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

By A. Dunton†, L. Jofre, A. Doostan† AND G. Iaccarino

1. Motivation and objectives

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 (1965) and Kryder's (Walter 2005) Laws, the different parts of supercomputers have improved extraordinarily over the past decades. However, memory capacity and bandwidth have failed to keep pace with the rate of generating data. This trend is not reverting and most likely will augment in the near future. For example, it is expected that the Exascale supercomputers (Ang *et al.* 2012) to be deployed during the next decade will provide 1-10k times increased floating-point performance but with a memory speed increase of only a factor of 100.

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. by temporal sub-sampling. This problem is of particular importance in the case of turbulent flows, as the spatial and time integration resolutions required to capture all the flow scales in Direct Numerical Simulation (DNS) increase exponentially with the Reynolds number; e.g., Lozano-Durán & Jiménez (2014), Ishihara *et al.* (2016). 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; e.g., Masquelet *et al.* (2017), Jofre *et al.* (2017*a*,*b*). The storage capacity and bandwidth limitations also complicate the applicability of time-decoupled strong recycling turbulence inflow methods (Wu 2017), 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-wounded flows, are stored to disk to be reused later as inflow in spatially developing flows.

If the prediction described above materializes, flow solvers will need to pursue new strategies in which the data size at each time slice and sub-sampling frequency are greatly reduced before writing to disk. Obviously, computational scientists prefer to perform visualization and analysis directly on raw data with minimal error resulting from observation as opposed to compression. However, as a result of the aforementioned limitations, the community will have to accept the compromises inherent in compression and adapt their

† University of Colorado at Boulder

numerical solvers accordingly. Several compression methods for mesh-based flow solvers have been investigated in the past. For instance, some of the most popular approaches are lossless compression (Burtscher *et al.* 2016), wavelet-type transforms (Farge 1992), and compressed sensing (Bourguignon *et al.* 2014).

In this work, alternative methods to these approaches are explored, particularly the interpolative decomposition (ID) and blocked single-pass singular value decomposition (SVD), where single-pass means the algorithm reads the input data once. More generally, pass-efficient algorithms are algorithms which limit the number of times the input data is read. These two pass-efficient methods generate approximations of high-dimensional matrices using a small fraction of the original matrix (Martinsson 2016; Yu *et al.* 2017). Their performance is analyzed based on their compression efficiency and reconstruction accuracy of three-dimensional (3-D) turbulent velocity fields for an incompressible fluid, denoted U, V, and W. Flow data for the fields U, V, and W are collected on a two-dimensional (2-D) grid slice in the y-z plane for several characteristic integral times requiring thousands of time-steps. The temporal, planar data are then reshaped into vectors and assembled into one single matrix. For example, the U, V, and W velocity fields for a channel flow at $Re_\tau = 180$ measured on a 130×130 grid slice are converted into row vectors of size 16900 and captured over 25100 time steps, resulting in 25100×16900 matrices. Then, the corresponding matrices $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{W}$ are vertically concatenated into a 50700×25100 matrix of the form

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{U}^\intercal \\ \boldsymbol{V}^\intercal \\ \boldsymbol{W}^\intercal \end{bmatrix}, \tag{1.1}$$

which is a high-dimensional matrix comprised of $\mathcal{O}(10^9)$ entries, and consequently an excellent candidate for the aforementioned decompositions.

In this research brief, compression methods for high-dimensional data are presented and a preliminary numerical investigation of their performance for a turbulent flow is discussed. The report is organized as follows. In Section 2, strategies for efficiently compressing turbulence data based on matrix decomposition approaches are summarized. Next, exploratory numerical results of their compression efficiency and reconstruction accuracy for turbulent channel flow are reported in Section 3. Finally, conclusions are drawn and future work is outlined in Section 4.

2. Decomposition methods for data compression

Data compression is the transformation of data into a format which requires fewer bits than the original representation, and can be divided into two main categories: lossless and lossy. Lossless data compression is more widely accepted in the scientific community. When using lossless compression methods, it is possible to perform an inverse transformation to recover the original data without loss of accuracy. However, their compression ratio is limited; e.g., Engelson *et al.* (2000), Ratanaworabhan *et al.* (2006). A higher compression ratio can be obtained by using lossy data compression algorithms, but at the expense that the inverse transformation produces at best an approximation of the original data. In this section, the lossy data compression algorithms investigated in this work are introduced. All the approaches discussed achieve a relative Frobenius norm error of order 10^{-3} using 1% or less of the original data for the numerical experiments considered. The relative Frobenius norm error, e_F, for an approximation $\boldsymbol{B}$ of a matrix $\boldsymbol{A}$ is measured via the quotient

$$e_F = \frac{\|\boldsymbol{A}-\boldsymbol{B}\|_F}{\|\boldsymbol{A}\|_F}, \quad \text{where} \quad \|\boldsymbol{A}\|_F = \sqrt{\sum_{i,j=1}^{n} \boldsymbol{A}_{ij}^2}, \tag{2.1}$$

and $\boldsymbol{A}_{ij}$ denotes the entry in the i^{th} row and j^{th} column of $\boldsymbol{A}$.

2.1. *Review of numerical QR and SVD implementations*

The ID and blocked single-pass SVD methods rely on two canonical matrix decompositions: the QR decomposition (QR) and the full, i.e., non-compressive SVD. The QR yields a factorization of a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ 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 $\boldsymbol{A}$ (Golub & Van Loan 2012). The three main approaches for computing this decomposition include pivoted Gram-Schmidt orthonormalization of the columns, Householder reflections, and Givens rotations (Golub & Van Loan 2012). Of particular interest is the rank-revealing QR algorithm, which relies on the pivoted Gram-Schmidt procedure (Gu & Eisenstat 1996), and thereby induces the concept of a numerical rank, which is a generalization of rank from linear algebra (Hong & Pan 1992). In this work, a matrix $\boldsymbol{A}$ is said to be of numerical-rank k —also referred to as ϵ-rank— for some $\epsilon > 0$ if there exists a matrix $\boldsymbol{A}_k$ of rank-k such that $\|\boldsymbol{A}-\boldsymbol{A}_k\|_F \leq \epsilon$ (Martinsson 2016). This concept lies at the center of the ID and blocked single-pass SVD.

Also crucial to the compression methods explored in this brief is the SVD. The SVD of a matrix is a decomposition of the form $\boldsymbol{A} = \boldsymbol{USV}^\intercal$, where $^\intercal$ denotes the transpose of a matrix. 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 $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively. The matrix $\boldsymbol{S} \in \mathbb{R}^{n\times n}$ is a diagonal matrix whose entries are the singular values of $\boldsymbol{A}$, which are precisely the square roots of the non-zero eigenvalues of $\boldsymbol{A}^\intercal\boldsymbol{A}$ or $\boldsymbol{AA}^\intercal$. Of interest in the problem of generating rank-k approximations of a matrix $\boldsymbol{A}$ is the truncated SVD, which yields a decomposition of the form $\boldsymbol{U_k S_k V_k^\intercal}$, with $\boldsymbol{U} \in \mathbb{R}^{m\times k}, \boldsymbol{V} \in \mathbb{R}^{n\times k}$ and $\boldsymbol{S} \in \mathbb{R}^{k\times k}$. 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 $\boldsymbol{A}$ corresponding to its k largest singular values, and $\boldsymbol{S}$ is once more a diagonal matrix containing the largest k singular values of $\boldsymbol{A}$. The product $\boldsymbol{B} = \boldsymbol{U_k S_k V_k^\intercal}$ forms a rank-k approximation of the matrix $\boldsymbol{A}$. Moreover, the truncated SVD is the theoretically best rank-k approximation of a matrix $\boldsymbol{A}$ in Frobenius norm (Eckart & Young 1936), i.e.,

$$\inf_{rank(\boldsymbol{B})=k} \|\boldsymbol{A}-\boldsymbol{B}\|_F = \|\boldsymbol{A}-\boldsymbol{U_k S_k V_k^\intercal}\|_F = \sigma_{k+1}. \tag{2.2}$$

A family of popular methods for generating rank-k SVD approximations are the randomized SVD algorithms (Halko *et al.* 2011). These methods approximate $\boldsymbol{A}$ following a general structure of the form $\boldsymbol{A} \approx \boldsymbol{U_k S_k V_k^\intercal}$. In the first step, a Gaussian matrix $\boldsymbol{\Omega} \in \mathbb{R}^{n\times l}$ where $k < l \ll n$. is generated and the $\boldsymbol{QR}$ decomposition of $\boldsymbol{A\Omega}$ is computed. Then the matrix $\boldsymbol{B} = \boldsymbol{Q}^\intercal\boldsymbol{A} \in \mathbb{R}^{k\times n}$ is stored. Next, the SVD of the small matrix $\boldsymbol{B}$ is calculated, yielding $\boldsymbol{B} \approx \tilde{\boldsymbol{U}}_{\boldsymbol{k}}\boldsymbol{S_k V_k^\intercal}$. Finally, the matrix $\boldsymbol{U_k} = \boldsymbol{Q}\tilde{\boldsymbol{U}}_{\boldsymbol{k}}$ is formed, allowing for the construction of the SVD $\boldsymbol{A} = \boldsymbol{U_k S_k V_k^\intercal}$ (Halko *et al.* 2011). In this work, the blocked single-pass SVD developed by Yu *et al.* (2017) is the algorithm selected from this broader class of methods.

Method	Computational complexity	Disk storage	RAM usage
ID	$\mathcal{O}(mnk)$	$k(m+n)$	$k(m+n)+mn$
Sub-sampled ID	$\mathcal{O}(m_1nk)$	$k(m+n)$	$k(m_1+n)+m_1n$
Single-Pass SVD	$\mathcal{O}(mnk)$	$k(m+n+k)$	$\ell(m+2n)$

TABLE 1. Computational complexity of the ID, sub-sampled ID, and single-pass SVD. Variable $m_1 \ll m$ represents the dimension of the input matrix after sub-sampling in sub-sampled ID. Variable $\ell \approx k$ in column 3 is the sub-sampled dimension of the matrix in single-pass SVD.

2.2. *Blocked single-pass SVD*

The blocked single-pass SVD developed by Yu *et al.* (2017) is a randomized SVD method which generates a truncated SVD of the solution matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ of rank k, i.e., $\boldsymbol{A} \approx \boldsymbol{U_k S_k V_k^\mathsf{T}}$. This yields a compression factor of approximately k/m when $m \gg n$, k/n when $n \gg m$, and $2k/m$ when $m \approx n$. The truncated SVD is the theoretically most accurate rank k approximation of a matrix, but requires $\mathcal{O}(k(m+n+k))$ storage, which is slightly more than that needed by the ID. Like the ID, the other matrix decomposition examined, this method has computational complexity $\mathcal{O}(mnk)$, but an approximate processing storage requirement of $k(m+2n)$ floats (Yu *et al.* 2017), which is lower than that of the ID which only uses $k(m+n)+mn$ floats (see Table 1). Complete implementation details of the blocked single-pass randomized algorithm are given in Algorithm 2 of the Appendix.

The authors in Yu *et al.* (2017) note that their blocked algorithm, which was inspired by algorithms of similar structure presented in Halko *et al.* (2011), resolves the accuracy issues associated with the single-pass algorithms and maintains accuracy at high target rank values. With respect to compression of turbulent flow data, using this blocked single-pass algorithm is quite effective — similar to the myriad variations of SVD algorithms. This algorithm is considerably RAM efficient, allowing for compression of files as large as 150 GB using less than 1 GB of RAM (Yu *et al.* 2017). The primary drawbacks of the SVD is its runtime requirement relative to the subsampled ID, which is discussed in this next section.

2.3. *Interpolative decomposition*

The ID produces a factorization of a matrix $\boldsymbol{A}$ of the form

$$\boldsymbol{A} \approx \boldsymbol{A}(:,\mathcal{I})\boldsymbol{P}, \tag{2.3}$$

where $\boldsymbol{A}(:,\mathcal{I}) \in \mathbb{R}^{m\times k}$ is a set of columns of $\boldsymbol{A}$, referred to as the column skeleton, and $\boldsymbol{P} \in \mathbb{R}^{k\times n}$ is a coefficient matrix such that $\boldsymbol{P}(\mathcal{I},:) = \boldsymbol{I}$, with $\boldsymbol{I}$ the identity matrix. The computational complexity of the method for compressing an input matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ of numerical rank k is $\mathcal{O}(mnk)$. Thus, when trying to compress the column space of low-rank data with a large row space, efficiency is greatly enhanced by reducing the dimension of the row space, i.e., reducing m. The process of reducing this dimension is made simple when computing an ID. Because the ID is computed using information from the column space of $\boldsymbol{A}$, a coarsened version of $\boldsymbol{A}$, $\boldsymbol{A_c} = \boldsymbol{A}(\mathcal{J},:)$, may be passed as input into the algorithm. This generates an ID $\boldsymbol{A_c} \approx \boldsymbol{A_c}(:,\mathcal{I}_c)\boldsymbol{P_c}$, which is also a valid ID for the original matrix $\boldsymbol{A}$, i.e., $\boldsymbol{A} \approx \boldsymbol{A}(:,\mathcal{I}_c)\boldsymbol{P_c}$. Since the ID is of computational complexity

Sub-sampling factor	Sub-sampled ID error	Sub-sampled ID runtime
1	2.3×10^{-5}	1
21	1.2×10^{-4}	8.8×10^{-3}
41	7.2×10^{-4}	7.6×10^{-3}
61	2.4×10^{-3}	7.4×10^{-3}

TABLE 2. Relative error in Frobenius norm and runtime (normalized to that of the full ID) of the sub-sampled ID for a channel flow at $Re_\tau = 180$ on a 130×130 grid corresponding to the y-z plane. For a fixed target rank value, 20 trials were run with runtimes measured independently of one another.

$\mathcal{O}(mnk)$, reducing the row space via the index vector $\mathcal{J}$ to dimension m_1, where $m_1 \ll m$ and m_1 is the length of $\mathcal{J}$, yields a complexity of $\mathcal{O}(m_1 nk)$ as opposed to $\mathcal{O}(mnk)$. In this application of column ID, this is achieved through uniform sub-sampling of the row space of $\boldsymbol{A}$, which substantially expedites the computation of $\mathcal{I}$ and $\boldsymbol{P}$ (see Tables 2, 3). This methodology entails extracting every s^{th} entry in the row space of $\boldsymbol{A}$ in Eq. (1.1), viz. every s^{th} grid-point in the mesh prior to reshaping of the data. The algorithmic implementation of the full ID is detailed in Algorithm 1 of the Appendix.

The sub-sampled ID theoretically outperforms the other methods explored in this work in terms of runtime and storage requirements. Its superior performance with respect to disk storage is particularly pronounced in this specific application, as the entire skeletonized matrix is not stored in memory, just the corresponding index set $\mathcal{I}$ and coefficient matrix $\boldsymbol{P}$, from which velocity data can be reconstructed. The ID without sub-sampling is significantly less time and RAM efficient than the blocked single-pass SVD, and therefore requires aggressive sub-sampling prior to compression in all cases. Though certain optimizations for random compression exist, namely the use of structured random matrices such as the sub-sampled random Fourier transform (SRFT) and randomized Givens rotations, they require $\mathcal{O}(m^2)$ storage in RAM, where m is the dimension of the space being compressed, i.e., the row space of $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, making them scale poorly (Halko *et al.* 2011) in the problem considered in this work.

3. Numerical experiments

The compression efficiency and reconstruction accuracy of the full ID, sub-sampled ID, and blocked single-pass SVD methods are investigated using data extracted from DNS of wall-bounded turbulent flow using the Soleil-MPI low-Mach-number flow solver (Esmaily-Moghadam *et al.* 2015). In particular, the canonical periodic channel flow at friction Reynolds number $Re_\tau = 180$ is selected as the test case. 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. 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

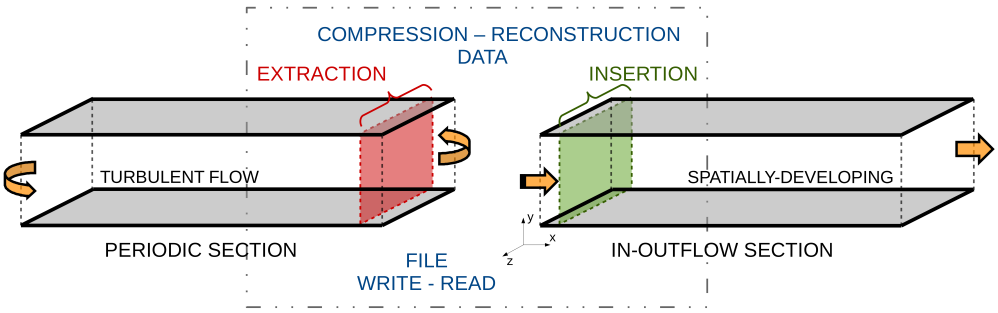

FIGURE 1. Computational setup of the extraction, compression, reconstruction, and insertion data steps. Left domain: periodic section from which planar data are extracted and compressed. Right domain: inflow-outflow section where data are reconstructed and inserted.

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 $y^+ = yu_\tau/\nu = 0.1$ and with resolutions in the range $0.1 < \Delta y^+ < 8$. This grid arrangement corresponds to a DNS of size $256 \times 128 \times 128$.

3.1. *Extraction and reinsertion of planar flow data*

As introduced in Section 1, limitations in disk storage and bandwidth with respect to floating-point capacity are encountered in different problems involving high-dimensional flow data, e.g., turbulence dynamics, outer-loop studies, recycling inflow methods, etc. Motivated by the interest in performing uncertainty quantification of spatially developing turbulent flows by means of Monte Carlo-type (MC) sampling approaches, in which large numbers of model evaluations, e.g., $\mathcal{O}(10^3)$, for different input values are required, this work focuses on the investigation of the performance of compression methods for time-decoupled strong recycling turbulent inflow boundary conditions. As illustrated in Figure 1, the idea is to compress and save to file the temporal evolution of the velocity field at the outlet plane of the channel for a few FTTs to be later used as inflow conditions for different spatially developing channel flow samples. Based on the bulk velocity, $u_b = 1/\delta \int_0^\delta \langle u \rangle \, dy$, and the length of the channel, $L = 4\pi\delta$, a FTT is defined as $t_b = L/u_b$.

The final goal is to efficiently compress and store to disk the temporal evolution of the outlet velocity during runtime. However, as a first exploratory stage, the outlet velocity data are (1) written to file uncompressed for each time-integration step, (2) externally compressed, and (3) reconstructed by the flow solver and inserted at the inlet of the inflow-outflow channel section. This strategy facilitates analysis of the compression efficiency and reconstruction accuracy of the different compression algorithms considered. First, a periodic channel flow (left section in Figure 1) is initiated with a sinusoidal velocity field and advanced in time to reach turbulent steady-state conditions after several FTTs. 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 written to file for an entire FTT resulting in 25100 time slices. Next, the temporal data are externally compressed using the full ID, sub-sampled ID, and blocked single-pass SVD methods generating new data files that are orders of magnitude smaller in size. Then, for each method, the corresponding compressed data file is read at runtime by the flow solver, uncompressing

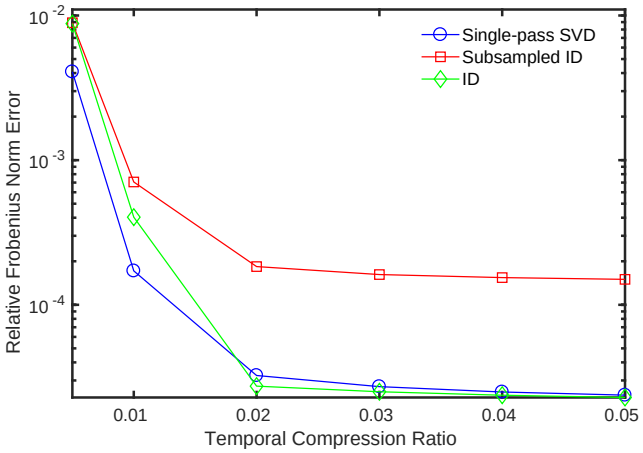

FIGURE 2. Compression accuracy of the blocked single-pass SVD, sub-sampled ID, and full ID for different target ranks. Data in this experiment are extracted from a channel flow at $Re_\tau = 180$.

Fraction of columns used	Sub-sampled ID speed-up	Single-pass SVD speed-up
5×10^{-3}	19.6	8.3
1×10^{-2}	41.7	16.4
5×10^{-2}	55.6	30.3

TABLE 3. Speed-up in runtime using the sub-sampled ID and single-pass SVD relative to the full ID for compressing data extracted from turbulent channel flow at $Re_\tau = 180$

the planar velocity field at each time step to use it as inflow for the spatially developing channel flow (right section in Figure 1). Finally, the flow field is advanced in time and first- and second-order statistics are collected over the reconstructed FTT to analyze compression accuracy.

3.2. *Data compression efficiency and accuracy*

The three algorithms studied in compression of turbulent flow data are the two variations of ID and the blocked single-pass SVD. The compressed matrices generated using both approaches require less than 1% of the storage of the original matrix. In the blocked single-pass SVD algorithm, the column and row space of the flow solution matrix, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ in Eq. (1.1), are compressed in a single pass over the data via blocked multiplication by Gaussian matrices. The blocked computations are computed on blocks of size 10 (parameter b in Algorithm 2). Using the single-pass SVD, accuracy in relative Frobenius-norm error of 10^{-5} is achieved. In the ID, the row space of the flow solution matrix $\boldsymbol{A}$, i.e., the reshaped flow spatial mesh, is compressed via uniform sub-sampling. On the 130×130 grid, the sub-sampling parameter used is $s = 12$, indicating that every 12^{th} grid point is sub-sampled to construct a coarse representation of $\boldsymbol{A}$. This choice of sub-sampling parameter balances optimal runtime while maintaining acceptable accuracy. For example,

	ID	Sub-sampled ID	Single-pass SVD
Compression factor	7.19×10^{-3}	7.19×10^{-3}	0.0125
Error tolerance in Frobenius norm	10^{-4}	10^{-4}	N/A
Over-sampling parameter	0	0	10
Fraction of time slices used	5.78×10^{-3}	5.78×10^{-3}	N/A
Runtime speedup (normalized to ID)	1	55.6	30.3

TABLE 4. Compression factor, stopping criteria, over-sampling parameter, time slices used, and runtime for the ID and blocked single-pass SVD for a channel flow at $Re_\tau = 180$.

as summarized in Table 2, aggressive sub-sampling of a 130×130 grid – corresponding to 16900 rows in the reshaped matrix $\boldsymbol{A}$ in Eq. (1.1) – achieves approximate optimal runtime while yielding a compression accuracy of 10^{-4}. With respect to the blocked single-pass SVD, accuracy is an order of magnitude greater than the sub-sampled ID for compressions involving a higher target rank, as demonstrated in Figure 2. The experiment referred to in Figure 2 was performed on the channel flow configuration over 2000 time steps, with 20 trials per target rank value run with the random matrices used to compress the matrix as it is streamed when regenerated in each run. The parameter k/n represents the fraction of columns sampled to construct the decomposition, i.e., the target rank k divided by n where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, and the error is measured using the relative Frobenius error of the decomposition, Eq. (2.1). The results demonstrate that the full ID and single-pass SVD are clearly the most accurate methods as target rank is increased. The sub-sampled ID, however, is the fastest of the three; it compresses a matrix roughly 60 times faster than the full ID and almost twice as fast as the single-pass SVD.

An important difference between the two algorithms is how their respective stopping criteria are defined. The blocked single-pass SVD requires knowledge of the target rank, i.e., compression factor, whereas the ID does not. The single-pass SVD takes as input a matrix, a target rank corresponding to compression dimension, and a block size. Consequently, in the blocked single-pass SVD the desired compression dimension of a matrix must be known *a-priori* when using the single-pass SVD. Without extensive knowledge of the flow being compressed, the accuracy of a decomposition generated via the single-pass 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 become a two-pass method. The ID, on the other hand, may take a compression factor or an error tolerance as input, and therefore does not require knowledge of the numerical rank of a matrix. However, this comes at the cost of the ID requiring more passes at the input. In this work, an error tolerance of 10^{-4} was used as a stopping criterion when computing the ID in cases in which the approximate numerical rank was not known in advance.

3.3. *Flow reconstruction accuracy*

The accuracy of compression methods is analyzed by using the compressed data as inlet velocity for the inflow-outflow channel flow configuration. As a first approximation to this study, flow statistics are averaged in time for a single reconstructed FTT, and in space along the stream-wise and span-wise directions. The quantities of interest (QoI) considered are the numerical friction, Re_τ, and bulk, $Re_b = 2u_b\delta/\nu$, Reynolds numbers, the skin-friction coefficient, $C_f = \tau_w/(1/2\rho u_b^2)$, the mean stream-wise velocity profile,

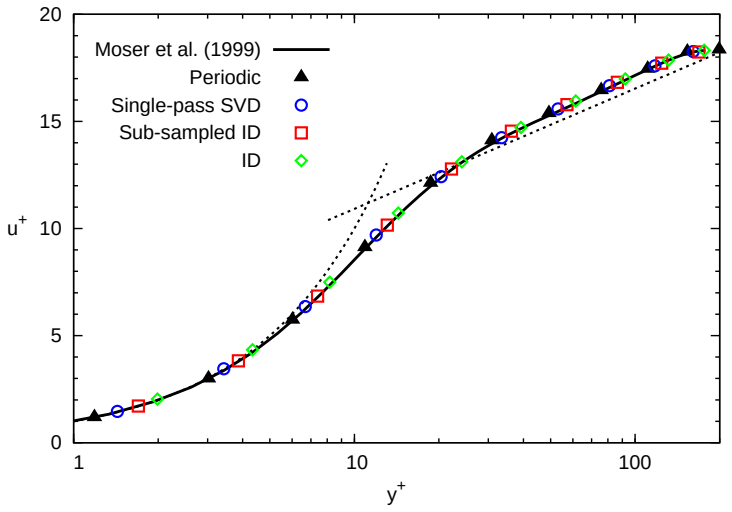

FIGURE 3. Mean stream-wise velocity profile (quantities in wall units). Moser *et al.* (1999) (black solid lines), periodic (black triangles), Single-pass SVD (blue circles), sub-sampled ID (red squares), ID (green diamonds).

	Analytical	Periodic	ID	Sub-sampled ID	Single-pass SVD
Re_τ	180	180	183	181	182
Re_b	≈ 5639	5643	5677	5675	5676
C_f	≈ 0.0082	0.0082	0.0083	0.0082	0.0083

TABLE 5. Friction, Re_τ, and bulk, Re_b, Reynolds numbers, and skin-friction coefficient, C_f, obtained from periodic and compressed data inflow-outflow channel flow configurations at $Re_\tau = 180$

$u^+ = \langle u \rangle / u_\tau$, 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 based on $Re_\tau \approx 0.09 Re_b^{0.88}$ (Pope 2000). 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 are imposed by Re_τ. Regarding mean and fluctuation velocity profiles, reference DNS data are widely available, for example, from Moser *et al.* (1999).

Compression accuracy results for the periodic and inflow-outflow (compressed data) channel flow configurations are shown in Table 5 and Figures 3 and 4. The first observation is that the results obtained from the periodic configuration agree well with the analytical solutions and reference DNS data. This outcome is important as the inflow-outflow channel calculations utilize compressed data from the periodic case as inlet velocity, and therefore analysis of compression accuracy significantly depends on the quality of the input data. The second observation is that the three strategies considered provide similar compression accuracies — evaluated through the reconstructed velocity field — and

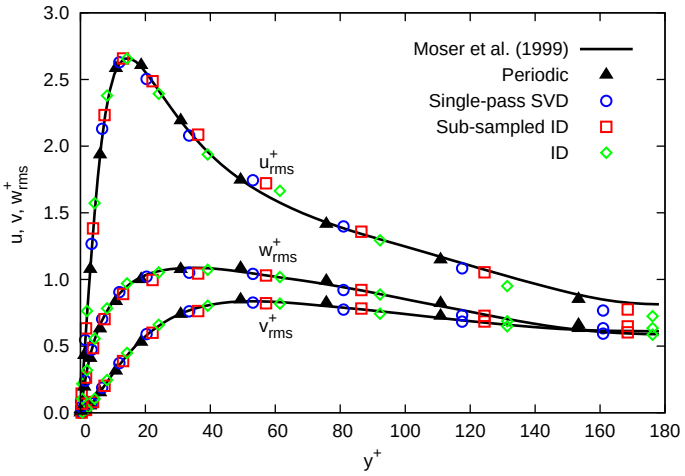

FIGURE 4. Root mean square velocity fluctuations (quantities in wall units). Moser *et al.* (1999) (black solid lines), periodic (black triangles), single-pass SVD (blue circles), sub-sampled ID (red squares), ID (green diamonds).

are in agreement with the analytical and periodic configuration values. In particular, the relative errors for the quantities in Table 5 are below 2%, and the mean stream-wise velocity and rms fluctuations perfectly capture the DNS reference solution. These results are very promising and motivate to continue exploring and analyzing this class of compression approaches. Future studies will consider the stream-wise evolution of the QoIs to characterize the development length required to fully recover periodic channel flow DNS statistics, as well as kinetic energy spectra analysis to quantify the sources of error as a function of turbulent flow scale.

4. Conclusions

Pass-efficient matrix decomposition methods for compression of high-dimensional turbulent flow velocity data have been investigated in this work: the ID, sub-sampled ID and blocked single-pass SVD. Results show that the blocked single-pass SVD algorithm (Yu *et al.* 2017) can be used effectively to obtain highly accurate compressions of turbulent flow data. On the other hand, the sub-sampled ID, the fastest and most disk-memory efficient of the algorithms, also performs well in compression of the flow data. Although the ID requires two passes at the input data and consequently consumes more RAM, the row space or column of the input matrix can be sub-sampled as it is read in order to mitigate the issues associated with overuse of memory. The ID, unlike the single-pass SVD, takes as input an error tolerance which functions as a stopping criterion, and therefore does not require *a-priori* knowledge of the compressibility of a dataset. The ID is also an intuitive decomposition of a matrix, interpolating its column space via the construction of a column skeleton of the input data matrix. This makes it a useful tool for domain experts who may find the SVD harder to interpret in the context of their raw data. The preliminary results suggest that both the blocked single-pass SVD and sub-sampled ID are suitable tools for compression of turbulent flow data. As these methods allow for *in situ* compression of RAM-prohibitive input data as they are streamed, they may help address scalability issues associated with large-scale turbulent flow data.

In terms of accuracy in compression of turbulent flows, evaluated by utilizing the compressed velocity field as inflow for a channel flow at $Re_\tau = 180$, all three methods proved successful in recovering DNS-like first- and second-order flow statistics. Note, however, that the QoIs are extracted by spatially averaging in the stream-wise and span-wise directions. This methodology is sufficient for an initial exploration of the different compression algorithms. However, as it is likely that a feeble development region may be generated at the inlet, the QoIs need to be further investigated at different points in the stream-wise direction of the flow to better assess compression accuracy. In addition, a useful exercise is to characterize compression error as a function of turbulent flow scale as a way to understand how the algorithms operate from a physical point of view. In this regard, ongoing work is focused on further exploring these methods for the channel flow case. Additionally, the application of the compression algorithms studied to more complex problems, e.g., higher Reynolds numbers, variable-density fluids and particle-laden flows, is being investigated.

Acknowledgments

This investigation was funded by the Advanced Simulation and Computing (ASC) program of the US Department of Energy's National Nuclear Security Administration via the PSAAP-II Center at Stanford, Grant #DE-NA-0002373. The third author 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.

Appendix

Column ID approximates a matrix as $\boldsymbol{A} \approx \boldsymbol{A}(:, \mathcal{I})\boldsymbol{P}$, where $\boldsymbol{A}(:, \mathcal{I})$ is a sub-sampled set of columns of $\boldsymbol{A}$ obtained using rank-revealing QR (Martinsson 2016). In the description of the algorithm, *mgsqr* refers to pivoted QR using a modified Gram-Schmidt procedure adapted from Golub & Van Loan (2012).

Algorithm 1 General column ID $\boldsymbol{A} \approx \boldsymbol{A}(:, \mathcal{I})\boldsymbol{P}$ (Martinsson 2016)

procedure ID($\boldsymbol{A} \in \mathbb{R}^{m \times n}$)
  $tol \leftarrow$ stopping tolerance
  $\boldsymbol{Q}$, $\boldsymbol{R}$, $\mathcal{I}, k \leftarrow mgsqr(\boldsymbol{A})$
  $\boldsymbol{T} \leftarrow (\boldsymbol{R}(1:k, 1:k))^{-1}\boldsymbol{R}(1:k, (k+1):n)$
  $\boldsymbol{P} \leftarrow zeros(k, n)$
  $\boldsymbol{P}(:, \mathcal{I}) \leftarrow [\boldsymbol{I}_k \quad \boldsymbol{T}]$
  $\mathcal{I} \leftarrow \mathcal{I}(1:k)$
  return $\mathcal{I}, \boldsymbol{P}$

The blocked single-pass SVD developed by Yu *et al.* (2017) returns a decomposition of the form $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\intercal$ in one pass by compressing the input matrix $\boldsymbol{A}$ *in situ*. This algorithm was developed as an improvement upon the single-pass method developed by Halko *et al.* (2011).

Algorithm 2 Blocked single-pass SVD $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\intercal$ (Yu *et al.* 2017)

procedure rSVD($\boldsymbol{A} \in \mathbb{R}^{m \times n}$)
  $k \leftarrow$ target rank
  $b \leftarrow$ block size
  instantiate $\boldsymbol{Q}, \boldsymbol{B}$
  $\boldsymbol{W} \leftarrow randn(n, l)$
  $\boldsymbol{G} \leftarrow []$
  $\boldsymbol{H} \leftarrow zeros(n, l)$
  while $\boldsymbol{A}$ is not entirely read through ***do***
  read the next set of rows in RAM $\boldsymbol{a}$
  $\boldsymbol{g} \leftarrow \boldsymbol{a}\boldsymbol{W}$
  $\boldsymbol{G} \leftarrow [\boldsymbol{G}; \boldsymbol{g}]$
  $\boldsymbol{H} \leftarrow \boldsymbol{H} + \boldsymbol{a}^\intercal \boldsymbol{g}$
  end while
  for i = 1, 2, ..., t ***do***
  $\boldsymbol{W}_i \leftarrow \boldsymbol{W}(:, (i-1)b + 1 : ib)$
  $\boldsymbol{Y}_i \leftarrow G(:, (i-1)b + 1 : ib) - \boldsymbol{Q}(\boldsymbol{B}\boldsymbol{W}_i)$
  $\boldsymbol{Q}_i, \boldsymbol{R}_i \leftarrow qr(\boldsymbol{Y}_i)$
  $\boldsymbol{Q}_i, \tilde{\boldsymbol{R}}_i \leftarrow qr(\boldsymbol{Q}_i - \boldsymbol{Q}(\boldsymbol{Q}^\intercal \boldsymbol{Q}_i))$
  $\boldsymbol{R}_i \leftarrow \tilde{\boldsymbol{R}}_i \boldsymbol{R}_i$
  $\boldsymbol{B}_i \leftarrow \boldsymbol{R}_i^{-\intercal}(\boldsymbol{H}(:, (i-1)b + 1 : ib)^\intercal - \boldsymbol{Y}_i^\intercal \boldsymbol{Q}\boldsymbol{B} - \boldsymbol{W}_i^\intercal \boldsymbol{B}^\intercal \boldsymbol{B})$
  $\boldsymbol{Q} \leftarrow [\boldsymbol{Q}, \boldsymbol{Q}_i]$ $\boldsymbol{B} \leftarrow [\boldsymbol{B}^\intercal, \boldsymbol{B}_i^\intercal]^\intercal$
  end for
  $\tilde{\boldsymbol{U}}, \boldsymbol{S}, \boldsymbol{V} \leftarrow svd(\boldsymbol{B})$;
  $\boldsymbol{U} \leftarrow \boldsymbol{Q}\tilde{\boldsymbol{U}}$;
  $\boldsymbol{U}(:, 1 : k)$; $\boldsymbol{V}(:, 1 : k)$; $\boldsymbol{S} \leftarrow \boldsymbol{S}(1 : k, 1 : k)$;
  return $\boldsymbol{U}, \boldsymbol{S}, \boldsymbol{V}$

REFERENCES

Ang, J., Evans, K., Geist, A., Heroux, M., Hovland, P., Marques, O., Curfman, L., Ng, E. & Wild, S. 2012 Workshop on extreme-scale solvers: transition to future architectures. *Tech. Rep.*. U.S. Department of Energy, Office of Advanced Scientific Computing Research.

Bourguignon, J.-L., Tropp, J. A., Sharma, A. S. & McKeon, B. J. 2014 Compact representation of wall-bounded turbulence using compressive sampling. *Phys. Fluids* **26**, 015109.

Burtscher, M., Mukka, H., Yang, A. & Hesaaraki, F. 2016 Real-time synthesis of compression algorithms for scientific data. In *Proc. Int. Conf. High Perfom. Comput. Netw. Stor. Anal.*

Eckart, C. & Young, G. 1936 The approximation of one matrix by another of lower rank. *Psychometrika* **1**, 211–218.

Engelson, E., Fritzson, D. & Fritzson, P. 2000 Lossless compression of high-volume numerical data from simulations. In *Proc. Data Compress. Conf.*

Esmaily-Moghadam, M., Jofre, L., Iaccarino, G. & Mani, A. 2015 A scalable geometric multigrid method for non-symmetric linear systems arising from elliptic equations. *Annual Research Briefs*, Center for Turbulence Research, Stanford University, pp. 211-223.

Farge, M. 1992 Wavelet transforms and their application to turbulence. *Annu. Rev. Fluid Mech.* **24**, 395–457.

Golub, G. H. & Van Loan, C. F. 2012 *Matrix Computations*. JHU Press.

Gu, M. & Eisenstat, S. C. 1996 Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM J. Sci. Comput.* **17**, 848–869.

Halko, N., Martinsson, P.-G. & Tropp, J. A. 2011 Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.* **53**, 217–288.

Hong, Y. P. & Pan, C.-T. 1992 Rank-revealing QR factorizations and the singular value decomposition. *Math. Comput.* **58**, 213–232.

Ishihara, T., Morishita, K., Yokokawa, M., Uno, A. & Kaneda, Y. 2016 Energy spectrum in high-resolution direct numerical simulations of turbulence. *Phys. Rev. Fluids* **1**, 082403.

Jofre, L., Domino, S. P. & Iaccarino, G. 2017*a* A framework for characterizing structural uncertainty in large-eddy simulation closures. *Flow Turbul. Combust.* **99**, 1–23.

Jofre, L., Geraci, G., Fairbanks, H. R., Doostan, A. & Iaccarino, G. 2017*b* Multi-fidelity uncertainty quantification strategies for large-scale multiphysics applications: PSAAP II particle-based solar energy receiver. In *Proc. 20th APS-SCCM Biennial Conf.*

Lozano-Durán, A. & Jiménez, J. 2014 Time-resolved evolution of coherent structures in turbulent channels: characterization of eddies and cascades. *J. Fluid Mech.* **759**, 432–471.

Martinsson, P.-G. 2016 Randomized methods for matrix computations and analysis of high dimensional data. In arXiv preprint, arXiv:1607.01649.

Masquelet, M., Yan, J., Dord, A., Laskowski, G., Shunn, L., Jofre, L. & Iaccarino, G. 2017 Uncertainty quantification in large eddy simulations of a rich-dome aviation gas turbine. *ASME Turbo Expo*, GT2017-64835.

Moore, G. E. 1965 Cramming more components onto integrated circuits. *Electronics* **38**, 114–117.

Moser, R. D., Kim, J. & Mansour, N. N. 1999 Direct numerical simulation of turbulent channel flow up to $Re_\tau = 590$. *Phys. Fluids* **11**, 943–945.

Pope, S. B. 2000 *Turbulent flows*. Cambridge University Press.

Ratanaworabhan, P., Ke, J. & Burtscher, M. 2006 Fast lossless compression of scientific floating-point data. In *Proc. Data Compress. Conf.*

Walter, C. 2005 Kryder's Law. *Sci. Am.* **293**, 20–21.

Wu, X. 2017 Inflow turbulence generation methods. *Annu. Rev. Fluid Mech.* **49**, 23–49.

Yu, W., Gu, Y., Li, J., Liu, S. & Li, Y. 2017 Single-pass PCA of large high-dimensional data. In arXiv preprint, arXiv:1704.07669.