{R}^{n \times l}$, where $k < l \ll n$
(2) Obtain the matrices $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$ and $\boldsymbol{B} = \boldsymbol{A}^T\boldsymbol{Y}$ in a single-pass over $\boldsymbol{A}$ (Steps 11-15 in Algorithm 2)
(3) Compute a QR decomposition of $\boldsymbol{Y} = \boldsymbol{Q}\boldsymbol{R}$, set $\boldsymbol{B} = \boldsymbol{B}\boldsymbol{R}^{-1}$ so $\boldsymbol{B} \approx \boldsymbol{A}^T\boldsymbol{Q}$ (Steps 16-24 in Algorithm 2).
(4) Compute the SVD of the small matrix $\boldsymbol{B}^T \approx \tilde{\boldsymbol{U}}\boldsymbol{S}\boldsymbol{V}^T$

Algorithm 2 Single-pass Blocked Randomized SVD (SBR-SVD) $\boldsymbol{A} \approx \boldsymbol{U}_k \boldsymbol{S}_k \boldsymbol{V}_k^T$ [42]

procedure SBR-SVD($\boldsymbol{A} \in \mathbb{R}^{m \times n}$)
  $k \leftarrow$ target rank
  $p \leftarrow$ over-sampling parameter
  $l \leftarrow k + p$
  $b \leftarrow$ block size
  $s \leftarrow$ number of blocks such that $s \times b = l$
  instantiate $\boldsymbol{Q}, \boldsymbol{B}$
  $\boldsymbol{\Omega} \leftarrow randn(n, l)$
  instantiate $\boldsymbol{G}$
  $\boldsymbol{H} \leftarrow zeros(n, l)$
  while $\boldsymbol{A}$ is not entirely read through ***do***
    read the next row $\boldsymbol{a}$ of $\boldsymbol{A}$
    $\boldsymbol{g} \leftarrow \boldsymbol{a}\boldsymbol{\Omega}$ $\boldsymbol{G} \leftarrow [\boldsymbol{G}; \boldsymbol{g}]$
    $\boldsymbol{H} \leftarrow \boldsymbol{H} + \boldsymbol{a}^T \boldsymbol{g}$
  end while
  for $i = 1, 2, \ldots, s$ ***do***
    $\boldsymbol{\Omega}_i \leftarrow \boldsymbol{\Omega}(:, (i-1)b + 1 : ib)$
    $\boldsymbol{Y}_i \leftarrow \boldsymbol{G}(:, (i-1)b + 1 : ib) - \boldsymbol{Q}(\boldsymbol{B}\boldsymbol{\Omega}_i)$
    $\boldsymbol{Q}_i, \boldsymbol{R}_i \leftarrow qr(\boldsymbol{Y}_i)$
    $\boldsymbol{Q}_i, \tilde{\boldsymbol{R}}_i \leftarrow qr(\boldsymbol{Q}_i - \boldsymbol{Q}(\boldsymbol{Q}^T \boldsymbol{Q}_i))$
    $\boldsymbol{R}_i \leftarrow \tilde{\boldsymbol{R}}_i \boldsymbol{R}_i$
    $\boldsymbol{B}_i \leftarrow \boldsymbol{R}_i^{-T}(\boldsymbol{H}(:, (i-1)b + 1 : ib)^T - \boldsymbol{Y}_i^T \boldsymbol{Q}\boldsymbol{B} - \boldsymbol{\Omega}_i^T \boldsymbol{B}^T \boldsymbol{B})$
    $\boldsymbol{Q} \leftarrow [\boldsymbol{Q}, \boldsymbol{Q}_i]$ $\boldsymbol{B} \leftarrow \left[\boldsymbol{B}^T, \boldsymbol{B}_i^T\right]^T$
  end for
  $\tilde{\boldsymbol{U}}, \boldsymbol{S}, \boldsymbol{V} \leftarrow svd(\boldsymbol{B})$;
  $\boldsymbol{U} \leftarrow \boldsymbol{Q}\tilde{\boldsymbol{U}}$;
  $\boldsymbol{U}_k \leftarrow \boldsymbol{U}(:, 1 : k)$, $\boldsymbol{S}_k \leftarrow \boldsymbol{S}(1 : k, 1 : k)$, $\boldsymbol{V}_k \leftarrow \boldsymbol{V}(:, 1 : k)$
  return $\boldsymbol{U}_k, \boldsymbol{S}_k, \boldsymbol{V}_k$

(5) Construct $\boldsymbol{U} = \boldsymbol{Q}\tilde{\boldsymbol{U}}$ and set $\boldsymbol{U}_k = \boldsymbol{U}(:, 1 : k)$, $\boldsymbol{S}_k = \boldsymbol{S}(1 : k, 1 : k)$, $\boldsymbol{V}_k = \boldsymbol{V}(:, 1 : k)$ to extract the truncated SVD $\boldsymbol{A} \approx \boldsymbol{U}_k \boldsymbol{S}_k \boldsymbol{V}_k^T$

As discussed in [42], Step (3) of this implementation can be performed in a blocked and, more importantly, single-pass mode resulting in Algorithm 2, herein referred to as single-pass blocked randomized SVD (SBR-SVD) and employed in the numerical examples.

For sufficiently large p (hence l), e.g., p between 5 and 20 [41], the output of this algorithm is an approximately optimal k-rank approximation for a matrix per the Eckart-Young theorem [53]. More specifically,

$$\mathbb{E}(\|\boldsymbol{A} - \boldsymbol{U}_k \boldsymbol{S}_k \boldsymbol{V}_k^T\|_F) \leq \left(1 + \frac{k}{p-1}\right)^{1/2} \left(\sum_{j=k+1}^{\min(m,n)} \sigma_j^2\right)^{1/2}. \tag{5}$$

In words, the error is on average only worse than the optimal solution by a factor of $[1 + k/(p-1)]^{1/2}$ [42].

The SBR-SVD algorithm has computational complexity $\mathcal{O}(mnk)$, which can be reduced to $\mathcal{O}(mn \log k)$ when implemented with certain optimized matrix sketches including the sub-sampled random Fourier transform [41, 60, 61]. In addition, it has an approximate processing storage requirement of $l(m + 2n)$ elements in RAM during execution [42]. Other single-pass implementations of randomized SVD are available in the literature, see, e.g., [62, 63, 64, 65, 66], which are not considered in this work.

2.4. Interpolative decomposition (ID) and its randomized variant

The row ID, a two-pass algorithm, generates a decomposition of a matrix $\boldsymbol{A}$ following the form, [43],

$$\boldsymbol{A} \approx \boldsymbol{P}\boldsymbol{A}(\mathcal{I},:), \tag{6}$$

where the *row skeleton* $\boldsymbol{A}(\mathcal{I},:) \in \mathbb{R}^{k\times n}$ consists of a set of rows of $\boldsymbol{A}$ indexed by $\mathcal{I} \subseteq \{1,\ldots,m\}$ with size $|\mathcal{I}| = k$. Further, $\boldsymbol{P} \in \mathbb{R}^{m\times k}$ is a coefficient matrix such that $\boldsymbol{P}(\mathcal{I},:) = \boldsymbol{I}$, with $\boldsymbol{I}$ the identity matrix. Row ID earns its name from the fact that it *interpolates* $\boldsymbol{A}$ in a basis consisting of a subset of its rows. The core procedure in the algorithm used to generate this decomposition are the column-pivoted (rank-revealing) QR algorithm [51], which yields the index vector $\mathcal{I}$ along with a least squares problem to compute the coefficient matrix $\boldsymbol{P}$. In more detail, first, the rank k column-pivoted QR decomposition of $\boldsymbol{A}^T$ is computed.

$$\boldsymbol{A}^T\boldsymbol{Z} \approx \boldsymbol{Q}\boldsymbol{R}, \tag{7}$$

where $\boldsymbol{Z} \in \mathbb{R}^{m\times m}$ is a permutation matrix encoding the pivoting done in the algorithm, $\boldsymbol{Q} \in \mathbb{R}^{n\times k}$ has orthonormal columns, and $\boldsymbol{R} \in \mathbb{R}^{k\times m}$ is upper triangular. Separating the matrix $\boldsymbol{R}$ by its columns into two sub-matrices $\boldsymbol{R} = [\boldsymbol{R}_1 \,|\, \boldsymbol{R}_2]$, where $\boldsymbol{R}_1 \in \mathbb{R}^{k\times k}$ and $\boldsymbol{R}_2 \in \mathbb{R}^{k\times(m-k)}$, and approximating $\boldsymbol{R}_2 \approx \boldsymbol{R}_1\boldsymbol{C}$ yields

$$\boldsymbol{A}^T \approx \boldsymbol{Q}\boldsymbol{R}_1\,[\boldsymbol{I} \,|\, \boldsymbol{C}]\,\boldsymbol{Z}^T = \boldsymbol{A}^T(:,\mathcal{I})\,[\boldsymbol{I} \,|\, \boldsymbol{C}]\,\boldsymbol{Z}^T = \boldsymbol{A}^T(:,\mathcal{I})\boldsymbol{P}^T, \tag{8}$$

and hence (6). As shown in [43], the rank k row ID of an $m \times n$ matrix features a spectral error bound of

$$\|\boldsymbol{A} - \boldsymbol{P}\boldsymbol{A}(\mathcal{I},:)\|_2 \leq \sqrt{1 + k(m-k)}\,\sigma_{k+1}, \tag{9}$$

and the computational complexity of $\mathcal{O}(mnk)$. Algorithm 3 summarizes the steps involved in ID, which are used in this study. To improve the stability of the QR factorization step, the modified Gram-Schmidt ($mgsqr$) procedure of [50] is employed in Step 3.

Remark 2.1. *The ID algorithms presented in this study do not constitute a complete list of methods for generating decomposition of matrices which interpolate a subset of their rows. Other approaches for generating similar decompositions can be found in, e.g., [67, 68, 69]. It is also important to note that in the numerical results presented in Section 3, the ID refers to a specific variation of the decomposition, the row ID. Analogous definitions of a column ID or double-sided ID are also available in the literature [57].*

Algorithm 3 Row ID $\boldsymbol{A} \approx \boldsymbol{P}\boldsymbol{A}(\mathcal{I},:)$ [43]

1: **procedure** ID($\boldsymbol{A} \in \mathbb{R}^{m\times n}$)
2: $k \leftarrow$ approximation rank
3: $\boldsymbol{Q}, \boldsymbol{R}, \mathcal{I} \leftarrow mgsqr(\boldsymbol{A}^T, k)$ (First pass through $\boldsymbol{A}$)
4: $\boldsymbol{C} \leftarrow (\boldsymbol{R}(1:k, 1:k))^{+}\boldsymbol{R}(1:k, (k+1):m)$ ($^+$ denotes pseudo-inverse)
5: $\boldsymbol{Z} \leftarrow \boldsymbol{I}_m(:, [\mathcal{I}\ ,\ \mathcal{I}^c])$ ($\mathcal{I}^c$ is the complement of $\mathcal{I}$ in $\{1,\ldots,m\}$)
6: $\boldsymbol{P} \leftarrow \boldsymbol{Z}\,[\boldsymbol{I}_k \,|\, \boldsymbol{C}]^T$
7: ***return*** $\boldsymbol{A}(\mathcal{I},:)$, $\boldsymbol{P}$ (Collecting $\boldsymbol{A}(\mathcal{I},:)$ requires second pass over $\boldsymbol{A}$)

Notice that the row ID uses the entire columns of $\boldsymbol{A}$ – corresponding to all n degrees-of-freedom of the PDE data – to find the row skeleton $\boldsymbol{A}(\mathcal{I},:)$ and the coefficient matrix $\boldsymbol{P}$. Therefore, the computational cost of the QR factorization step depends explicitly on n. To reduce the cost of ID, similar random projections of Section 2.3 have been proposed in [70] to generate a sketch of $\boldsymbol{A}$ using which an approximate ID of $\boldsymbol{A}$ is computed. Specifically, randomized ID (Algorithm 4) performs the ID of the sketch matrix $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$, with a random matrix $\boldsymbol{\Omega} \in \mathbb{R}^{n\times l}$, as

$$\boldsymbol{Y} \approx \tilde{\boldsymbol{P}}\boldsymbol{Y}(\tilde{\mathcal{I}},:). \tag{10}$$

It then uses the row indices $\tilde{\mathcal{I}}$ to set the row skeleton $\boldsymbol{A}(\tilde{\mathcal{I}},:)$ and applies the same coefficient matrix $\tilde{\boldsymbol{P}}$; that is,

$$\boldsymbol{A} \approx \tilde{\boldsymbol{P}}\boldsymbol{A}(\tilde{\mathcal{I}},:). \tag{11}$$

As shown in [70], with high probability depending on l, the randomized ID leads to a low-rank factorization of $\boldsymbol{A}$ admitting bounds comparable to that of the standard ID in (9) but with larger constants.

Algorithm 4 Randomized (Row) ID (Gaussian) [56] $\boldsymbol{A} \approx \tilde{\boldsymbol{P}}\boldsymbol{A}(\tilde{\mathcal{I}}, :)$

procedure RANDID($\boldsymbol{A} \in \mathbb{R}^{m\times n}$)
  $k \leftarrow$ approximation rank
  Generate $\boldsymbol{\Omega} \in \mathbb{R}^{n\times l}$ with i.i.d. Gaussian entries and $l = k + p$
  $\boldsymbol{Y} \leftarrow \boldsymbol{A}\boldsymbol{\Omega}$ (First pass through $\boldsymbol{A}$)
  $\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{R}}, \tilde{\mathcal{I}} \leftarrow mgsqr(\boldsymbol{Y}^T, k)$
  $\tilde{\boldsymbol{C}} \leftarrow (\tilde{\boldsymbol{R}}(1:k, 1:k))^{+}\tilde{\boldsymbol{R}}(1:k, (k+1):m)$ ($^+$ denotes pseudo-inverse)
  $\tilde{\boldsymbol{Z}} \leftarrow \boldsymbol{I}_m(:, [\tilde{\mathcal{I}}\ ,\ \tilde{\mathcal{I}}^c])$ ($\tilde{\mathcal{I}}^c$ is the complement of $\tilde{\mathcal{I}}$ in $\{1, \ldots, m\}$)
  $\tilde{\boldsymbol{P}} \leftarrow \tilde{\boldsymbol{Z}}\left[\boldsymbol{I}_k \,|\, \tilde{\boldsymbol{C}}\right]^T$
  return $\boldsymbol{A}(\tilde{\mathcal{I}}, :)$, $\tilde{\boldsymbol{P}}$ (Collecting $\boldsymbol{A}(\tilde{\mathcal{I}}, :)$ requires second pass over $\boldsymbol{A}$)

2.5. Sub-sampled interpolative decomposition

Motivated by the randomized ID approach, a faster variant of ID that relies on grid (or data) sub-sampling to generate a deterministic sketch of the original data matrix $\boldsymbol{A}$ is proposed. Such a sketch is of particular interest, as it can be generated faster that the random projection – at least when $\boldsymbol{\Omega}$ is dense – by relying on the simple observation that for *smooth* solutions, a coarse representation of data is able to capture the low-rank subspace of the full solution.

Let $\mathcal{J} \subseteq \{1, \ldots, n\}$, with $k < |\mathcal{J}| = n_c \ll n$, denote the index of the degrees-of-freedom associated with a coarse subset of the original degrees-of-freedom, i.e. coarse grid or sub-sampled representation of data. The sub-sampled ID generates the rank k ID of the sub-sampled data $\boldsymbol{A}_c$,

$$\boldsymbol{A}_c = \boldsymbol{A}(:, \mathcal{J}), \tag{12}$$

as

$$\boldsymbol{A}_c \approx \hat{\boldsymbol{A}}_c = \boldsymbol{P}_c\boldsymbol{A}_c(\mathcal{I}_c, :), \tag{13}$$

which then induces an ID for the original matrix $\boldsymbol{A}$ as

$$\boldsymbol{A} \approx \hat{\boldsymbol{A}} = \boldsymbol{P}_c\boldsymbol{A}(\mathcal{I}_c, :). \tag{14}$$

In words, the indices and coefficient matrix obtained from the ID decomposition of the coarse grid data in (13) are used to generate an interpolation rule for the full data matrix $\boldsymbol{A}$, a procedure called lifting in [71]. This form of dimension reduction (i.e., directly sub-sampling the columns of the input matrix) is referred to as direct injection in the geometric multi-grid literature [72].

In addition to reducing the number of columns to be stored from n to n_c, the complexity of computing the coarse-grid ID (13) is $\mathcal{O}(mn_ck)$ instead of $\mathcal{O}(mnk)$ needed for $\boldsymbol{A}$ (see Table 1). Moreover, the sketch time required to obtain $\boldsymbol{A}_c$ is $\mathcal{O}(1)$, a far better complexity than that of, e.g., the sub-sampled random Fourier transform ($\mathcal{O}(mn\log(k))$) [56, 60]. The main steps of the sub-sampled ID are presented in Algorithm 5.

In the context of reduced order modeling and uncertainty quantification, the use of coarse grid data to *guide* the low-rank approximation of a fine grid quantity of interest has been successfully considered in several recent work; see, e.g., [73, 71, 74, 75, 76]. One such result is employed from [76, Theorem 1] to bound the error of sub-sampled ID.

Theorem 2.1. *Let $\boldsymbol{A}$ be the original data matrix, $\boldsymbol{A}_c$ the sub-sampled (coarsened) matrix as in (12), $\hat{\boldsymbol{A}}_c$ the rank k ID approximation to $\boldsymbol{A}_c$ as in (13), and $\hat{\boldsymbol{A}}$ the sub-sampled ID approximation as in (14). For any $\tau \geq 0$, let*

$$\epsilon(\tau) := \lambda_{\max}(\boldsymbol{A}\boldsymbol{A}^T - \tau\boldsymbol{A}_c\boldsymbol{A}_c^T), \tag{15}$$

where $\lambda_{\max}$ denotes the largest eigenvalue. Then,

$$\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_2 \leq \min_{\tau, k\leq \mathrm{rank}(\boldsymbol{A}_c)} \rho_k(\tau), \tag{16}$$

$$\rho_k(\tau) := \left[(1 + \|\boldsymbol{P}_c\|_2)\sqrt{\tau\sigma_{k+1}^2 + \epsilon(\tau)} + \|\boldsymbol{A}_c - \hat{\boldsymbol{A}}_c\|_2\sqrt{\tau + \epsilon(\tau)\sigma_k^{-2}}\right], \tag{17}$$

Algorithm 5 Sub-sampled ID $\boldsymbol{A} \approx \boldsymbol{P}_c \boldsymbol{A}(\mathcal{I}_c, :)$

procedure SUBID($\boldsymbol{A} \in \mathbb{R}^{m \times n}$)
  $k \leftarrow$ approximation rank
  $\boldsymbol{A}_c^T \leftarrow subsample(\boldsymbol{A}^T)$ (First pass through $\boldsymbol{A}$)
  $\boldsymbol{Q}_c, \boldsymbol{R}_c, \mathcal{I}_c \leftarrow mgsqr(\boldsymbol{A}_c^T, k)$
  $\boldsymbol{C}_c \leftarrow (\boldsymbol{R}_c(1:k, 1:k))^+ \boldsymbol{R}_c(1:k, (k+1):m)$ ($^+$ denotes pseudo-inverse)
  $\boldsymbol{Z}_c \leftarrow \boldsymbol{I}_m(:, [\mathcal{I}_c \ , \ \mathcal{I}_c^c])$ ($\mathcal{I}_c^c$ is the complement of $\mathcal{I}_c$ in $\{1, \ldots, m\}$)
  $\boldsymbol{P}_c \leftarrow \boldsymbol{Z}_c \left[\boldsymbol{I}_k \mid \boldsymbol{C}_c\right]^T$
  return $\boldsymbol{A}(\mathcal{I}_c, :)$, $\boldsymbol{P}_c$ (Collecting $\boldsymbol{A}(\mathcal{I}_c, :)$ requires second pass over $\boldsymbol{A}$)

where σ_k and σ_{k+1} are the k^{th} and $(k+1)^{th}$ largest singular values of $\boldsymbol{A}_c$, respectively.

The interested reader is referred to [76, Theorem 1] for the details of the proof of this theorem. Instead, some remarks regarding the results are provided next. Firstly, following [70], $\|\boldsymbol{P}_c\|_2 \leq \sqrt{k(m-k)+1}$. The effectiveness of the sub-sampled ID approximation $\hat{\boldsymbol{A}}$ depends on the assumptions that $\boldsymbol{A}_c$ is low-rank and the optimal $\epsilon(\tau)$ is small. The former assumption follows from the assumption that $\boldsymbol{A}$ is low-rank. To investigate $\epsilon(\tau)$, consider the case in which the physical domain of the PDE is a subset of $\mathbb{R}^3$ and the original data is generated via a uniform grid of size h in each direction. It is also assumed the sub-sampled data corresponds to a coarse subset of this grid with uniform size $H \gg h$ in each direction. It can be shown that, for $\tau^* = n/n_c$,

$$\epsilon(\tau^*) \lesssim H, \tag{18}$$

where $\lesssim$ denotes a smaller inequality with a bounded constant. To see this, we observe that the entry (i, j) of the Gramians $\boldsymbol{A}\boldsymbol{A}^T/n$ and $\boldsymbol{A}_c\boldsymbol{A}_c^T/n_c$ are the approximations of the Euclidean inner-product of the solution at times i and j via a rectangular (piece-wise constant) rule of size h and H, respectively. Therefore, with $\tau^* = n/n_c$ and $\|\cdot\|_{\max}$ the maximum entry in terms of absolute value in the argument, $\|\boldsymbol{A}\boldsymbol{A}^T - \tau^* \boldsymbol{A}_c\boldsymbol{A}_c^T\|_{\max} \lesssim H$, assuming the data snapshots have bounded first derivatives. Then, (18) follows given that $\epsilon(\tau) = \|\boldsymbol{A}\boldsymbol{A}^T - \tau\boldsymbol{A}_c\boldsymbol{A}_c^T\|_2 \leq m\|\boldsymbol{A}\boldsymbol{A}^T - \tau\boldsymbol{A}_c\boldsymbol{A}_c^T\|_{\max}$. When the singular values of $\boldsymbol{A}_c$ decay rapidly, together with (16), this dependence of $\epsilon(\tau^*)$ on H suggests the error estimate $\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_2 \lesssim H^{1/2}$ for the sub-sampled ID approximation.

2.6. Single-pass interpolative decomposition

In this section, a simple, single-pass algorithm for generating ID, dubbed single-pass ID (Algorithm 6), is presented. To the best of our knowledge, this is the first single-pass ID algorithm. The standard, randomized, or sub-sampled ID algorithms described in Sections 2.4-2.5 require a second pass through the data to extract the row skeleton matrix $\boldsymbol{A}(\mathcal{I}_c, :)$ after identifying the row indices given in $\mathcal{I}_c$ in the first pass. Instead, in single-pass ID, the row skeleton of the coarse data matrix, i.e., $\boldsymbol{A}_c(\mathcal{I}_c, :)$, are interpolated back to the original grid in order to form an approximation to $\boldsymbol{A}(\mathcal{I}_c, :)$. Specifically,

$$\boldsymbol{A} \approx \hat{\boldsymbol{A}} = \boldsymbol{P}_c \boldsymbol{A}_c(\mathcal{I}_c, :)\boldsymbol{M}, \tag{19}$$

where $\boldsymbol{P}_c$ and $\boldsymbol{A}_c(\mathcal{I}_c, :)$ are as in (13) and $\boldsymbol{M} \in \mathbb{R}^{n_c \times n}$ is a coarse to fine interpolation operator. Such a construction is similar to the prolongation step of multi-grid solvers, and is naturally motivated by the observation that the rows of $\boldsymbol{A}_c(\mathcal{I}_c, :)$ are coarse grid representation of those of $\boldsymbol{A}(\mathcal{I}_c, :)$. The trade-off for one fewer pass reduces loading and simulation time, but sacrifices accuracy as, depending on the sub-sampling factor n/n_c, this method may incur large error when interpolating back onto the fine grid. Therefore, the use of the single-pass ID is justified when rerunning the PDE solver or a second pass through the data to set $\boldsymbol{A}(\mathcal{I}_c, :)$ is not desirable. Notice that the interpolation step $\boldsymbol{A}_c(\mathcal{I}_c, :)\boldsymbol{M}$ in (19) is more computationally complex than simply accessing the rows of $\boldsymbol{A}$ indexed by $\mathcal{I}_c$ to set $\boldsymbol{A}(\mathcal{I}_c, :)$ in the randomized or sub-sampled ID. However, the interpolation can be done when the data matrix $\boldsymbol{A}$ is to be reconstructed as opposed to

Algorithm 6 Single-pass ID $\boldsymbol{A} \approx \boldsymbol{P}_c \boldsymbol{A}_c(\mathcal{I}_c, :)\boldsymbol{M}$

1: **procedure** SPID($\boldsymbol{A} \in \mathbb{R}^{m\times n}, Mesh$)
2: $k \leftarrow$ approximation rank
3: $\boldsymbol{A}_c^T \leftarrow subsample(\boldsymbol{A}^T)$ (Only pass through $\boldsymbol{A}$)
4: Form the interpolation matrix $\boldsymbol{M}$
5: $\boldsymbol{Q}_c$, $\boldsymbol{R}_c$, $\mathcal{I}_c \leftarrow mgsqr(\boldsymbol{A}_c^T, k)$
6: $\boldsymbol{C}_c \leftarrow (\boldsymbol{R}_c(1:k, 1:k))^+ \boldsymbol{R}_c(1:k, (k+1):m)$ ($^+$ denotes pseudo-inverse)
7: $\boldsymbol{Z}_c \leftarrow \boldsymbol{I}_m(:, [\mathcal{I}_c \ , \ \mathcal{I}_c^c])$ ($\mathcal{I}_c^c$ is the complement of $\mathcal{I}_c$ in $\{1, \ldots, m\}$)
8: $\boldsymbol{P}_c \leftarrow \boldsymbol{Z}_c \left[\boldsymbol{I}_k \mid \boldsymbol{C}_c\right]^T$
9: ***return*** $\boldsymbol{A}_c(\mathcal{I}_c, :)$, $\boldsymbol{P}_c$, $\boldsymbol{M}$

during the PDE simulation or data compression steps. This in turn leads to overall savings in memory movement and simulation time.

The computational complexity of single-pass ID is the same as that of the sub-sampled ID, i.e., $\mathcal{O}(mn_ck)$. Let t denote the number of coarse grid data points used to construct each of the n fine grid data points. The cost of building and storing the interpolation operator $\boldsymbol{M}$, a sparse matrix, is $\mathcal{O}(tn)$, which is negligible relative to any asymptotic dependence on mn_ck. In lieu of any spatial compression, the total disk memory required to store the output of the algorithm is far less than the other ID (or SVD) methods described in the prior sections. This is because the coarse skeleton matrix $\boldsymbol{A}_c(\mathcal{I}_c, :)$ is stored instead of its fine counterpart. This variant of ID therefore yields a spatio-temporally compression of a data set, as opposed to only temporal compression achieved by the other ID algorithms. Additionally, the total RAM usage of the algorithm is $k(m + n_c) + mn_c + tn$, where tn corresponds to the interpolation matrix $\boldsymbol{M}$. Notice that each step of the single-pass ID is the same as those of sub-sampled ID excluding the interpolation step. The interpolation operator can be represented as a sparse matrix $\boldsymbol{M}$ (constructed in Step 4 of Algorithm 6), which can then be stored to disk and applied to the coarse row skeleton $\boldsymbol{A}_c(\mathcal{I}_c, :)$ following execution of the algorithm.

A bound on the error of an approximation generated using single-pass ID is presented in the following theorem.

Theorem 2.2. *Let $\boldsymbol{A}$ be a data matrix, $\boldsymbol{A}_c$ the sub-sampled (coarsened) matrix as in (12), $\boldsymbol{M}$ the interpolation operator as in (19) and with associated interpolation error $\boldsymbol{E}_I := \boldsymbol{A} - \boldsymbol{A}_c\boldsymbol{M}$, and σ_{k+1} the $(k+1)^{th}$ largest singular value of $\boldsymbol{A}_c$. The error of the single-pass ID approximation $\hat{\boldsymbol{A}}$ in (19) is bounded as follows*

$$\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_2 \leq \|\boldsymbol{E}_I\|_2 + \|\boldsymbol{M}\|_2\sqrt{1 + k(m-k)}\sigma_{k+1}. \tag{20}$$

Proof. Note that,

$$\begin{aligned} \|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_2 &= \|\boldsymbol{A} - \boldsymbol{P}_c\boldsymbol{A}_c(\mathcal{I}_c, :)\boldsymbol{M}\|_2 \\ &\leq \|\boldsymbol{A} - \boldsymbol{A}_c\boldsymbol{M}\|_2 + \|\boldsymbol{A}_c\boldsymbol{M} - \boldsymbol{P}_c\boldsymbol{A}_c(\mathcal{I}_c, :)\boldsymbol{M}\|_2 \\ &\leq \|\boldsymbol{E}_I\|_2 + \|\boldsymbol{A}_c - \boldsymbol{P}_c\boldsymbol{A}_c(\mathcal{I}_c, :)\|_2\|\boldsymbol{M}\|_2 \\ &\leq \|\boldsymbol{E}_I\|_2 + \|\boldsymbol{M}\|_2\sqrt{1 + k(m-k)}\sigma_{k+1}, \end{aligned} \tag{21}$$

where the last inequality follows from (9) applied to the ID of $\boldsymbol{A}_c$. □

Stated differently, Theorem 2.2 suggests the error of single-pass ID depends on the low-rank structure of $\boldsymbol{A}_c$ as well as the error incurred in the interpolation step. When the data is considerably low-rank, the single-pass ID error is dominated by the interpolation error $\|\boldsymbol{E}_I\|$. Considering piece-wise linear interpolation, when data has bounded second derivative, $\|\boldsymbol{E}_I\| \lesssim H^2$, where H is the size of the course grid [77]. It is worthwhile to note that the interpolation error is independent of the target rank, instead being a function of H, or the sub-sampling parameter n/n_c, and the smoothness of the data.

The sub-sampled ID, on the other hand, is less sensitive to sub-sampling, with an error growing asymptotically like $H^{1/2}$. As the sub-sampling factor is increased, a significantly smaller increase in error when using sub-sampled ID than when using single-pass ID is to be expected. Moreover, the error of sub-sampled ID can be bounded above as a function of the singular values of the coarse matrix $\boldsymbol{A}_c$ (see Theorem 2.1), while the error of single-pass ID saturates once the interpolation error dominates the low-rank approximation error. A consequence of this is that the accuracy of single-pass ID will cease to improve despite increasing the target rank. Consequently, sub-sampled ID outperforms single-pass ID in many cases, as examined in Section 3. Despite the general superiority of sub-sampled ID in constructing row-rank factorizations, single-pass ID may achieve greater accuracy in certain cases. In particular, when an insufficiently small approximation rank is used, e.g., $k \sim \mathcal{O}(1)$, the absolute interpolation error from generating $\boldsymbol{A}_c(\mathcal{I},:)\boldsymbol{M}$ can be negligible compared to the error incurred by the low-rank approximation in and of itself. Moreover, when the coarse data is generated via minimal sub-sampling, e.g., $n/n_c \sim \mathcal{O}(1)$, the interpolation error is small enough to compete with lifting in the reconstruction of $\boldsymbol{A}$ in the sub-sampled ID. This is corroborated by the result reported in the left panel of Figure 4 in Section 3.1.

Remark 2.2. *An important difference between single-pass ID and SBR-SVD is how their respective stopping criteria are defined. SBR-SVD requires knowledge of the target rank, i.e., compression factor (CF), whereas ID adaptively evaluates a stopping criterion based on the decay of the singular values of the input matrix via a rank-revealing QR algorithm. SBR-SVD takes as input a matrix, a target rank corresponding to compression dimension, and a block size (see Algorithm 3). Consequently, in SBR-SVD the desired compression dimension of a matrix must be known a priori. Without extensive knowledge of the data being compressed, the accuracy of a decomposition generated via SBR-SVD cannot be known without revisiting the entire matrix following compression. If the numerical rank is not known in advance, the algorithm must be augmented to make more passes over the input matrix.*

2.7. Computational complexity and storage comparison

Table 1 provides the computational complexity, disk storage, and RAM usage required by the SBR-SVD, full ID, sub-sampled ID, and the single-pass ID algorithm with input matrix of size $m \times n$ and target rank k. Recall that $l = k + p$ is slightly larger than the target rank k by the oversampling parameter p, e.g., p between 10 and 20. In addition, the size of the sub-sampled matrix is considerably smaller than that of the full data matrix, i.e., $n_c \ll n$. t is defined to be the size of the stencil of the interpolation scheme used in to map the coarse grid data onto the fine grid. RAM usage and disk storage are given as the total number of matrix entries stored during and following the execution of the algorithm, respectively.

Method	Computational complexity	Disk storage	RAM usage
SBR-SVD (Algorithm 2)	$\mathcal{O}(mnk)$	$k(m+n)$	$l(m+2n)$
Full ID (Algorithm 3)	$\mathcal{O}(mnk)$	$k(m+n)$	$k(m+n)+mn$
Randomized ID (Algorithm 4)	$\mathcal{O}(mnk)$	$k(m+n)$	$k(m+n_c)+mn_c$
Sub-sampled ID (Algorithm 5)	$\mathcal{O}(mn_ck)$	$k(m+n)$	$k(m+n_c)+mn_c$
Single-pass ID (Algorithm 6)	$\mathcal{O}(mn_ck)$	$k(m+n_c)+tn$	$k(m+n_c)+mn_c+tn$

Table 1: Computational complexity of the SBR-SVD, full ID, sub-sampled ID and single-pass ID with input matrix of size $m \times n$ and target rank k. Variable $l \approx k$ in column 3 is the sub-sampled dimension of the matrix in SBR-SVD. Variable $n_c \ll n$ represents the column dimension of the input matrix after sub-sampling. Here, t is the size of the stencil for the interpolation scheme used in mapping the coarse grid onto the fine grid, e.g., $t = 2$ for piece-wise linear interpolation in one dimension. Disk storage is given as the total number of matrix entries stored following execution of the algorithm.

3. Numerical experiments

The compression efficiency and reconstruction accuracy of SBR-SVD, ID, sub-sampled ID, and single-pass ID methods are investigated in compressing data extracted from the DNS of wall-bounded turbulent flows using the Soleil-MPI low-Mach-number flow solver [78]. In particular, the canonical periodic channel

flow at friction Reynolds number $Re_\tau = 180$ is selected for two test cases: single-phase turbulence (Figure 2) and particle-laden flow with Stokes numbers $St^+ = 0, 1, 10$ (Figure 3). As is customary, $Re_\tau = u_\tau \delta/\nu$, where u_τ is the friction velocity, δ is the channel half-height, and ν is the kinematic viscosity of the fluid; $\nu = \mu/\rho$ with μ the dynamic viscosity and ρ the density. The mass flow rate is determined through a mean stream-wise pressure gradient $\langle dp/dx \rangle = -\tau_w/\delta$, where p is the pressure and $\tau_w = \rho\nu \left(d\langle u \rangle/dy\right)_{y=0} = \rho u_\tau^2$ is the wall shear stress, with $\langle u \rangle$ the mean stream-wise velocity.

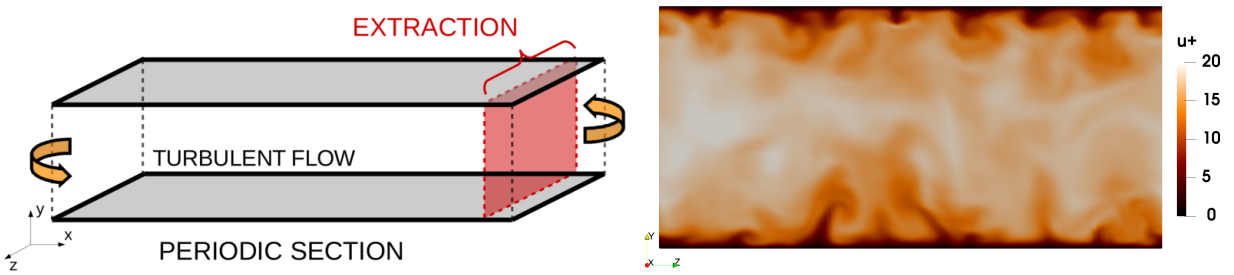

Figure 2: Left: Data extraction computational setup for compression and reconstruction. Right: Stream-wise velocity (wall units) on the extraction plane (instantaneous, span-wise y-z plane snapshot).

In wall-bounded particle-laden turbulent flows, preferential concentration of the disperse phase — inertial particles are expelled from intense vortical structures and concentrate in regions of the flow dominated by strain — is characterized by the viscous Stokes number $St^+ = \tau_p/\tau_f$, defined as the ratio between particle relaxation, $\tau_p = \rho_p d_p^2/(18\rho\nu)$ with ρ_p the particle density and d_p its diameter, and flow, $\tau_f = \nu/u_\tau^2$, time scales. For the three St^+ numbers considered, the flow is laden with $200,000$ particles resulting in a dilute mixture, i.e., one-way coupling with no particle-particle collisions. The particle sizes are several orders of magnitude smaller than the smallest (i.e., Kolmogorov) flow scales, and the density ratio between particles and fluid is $\rho_p/\rho \gg 1$. As a result, particles are modeled following a Lagrangian point-particle (PP) approach with Stokes' drag as the most important force [79]. A detailed description of the physics modeling and mathematical formulation can be found in [4, 5, 6]. The computational domain is $4\pi\delta \times 2\delta \times 4/3\pi\delta$ in the stream-wise (x), vertical (y), and span-wise (z) directions, respectively. The stream-wise and span-wise boundaries are set periodic, and no-slip conditions are imposed on the horizontal boundaries (x-z planes). The grid is uniform in the stream-wise and span-wise directions with spacings in wall units equal to $\Delta x^+ = 9$ and $\Delta z^+ = 6$, and stretched toward the walls in the vertical direction with the first grid point at $y^+ = y u_\tau/\nu = 0.1$ and with resolutions in the range $0.1 < \Delta y^+ < 8$. This grid arrangement corresponds to a DNS of size $256 \times 128 \times 128$ grid-points.

The problem under study is illustrated in Figure 3. The simulation strategy starts from a sinusoidal velocity field, seeded with randomly distributed particles, which is advanced in time to reach turbulent steady-state conditions after several FTTs. Based on the bulk velocity, $u_b = 1/\delta \int_0^\delta \langle u \rangle \, dy$, and the length of the channel, $L = 4\pi\delta$, the FTT is defined as $t_b = L/u_b$. Once a sufficiently long transient period is surpassed — approximately 10 eddy-turnover times, $t_l \sim \delta/u_\tau$ — the temporal evolution of the velocity field at the outlet plane (130×130 grid: 128 inner + 2 boundary points) is extracted for an entire FTT resulting in $25,100$ time snapshots. Similarly, disperse-phase data, viz. time-dependent positions and velocities, for all $200,000$ particles are collected over $10,000$ time steps. These two sets of data are utilized to investigate the performance of the compression methods.

3.1. Test case 1: Outlet flow data compression

The u (stream-wise), v (wall-normal), and w (span-wise) velocity fields on the 130×130 grid slice are captured over $25,100$ time steps to form the data to be used in the numerical experiments in this section. The four methods utilized for the compression of this flow data are SBR-SVD, ID, sub-sampled ID, and single-pass ID. In all test cases, the interpolation step in single-pass ID is taken to be a piece-wise linear interpolation scheme, though other approaches, e.g., spline interpolation, may be used in its place.

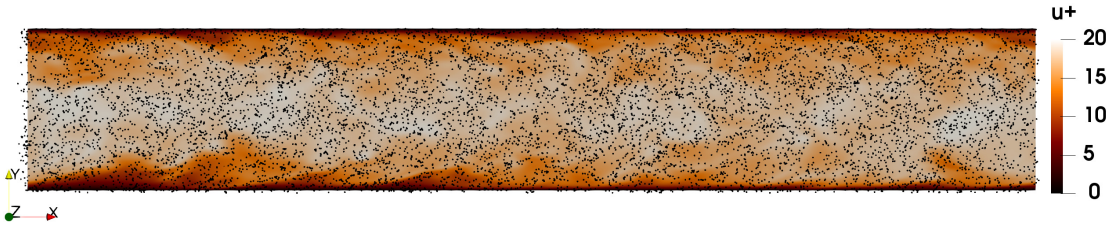

Figure 3: Particle-laden turbulent flow in a periodic channel (instantaneous, stream-wise x-y plane snapshot). The quantity represented is stream-wise velocity (wall units) of the fluid phase with particles colored in black. The friction Reynolds number is $Re_\tau = 180$ and the viscous Stokes number is $St^+ = 1$.

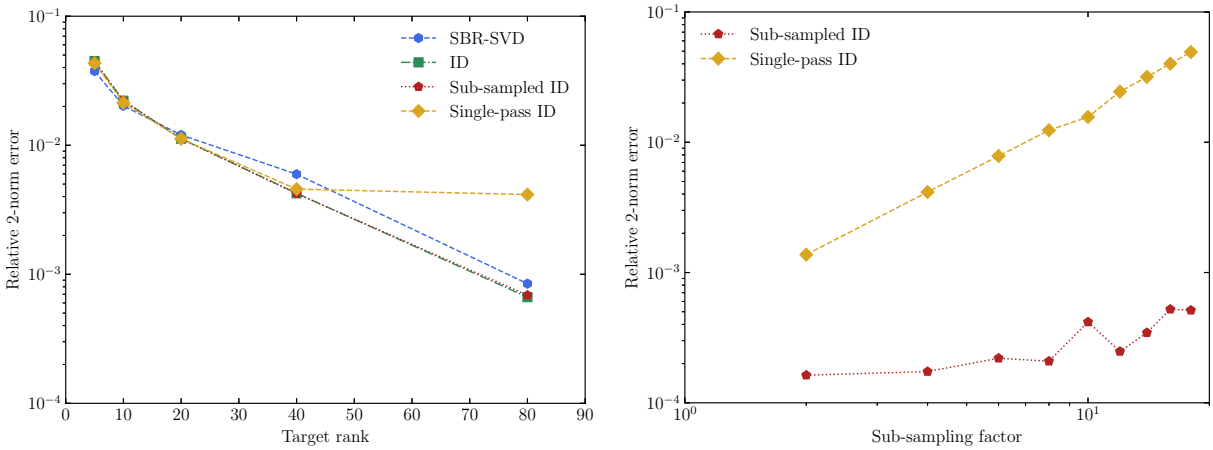

Figure 4: Left: Relative 2-norm error of the four methods for different target ranks for stream-wise velocity data from a turbulent channel flow at $Re_\tau = 180$. Right: Relative 2-norm error of single-pass ID and sub-sampled ID with respect to subsampling factor for fixed rank $k = 100$.

In the left panel of Figure 4, accuracy results from an experiment performed on u-velocity data collected from the channel flow configuration at 2,600 spatial grid points (sampled from the full 16,900 in the original data set) over 500 time steps (sampled from the full 25,100 in the original data set) are shown. The results demonstrate that ID and sub-sampled ID are the most accurate methods as the target rank is increased. SBR-SVD performs almost as well, while the error of single-pass ID saturates after the target rank surpasses 40 due to the interpolation error $\|\boldsymbol{E}_I\|_2$ in Theorem 2.2 not decreasing as the target rank is increased. For target rank values 10 and 20, single-pass ID achieves accuracy on par with the other methods. Due to the nearly indistinguishable performances of the sub-sampled ID and ID algorithms, comparisons against randomized ID are omitted in this test problem. As ID involves no randomization nor sketching of any kind, it is reasonable to expect that any randomized ID algorithm would invariably perform at best the same as ID.

In the right panel of Figure 4, the errors of single-pass and sub-sampled ID are shown for a range of sub-sampling parameter values. Results clearly indicate that single-pass ID does not match up well to sub-sampled ID in this test case due to the large target rank of 100. The plot is on log-log scale, which also indicates that the rate of convergence of single-pass ID is greater than that of sub-sampled ID with respect to the sub-sampling parameter n/n_c. In particular, single-pass ID exhibits a convergence rate of $\mathcal{O}(H^2)$, where H is the grid size used in the piece-wise linear interpolation step of single-pass ID.

In Table 2, the speed-up of sub-sampled ID, single-pass ID, and SBR-SVD are reported for different compression factors, where the speedup of each algorithm is defined as

$$\text{speedup} := \frac{\text{runtime of slowest algorithm}}{\text{runtime of algorithm}}, \tag{22}$$

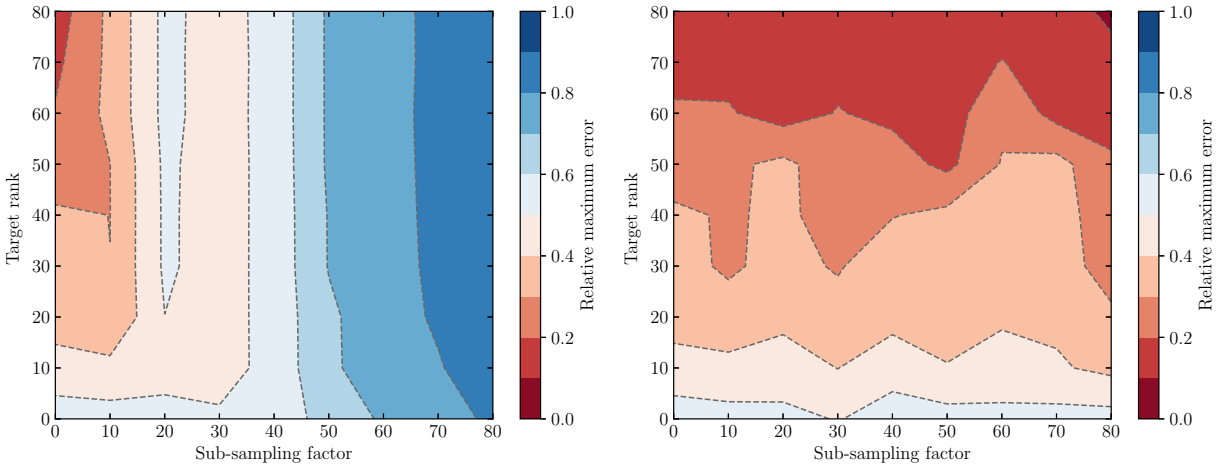

Figure 5: Relative maximum error as function of sub-sampling factor and target rank. Data corresponds to outflow stream-wise velocity, u, extracted from channel flow at $Re_\tau = 180$. Left: Single-pass ID. Right: Sub-sampled ID.

Temporal compression factor (m/k)	SBR-SVD	Sub-sampled ID	Single-pass ID without (and with) interpolation
200	8.3	19.6	19.6 (5.4)
100	16.4	41.7	41.7 (8.1)
20	30.3	55.6	55.6 (13.2)

Table 2: Speedup in runtime using the SBR-SVD, sub-sampled ID, and single-pass ID relative to that of the full ID algorithm for compressing outlet stream-wise velocity data from channel flow at $Re_\tau = 180$. The cost of second pass over the data is ignored for Sub-sampled ID. The results reported in the parentheses include the interpolation cost of single-pass ID needed only for data reconstruction.

and the compression factor is generally defined as

$$\text{compression factor} := \frac{\text{number of matrix entries in original data}}{\text{number of matrix entries in compressed data}}, \tag{23}$$

while the temporal compression factor reported in Table 2 is

$$\text{temporal compression factor} := \frac{\text{number of rows in original data}}{\text{target rank of approximation}}. \tag{24}$$

Examining the table, sub-sampled ID and single-pass ID (without interpolation) are by far the fastest of the methods considered; they compresses a matrix roughly 60× faster than ID and almost twice as fast as SBR-SVD in the lower compression regimes. Single-pass ID with the interpolation step included, i.e., multiplication by $\boldsymbol{M}$, offers the smallest speedup in all three cases. However, in the context of *in situ* compression of simulation data, the interpolation step is an *a posteriori* operation in that it is needed when reconstructing the data. If the interpolation step is ignored in this study, the runtimes of the sub-sampled ID and single-pass ID are identical. While the cost of the runtimes of single-pass ID (without interpolation) and sub-sample ID are the same, we remind that sub-sampled ID require an entire simulation to be run again to extract fine-grid row-skeleton snapshots. Our experiment does not take this into account; if a simulation takes days to complete while interpolation takes minutes, the computational savings of single-pass ID over sub-sampled ID will be significant. Note that the difference in performance time between the four methods increases as the compression factor is decreased.

In Figure 5, the relationship between the sub-sampling parameter, target rank, and relative entry-wise error of single-pass ID and sub-sampled ID is shown. For a data matrix $\boldsymbol{A}$ and its approximation $\hat{\boldsymbol{A}}$, the

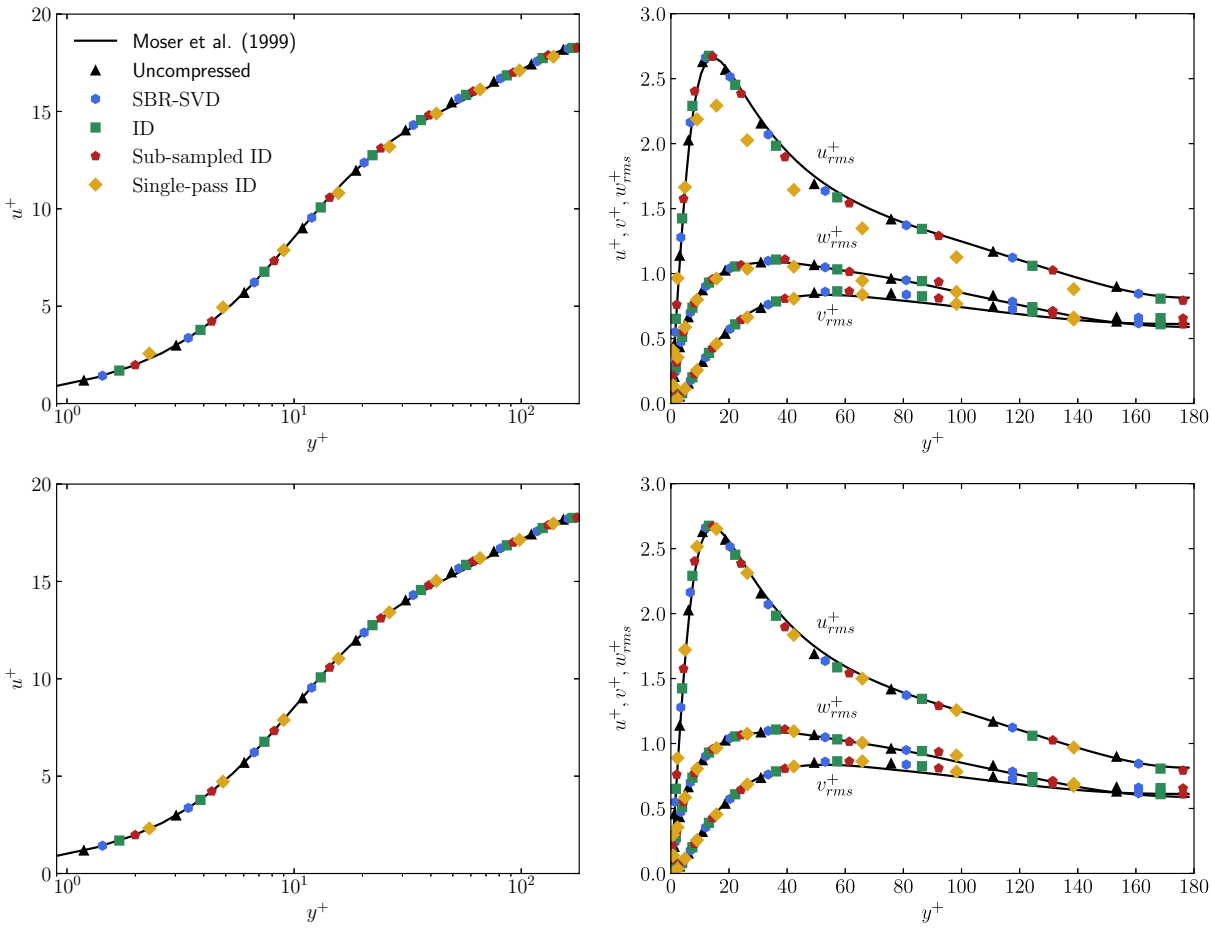

Figure 6: Left column: Mean stream-wise velocity profile (wall units). Right column: Root mean square velocity fluctuations (wall units). Top row: sub-sampling parameter of 12. Bottom row: sub-sampling parameter of 4. Moser et al. DNS [80] (black solid lines), uncompressed (black triangles), SBR-SVD (blue hexagons), full ID (green squares), sub-sampled ID (red pentagons) and single-pass ID (yellow diamonds).

relative entry-wise error is defined as

$$\max_{i,j} \frac{|\boldsymbol{A}_{i,j} - \hat{\boldsymbol{A}}_{i,j}|}{|\boldsymbol{A}_{i,j}|}. \tag{25}$$

The data matrix selected for the test is a $5,000 \times 16,900$ u-velocity outflow data obtained via sub-sampling time realization of the full-scale channel flow data. The plots show that the error of the single-pass method is heavily dependent on the sub-sampling parameter, which is reflected in the vertical stripe contours in the plot. The error of the sub-sampled ID, on the other hand, depends primarily on the target rank of the approximation, viz. the horizontal stripe contours in the right panel of Figure 5.

To investigate the impact of the accuracy loss, we use the compressed data (with 25,100 time slices) to generate inflow velocity and calculate time- and space-averaged (x and z directions) first- and second-order flow statistics. The quantities of interest considered are the numerical friction Reynolds number, Re_τ, the numerical bulk Reynolds number, $Re_b = 2u_b\delta/\nu$, the numerical skin-friction coefficient, $C_f = \tau_w/(1/2\rho u_b^2)$, the mean stream-wise velocity profile, $u^+ = \langle u \rangle / u_\tau$, and the root mean square (rms) velocity fluctuations, u^+_{rms}, v^+_{rms}, and w^+_{rms}. Reference solutions for all these QoIs are available in the literature. For instance, Re_b can be analytically approximated from $Re_\tau \approx 0.09 Re_b^{0.88}$ [81]. Once Re_b is known, C_f is directly obtained by calculating u_b from the bulk Reynolds number definition and noticing that τ_w and ρ depend on $\langle dp/dx \rangle$

	Analytical	Uncompressed	SBR-SVD	Full ID	Sub-sampled ID	Single-pass ID
Re_τ	180	180.3	180.4	180.3	180.3	180.3
Re_b	≈ 5600	5569.4	5569.5	5569.4	5569.4	5569.4
C_f	$\approx 8.2 \cdot 10^{-3}$	$8.3 \cdot 10^{-3}$	$8.3 \cdot 10^{-3}$	$8.3 \cdot 10^{-3}$	$8.3 \cdot 10^{-3}$	$8.3 \cdot 10^{-3}$

Table 3: Friction, Re_τ, and bulk, Re_b, Reynolds numbers, and skin-friction coefficient, C_f, of channel flow at $Re_\tau = 180$ obtained from uncompressed and compressed data. The compressed data has a compression factor of approximately 150 for all the methods considered.

and Re_τ. Regarding mean and fluctuation velocity profiles, reference DNS data are widely available, for example, from Moser et al. [80].

Values computed from compressed data, as well as reference and uncompressed solutions, are shown in Table 3 and Figures 6 and 7, with the compressed data requiring approximately 150× less storage than the original. Examining the results shown in Table 3 and Figures 6 and 7, the first observation is that the uncompressed and compressed results agree well with the analytical and DNS reference data. The second observation is that the four strategies considered provide similar compression accuracies — evaluated through the reconstructed velocity field — and are in agreement with the uncompressed data values. In particular, the relative errors for the quantities in Table 3 are below $6 \cdot 10^{-3}$ %, and the mean stream-wise velocity and rms fluctuations accurately reconstruct the uncompressed data solution. These results indicate that in applications involving spatially developing turbulent wall-bounded flow, this compressed data can be reused as inflow in subsequent simulations. The disk memory required to store inflows in these simulations can be reduced by factors exceeding 100, thereby placing less strain on the memory and I/O of the system in use.

In Figure 6, the reconstruction of the mean stream-wise velocity and root-mean-squared (rms) velocity fluctuations using the four methods described in Section 2 is shown. In the top two panels a sub-sampling factor of 12 in single-pass ID is used, whereas the bottom two panels correspond to a sub-sampling factor of 4. The results clearly demonstrate that the mean stream-wise velocity profile can be recovered in light of substantial interpolation error in single-pass ID. On the other hand, the rms velocity fluctuations, specifically u^+_{rms}, are much more sensitive to the reconstruction accuracy achieved by the methods. This highlights the delicate balance between interpolation error and computational performance inherent to the single-pass ID, as well as its application dependence.

In Figure 7 the same statistics as those presented in Figure 6 are shown for three different compression factor values. In this series of tests, the sub-sampling parameter used in the single-pass ID is set to 4 in all cases, as is presented in the bottom panels of Figure 6. The compression factor values chosen are 240 (bottom panels), 420 (middle panels) and 840 (top panels). The experiment demonstrates that all four methods can recover first- and second-order statistics to within reasonable accuracy while achieving compression factors exceeding 400. Beyond these values, the accuracy of the statistics recovered from the compressed data begins to drop off significantly. Noteworthy is the performance of the single-pass ID, which performs as well as the other three methods presented, indicating its utility as an online data compression algorithm.

3.2. Test case 2: Volumetric flow data compression

The volumetric, stream-wise velocity u field of the flow described in Section 3.1 is captured over 125 time steps to form the dataset of this experiment. The u-velocity snapshots are collected for the entire volume, $(256+2) \times (128+2) \times (128+2)$ 3D spatial mesh (4,360,200 grid points), at each time level resulting in a total $125 \times 4,360,200$ data matrix. The efficiency and accuracy of sub-sampled ID, SBR-SVD, and single-pass ID are analyzed for this test problem. Given the scale of the data matrix, analysis of the performance of the ID algorithm is omitted due to memory overflow on the machine used for the computations.

Figure 8 presents a 3D (volumetric) visualization of the turbulent flow at a single time instance from the original simulation, as well as three reconstructed using SBR-SVD, sub-sampled ID, and single-pass ID. All three methods provide a qualitatively accurate reconstruction of the full simulation, though single-pass ID presents some noticeably different features at the small-scale level (e.g., near-wall turbulent structures) compared to those in the other three velocity fields. One should note, however, that the single-pass ID

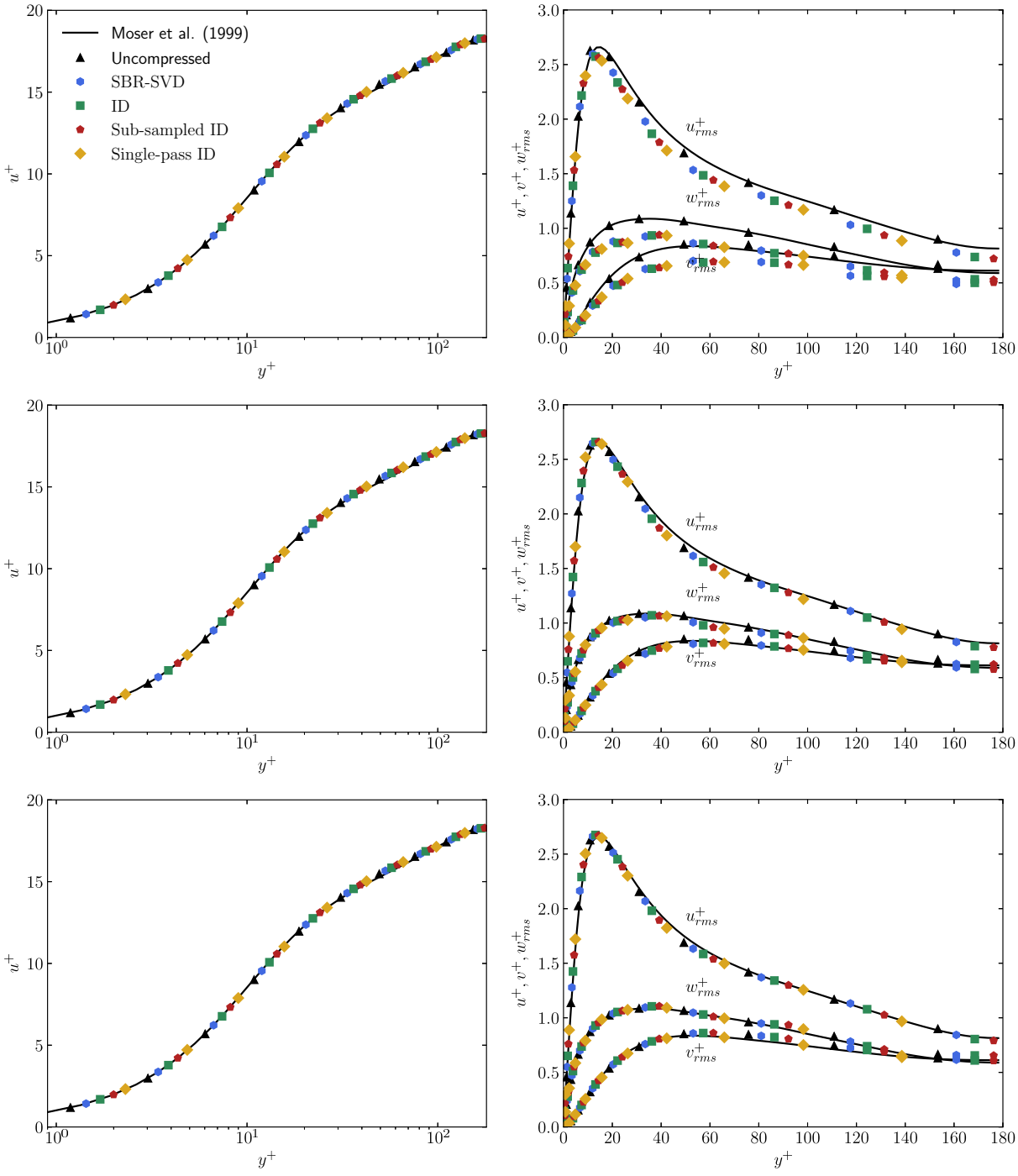

Figure 7: Left column: Mean stream-wise velocity profile (wall units). Right column: Root mean square velocity fluctuations (wall units). Top row: target rank of 20 (CF = 840). Center row: target rank of 40 (CF = 420). Bottom row: target rank of 70 (CF = 240). Moser et al. DNS [80] (black solid lines), uncompressed (black triangles), SBR-SVD (blue hexagons), full ID (green squares), sub-sampled ID (red pentagons) and single-pass ID (yellow diamonds).

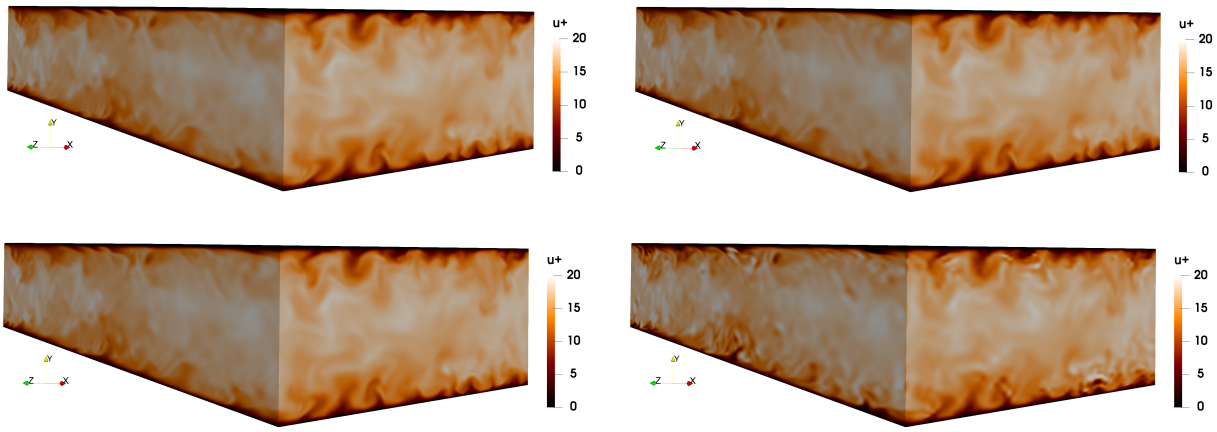

Figure 8: Instantaneous snapshots of channel flow stream-wise velocity u from the uncompressed data (top left), reconstructed using SBR-SVD (top right), reconstructed using sub-sampled ID (bottom left), and reconstructed using single-pass ID (bottom right).

reconstruction is achieved with a much higher compression factor — 8 times higher to be precise — as the method compresses not only the time domain of the data but the spatial domain as well.

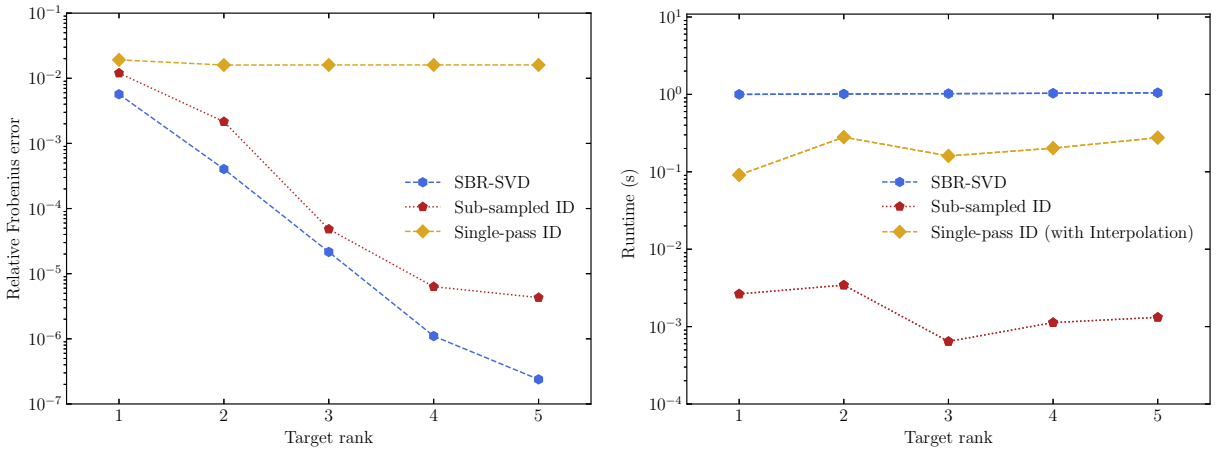

Figure 9: Left: Errors of three compression methods on volumetric data versus target rank. Right: Runtimes (normalized to the slowest trial of the SBR-SVD) of the three schemes versus target rank. In the right panel, the red line corresponds to the runtimes for both sub-sampled ID and single-pass ID *without* interpolation, while the yellow line reports the runtime of single-pass ID including the interpolation step.

The accuracy and runtime for the different methods are compared in Figure 9. Runtime is reported as the average over 20 independent trials per target rank value, and the oversampling parameter for SBR-SVD p is set to 20. In single-pass ID, each dimension is sub-sampled by a factor of 2, leading to a spatial compression factor of 8, which compounds on the temporal compression factor achieved via low-rank approximation. The rank $k = 1$ approximation for all three methods is roughly equally effective. For higher rank values, sub-sampled ID and SBR-SVD achieve far superior accuracy. Examining the singular vector corresponding to the largest singular value of the data matrix shown in Figure 10, it can be seen that the dominant modes in the system are significantly complex in the spatial dimension. This ensures that the interpolation step in single-pass ID will cause significant loss of accuracy, even at the highest mode.

Runtime performances are analyzed relative to the slowest trial of SBR-SVD. As depicted in Figure 9, sub-sampled ID is the fastest across all target rank values, further demonstrating its efficacy as a compression

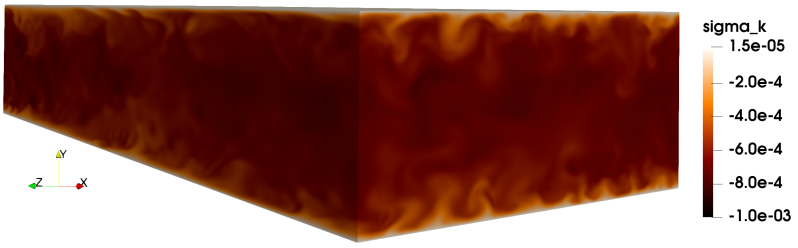

Figure 10: Singular vector associated with the largest singular value of the data matrix in Test case 2 computed via SBR-SVD.

method for turbulent flow data. SBR-SVD is invariably the slowest of the three. Notably, single-pass ID is faster than SBR-SVD for all target ranks, even including the interpolation cost. This demonstrates that with sufficient optimizations, namely the incorporation of random projection prior to the column selection set, the additional cost of the interpolation step will not necessarily make single-pass ID slower than its SVD counterpart.

Though sub-sampled ID is clearly more accurate than single-pass ID, it is important to remember that in an application where a simulation may take days to complete, or in which the data generated from such a simulation cannot be stored in RAM, a single-pass method, relative to a double-pass counterpart (e.g., sub-sampled ID) will considerably reduce the computational cost of simulation or loading time. Thus the trade-off in accuracy, as well as the additional runtime to interpolate back onto the full mesh —negligible relative to the cost of a high-fidelity simulation of a turbulent flow— may be worth it to practitioners faced with the dilemma of balancing accuracy and computational cost.

3.3. Test case 3: Particle data compression efficiency and accuracy

Wall-normal velocity, v, for $200,000$ particles traced over 10,000 time-steps is selected as test data. Three cases are considered in this section, corresponding to Stokes numbers $St^+ = 0, 1, 10$. In particle-laden turbulent flows, the Stokes number characterizes the particle response time with respect to the flow time-scale. Particles follow the turbulent motions at small St^+, whereas inertial forces dominate particle trajectories at large St^+. As a result, particles at high Stokes numbers are less prone to multi-scale behavior, which leads to enhanced compression accuracy for the algorithms.

SBR-SVD, ID, randomized ID (Algorithm 4), and sub-sampled ID are applied to the individually traced particle data extracted from a particle-laden turbulent flow. In the randomized ID, random projection via multiplication by a matrix with i.i.d. Gaussian entries is used as opposed to the deterministic sub-sampling step in sub-sampled ID to reduce the dimension of the input data (see Step 4 in Algorithm 4). Single-pass ID is not utilized in this section because the individually traced particle data lacks an associated mesh for interpolation, which single-pass ID requires.

In Table 4, compression error and runtime results are reported for the four methods for y-position data of $5,000$ particles tracked over $10,000$ time-steps. In all three Stokes regimes, SBR-SVD is the fastest and most accurate. ID is the slowest, while the second most accurate, and randomized ID and sub-sampled ID are the least accurate but achieve runtimes close to those of SBR-SVD. Notice that in Algorithm 5 for sub-sampled ID, the standard ID is performed on the sketch matrix $\boldsymbol{A}_c$ (Step 4). We anticipate the runtime of the sub-sampled ID presented in Table 4 and in the subsequent results will be improved by performing randomized ID on $\boldsymbol{A}_c$ instead. Moreover, randomized ID and sub-sampled ID achieve similar accuracy, except in the $St^+ = 10$ case, where randomized ID is more accurate. In Table 5, the same metrics are applied to access the compression performance of the algorithms on v-velocity data collected from the same $5,000$ particles as in Table 4. The same relative performances are observed as before; the performance gap between sub-sampled ID and the other three methods is particularly pronounced when $St^+ = 10$. Again, SBR-SVD is the fastest and most accurate method, while ID is by far the slowest.

In Figure 11, accuracy and runtime (normalized by the largest value across all algorithms and test cases) of SBR-SVD, randomized ID, and sub-sampled ID algorithms are shown in two contour plots for the case

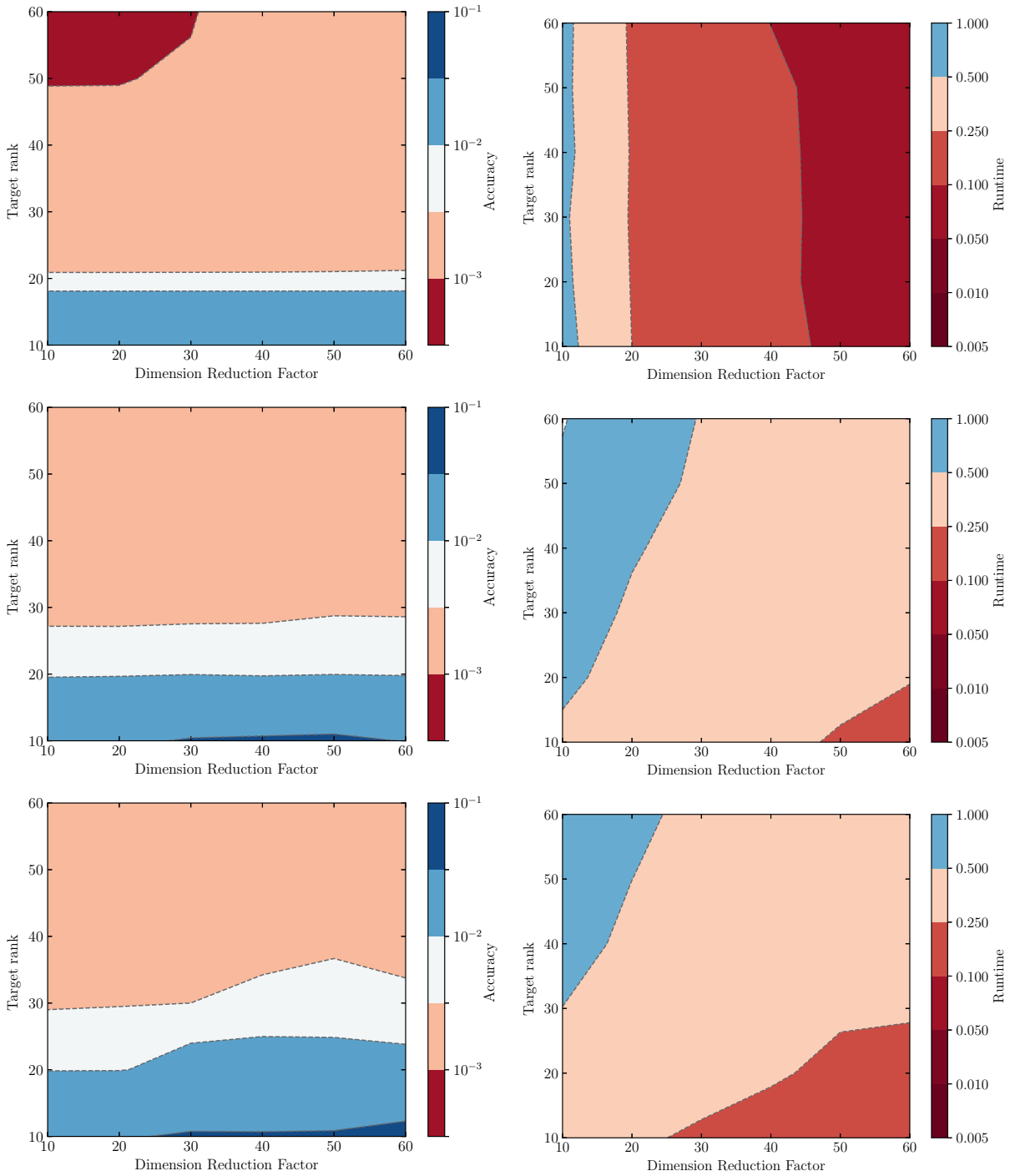

Figure 11: Accuracy (relative Frobenius error) and runtime (normalized by the largest value, i.e., slowest runtime: Randomized ID, dimension reduction factor of 10, target rank of 60) for individually traced particle v-velocity data at $St^+ = 1$. The dimension reduction factor is defined as the ratio n/l for SBR-SVD and Randomized ID, and n/n_c for sub-sampled ID. Top row: Accuracy (left) and runtime (right) of SBR-SVD. Center: Randomized ID. Bottom: Sub-sampled ID.

of $St^+ = 1$ with $5,000$ particles selected from the original $200,000$. The dimension reduction factor given

Stokes number	Method	Y-position data compression error	Runtime (s)
$St^+ = 0$	SBR-SVD	$3.6 \cdot 10^{-6}$	1.3
	ID	$4.6 \cdot 10^{-6}$	72.2
	Randomized ID	$6.0 \cdot 10^{-6}$	4.6
	Sub-sampled ID	$7.6 \cdot 10^{-6}$	4.3
$St^+ = 1$	SBR-SVD	$1.1 \cdot 10^{-6}$	1.2
	ID	$1.2 \cdot 10^{-6}$	73.3
	Randomized ID	$1.5 \cdot 10^{-6}$	5.1
	Sub-sampled ID	$2.3 \cdot 10^{-6}$	4.5
$St^+ = 10$	SBR-SVD	$2.6 \cdot 10^{-6}$	1.2
	ID	$2.7 \cdot 10^{-6}$	82.5
	Randomized ID	$3.4 \cdot 10^{-6}$	5.0
	Sub-sampled ID	$2.1 \cdot 10^{-5}$	4.4

Table 4: Compression error and runtime achieved by the SBR-SVD, ID, randomized ID, and sub-sampled ID for $St^+ = 0, 1, 10$ Y-position data for target rank 100 (temporal compression factor 100). Compression errors are computed for $10,000 \times 5,000$ matrices of the particles wall-normal position in terms of relative Frobenius norm error. Runtimes are given in seconds. In sub-sampled ID, a sub-sampling factor of $n/n_c = 20$ is used. In randomized ID and SBR-SVD, the matrix is projected to the same dimension as in sub-sampled ID, using a Gaussian sampling matrix instead. In SBR-SVD, the QR decomposition is computed in blocks of size 110.

Stokes number	Method	V-velocity data compression error	Runtime (s)
$St^+ = 0$	SBR-SVD	$4.9 \cdot 10^{-3}$	1.2
	ID	$6.0 \cdot 10^{-3}$	72.7
	Randomized ID	$7.7 \cdot 10^{-3}$	5.3
	Sub-sampled ID	$9.4 \cdot 10^{-3}$	4.5
$St^+ = 1$	SBR-SVD	$6.2 \cdot 10^{-4}$	1.3
	ID	$8.2 \cdot 10^{-4}$	73.4
	Randomized ID	$1.0 \cdot 10^{-3}$	4.7
	Sub-sampled ID	$1.7 \cdot 10^{-3}$	4.1
$St^+ = 10$	SBR-SVD	$6.9 \cdot 10^{-5}$	1.1
	ID	$9.6 \cdot 10^{-5}$	75.7
	Randomized ID	$1.2 \cdot 10^{-4}$	5.0
	Sub-sampled ID	$4.7 \cdot 10^{-3}$	4.4

Table 5: Compression error and runtime achieved by the SBR-SVD, ID, randomized ID, and sub-sampled ID for $St^+ = 0, 1, 10$ V-velocity data for target rank 100 (temporal compression factor 100). Compression errors are computed for $10,000 \times 5,000$ matrices of the particles v-velocities in terms of relative Frobenius norm error. Runtimes are given in seconds. In sub-sampled ID, a sub-sampling factor of $n/n_c = 20$ is used. In randomized ID and SBR-SVD, the matrix is projected to the same dimension as in sub-sampled ID, using a Gaussian sampling matrix instead. In SBR-SVD, the QR decomposition is computed in blocks of size 110.

as the horizontal axes of the plots is defined as the ratio n/l, for randomized ID and SBR-SVD, and n/n_c for sub-sampled ID. All three methods prove to be robust with respect to dimension reduction, which leads to savings in runtime with insignificant loss of accuracy. In this test case, SBR-SVD outperforms all other algorithms in terms of accuracy as well as in terms of runtime.

In Table 6, compression error and runtime results are reported for the four methods for y-position data of $20,000$ particles tracked over $10,000$ time-steps. In all three Stokes regimes, sub-sampled ID is the fastest of the four methods, while SBR-SVD is the most accurate. ID is by far the slowest, and the much faster randomized ID and sub-sampled ID are the least accurate. Randomized ID and sub-sampled ID achieve similar errors across Stokes regimes, with randomized ID slightly more accurate in each of the three test cases, in particular $St^+ = 10$. In Table 7, the same metrics are used to access the performance of the algorithm applied to the compression of v-velocity data collected from the same $20,000$ particles as in Table 6. In

this case, sub-sampled ID is again the fastest and SBR-SVD the most accurate of the three methods for all Stokes regimes. The performance gap between sub-sampled ID and the other schemes is again significant when $St^+ = 10$, as was also observed in the $5,000$ particle test case (see Table 5).

A key difference between the $5,000$ particle and $20,000$ particle test cases is that sub-sampled ID is faster than all three other methods in the $20,000$ particle case, while it is slower than SBR-SVD in the $5,000$ particle particle test case. This is due to the fact that in the $5,000$ particle case (considered in Tables 4 and 5), the spatial domain of the matrix is much smaller ($1/2$ the size of the temporal domain), limiting the speedup to be gained by sub-sampling the matrix – instead of using random projection – prior to compression. When the ratio of the number of particles to the number of timesteps is increased to 2 as in the $20,000$ particle test case reported in Tables 6 and 7, sub-sampled ID offers greater speedup relative to the other schemes. More generally, sub-sampled ID is ideally suited to PDE data in which the size of the spatial domain greatly exceeds that of the temporal domain, as was observed in the results of Test Case 2.

Stokes number	Method	Y-position data compression error	Runtime (s)
$St^+ = 0$	SBR-SVD	3.0×10^{-6}	4.3
	ID	3.9×10^{-6}	284.3
	Randomized ID	5.0×10^{-6}	5.9
	Sub-sampled ID	6.7×10^{-6}	4.0
$St^+ = 1$	SBR-SVD	5.8×10^{-7}	4.2
	ID	6.4×10^{-7}	319.2
	Randomized ID	8.2×10^{-7}	5.9
	Sub-sampled ID	1.0×10^{-6}	4.0
$St^+ = 10$	SBR-SVD	6.0×10^{-7}	4.3
	ID	6.2×10^{-7}	337.6
	Randomized ID	8.0×10^{-7}	5.9
	Sub-sampled ID	8.3×10^{-6}	4.0

Table 6: Compression error and runtime achieved by the SBR-SVD, ID, randomized ID, and sub-sampled ID for $St^+ = 0, 1, 10$ Y-position data for target rank 100 (temporal compression factor 100). Compression errors are computed for $10,000 \times 20,000$ matrices of the particles y-positions in terms of relative Frobenius norm error. Runtimes are given in seconds. In sub-sampled ID, a sub-sampling factor of $n/n_c = 80$ is used. In randomized ID and SBR-SVD, the matrix is projected to the same dimension as in sub-sampled ID, using a Gaussian sampling matrix instead. In SBR-SVD, the QR decomposition is computed in blocks of size 110.

Next, the compression accuracy of SBR-SVD, randomized ID, and sub-sampled ID applied to the reconstruction of time-averaged wall-normal particle concentrations and velocities for the full 200,000 particle data is investigated, depicted in Figure 12. The $10,000 \times 200,000$ matrix is first separated into blocks of size $10,000 \times 5,000$ and compressed individually due to our serial implementation of Algorithms 2, 4, and 5.

SBR-SVD, randomized ID and sub-sampled ID are the three methods considered in this case, with the target rank for each block set to $k = 100$ in all test cases. The simulation in question consists of 10,000 timesteps, therefore, all of the algorithms attain temporal compression factors of 100. The accuracy in compressing particle positions is quantified by the normalized concentration (with respect to the centerline of the channel), C/C_{cl}, while the compression of particles velocity, $v_p^+ = v_p/u_\tau$, is evaluated by means of scaled probability density functions (PDF) in the viscous sublayer ($y^+ < 5$), buffer layer ($5 < y^+ < 30$), and log-law region ($y^+ > 30$). The results shown in the plots demonstrate that SBR-SVD, randomized ID, and sub-sampled ID are able to compress the Lagrangian data without significant loss of reconstruction accuracy.

4. Conclusions

Inspired by growing issues in memory capacity and I/O emerging due to drastically increased floating-point performance on modern high-performance computers, pass-efficient matrix decomposition algorithms for the utility of compression of large-scale streaming PDE data are investigated in this work. Within

Stokes number	Method	V-velocity data compression error	Runtime (s)
$St^+ = 0$	SBR-SVD	5.1×10^{-3}	4.2
	ID	6.4×10^{-3}	331.4
	Randomized ID	8.2×10^{-3}	5.8
	Sub-sampled ID	1.0×10^{-2}	4.1
$St^+ = 1$	SBR-SVD	4.4×10^{-4}	4.6
	ID	5.9×10^{-4}	316.5
	Randomized ID	7.6×10^{-4}	6.4
	Sub-sampled ID	9.2×10^{-4}	4.4
$St^+ = 10$	SBR-SVD	1.4×10^{-4}	4.9
	ID	2.9×10^{-4}	344.1
	Randomized ID	3.9×10^{-4}	6.1
	Sub-sampled ID	7.3×10^{-3}	4.1

Table 7: Compression error and runtime achieved by the SBR-SVD, ID, randomized ID, and sub-sampled ID for $St^+ = 0, 1, 10$ V-velocity data for target rank 100 (temporal compression factor 100). Compression errors are computed for $10,000 \times 20,000$ matrices of the particles v-velocities in terms of relative Frobenius norm error. Runtimes are given in seconds. In sub-sampled ID, a sub-sampling factor of $n/n_c = 80$ is used. In randomized ID and SBR-SVD, the matrix is projected to the same dimension as in sub-sampled ID, using a Gaussian sampling matrix instead. In SBR-SVD, the QR decomposition is computed in blocks of size 110.

the broad field of matrix decomposition methods, this work focused on four algorithms: a single-pass randomized SVD-based algorithm (SBR-SVD), and three variants of interpolative decomposition, namely, randomized ID, (grid) sub-sampled ID, and (grid sub-sampled) single-pass ID. Prior to this work, all existing ID algorithms required at least two passes over the input data. The novel single-pass ID addresses this gap within this class of matrix factorization. At the core of the new ID methods, sub-sampled ID and single-pass ID, is the utility of a sketch based on coarse description of the data associated with the computational grid which enables a faster construction of ID. Bounds on the approximation error of these variants of ID are derived and insight into the role of the coarse grid sketch is presented.

On the empirical side, these low-rank factorization techniques are employed for the compression of two turbulent channel flows, one unladen and another laden with particles. Planar, volumetric, and discrete particle data are considered. In all cases the data admits low-rank representation and these methods are able to exploit this structure to achieve significant data compression. Among the one-pass (streaming) algorithms in the unladen turbulent channel flow test case, SBR-SVD is typically faster and more accurate than single-pass ID, but both are effective in the compression of the outlet velocity dataset as well as the volumetric dataset. Among the double-pass ID algorithms, sub-sampled ID performs as well as the ID in the recovery of flow statistics in the first two test cases, indicating promise for its application in other problems with similar low-rank structure and smooth spatial variation. SBR-SVD and sub-sampled ID feature significant reductions in runtime with insignificant losses in accuracy as compared to ID in all test cases. Individually traced particle data, lacking an associated mesh, remains an application to which single-pass ID is not adapted. In situations where the size of the spatial grid or the number of particles is large (relative to the number of time snapshots), we observe faster runtime achieved by sub-sampled ID as compared with randomized ID or SBR-SVD, thus justifying the utility of sub-sampled ID in high-dimensional data compression.

Despite their generally inferior performance in terms of runtime and accuracy on the three test cases considered in this work, the ID algorithms have key benefits over SBR-SVD. One is that all three variants of ID examined in this work may take as input an error tolerance. As a result, the three ID algorithms do not require *a priori* knowledge of the compressibility, i.e., the numerical rank, of a dataset. Work has been done recently to develop randomized SVD algorithms which adaptively determine the rank of a dataset [82], though these methods have yet to be developed in a single-pass framework to the best of the authors' knowledge. ID also generates an intuitive decomposition of a matrix, as it expresses the row space of a matrix in a basis of a subset of the rows themselves. This makes it a useful tool for domain experts who may

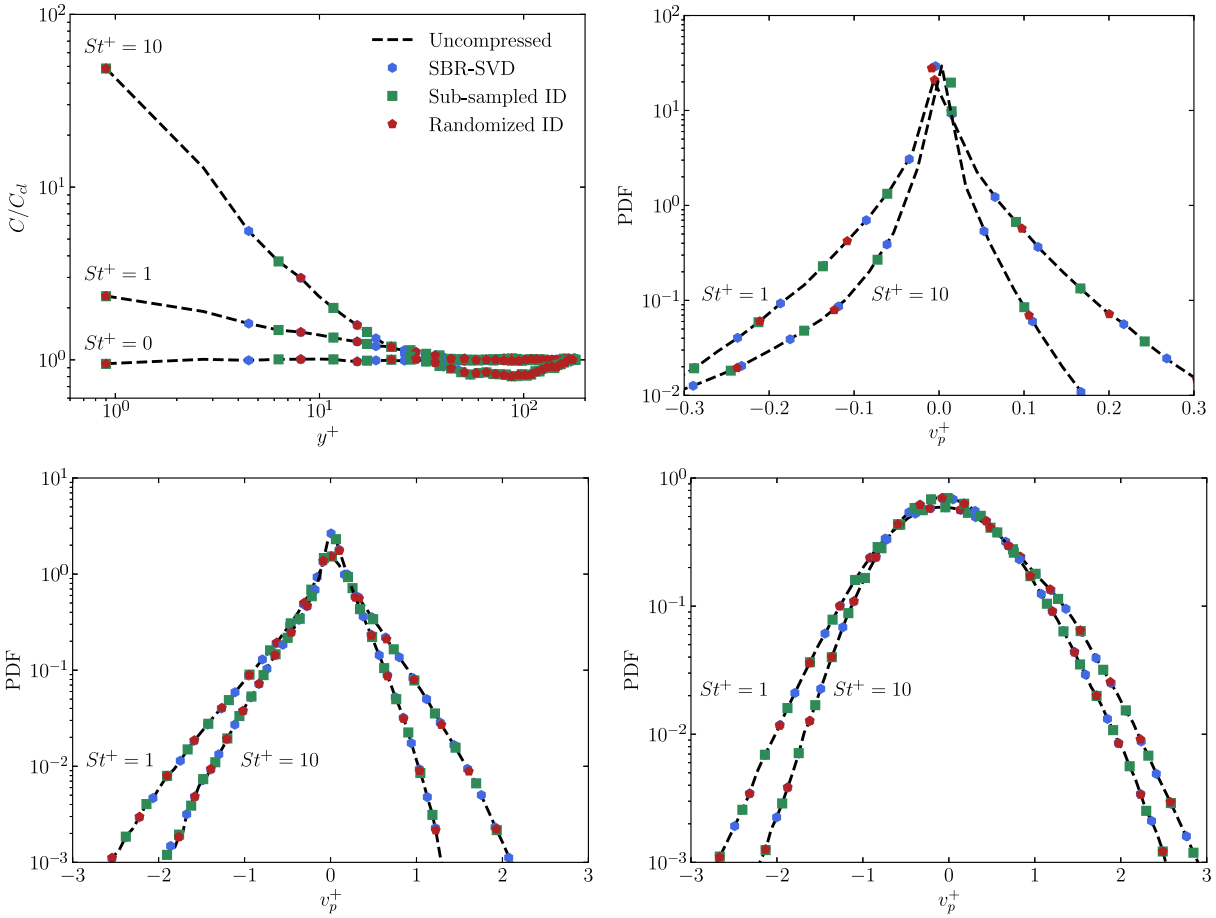

Figure 12: Top left: Steady-state mean normalized particle concentration in the wall-normal direction (wall-units). Top right: Scaled PDF of wall-normal particle velocity (wall-units) in the viscous sublayer ($y^+ < 5$). Bottom left: Scaled PDF of wall-normal particle velocity (wall-units) in the buffer layer ($5 < y^+ < 30$). Bottom right: Scaled PDF of wall-normal particle velocity (wall-units) in the log-law region ($y^+ > 30$). For SBR-SVD, sub-sampled ID, and randomized ID, results are recovered from compressed data with temporal compression factor 100, corresponding to a target rank of $k = 100$ used on each $10,000 \times 5,000$ matrix block.

find the SVD more difficult to interpret in the context of their raw data. For example, in a turbulent flow simulation whose time realizations arrive in RAM as row vectors, row ID can be interpreted as identifying the *most important* snapshots of the entire solution, in the sense that they form a near-optimal self-expressive basis to reconstruct the data. The SVD, on the other hand, does not provide exact snapshot reconstruction at any point in the temporal domain.

An ongoing extension of this work is the development of scalable, parallel implementations of these methods directly withing a flow solver, with the aim to enable *in situ* compression. We anticipate some of the observations made regarding the relative runtime of the compression algorithms considered here may change when executed in parallel. Other future avenues of research directions include the utility of i) higher order interpolation schemes (potentially on unstructured grids) to improve the accuracy of single-pass ID, and ii) spatially compressing the row skeleton generated by the ID using, e.g., compressed sensing algorithms [83, 84] to further enhance current compression factors.

Acknowledgments

This work was funded by the United States Department of Energy's National Nuclear Security Administration under the Predictive Science Academic Alliance Program (PSAAP) II at Stanford University, Grant DE-NA-0002373. A.D. acknowledges funding by the US Department of Energy's Office of Science Advanced Scientific Computing Research, Award DE-SC0006402, and National Science Foundation, Grant CMMI-145460. L.J. acknowledges funding by the Beatriz Galindo Program (Distinguished Researcher, BGP18/00026) of the Ministerio de Ciencia, Innovación y Universidades, Spain.

An award of computer time was provided by the ASCR Leadership Computing Challenge program. This research used resources of the Argonne Leadership Computing Facility, which is a Department of Energy's Office of Science User Facility supported under contract DE-AC02-06CH11357. This research also used resources of the Oak Ridge Leadership Computing Facility, which is a Department of Energy's Office of Science User Facility supported under contract DE-AC05-00OR22725. A.D. acknowledges the fruitful discussions with Prof. Kenneth Jansen and Stephen Becker from CU Boulder.

References

[1] G. E. Moore, Cramming more components onto integrated circuits, Electronics 38 (1965) 114–117.
[2] C. Walter, Kryder's law, Scientific American 293 (2005) 32–33.
[3] J. Ang, K. Evans, A. Geist, M. Heroux, P. Hovland, O. Marques, L. Curfman, E. Ng, S. Wild, Workshop on extreme-scale solvers: transition to future architectures, Technical Report, U.S. D.O.E., Office of Advanced Scientific Computing Research, 2012.
[4] L. Jofre, G. Geraci, H. R. Fairbanks, A. Doostan, G. Iaccarino, Multi-fidelity uncertainty quantification of irradiated particle-laden turbulence, CTR Annu. Res. Briefs (2017) 21–34.
[5] H. R. Fairbanks, L. Jofre, G. Geraci, G. Iaccarino, A. Doostan, Bi-fidelity approximation for uncertainty quantification and sensitivity analysis of irradiated particle-laden turbulence, J. Comp. Phys. 402 (2020) 108996.
[6] L. Jofre, Z. R. del Rosario, G. Iaccarino, Data-driven dimensional analysis of heat transfer in irradiated particle-laden turbulent flow, Int. J. Multiph. Flow 125 (2020) 103198.
[7] X. Wu, Inflow turbulence generation methods, Annu. Rev. Fluid Mech. 49 (2017) 23–49.
[8] S. Li, N. Marsaglia, C. Garth, J. Woodring, J. Clyne, H. Childs, Data reduction techniques for simulation, visualization and data analysis, in: Comput. Graphics Forum, Wiley Online Library.
[9] E. Engelson, D. Fritzson, P. Fritzson, Lossless compression of high-volume numerical data from simulations, in: Proc. Data Compress. Conf., pp. 574–586.
[10] P. Ratanaworabhan, J. Ke, M. Burtscher, Fast lossless compression of scientific floating-point data, in: Proc. Data Compress. Conf., pp. 1–10.
[11] J.-L. Gailly, M. Adler, The gzip home page, URL: http://www. gzip. org/(April 4, 2013) (2003).
[12] C. E. Shannon, A mathematical theory of communication, ACM SIGMOBILE Mob. Comput. Commun. Rev. 5 (2001) 3–55.
[13] D. A. Huffman, A method for the construction of minimum-redundancy codes, Proc. IRE 40 (1952) 1098–1101.
[14] J. Ziv, A. Lempel, A universal algorithm for sequential data compression, IEEE Trans. Inf. Theory 23 (1977) 337–343.
[15] J. Ziv, A. Lempel, Compression of individual sequences via variable-rate coding, IEEE Trans. Inf. Theory 24 (1978) 530–536.
[16] M. Burtscher, P. Ratanaworabhan, FPC: A high-speed compressor for double-precision floating-point data, IEEE Trans. Comput. 58 (2009) 18–31.
[17] Y. Sazeides, J. E. Smith, The predictability of data values, in: Proc. 30th Annu. ACM/IEEE Int. Symp. Microarchitecture, IEEE Computer Society, pp. 248–258.
[18] P. Lindstrom, M. Isenburg, Fast and efficient compression of floating-point data, IEEE Trans. Visual Comput. Graphics 12 (2006) 1245–1250.
[19] G. Strang, T. Nguyen, Wavelets and filter banks, SIAM, 1996.
[20] H. Lehmann, B. Jung, In-situ multi-resolution and temporal data compression for visual exploration of large-scale scientific simulations, in: Large Data Anal. Visual (LDAV), 2014 IEEE 4th Symp., IEEE, pp. 51–58.
[21] H. Lehmann, E. Werzner, C. Degenkolb, Optimizing in-situ data compression for large-scale scientific simulations, in: Proc. 24th High Performance Comput. Symp., Soc. Comput. Simul. Int., p. 5.
[22] X. Tong, T.-Y. Lee, H.-W. Shen, Salient time steps selection from large scale time-varying data sets with dynamic time warping, in: Large Data Anal. and Visual (LDAV), 2012 IEEE Symposium on, IEEE, pp. 49–56.
[23] W. Austin, G. Ballard, T. G. Kolda, Parallel tensor compression for large-scale scientific data, in: Parallel and Distrib. Process. Symp., 2016 IEEE Int., IEEE, pp. 912–922.
[24] K. Zhao, N. Sakamoto, K. Koyamada, Time-varying volume compression in spatio-temporal domain 1 (2015) 171–187.
[25] Z. Gong, T. Rogers, J. Jenkins, H. Kolla, S. Ethier, J. Chen, R. Ross, S. Klasky, N. F. Samatova, Mloc: Multi-level layout optimization framework for compressed scientific data exploration with heterogeneous access patterns, in: Parallel Process. (ICPP), 2012 41st Int. Conf.,, IEEE, pp. 239–248.

[26] J. E. Fowler, R. Yagel, Lossless compression of volume data, in: Proc. 1994 Symp. Volume Visual., ACM, pp. 43–50.
[27] S. Di, F. Cappello, Fast error-bounded lossy hpc data compression with sz, in: 2016 ieee international parallel and distributed processing symposium (ipdps), IEEE, pp. 730–739.
[28] D. Tao, S. Di, Z. Chen, F. Cappello, Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization, in: Parallel and Dist. Process. Symp. (IPDPS), 2017 IEEE Int., IEEE, pp. 1129–1139.
[29] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, F. Cappello, Error-controlled lossy compression optimized for high compression ratios of scientific datasets, in: 2018 IEEE International Conference on Big Data (Big Data), IEEE, pp. 438–447.
[30] L. Ibarria, P. Lindstrom, J. Rossignac, A. Szymczak, Out-of-core compression and decompression of large n-dimensional scalar fields, in: Comput. Graphics Forum, volume 22, Wiley Online Library, pp. 343–348.
[31] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Klasky, R. Latham, R. Ross, N. F. Samatova, Compressing the incompressible with ISABELA: In-situ reduction of spatio-temporal data, in: Euro. Conf. Parallel Process., Springer, pp. 366–379.
[32] E. Otero, R. Vinuesa, O. Marin, E. Laure, P. Schlatter, Lossy data compression effects on wall-bounded turbulence: bounds on data reduction, Flow, Turbul. and Combust. (2018) 1–23.
[33] O. Marin, M. Schanen, P. Fischer, Large-scale lossy data compression based on an a priori error estimator in a spectral element code, Technical Report, ANL/MCS-P6024-0616, 2016.
[34] P. Lindstrom, Fixed-rate compressed floating-point arrays, IEEE Trans. Visual Comput. Graphics 20 (2014) 2674–2683.
[35] M. Loeve, Probability theory, vol. ii, Graduate texts in mathematics 46 (1978) 0–387.
[36] F. L. Hitchcock, The expression of a tensor or a polyadic as a sum of products, Stud. Appl. Math. 6 (1927) 164–189.
[37] L. R. Tucker, Some mathematical notes on three-mode factor analysis, Psychometrika 31 (1966) 279–311.
[38] N. Vannieuwenhoven, R. Vandebril, K. Meerbergen, A new truncation strategy for the higher-order singular value decomposition, SIAM J. Sci. Comput. 34 (2012) A1027–A1052.
[39] L. De Lathauwer, B. De Moor, J. Vandewalle, On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors, SIAM J. Matrix Anal. Appl. 21 (2000) 1324–1342.
[40] P. M. Kroonenberg, J. De Leeuw, Principal component analysis of three-mode data by means of alternating least squares algorithms, Psychometrika 45 (1980) 69–97.
[41] N. Halko, P.-G. Martinsson, J. A. Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. 53 (2011) 217–288.
[42] W. Yu, Y. Gu, J. Li, S. Liu, Y. Li, Single-pass PCA of large high-dimensional data, arXiv preprint arXiv:1704.07669 (2017).
[43] H. Cheng, Z. Gimbutas, P.-G. Martinsson, V. Rokhlin, On the compression of low rank matrices, SIAM J. on Sci. Comput. 26 (2005) 1389–1404.
[44] M. Azaïez, L. Lestandi, T. C. Rebollo, Low rank approximation of multidimensional data, in: High-Performance Computing of Big Data for Turbulence and Combustion, Springer, 2019, pp. 187–250.
[45] M. Brand, Fast low-rank modifications of the thin singular value decomposition, Linear algebra and its applications 415 (2006) 20–30.
[46] R. Zimmermann, B. Peherstorfer, K. Willcox, Geometric subspace updates with applications to online adaptive nonlinear model reduction, SIAM Journal on Matrix Analysis and Applications 39 (2018) 234–261.
[47] J. A. Tropp, A. Yurtsever, M. Udell, V. Cevher, Streaming low-rank matrix approximation with an application to scientific simulation, arXiv preprint arXiv:1902.08651 (2019).
[48] K. Weiss, L. De Floriani, Simplex and diamond hierarchies: Models and applications, in: Comput. Graphics Forum, volume 30, Wiley Online Library, pp. 2127–2155.
[49] J.-L. Bourguignon, J. A. Tropp, A. S. Sharma, B. J. McKeon, Compact representation of wall-bounded turbulence using compressive sampling, Phys. Fluids 26 (2014) 015109.
[50] G. H. Golub, C. F. Van Loan, Matrix computations, JHU Press, 2012.
[51] M. Gu, S. C. Eisenstat, Efficient algorithms for computing a strong rank-revealing QR factorization, SIAM J. Sci. Comput. 17 (1996) 848–869.
[52] Y. P. Hong, C.-T. Pan, Rank-revealing QR factorizations and the singular value decomposition, Math. Comput. 58 (1992) 213–232.
[53] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (1936) 211–218.
[54] W. Kahan, Numerical linear algebra, Canadian Mathematical Bulletin 9 (1966) 757–801.
[55] P.-G. Martinsson, V. Rockhlin, M. Tygert, A randomized algorithm for the approximation of matrices, Technical Report, Yale Univ Dept. of Computer Science, 2006.
[56] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin, M. Tygert, Randomized algorithms for the low-rank approximation of matrices, PNAS 104 (2007) 20167–20172.
[57] P.-G. Martinsson, Randomized methods for matrix computations, The Mathematics of Data 25 (2018) 187–231.
[58] W. B. Johnson, J. Lindenstrauss, Extensions of lipschitz mappings into a hilbert space, Contemporary mathematics 26 (1984) 1.
[59] P.-G. Martinsson, S. Voronin, A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices, SIAM J. Sci. Comput. 38 (2016) S485–S507.
[60] F. Woolfe, E. Liberty, V. Rokhlin, M. Tygert, A fast randomized algorithm for the approximation of matrices, Appl. Comput. Harmon. Anal. 25 (2008) 335–366.
[61] V. Rokhlin, M. Tygert, A fast randomized algorithm for overdetermined linear least-squares regression, PNAS 105 (2008)

13212–13217.
[62] C. Boutsidis, D. P. Woodruff, P. Zhong, Optimal principal component analysis in distributed and streaming models, in: Proceedings of the forty-eighth annual ACM symposium on Theory of Computing, ACM, pp. 236–249.
[63] K. L. Clarkson, D. P. Woodruff, Numerical linear algebra in the streaming model, in: Proceedings of the forty-first annual ACM symposium on Theory of computing, ACM, pp. 205–214.
[64] J. A. Tropp, A. Yurtsever, M. Udell, V. Cevher, Practical sketching algorithms for low-rank matrix approximation, SIAM Journal on Matrix Analysis and Applications 38 (2017) 1454–1485.
[65] J. Upadhyay, Fast and space-optimal low-rank factorization in the streaming model with application in differential privacy, arXiv preprint arXiv:1604.01429 (2016).
[66] D. P. Woodruff, et al., Sketching as a tool for numerical linear algebra, Foundations and Trends® in Theoretical Computer Science 10 (2014) 1–157.
[67] M. W. Mahoney, P. Drineas, Cur matrix decompositions for improved data analysis, PNAS 106 (2009) 697–702.
[68] E. Elhamifar, R. Vidal, Sparse subspace clustering: Algorithm, theory, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 35 (2013) 2765–2781.
[69] E. L. Dyer, T. A. Goldstein, R. Patel, K. P. Kording, R. G. Baraniuk, Self-expressive decompositions for matrix approximation and clustering, arXiv preprint arXiv:1505.00824 (2015).
[70] P.-G. Martinsson, V. Rokhlin, M. Tygert, A randomized algorithm for the decomposition of matrices, Applied and Computational Harmonic Analysis 30 (2011) 47–68.
[71] A. Narayan, C. Gittelson, D. Xiu, A stochastic collocation algorithm with multifidelity models, SIAM J. Sci. Comp. 36 (2014) A495–A521.
[72] W. Hackbusch, Multi-grid methods and applications, volume 4, Springer Science & Business Media, 2013.
[73] A. Doostan, R. G. Ghanem, J. Red-Horse, Stochastic model reduction for chaos representations, Computer Methods in Applied Mechanics and Engineering 196 (2007) 3951–3966.
[74] A. Doostan, G. Geraci, G. Iaccarino, A bi-fidelity approach for uncertainty quantification of heat transfer in a rectangular ribbed channel, in: ASME Turbo Expo 2016: Turbomachinery Tech. Conf. and Expo., American Society of Mechanical Engineers.
[75] H. R. Fairbanks, A. Doostan, C. Ketelsen, G. Iaccarino, A low-rank control variate for multilevel Monte Carlo simulation of high-dimensional uncertain systems, J. Comp. Phys. 341 (2017) 121–139.
[76] J. Hampton, H. R. Fairbanks, A. Narayan, A. Doostan, Practical error bounds for a non-intrusive bi-fidelity approach to parametric/stochastic model reduction, J. Comput. Phys. 368 (2018) 315–332.
[77] S. Brenner, R. Scott, The mathematical theory of finite element methods, volume 15, Springer Science & Business Media, 2007.
[78] M. Esmaily, L. Jofre, A. Mani, G. Iaccarino, A scalable geometric multigrid solver for nonsymmetric elliptic systems with application to variable-density flows, J. Comput. Phys. 357 (2018) 142–158.
[79] M. R. Maxey, J. J. Riley, Equation of motion for a small rigid sphere in a nonuniform flow, Physics of Fluids 26 (1983) 883–889.
[80] R. D. Moser, J. Kim, N. N. Mansour, Direct numerical simulation of turbulent channel flow up to $Re_\tau = 590$, Phys. Fluids 11 (1999) 943–945.
[81] S. B. Pope, Turbulent flows, Cambridge University Press, 2000.
[82] H. Ji, W. Yu, Y. Li, A rank revealing randomized singular value decomposition (r3svd) algorithm for low-rank matrix approximations, arXiv preprint arXiv:1605.08134 (2016).
[83] R. G. Baraniuk, V. Cevher, M. F. Duarte, C. Hegde, Model-based compressive sensing, IEEE Trans. Inf. Theory 56 (2010) 1982–2001.
[84] S. Jokar, V. Mehrmann, M. E. Pfetsch, H. Yserentant, Sparse approximate solution of partial differential equations, Appl. Numer. Math. 60 (2010) 452–472.